SEVENTH FRAMEWORK PROGRAMME
Research Infrastructures

INFRA-2012-2.3.1 – Third Implementation Phase of the European High Performance Computing (HPC) service PRACE

PRACE-3IP

PRACE Third Phase Implementation Project

Grant Agreement Number: RI-312763

D7.2.1
A Report on the Survey of HPC Tools and Techniques
Final

Version: 1.0
Author(s): Michael Lysaght, ICHEC
Bjorn Lindi, SIGMA-NTNU
Vit Vondrak, VSB
John Donners, SURFSARA
Marc Tajchman, GENCI-CEA

Date: 29.04.2013

Project and Deliverable Information Sheet

PRACE Project	**Project Ref. №: RI-312763**	
	Project Title: PRACE Third Phase Implementation Project	
	Project Web Site: http://www.prace-project.eu	
	Deliverable ID: < **D7.2.1**>	
	Deliverable Nature: <Report >	
	Deliverable Level: PU	**Contractual Date of Delivery:** 30 / 04 / 2013
		Actual Date of Delivery: 30 / 04 / 2013
	EC Project Officer: Leonardo Flores Añover	

* - The dissemination level are indicated as follows: **PU** – Public, **PP** – Restricted to other participants (including the Commission Services), **RE** – Restricted to a group specified by the consortium (including the Commission Services). **CO** – Confidential, only for members of the consortium (including the Commission Services).

Document Control Sheet

Document	**Title: A Report on the Survey of HPC Tools and Techniques**	
	ID: D7.2.1	
	Version: <1.0 >	**Status:** ***Final***
	Available at: http://www.prace-project.eu	
	Software Tool: Microsoft Word 2007	
	File(s): D7.2.1.docx	
Authorship	**Written by:**	Michael Lysaght, ICHEC Bjorn Lindi, SIGMA-NTNU Vit Vondrak, VSB John Donners, SURFSARA Marc Tajchman, GENCI-CEA

	Contributors:	Jorge Rodriguez, BSC Abdou Abdel Rehim, CASTORC Sami Saarinen, CSC Iain Bethune, EPCC Fiona Reid, EPCC Lorna Smith EPCC Konstantinos Nikas, Niko Anastopoulos, GRNET Emma Hogan, ICHEC Ben Eagan, ICHEC Buket Gursoy, ICHEC Petar Jovanovic, IPB Dusan Stankovic, IPB Thomas Ponweiser, JKU Peter Stadelmeyer, JKU Nevena Ilieva, NCSA Valentin Pavlov, NCSA Henrik Nagel, SIGMA-NTNU Jan Christian Meyer, SIGMA-NTNU Chandan Basu SNIC-LiU Ahmet Duran, ITU-UHeM Serdar Celebi ITU-UHeM Mehmet Tuncel ITU-UHeM Ata Turk, Bilkent-UHeM Cevdet Aykanat, Bilkent-UHeM Can Ozturan, Bogazici-UHeM Agnieszka Kwiecien, WCSS Mariusz Uchronski, WCSS Andrew Sunderland, STFC Xiaohu Guo, STFC Will Sawyer, CSCS
	Reviewed by:	Jarno Laitinen, CSC Florian Berberich, JUELICH
	Approved by:	MB/TB

ocument Status Sheet

Version	Date	Status	Comments
0.1	11/04/2013	Draft	First draft of reports completed.
0.2	25/04/2013	Draft	Response to reviewers comments
1.0	29/04/2013	Final version	

Document Keywords

Keywords:	PRACE, HPC, Research Infrastructure

Table of Contents

List of Figures

List of Tables

References and Applicable Documents

[1] European Exascale Software Initiative (EESI) homesite: http://www.eesi-project.eu/pages/menu/homepage.php
[2] TEXT exascale project homesite: http://www.project-text.eu/
[3] CRESTA exascale project homesite: http://cresta-project.eu/
[4] DEEP exascale project homesite: http://www.deep-project.eu/deep-project/EN/Home/home_node.html
[5] Mont-Blanc exascale project homesite: http://www.montblanc-project.eu/
[6] IESP project homesite: http://www.exascale.org/bdec/
[7] US Department of Energy Exascale Worskshop Panel Report (2010), pdf: http://www.exascale.org/mediawiki/images/4/48/TrivelpieceExascaleWorkshop.pdf
[8] Chapel homesite: http://chapel.cray.com/index.html
[9] B Chamberlian et al, 'Chapel Support for Heterogeneous Architectures via Hierarchical Locales', PGAS-X Workshop, Santa Barbara, USA Oct. 2012, pdf: http://chapel.cray.com/presentations/ChapelForPGASX-presented.pdf
[10] US Department of Energy Exascale Research Conference 2012 homesite: http://exascaleresearch.labworks.org/apr2012/conference
[11] MPI: A Message-Passing Interface Standard Version 3.0, pdf: http://www.mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf
[12] OpenMP version 3.0, complete specification, pdf: http://www.openmp.org/mp-documents/spec30.pdf
[13] NVIDIA CUDA homesite http://www.NVIDIA.com/object/cuda_home_new.html
[14] OpenCL Khronos Group homesite: http://www.khronos.org/opencl/
[15] OpenACC forum website: http://www.openacc-standard.org/
[16] Intel Thread Building Blocks (TBB) homesite: http://threadingbuildingblocks.org/
[17] Cilk Plus homesite: http://cilkplus.org/
[18] Intel SDK for OpenCL Applications 2013, product brief: http://software.intel.com/en-us/vcsource/tools/opencl-sdk
[19] OmpSs homesite: http://pm.bsc.es/ompss
[20] CoArray Fortran homesite: http://www.co-array.org/

[21] Unified Parallel C (UPC), 'official website', http://upc.gwu.edu/
[22] X10 Programming Language homesite: http://x10-lang.org/
[23] Global Arrays Toolkit homesite: http://www.emsl.pnl.gov/docs/global/
[24] MPI Forum homesite: http://www.mpi-forum.org/
[25] MPICH homesite: http://www.mpich.org/
[26] MVAPICH homesite: http://mvapich.cse.ohio-state.edu/
[27] OpenMPI homesite: http://www.open-mpi.org/
[28] MVAPICH2-X, Unified MPI+PGAS Communication Runtime over OpenFabrics/Gen2 for Exascale Systems', http://mvapich.cse.ohio-state.edu/overview/mvapich2x/, J. Jose, M. Luo, S. Sur and D. K. Panda, Unifying UPC and MPI Runtimes: Experience with MVAPICH, Fourth Conference on Partitioned Global Address Space Programming Model (PGAS10), Oct. 2010
[29] 'Introducing OpenSHMEM', B Chapman et al, published in PGAS '10 Proceedings of the Fourth Conference on Partitioned Global Address Space Programming Model, article 2, DOI: 10.1145/2020373.2020375
[30] H Wang et al, 'MVAPICH2-GPU: Optimized GPU to GPU Communication for InfiniBand Clusters', Int'l Supercomputing Conference (ISC), June 2011, pdf: http://nowlab.cse.ohio-state.edu/publications/conf-presentations/2011/hao-isc11-slides.pdf
[31] NVIDIA GPUDirect, 'Technical Overview', https://developer.NVIDIA.com/gpudirect
[32] S Potluri, et al, 'Intra-MIC MPI Communication using MVAPICH2: Early Experience', published in proceedings of TI-HPCS, 2012, http://nowlab.cse.ohio-state.edu/publications/conf-papers/2012/potluri-tihpcs2012-paper.pdf
[33] PRACE deliverable D6.4, PRACE-PP, 'Report on Approaches to Petascaling', http://www.prace-ri.eu/IMG/pdf/D6-4.pdf
[34] Jean-Marc Molines, Nicole Audiffren and Albanne Lecointre, 'High resolution ocean simulations with NEMO modeling system', PRACE whitepaper, PRACE-1IP, , http://www.prace-ri.eu/IMG/pdf/High_resolution_ocean_simulations_with_NEMO_modeling_system.pdf
[35] WP8 PRACE-2IP F2F /Workshop. Progress reports can be found at: https://hpcforge.org/plugins/mediawiki/wiki/pracewp8/index.php/F2f5
[36] 'Four applications sustain 1 PF on Blue Waters', More information can be found in pdf: http://www.ncsa.illinois.edu/News/Stories/BW1year/apps.pdf
[37] 'Record simulations conducted on Lawrence Livermore supercomputer', https://www.llnl.gov/news/newsreleases/2013/Mar/NR-13-03-05.html
[38] OSIRIS code overview: http://plasmasim.physics.ucla.edu/codes/osiris
[39] ParaStation MPI, product description in pdf: http://docs.par-tec.com/pdf/SPD-MPI2-5.0.4en.pdf
[40] The OpenMP API specification for parallel programming homesite: http://openmp.org/wp/
[41] OpenMP version 4.0, public release candidate: http://www.openmp.org/mp-documents/OpenMP_4.0_RC2.pdf
[42] Best Practice Guide-Intel Xeon Phi, PRACE website: http://www.prace-ri.eu/Best-Practice-Guide-Intel-Xeon-Phi
[43] PRACE deliverable D7.5, PRACE-1IP, http://www.prace-ri.eu/IMG/pdf/d7.5_1ip.pdf
[44] Fabio Affinito, Andrew Emerson, Leandar Litov, Peicho Petkov, Rossen Apostolov, Lilit Axner, Berk Hess, Erik Lindahl and Maria Francesca Iozzi, 'Performance Analysis and Petascale Enabling of GROMACS', PRACE whitepaper, PRACE-1IP , pdf: http://www.prace-ri.eu/IMG/pdf/Performance_Analysis_and_Petascaling_Enabling_of_GROMACS.pdf

[45] Michael Lange, Gerard Gorman, Michele Weiland, Lawrence Mitchell and James Southern, 'Achieving Efficient Strong Scaling with PETSc using Hybrid MPI/OpenMP Optimisation', preprint pdf: http://arxiv.org/pdf/1303.5275v1.pdf]
[46] Tim Cramer, Dirk Schmidl, Michael Klemmy and Dieter an Mey, 'OpenMP Programming on Intel Xeon Phi Coprocessors: An Early Performance Comparison', , Proceedings of the Many-core Applications Research Community (MARC) Symposium at RWTH Aachen University, 2012, pdf: http://www.lfbs.rwth-aachen.de/users/stefan/marc2012/07_Cramer.pdf
[47] MALI OpenCL SDK homesite: http://malideveloper.arm.com/develop-for-mali/sdks/mali-opencl-sdk/
[48] Michael Lysaght, Mariusz Uchronski, Agnieszka Kwiecien, Marcin Gebarowski, Peter Nash, Ivan Girotto and Ilian T.Todorov, 'Benchmarking and analysis of DL_POLY 4 on GPU clusters', PRACE whitepaper, PRACE-1IP, pdf: http://www.prace-ri.eu/IMG/pdf/benchmarking_and_analysis_of_dl_poly_4_on_gpu_clusters.pdf
[49] PRACE deliverable D9.2.2, ' Final Software Evaluation Report', PRACE-1IP, pdf: http://www.prace-ri.eu/IMG/pdf/D9-2-2_1ip.pdf
[50] Mariusz Uchronski, Marcin Gebarowski, Agnieszka Kwiecien, 'Optimization of SHAKE and RATTLE Algorithms', PRACE whitepaper,PRACE-2IP, pdf: http://www.prace-ri.eu/IMG/pdf/optimization_of_shake_and_rattle_algorithms.pdf
[51] B. Bland,'Titan-Early Experience with the Titan System at Oak Ridge National Laboratory', SC12 presentation, more details in pdf: http://developer.download.NVIDIA.com/GTC/PDF/GTC2012/PresentationPDF/BuddyBland_Titan_SC12.pdf
[52] W M Brown, A Kohlmeyer, S J Plimpton, and A N Tharrington, 'Implementing Molecular Dynamics on Hybrid High Performance Computers – Particle-Particle Particle-Mesh', Computer Physics Communications 183, p4 (2012).
[53] Geryon Library homesite: http://users.nccs.gov/~wb8/geryon/index.htm
[54] VexCL: Vector expression template library for OpenCL, homesite: https://github.com/ddemidov/vexcl
[55] ADAPT 2013 Workshop, homsite: http://homepages.inf.ed.ac.uk/cdubach/adapt2013/
[56] The OpenACC Application Programming Interface, Version 2.0 Public Comment Draft, March 13th 2013 http://openacc.org/sites/default/files/OpenACC-2.0-draft.pdf
[57] OpenMP Technical Report 1 on Directives for Attached Accelerators, pdf: http://www.openmp.org/mp-documents/TR1_167.pdf
[58] Ben Eagan and Gilles Civario, 'Investigating Performance Benefits from OpenACC Kernel Directives', PRACE whitepaper, PRACE-2IP, pdf: http://www.prace-ri.eu/IMG/pdf/wp64_investigating_openacc.pdf
[59] J M Levesque. R Sankaran and R Grout, 'Hybridizing S3D into an exascale application using OpenACC: an approach for moving to multi-petaflops and beyond', SC '12 Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, Salt Lake City, Utah, 2012.
[60] Private Communication with CRESTA team, CRESTA deliverable D3.5.1, 'Compiler support for exascale' - to be published.
[61] Private Communication with CRESTA team- A Feasibility Study on the Hybridisation of HemeLB, to be published
[62] PRACE deliverable, D9.2.1, 'First Report on Multi-Petascale to Exascale Software'PRACE-1IP, http://www.prace-ri.eu/IMG/pdf/d9.2.1_1ip.pdf
[63] Jiri Dokulil, Enes Bajrovic, Siegfried Benkner, Sabri Pllana, Martin Sandrieser, Beverly Bachmayer 'Efficient Hybrid Execution of C++ Applications using Intel Xeon Phi Coprocessor' preprint at http://arxiv.org/abs/1211.5530
[64] deal.II homesite: : http://www.dealii.org/

[65] Trilinos homesite: http://trilinos.sandia.gov/
[66] J. Eisenlor, D. E. Hudak, K. Tomko, and T. C. Prince, 'Dense linear algebra factorization in OpenMP and Cilk Plus on Intel MIC: Development experiences and performance analysis', In TACC-Intel Highly Parallel Computing Symposium, 2012.
[67] Alexei Strelchenko, Marcus Petschlies and Giannis Koutsou, 'Extending the QUDA library for Domain Wall and Twisted Mass fermions', PRACE whitepaper, PRACE-1IP, pdf: http://www.prace-ri.eu/IMG/pdf/extending_the_quda_library_for_domain_wall_and_twisted_mass_fermions.pdf
[68] NVIDIA report on Hyper-Q technology: http://blogs.nvidia.com/2012/08/unleash-legacy-mpi-codes-with-keplers-hyper-q/
[69] C. Baker, G. Davidson, T. M. Evans, S. Hamilton, J. Jarrell and W. Joubert, 'High Performance Radiation Transport Simulations - Preparing for TITAN', Paper at SC12 (2012) pdf: http://conferences.computer.org/sc/2012/papers/1000a069.pdf
[70] Jesus Labarta, 'StarSs: a Programming Model for the Multicore Era', PRACE website: http://www.prace-ri.eu/IMG/pdf/08_starss_jl.pdf
[71] GasNet homesite: http://gasnet.cs.berkeley.edu/
[72] Claudia Rosas, Vladimir Subotic, Jose Carlos Sancho and Jesus Labarta, 'Analysis and Optimization of a Hybrid Linear Equation Solver using Task-Based Parallel Programming Models', PRACE whitepaper, PRACE-2IP, pdf: http://www.prace-ri.eu/IMG/pdf/wp61_analysis_and_optimization_of_hybrid_linear_equations_solver_using_task-based_parallel_prog.pdf
[73] J Bueno et al, 'Productive cluster programming with OmpSs', Euro-Par 2011 Parallel Processing, 555-566 (2011)
[74] J Bueno et al, 'Productive Programming of GPU Clusters with OmpSs', Parallel & Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International (2012)
[75] Private Communication with the DEEP team
[76] Fortran Wiki: http://fortranwiki.org/fortran/show/HomePage
[77] Rice University Corray Fortran homesite: http://caf.rice.edu/
[78] CRESTA, Collaborative Research into Exascale Systemware Tools & Applications, George Mozdzynski (ECMWF) - IS-ENES 2nd HPC workshop;pdf: https://is.enes.org/the-project/presentation-on-is-enes-slides/is-enes-2nd-hpc-workshop-presentations-february-2013/session-1-status-of-eu-exascale-projects/CRESTA.pdf/at_download/file
[79] Unified Parallel C "official website": http://upc.gwu.edu/
[80] Pierre-Francois Lavallee, Guillaume Colin de Verdiere, Philippe Wautelet, Dimitri Lecas and Jean-Michel Dupays, 'Porting and Optimizing HYDRO to new platforms and programming paradigms - lessons learnt'PRACE whitepaper, PRACE-1IP, pdf: http://www.prace-ri.eu/IMG/pdf/porting_and_optimizing_hydro_to_new_platforms.pdf
[81] Intel Exascience Lab homesite: http://www.exascience.com/
[82] B. Verleyea et al, 'Implementation of a 2D electrostatic Particle-in-Cell algorithm in unified parallel C with dynamic load-balancing' Computers & Fluids, 2012, In Press, http://www.sciencedirect.com/science/article/pii/S0045793012003362
[83] James Dinan, Pavan Balaji, Ewing Lusk, P. Sadayappan, Rajeev Thakur. Proc. 7th ACM Conf. on Computing Frontiers (CF). Bertinoro, Italy. May 17-19, 2010.
[84] Alan Gray, An Evaluation of UPC in the Ludwig Application, Proceedings of CUG 2009 Conference, Atlanta, Georgia, May 2009, pdf: http://www2.epcc.ed.ac.uk/~alang/publications/GrayCUG2009.pdf
[85] Albert Sidelnik, Saeed Maleki, Bradford L. Chamberlain, María J. Garzarán, David Padua, 'Performance Portability with the Chapel Language', *IPDPS 2012*, May 2012, pdf: http://polaris.cs.uiuc.edu/~asideln2/ipdps12.pdf

[86] Vassily Litvinov et al, 'Multiresolution Parallel Programming with Chapel', HPC Advisory Council Conference, Sept. 2012, pdf: http://www.bsc.es/sites/default/files/public/mare_nostrum/hpcac2012-14_cray.pdf
[87] PRACE deliverable D6.6, PRACE-1IP, 'Report on petascale software libraries and programming models' http://www.prace-ri.eu/IMG/pdf/D6-6.pdf
[88] B Chamberlain et al, 'HPCC Benchmarks in Chapel' pdf: http://www.hpcchallenge.org/presentations/sc2009/ChapelAtHPCCBOF2009.pdf
[89] IBM DeveloperWorks homesite: http://www.ibm.com/developerworks/
[90] Marc Tajchman, 'Parallelization Using a PGAS Language such as X10 in HYDRO and Triton', PRACE Whitepaper, PRACE-2IP, http://www.prace-ri.eu/IMG/pdf/wp63_parallelization_using_a_pgas_language_such_as_x10_in_hydro_and_triton.pdf
[91] HPC Challenge homesite: www.hpcchallenge.org
[92] Olivier Tardieu et al, 'X10 for productivity and performance at scale' 2012 HPC Challenge Class 2, pdf: http://www.hpcchallenge.org/presentations/sc2012/x10-hpcc.pdf
[93] Anuchem homesite: http://cs.anu.edu.au/~Josh.Milthorpe/anuchem.html
[94] Jarek Nieplocha and Bryan Carpenter, 'ARMCI: A Portable Remote Memory Copy Library for Distributed Array Libraries and Compiler Run-Time Systems', Lecture Notes in Computer Science, Springer, 1999
[95] Pavlov V and Petkov P 'Data I/O Optimization in GROMACS using Global Arrays Toolkit', PRACE Whitepaper, PRACE-1IP, pdf: http://www.prace-ri.eu/IMG/pdf/Data_IO_Optimization_in_GROMACS_Using_the_Global_Arrays_Toolkit-2.pdf
[96] Apra, E, Harrison, R, deJong, W et al 2009, 'Liquid Water: Obtaining the Right Answer for the Right Reasons', in Janet Brown (ed.), Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, Association for Computing Machinery Inc (ACM), USA, pp. 7pp., 2009
[97] James Dinan et al 'Supporting the Global Arrays PGAS Model Using MPI One-Sided Communication', Proc. 26th Intl. Parallel and Distributed Processing Symp. (IPDPS). Shanghai, China. May 2012, pdf: http://www.mcs.anl.gov/~dinan/pubs/2012/dinan_ipdps12.pdf
[98] B Mohr. 'Survey of System Software Stacks in the IESP Community', link to excel sheet: http://www.exascale.org/mediawiki/images/0/07/Iesp-sfo-mohr-stack-survey-2.xls
[99] PRACE deliverable, D6.3.1, PRACE-PP, 'Report on available Performance Analysis and Benchmark Tools, Representative Benchmark', pdf: http://www.prace-ri.eu/IMG/pdf/D6-3-1.pdf
[100] HOPSA further information: http://www.vi-hps.org/projects/hopsa/overview/
[101] SCORE-P further information: http://www.vi-hps.org/projects/score-p/
[102] DEEP 'Midterm management report at month 6', deliverable D1.2, pdf: http://www.fz-juelich.de/SharedDocs/Downloads/DEEP-PROJECT/EN/Deliverables/deliverable-D1.2.pdf?__blob=publicationFile
[103] TAU Profiler homsite: http://www.cs.uoregon.edu/research/tau/home.php
[104] Sameer Shende , 'Scalability Improvements in the TAU Performance System' , Workshop on Extreme Scale Performance Tools SC'12 , pdf: http://www.vi-hps.org/upload/program/espt-sc12/vi-hps-espt-SC12-Workshop-Shende.pdf
[105] S. Shende and A. D. Malony, 'The TAU Parallel Performance System', International Journal of High Performance Computing Applications, SAGE Publications, 20(2):287-331, Summer 2006, pdf: http://www.cs.uoregon.edu/research/paracomp/papers/ijhpca05.tau/ijhpca_tau.pdf

[106] A D Malony, S Shende, W Spear, S Biersdorff, S Millstein, 'TAU Performance System and GPUs', Keeneland Tutorial, April 14-15, 2011, pdf: http://www.cc.gatech.edu/~vetter/keeneland/tutorial-2011-04-14/10-tau-gpu-tutorial-part1.pdf
[107] PAPI homsite: http://icl.cs.utk.edu/papi/
[108] CUPTI: for more information see: http://docs.NVIDIA.com/cuda/cupti/index.html
[109] Orio: An Annotation-Based Empirical Performance Tuning Framework, homesite: http://trac.mcs.anl.gov/projects/performance/wiki/Orio
[110] Jeffrey Vetter, Allen Malony, Philip Roth, Kyle Spafford, Jeremy Meredith,'Scalable Heterogeneous Computing on GPU Clusters', Tutorial at SC12
[111] A. Mametjanov, D. Lowell, C. Ma, B. Norris, 'Autotuning Stencil-Based Computations on GPUs', Preprint ANL/MCS-P2094-0512, May 2012, pdf: http://www.mcs.anl.gov/uploads/cels/papers/P2094-0512.pdf
[112] Jussi Enkovaara, Martti Louhivuori, Petar Jovanovic, Vladimir Slavnic, Mikael Rannar, 'Optimizing GIPAW', PRACE whitepaper, PRACE-1IP, pdf: http://www.prace-ri.eu/IMG/pdf/Optimizing_GPAW.pdf
[113] Chandan Basu, 'High Resolution EC Earth Porting and Benchmarking on Curie', PRACE whitepaper, PRACE-1IP, pdf: http://www.prace-ri.eu/IMG/pdf/High_Resolution_EC_Earth_Porting_and_Benchmarking_on_CURIE.pdf
[114] A Maloney, 'Targeting the TAU Performance System for Extreme Scale', LACSS 2008, pdf: http://www.lanl.gov/conferences/lacss/2008/slides/Malony_lacss_2008.pdf
[115] M Schulz, B Mohr and B Wylie, 'Supporting Performance Analysis and Optimization on Extreme-Scale Computer Systems' Tutorial at SC12
[116] Scalasca homesite: http://www.scalasca.org/
[117] D Bohme, M Geimer, F Wolf and L Arnold, 'Identifying the root causes of wait states in large-scale parallel applications', Published in Proceedings of the 39th International Conference on Parallel Processing (ICPP), San Diego, CA, September 2010, pdf: http://apps.fz-juelich.de/jsc-pubsystem/aigaion/attachments/rootcause.pdf-c6a59c39c21a36c19dcc01631d0c05f7.pdf
[118] PRACE deliverable, D7.4.1, PRACE-1IP, 'Applications and user requirements for Tier-0 systems', pdf: http://www.prace-ri.eu/IMG/pdf/D7-4-1_1ip.pdf
[119] PRACE deliverable, D7.1.2, 'Applications Enabling for Capability Science', pdf: http://www.prace-ri.eu/IMG/pdf/d7.1.2_1ip.pdf
[120] T Deloze, Y Hoarau and M Braza, 'Direct Numerical Simulation and Turbulence Modeling for Fluid-Structure Interaction in Aerodynamics', PRACE whitepaper, PRACE-1IP, pdf: http://www.prace-ri.eu/IMG/pdf/Direct_Numerical_Simulation_and_Turbulence_Modeling_for_Fluid.pdf
[121] A Schnurpfeil, A Schiller, F Janetzko, St. Meier and G Sutman, 'Semi-dilute polymer systems in shear flow - a particle based hydrodynamic approach', PRACE whitepaper, PRACE-1IP, pdf: http://www.prace-ri.eu/IMG/pdf/Semi-dilute_polymer_systems_in_shear.pdf
[122] R J. N. Wylie, M Geimer, B Mohr, 'Large-scale Performance Analysis of SWEEP3D with the Scalasca Toolset', Parallel Processing Letters, Vol. 20, No. 4 (2010) 397–414, pdf: http://www.google.ie/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&ved=0CDQQFjAB&url=http%3A%2F%2Fciteseerx.ist.psu.edu%2Fviewdoc%2Fdownload%3Fdoi%3D10.1.1.187.2873%26rep%3Drep1%26type%3Dpdf&ei=S51mUfSyGIKJhQfGj4DICQ&usg=AFQjCNHqKBA8WyUIi9dS5f9SoX9q7rvrDQ&bvm=bv.45107431,d.ZG4&cad=rja

[123] Dominic Eschweiler, Michael Wagner, Markus Geimer, Andreas Knüpfer, Wolfgang E. Nagel, Felix Wolf: Open Trace Format 2 - The Next Generation of Scalable Trace Formats and Support Libraries. In *Proc. of the Intl. Conference on Parallel Computing (ParCo), Ghent, Belgium, August 30 – September 2 2011*, volume 22 of *Advances in Parallel Computing*, pages 481–490, IOS Press, 2012. DOI: 10.3233/978-1-61499-041-3-481
[124] Private Communication with the DEEP team
[125] M-A Hermanns, S Krishnamoorthy and F Wolf, 'A scalable infrastructure for the performance analysis of passive target synchronization', Parallel Computing, Vol. 9 (3), p132 (2013) URL: http://www.sciencedirect.com/science/article/pii/S0167819112000762
[126] VampirTrace homesite: http://www.tu-dresden.de/die_tu_dresden/zentrale_einrichtungen/zih/forschung/software_werkzeuge_zur_unterstuetzung_von_programmierung_und_optimierung/vampirtrace
[127] Vampir homesite: http://www.vampir.eu/
[128] A. Knüpfer, H. Brunst, J. Doleschal, M. Jurenz, M. Lieber, H. Mickler, M. S. Müller, and W. E. Nagel: The Vampir Performance Analysis Tool-Set, Tools for High-Performance Computing, pp. 139-155, Springer 2008
[129] Open Trace Format, http://www.tu-dresden.de/die_tu_dresden/zentrale_einrichtungen/zih/forschung/software_werkzeuge_zur_unterstuetzung_von_programmierung_und_optimierung/otf, last accessed 21.03.2013
[130] IOFSL homesite: http://www.mcs.anl.gov/research/projects/iofsl/
[131] J. Domke, 'Trace Based Performance Analysis at Large Scale on Titan', pdf: http://www.olcf.ornl.gov/wp-content/uploads/2012/01/TitanWorkshop2012_Day3_Vampir.pdf
[132] More information can be found on the CRESTA website: http://cresta-project.eu/systemware/debugging-and-application-performance-tools.html
[133] TotalView homesite: http://www.roguewave.com/products/totalview.aspx
[134] More information can be found here: http://www.roguewave.com/company/news-events/press-releases/2012/scalability-milestone-for-totalview-debugger.aspx
[135] More information can be found here: http://www-hpc.cea.fr/en/Wotofe/docs/16-1210-Roguewave.pdf
[136] Allinea DDT homesite http://www.allinea.com/products/ddt/
[137] For more information see: http://content.allinea.com/news/allinea-software-sets-new-world-record-on-ornl-s
[138] For more information see: http://www.allinea.com/news/bid/88433/Allinea-Software-Helps-Launch-World-Class-Research
[139] For more information see: http://www.hpcwire.com/hpcwire/2012-05-14/blue_waters_supercomputer_debugs_with_allinea_ddt.html
[140] Intel Cluster Studio XE 2013, homesite: http://software.intel.com/en-us/intel-cluster-studio-xe
[141] Private Communication with DEEP team
[142] Intel Success Brief: "Creating a new standard in virtual crash testing" http://software.intel.com/sites/default/files/Altair_CS_052112.pdf
[143] Intel VTune Amplifier product site: http://software.intel.com/en-us/intel-vtune-amplifier-xe/
[144] Optimization and Performance Tuning for Intel Xeon Phi Coprocessors, Part 1 http://software.intel.com/en-us/articles/optimization-and-performance-tuning-for-intel-xeon-phi-coprocessors-part-1-optimization, http://software.intel.com/en-

us/articles/optimization-and-performance-tuning-for-intel-xeon-phi-coprocessors-part-2-understanding
[145] Intel Inspector Product site - http://software.intel.com/en-us/intel-inspector-xe/
[146] Intel Case Study (University of Kyoto) – "Delivering High-Speed Supercomputer Services":http://software.intel.com/sites/default/files/kyoto.pdf
[147] Private Communication with Intel Cluster Studio team
[148] 'Helping bring greater realism to manufacturing simulation': http://software.intel.com/sites/default/files/simulia_case_studyV2.pdf
[149] Nvidia Developer Zone site about Nsight http://www.nvidia.com/object/nsight.html
[150] PRACE deliverable D9.2.2, PRACE-1IP, 'Final Software Evaluation Report', pdf: (http://www.prace-ri.eu/IMG/pdf/D9-2-2_1ip.pdf)
[151] ScalaLife project deliverable D6.1, 'Report on the scalable techniques for life science software: Performance analysis and first parallel prototypes', pdf: http://www.scalalife.eu/system/files/ScalaLife-D6.1-final.pdf
[152] Using perfools for threaded and hybrid codes, Parallel Programming Workshops and Programming Language Courses, 2011, HLRS, pdf: https://fs.hlrs.de/projects/par/events/2011/parallel_prog_2011/2011XE6-1/04.2-Craypat.pdf
[153] IBM's Parallel Environment (PE) Developer Edition, homesite: https://www.ibm.com/developerworks/mydeveloperworks/wikis/home?lang=en#/wiki/Welcome%20to%20High%20Performance%20Computing%20%28HPC%29%20Central/page/Parallel%20Environment%20%28PE%29%20Developer%20Edit
[154] Paraver homesite: http://www.bsc.es/computer-sciences/performance-tools/paraver/general-overview
[155] Extrae homesite: http://www.bsc.es/computer-sciences/extrae
[156] Private Communication. For more information see: DEEP project deliverable, D5.1, 'Prototype programming environment in Booster Node', pdf: http://www.deep-project.eu/SharedDocs/Downloads/DEEP-PROJECT/EN/Deliverables/deliverable-D5.1.pdf?__blob=publicationFile
[157] IPM homesite: http://www.ipm2.org
[158] Effective Holistic PerformanceMeasurement at Petascale Using IPM", Karl Fürlinger, Nicholas J. Wright, and David Skinner. In Proceedings of The Sixteenth IEEE International Conference on Parallel and Distributed Systems (ICPADS 2010). Shanghai, China, December 2010.
[159] OpenSpeedShop homesite: http://www.openspeedshop.org/wp/
[160] Jim Galarowicz, "Look for Bottlenecks with OpenSpeedShop", article in the Internet magazine "Admin Magazine", http://www.admin-magazine.com/HPC/Articles/Look-for-Bottlenecks-with-Open-SpeedShop
[161] Jim Galarowicz, 'Understanding Performance of Parallel Codes Using OpenSpeedShop on BG/Q', ScicomP 2012, Toronto, May 17, 2012, http://spscicomp.org/wordpress/pages/understanding-performance-of-parallel-codes-using-openspeedshop/
[162] PAPI homesite: http://icl.cs.utk.edu/papi/
[163] Iain bethune, Adam Carter, Kevin Stratford, Paschalis Korosoglou, 'CP2K – Scalable Atomistic Simulation for the PRACE Community', PRACE-1IP report, 2012, pdf: http://www.era.lib.ed.ac.uk/bitstream/1842/6545/1/CP2K_-_Scalable_Atomistic_Simulation_for_the_PRACE_Community.pdf
[164] A short overview of Ayudame/Temanejo, with an explanation of its architecture.• "Ayudame/Temanejo Manual", Steffen Brinkmann (HLRS), pdf: http://www.project-text.eu/sites/default/files/AYUDAME_manual-1.pdf

[165] User guide for the debugger, including installation, and explanation of its functionalities. It also includes some examples, pdf: http://www.project-text.eu/sites/default/files/TEMANEJO_manual-1.pdf
[166] MAGMA Library homesite: http://icl.cs.utk.edu/magma/
[167] PETSc homesite: http://www.mcs.anl.gov/petsc/
[168] ELPA Library homesite: http://elpa.rzg.mpg.de/software
[169] ScaLAPACK homesite: http://www.netlib.org/scalapack/
[170] R. Johanni et al, 'Scaling of Eigenvalue Solver Dominated Simulations',Technical Report FZJ-JSC-IB-2011-02, p. 27-30, April 2011, pdf: http://www2.fz-juelich.de/jsc/docs/printable/ib/ib-11/ib-2011-02.pdf
[171] A Sunderland, 'Numerical Library Eigensolver Performance on PRACE Tier-0 Systems' PRACE whitepaper, PRACE-1IP, http://www.prace-ri.eu/IMG/pdf/numerical_library_eigensolver_performance_on_prace_tier-0_systems.pdf
[172] Dongarra, J., Dong, T., Gates, M., Haidar, A., Tomov, S., and Yamazaki, I. 'MAGMA: a New Generation of Linear Algebra Library for GPU and Multicore Architectures', SC12, Salt Lake City, Utah, November 14, 2012. pdf: http://icl.cs.utk.edu/projectsfiles/magma/pubs/25-MAGMA_1.3_SC12.pdf
[173] Song, F. and Dongarra, J. "A Scalable Framework for Heterogeneous GPU-Based Clusters," The 24th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2012), ACM, Pittsburgh, PA, USA, June 25, 2012, pdf: http://icl.cs.utk.edu/news_pub/submissions/spaa12_appendix.pdf
[174] Dongarra, J., Gates, M., Jia, Y., Kabir, K., Luszczek, P., Tomov, S. "MAGMA MIC: Linear Algebra Library for Intel Xeon Phi Coprocessors," SC12, Salt Lake City, Utah, November 12-15, 2012, pdf: http://icl.cs.utk.edu/projectsfiles/magma/pubs/24-MAGMA_MIC_03.pdf
[175] CULA homesite: http://www.culatools.com/
[176] ArrayFire homesite: http://www.accelereyes.com/products/arrayfire
[177] SuperLU homesite: http://crd-legacy.lbl.gov/~xiaoye/SuperLU/
[178] Xiaoye S. Li, 'An Overview of SuperLU: Algorithms, Implementation, and User Interface', Transactions on Mathematical Software, v31, p302 (2005)
[179] MUMPS homesite: http://graal.ens-lyon.fr/MUMPS/index.php?page=home
[180] (Par)METIS homesite: http://www.lrz.de/services/software/mathematik/metis/
[181] PRACE deliverable, D12.2, 'Exploration of Scalable Numerical Algorithms' PRACE-2IP, pdf: http://www.prace-ri.eu/IMG/pdf/D12-2_2ip.pdf
[182] A. Duran, M.S. Celebi, M. Tuncel and B. Akaydin, Design and implementation of new hybrid algorithm and solver on CPU for large sparse linear systems, PRACE-2IP white paper 2012, pdf: http://www.prace-ri.eu/IMG/pdf/wp43-newhybridalgorithmfo_lsls.pdf
[183] Xuefei Yuan et al 'Application of PDSLin to the magnetic reconnection problem', 2013 Comput. Sci. Disc. 6 014002, pdf: http://iopscience.iop.org/1749-4699/6/1/014002/pdf/1749-4699_6_1_014002.pdf
[184] Valeria Simoncini and Daniel B. Szyld, 'Recent computational developments in Krylov subspace methods for linear systems', Numer. Linear Algebra Appl.vol. 14, p1 (2007)
[185] Hypre homesite: http://acts.nersc.gov/hypre/
[186] Y Shapira, "Algebraic multigrid". Matrix-based multigrid: theory and applications. Springer. p. 66. (2003)
[187] CRESTA deliverable, D4.1.1, 'Overview of major limiting factors of existing algorithms and libraries', http://cresta-project.eu/images/docs/deliverables/D4.1.1_Overview_of_major_limiting_factors_of_existing_algorithms_and_libraries.pdf

[188] EESI Report D4.5 'Working Group report on numerical libraries, solvers and algorithms', pdf: http://www.google.ie/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CC8QFjAA&url=http%3A%2F%2Fwww.eesi-project.eu%2Fmodules%2Fdownload_pictures%2Fdlc.php%3Ffile=111%26id=1349445649%26sid=17&ei=rLFnUZHlOJK5hAeqrIHABA&usg=AFQjCNFXpdaTWCd4wchODFKIIfZR_h
[189] DUNE homesite: http://www.dune-project.org/dune.html
[190] FEAST homesite: http://www.feast.tu-dortmund.de/
[191] PSBLAS homesite:http://www.ce.uniroma2.it/psblas/
[192] PARPACK homesite: http://www.caam.rice.edu/~kristyn/parpack_home.html
[193] SLEPc website: http://www.grycap.upv.es/slepc/
[194] V Hernandez, J E Roman, A Tomas and V Vidal, SLEPc Technical Report STR-7 (2007), pdf: http://www.grycap.upv.es/slepc/documentation/reports/str7.pdf
[195] University of Florida Spares Matrix Collection homesite: http://www.cise.ufl.edu/research/sparse/matrices/
[196] K. Georgiev, N. Kosturski, I. Lirkov, S. Margenov, Y. Vutov, 'Parallel Solvers for Incompressible Navier-Stokes Equations and Scalable Tools for FEM Applications', PRACE Whitepaper, PRACE-1IP, pdf: http://www.prace-ri.eu/IMG/pdf/scalable-tools-for-fem-applications.pdf
[197] Pieter Ghysels, Tom Ashby, Karl Meerbergen, Wim Vanroose, 'Hiding global communication latency in the GMRES algorithm on massively parallel machines, To appear in SIAM Journal on Scientific Computing (SISC), pdf: http://www.exascience.com/wp-content/uploads/2012/02/Ghysels_latency_gmres.pdf
[198] Thomas J. Ashby, Pieter Ghysels, Wim Heirman, Wim Vanroose, 'Implementing Krylov Methods by Pipelining to Tolerate Global Communication Latency at Extreme Scales', 12th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP-12), pdf: http://www.exascience.com/wp-content/uploads/2012/12/Ashby_ICA3PP-12.pdf
[199] W Hackbusch, 'Hierarchische Matrizen', Springer (2009)
[200] Hlibpro homesite: http://www.hlibpro.com/
[201] M. Berzins, J. Luitjens, Q. Meng, T. Harman, C.A. Wight, and J.R. Peterson. Uintah - a scalable framework for hazard analysis. In TG '10: Proceedings of the 2010 TeraGrid Conference, New York, NY, USA, 2010. ACM.
[202] Schmidt, John, et al, 'Large scale parallel solution of incompressible flow problems using uintah and Hypre', Technical Report UUSCI-2012-002, Scientific Computing and Imaging Institute, 2012.
[203] I Bush, 'DaFT: A DaFT (Mixed Radix) FFT for DL_POLY_4', HECToR dCSE Technical Report, pdf: http://www.hector.ac.uk/cse/distributedcse/reports/DL_POLY03/DL_POLY03_domain/index.html
[204] FFTW homesite: http://www.fftw.org/
[205] A. Sunderland et al, "An Analysis of FFT Performance in PRACE Application Codes", PRACE White Paper, PRACE-1IP, pdf: http://www.prace-ri.eu/IMG/pdf/An_Analysis_of_FFT_Performance_in_PRACE_Application_Codes.pdf
[206] FFTE home site: http://www.ffte.jp/
[207] Futher information on cuFFT performance can be found here: https://developer.nvidia.com/cufft.
[208] Kenneth Czechowski et al. 'On the communication complexity of 3D FFTs and its implications for exascale'. In *Proceedings of the ACM International Conference on*

Supercomputing, San Servolo Island, Venice, Italy, June 2012. (*to appear*) pdf preprint: http://vuduc.org/pubs/czechowski2012-ics-xfft.pdf

[209] D. Stanković, A. Jović, P. Jovanović, D. Vudragović, V. Slavnić, "Enabling FFTE Library and FFTW3 Threading in Quantum ESPRESSO", PRACE White paper, PRACE-1IP, pdf: http://www.prace-ri.eu/IMG/pdf/enabling_ffte_library_and_fftw3_threading_in_quantum_espresso.pdf

[210] M. Guarrasi, G. Erbacci and A. Emerson, "Auto-tuning of the FFTW Library for Massively Parallel Supercomputers", PRACE whitepaper, PRACE-2IP, pdf: https://bscw.zam.kfa-juelich.de/bscw/bscw.cgi/d895012/Auto-tuning_of_FFTW_library_for_massively_Parallel_Supercomputers-CINECA-PRACE2IP-WP12.1.pdf

[211] T. Kozubek, M. Jarosov, M. Mensik and A. Markopoulos, 'Hybrid Total FETI Method', PRACE whitepaper, PRACE-1IP, pdf: http://www.prace-ri.eu/IMG/pdf/hybridtotalfetimethod.pdf

[212] T. Kozubek, D. Horak, V. Hapla, 'FETI Coarse Problem Parallelization Strategies and Their Comparison', PRACE whitepaper, PRACE-1IP, pdf: http://www.prace-ri.eu/IMG/pdf/feticoarseproblemparallelization.pdf

[213] M Hoemmen, 'A Communication-Avoiding, Hybrid-Parallel, Rank-Revealing Orthogonalization Method', Parallel & Distributed Processing Symposium (IPDPS), 2011 IEEE International

[214] E Turan and P Arbenz 'Preconditioning for large scale micro finite element analyses of 3D poroelasticity using Trilinos' European Trilinos User Group Meeting 2012 June 4th - June 6th, EPFL Lausanne, pdf: http://trilinos.sandia.gov/events/eurotug_2012/presentations/turan_eurotug.pdf

[215] Zoltan User Guide: http://www.cs.sandia.gov/Zoltan/

[216] Ata Turk, Cevdet Aykanat, G Vehbi Demirci, Sebastian von Alfthan and Ilja Honkonen, 'Investigation of load balancing scalability in space plasma, Simulations', PRACE whitepaper, PRACE-1IP, pdf: http://www.prace-ri.eu/IMG/pdf/Investigation_of_load_balancing_scalability_in_space_plasma_simulations.pdf

[217] Sebastian von Alfthan, Dusan Stankovic and Vladimir Slavnic, 'Scaling Vlasiator using Hybrid MPI and OpenMP parallelization', PRACE whitepaper, PRACE-1IP, pdf:, http://www.prace-ri.eu/IMG/pdf/Scaling_Vlasiator_using_Hybrid_MPI_and_OpenMP_parallelization.pdf

[218] SuperLU_Dist Version 3.2 documentation. http://crd-legacy.lbl.gov/~xiaoye/SuperLU/ - superlu_dist

[219] Ali Cevahir, Cevdet Aykanat, Ata Turk, Berkant Barla Cambazoglu: Site-Based Partitioning and Repartitioning Techniques for Parallel PageRank Computation. IEEE Trans. Parallel Distrib. Syst. 22(5): 786-802, 2011.

[220] Jeremy T. Bradley, Douglas V. de Jager, William J. Knottenbelt, Aleksandar Trifunović, Hypergraph Partitioning for Faster Parallel PageRank Computation, EPEW'05, Proceedings of the 2nd European Performance Evaluation Workshop, Volume 3670, pp.155–171, 2005.

[221] S.L. Yılmaz, M.B. Nik, M.R.H. Sheikhi, P.A. Strakey, P. Givi, An Irregularly Portioned Lagrangian Monte Carlo Method for Turbulent Flow Simulation, Journal of Scientific Computing, vol. 47(1), pp 109-125, 2011.

[222] ParMeTiS Manual, pdf: http://glaros.dtc.umn.edu/gkhome/fetch/sw/ParMeTiS/manual.pdf

[223] C. Moulinec, A.G. Sunderland, P. Kabelikova, A. Ronovsky, V. Vondrak, A. Turk, C. Aykanat and C. Theodosiou 'Optimisation of Code_Saturne for Petascale Simulations',

PRACE whitepaper, PRACE-1IP, pdf: http://www.prace-ri.eu/IMG/pdf/Optimisation_of_Code_Saturne_for_Petascale_Simulations.pdf
[224] Charles Moulinec, Yoann Audouin and Andrew Sunderland, 'Optimizing TELEMAC-2D for Large-scale Flood Simulations', PRACE whitepaper, PRACE-1IP, pdf: http://www.prace-ri.eu/IMG/pdf/Optimizing_TELEMAC-2D_for_Large-scale_Flood_Simulations.pdf
[225] Y Yilmaz, C Ozturan, O Tosun, A Haydar Ozer and S Soner, 'Parallel Mesh Generation, Migration and Partitioning for the Elmer Application' PRACE whitepaper, PRACE-1IP, pdf:, http://www.prace-ri.eu/IMG/pdf/Parallel_Mesh_Generation_Migration_and_Partitioning_for_the_Elmer_Application.pdf
[226] T. Coupez, H. Digonnet, R. Ducloux, Parallel meshing and remeshing, Applied Mathematical Modelling, Volume 25, Issue 2, pp 153-175, 2000.
[227] François Guibault, "Applying ParMeTiS to Structured Remeshing for Industrial CFD Applications", International Journal of High Performance Computing Applications vol. 17, no. 1, pp. 63-76, 2003.
[228] O.B. Fringer, M. Gerritsen, R.L. Street, "An unstructured-grid, finite-volume, nonhydrostatic, parallel coastal ocean simulator", Ocean Modelling, Volume 14, Issues 3–4, 2006, Pages 139–173.
[229] PETSc 3.3 Manual, http://www.mcs.anl.gov/petsc/petsc-3.3/docs/manual.pdf
[230] Cruz, F. A., Knepley, M. G. and Barba, L. A, PetFMM—A dynamically load-balancing parallel fast multipole library. Int. J. Numer. Meth. Engng. 85: 403–428, 2011.
[231] Scotch and PT-Scotch homepage http://www.labri.fr/perso/pelegrin/scotch
[232] PT-Scotch Manual, Version 6.0, (Accessed 23/02/2013), https://gforge.inria.fr/docman/view.php/248/8261/ptscotch_user6.0.pdf
[233] PRACE deliverable 7.6:, PRACE-1IP, 'Efficient handling of petascale data', pdf: http://www.prace-ri.eu/IMG/pdf/d7.6_1ip.pdf
[234] C. Chevalier and F. Pellegrini. 2008. PT-Scotch: A tool for efficient parallel graph ordering. Parallel Comput. 34, 6-8 (July 2008), 318-331. DOI=10.1016/j.parco.2007.12.001, http://dx.doi.org/10.1016/j.parco.2007.12.001
[235] Hom Nath Gharti et al, 'Application of an elastoplastic spectral-element method to 3D slope stability analysis', International Journal for Numerical Methods in Engineering Volume 91, Issue 1, pages 1–26, 2012
[236] Phil Ridley, Guide to Partitioning Unstructured Meshes for Parallel Computing, http://www.hector.ac.uk/cse/reports/unstructured_partitioning.pdf
[237] Daniel Peter et al, 'Forward and adjoint simulations of seismic wave propagation on fully unstructured hexahedral meshes', Geophysical Journal International, Volume 186, Issue 2, pages 721–739, 2011
[238] NetGen homesite: http://www.hpfem.jku.at/netgen/
[239] Parallel NetGen homesite: http://code.google.com/p/parallel-netgen/
[240] Y Yilmaz, Can Ozturan, Oguz Tosun, Ali Haydar Ozer, Seren Soner, 'Parallel Mesh Generation, Migration and Partitioning for the Elmer Application', PRACE whitepaper, PRACE-1IP, pdf: http://www.prace-ri.eu/IMG/pdf/Parallel_Mesh_Generation_Migration_and_Partitioning_for_the_Elmer_Application-2.pdf.
[241] Ioan Raicu et al, 'Towards Loosely Coupled Programming on Petascale Systems', pdf: http://arxiv.org/ftp/arxiv/papers/0808/0808.3540.pdf
[242] Saman Amarasinghe et al 'ExaScale Software Study: Software Challenges in Extreme Scale Systems', pdf: http://users.ece.gatech.edu/mrichard/ExascaleComputingStudyReports/ECSS%20report%20101909.pdf

[243] HDF5 homesite: http://www.hdfgroup.org/HDF5/
[244] PNetCDF homesite: http://www.mcs.anl.gov/parallel-netcdf
[245] NetCDF Operator (NCO) homepage: http://nco.sourceforge.net/
[246] Climate Data Operators (CDO) homepage: https://code.zmaw.de/projects/cdo
[247] FastBit homesite: https://sdm.lbl.gov/fastbit/
[248] Jack Dongarra et al, 'The International Exascale Software Project Roadmap', pdf: http://www.exascale.org/mediawiki/images/2/20/IESP-roadmap.pdf
[249] Prabhat et al, 'ExaHDF5: An I/O Platform for Exascale Data Models, Analysis and Performance', SciDac Conference 2011 pdf: http://www.mcs.anl.gov/uploads/cels/papers/scidac11/final/Prabhat.pdf
[250] A. Mignone, G. Muscianisi, M. Rivi and G. Bodo, 'I/O Optimization Strategies in the PLUTO Code', PRACE white paper, PRACE-1IP, pdf: http://www.prace-ri.eu/IMG/pdf/IO_Optimization_Strategies_in_the_PLUTO_Code.pdf
[251] Raul de la Cruz, Hadrien Calmet and Guillaume Houzeaux, 'Implementing a XDMF/HDF5 Parallel File System in Alya', PRACE white paper, PRACE-1IP, pdf: http://www.prace-ri.eu/IMG/pdf/Implementing_a_XDMF_HDF5_Parallel_File_System_in_Alya-2.pdf
[252] Philippe Wautelet and Pierre Kestener, 'Parallel IO performance and scalability study on the PRACE CURIE supercomputer', PRACE white paper, PRACE-1IP, pdf: http://www.prace-ri.eu/IMG/pdf/Parallel_IO_performance_and_scalability_study_on_the_PRACE_CURIE_supercomputer-2.pdf
[253] J. Ruokolainen, P. Raback, M Lyly, T. Kozubek, V. Vondrak, V. Karakasis and G. Goumas,'Improving the scalability of Elmer finite element software', PRACE white paper, PRACE-1IP, pdf: http://www.prace-ri.eu/IMG/pdf/Improving_the_scalability_of_Elmer_finite_element_software.pdf
[254] Claudio Gheller, Graziella Ferini, Maciej Cytowski and Franco Vazza ,'Large Scale Simulations of the Non-Thermal Universe', PRACE white paper, PRACE-1IP, pdf: http://www.prace-ri.eu/IMG/pdf/Large_Scale_Simulations_of_the_Non-Thermal_Universe.pdf
[255] J Shalf, Howison and Q Koziol, 'Tuning HDF5 for Lustre', SC'09, HDF5 BOF pdf: http://www.hdfgroup.org/pubs/presentations/LBNL_SC09_HDF5_BoF.pdf
[256] NetCDF homesite: http://www.unidata.ucar.edu/software/netcdf/
[257] R Latham et al 'A case study for scientific I/O: improving the FLASH astrophysics code', Comput. Sci. Disc. 5 015001 (2012), pdf: http://iopscience.iop.org/1749-4699/5/1/015001
[258] 'On Data Intensive Computing and Exascale', Presentation, IESP, http://www.exascale.org/mediawiki/images/6/64/Talk-12-Choudhary.pdf
[259] XIOS information can be found here: https://verc.enes.org/computing/hpc-collaborations/parallel-i-o/workshop-scalable-io-in-climate-models/presentations/XIOS_Yann_Meurdesoif.ppt
[260] More information can be found at: https://is.enes.org/the-project/presentation-on-is-enes-slides/is-enes-2nd-hpc-workshop-presentations-february-2013/session-4-report-from-large-numerical-climate-experiments-on-prace-platforms/Is-ENES_HPC_masson.pdf/at_download/file
[261] ADIOS homesite: http://www.olcf.ornl.gov/center-projects/adios/
[262] For more information see the HPCWire article: http://www.hpcwire.com/hpcwire/2009-10-29/adios_ignites_combustion_simulations.html
[263] For more information see the HPCWire article: http://www.hpcwire.com/hpcwire/2009-07-27/fusion_gets_faster.html
[264] J Tromp et al, 'Seismic Imagining: Modeling earthquakes and Earth's interior based on

Exascale simultations of seismic wave propagation', G8 Exascale Projects Workshop, Salt Lake City, USA 2012 pdf: http://www.agence-nationale-recherche.fr/Colloques/G8HORCs-workshop/presentations/05_Session3_G8ExascaleWorkshop_SEISMIC_IMAGING.pdf
[265] SIONLib homesite: http://www.fz-juelich.de/jsc/sionlib
[266] A. Schnurpfeil, A. Schiller, F. Janetzko, St. Meier and G. Sutmann ,'Semi-dilute polymer systems in shear flow – a particle based hydrodynamics approach', PRACE white paper, PRACE-1IP, pdf: http://www.prace-ri.eu/IMG/pdf/Semi-dilute_polymer_systems_in_shear.pdf
[267] "Parallel task-local I/O", 25. April 2012, JICS/GRS Workshop on Large-scale Computer Simulation, Frings and Wolf, http://computing.ornl.gov/workshops/JICSGRS2012/presentations/w_frings.pdfSIONLI
[268] High-Throughput Parallel-I/O using SIONlib for Mesoscopic Particle Dynamics Simulations on Massively Parallel Computers", Freche et al, in "Parallel computing: From multicores and GPU's to petascale", Chapman et al Eds.
[269] Darshan homesite: http://www.mcs.anl.gov/research/projects/darshan
[270] Mor info at: http://www.mcs.anl.gov/uploads/cels/papers/ANL:MCS-TM-331.pdf
[271] B Lindi, 'I/O Profiling with Darshan', PRACE whitpepaper, PRACE-1IP, pdf: http://www.prace-ri.eu/IMG/pdf/IO-profiling_with_Darshan-2.pdf
[272] J Donners et al, 'Performance Analysis of EC-EARTH', PRACE whitepaper, PRACE-1IP, pdf: http://www.prace-ri.eu/IMG/pdf/Performance_Analysis_of_EC-EARTH_3-1-2.pdf

List of Acronyms and Abbreviations

ABINIT	Density functional theory package
ACL	Adaptive OpenCL
ADIOS	Adaptable I/O System
AIX	Advanced Interactive eXective
ALYA	Computational Mechanics Code
AMBER	Molecular simulations programs package
AMG	Algebraic MultiGrid
AMR	Adaptive Mesh Refinement
ANL	Argonne National Laboratory, DOE, Illinois, USA
ANSI	American National Standards Institute
API	Application Programming Interface
ARM	Advanced RISC Machines
ARMCI	Aggregate Remote Memory Copy Interface
ARPACK	A collection of Fortran77 subroutines designed to solve large scale eigenvalue problems
AVX	Advanced Vector Instructions
BG/P	Blue Gene/P, Second generation of Blue Gene series of supercomputers
Bi-CG	Biconjugate gradient stabilized method
BLAS	Basic Linear Algebra Subprograms
BSC	Barcelona Supercomputing Center (Spain)
BSD(BSD-3)	Berkley Software Distribution (BSD- 3 clause License)
CABARET	Compact Accurately Boundary Adjusting High-Resolution Technique
CAF	Co-Array Fortran

CAPS	A Many-Core Programing Company
CASK	Cray Adaptive Sparse Kernels
CASTORC	Computation-based Science and Technology Research Center, The Cyprus Institute, Cyprus
CCSM	Community Climate System Model
CDO	Climate Data Operators
CEA	Commissariat à l'énergie atomique et aux énergies alternatives (France)
CESM	Community Earth Systems Model
CFD	Computational Fluid Dynamics
CG	Conjugate Gradient
CHAPEL	Cascade High-Productivity Language
CINECA	Consorzio Interuniversitario, the largest Italian computing centre (Italy)
CMCC	Centro Euro-Mediterraneo sui Cambiamenti Climatici (Italy)
CPU	Central Processing Unit
CRESTA	Collaborative Research into Exascel Systemware, Tools and Applications
CRS	Check Point and Restart Service
CSC	Finnish IT Centre for Science (Finland)
CSCS	The Swiss National Supercomputing Centre (represented in PRACE by ETHZ, Switzerland)
cuBLAS	CUDA Basic Linear Algebra Subroutines (NVIDIA)
CUDA	Compute Unified Device Architecture (NVIDIA)
cuFFT	CUDA Fast Fourier Transform Library (NVIDIA)
CUPTI	CUDA Profiling Tools Interface (NVIDIA)
cuSPARSE	CUDA Sparse Matrix library (NVIDIA)
DAFT	Daresbury Advanced Fourier Transform, DL_POLY library
DARPA	Defense Advanced Research Projects Agency
DBMS	DataBase Management System
DCMF	Deep Computing Messaging Framework
DEEP	EU Exascale-enabling supercomputing platform project
DFT	Discrete Fourier Transform
DGEMM	Double precision General Matrix Multiply
DiGPUFFT	Distributed GPU FFT
DL_POLY	Molecular simulation package
DOD	Department of Defence
DOE	Department of Energy
DoW	Description of Work
DP	Double Precision, usually 64-bit floating point numbers
DUNE	Distributed and Unified Numeric Environment
EC-EARTH	Consortium of national weather services and universities
ECMWF	European Centre for Medium-Range Weather Forecasts
EESI	European Exascale Software Initiative
ELPA	Distributed parallel direct eigenvalue solver for symmetric matrices
ENZO	Adaptive mesh refinement simulation code
EPCC	Edinburg Parallel Computing Centre (represented in PRACE by EPSRC, United Kingdom)

ERGO	Quantum chemistry programme for large-scale self-consistent field calculations
ESSL	European Severe Storms Laboratory
FD	Finite Differences
FE	Finite Elements
FEAST	Finite Element Analysis and Solution Tools
FEM	Finite Element Method
FFT	Fast Fourier Transform
FFTE	FORTRAN subroutine library for computing FFT
FFTW	Fastest Fourier Transform in the West
FLOP	FLoating-point Operations Per Second
Fluidity-ICOM	Finite Element ocean modelling software framework
FV	Finite Volumes
FZJ	Forschungszentrum Jülich (Germany)
GA	Global Arrays
GASNet	Global Address Space Network
GCC	GNU compiler
GCRM	Global cloud-resolving Model
GFLOPS/ Gflop/s	Giga (= 109) Floating point operations (usually in 64-bit, i.e. DP) per second, also GF/s
GMRES	Generalized Minimal Residual Method
GNU	GNU's not UNIX, Unix like operating system
GPFS	General Parallel File System
GPGPU	General Purpose GPU
GPU	Graphic Processing Unit
GRNET	Greek Research and Technology Network
GROMACS	Molecular dynamics package
GTC	GPU Technology Conference
GUI	Graphical User Interface
HDF5	Library used to read and write platform-independent files
HECToR	High End computing Terascale Resources (British Supercomputer)
HOPSA	Holistic Performance System Analysis Project
HPC	High Performance Computing
HPCS	High Productivity Computing Systems
HPCT	High Performance Computing Toolkit
I/O	Input/Output
IB	InfiniBand
IBM	Formerly known as International Business Machines
ICHEC	Irish Centre for High End Computing
IDE	Integrated Development Environment
IDIRIS	Institut du Développement et des Resources en Informatique Scientifique (represented in PRACE by GENCI, France)
IESP	International Exascale Software Project
IEST	Institute of Environmental Sciences and Technology
IFS	Integrated Forecast System

ILU	Incomplete LU-decomposition
INRIA	A public science and technology institution dedicated to computational sciences, France
IPB	Institute of Physics Belgrade, Serbia
IPC	Inter Process Communication
iPIC3D	Space weather application
IPM	Integrated Performance Monitoring
ISO	International Organization for Standardization
ITAC	Intel Trace Analyser and Collector
JKU	Johannes Kepler University, Linz, Austria
LaBRI	Laboratoire Bordelais de Recherche en Informatique, University of Bordeaux, France
LAMMPS	Molecular Dynamics Simulator
LAPACK	Linear Algebra PACKage
LB3D	Lattice-Boltzmann code
LGPL	Lesser General Public License
LLAPI	Low Level API
LLNL	Lawrence Livermore National Lab
LRZ	Leibniz Supercomputing Centre (Garching, Germany)
MAGMA	Matrix Algebra on GPU and Multicore Architectures
MIC	Many Integrated Core
MIMD	Multiple Instruction, Multiple Data
MIPS	Microprocessor without Interlocked Pipeline Stages
MIT	Massachusetts Institute of Technology
MKL	Math Kernel Library (Intel)
MPI	Message Passing Interface
MPICH	Portable implementation of MPI
MSP	Multi-Streaming Processors
MUMPS	Massachusetts General Hospital Utility Multi-Programming System
MUSCL	Finite Volume method for solving PDEs
MVAPICH	MPI software package
NAMD	A parallel molecular dynamics code
NCL	NCAR Command Language
NCO	NetCDF Operator
NCSA	National Center for Supercomputing Applications, Illinois, USA
NDA	Non-Disclosure Agreement. Typically signed between vendors and customers working together on products prior to their general availability or announcement.
NEC	Nippon Denki Kabushiki Gaisha
NEC SX	NEC vector supercomputer series
NEMO	UCLA cluster
NERSC	National Energy Research Scientific Computing Centre, Berkeley, CA, USA
NETGEN	Automatic 3D Tetrahedral mesh generator
NSF	National Science Foundation
NSMB	Navier-Stockes Multi-Block
NTNU	Norwegian University of Science and Technology

NUMA	Non-Uniform Memory Access or Architecture
ODE	Ordinary Differential Equations
OLEs	Offload Language Extensions, Intel
OOFEM	Free Object Oriented Finite Elemenet code
OOP	Object-Oriented Programming
OpenACC	Open Accelerate
OpenCL	Open Computing Language
OpenFOAM	Open source CFD software package
OpenFVM	CFD solver
OpenMP	Open Multi Processing
openSHMEM	Open Shared MEMory
ORNL	Oak Ridge National Laboratory
OSIRIS	PIC code
OTF	Open Trace Format
OTF-2	Open Trace Format 2
P3DFFT	Parallel Three-Dimensional Fast Fourier Transforms
PAPI	Performance Application Programming Interface
ParMeTis	MPI-based parallel library for partitioning and repartitioning unstructured graphs
ParNCL	Parallelized NCL tool
PARPACK	Parallel ARPACK
PBB	Parallel Building Blocks
PCI	Peripheral Component Interconnect
PCIe	Peripheral Component Interconnect Express
PDC	Centre for HPC, Sweden
PDE	Partial Differential Equations
PDT	Program Database Toolkit
PEPC	Pretty Efficient Parallel Coulomb Solver
PerfDMF	Performance Data Management Framework
PetFMM	Parallel fast multipole library
PETSc	Portable, Extensible Toolkit for Scientific computation
PGAS	Partitioned Global Address Space
PGI	Portland Group, Inc.
PHG	Zoltan hypergraph partitioning tool
PIC	Particle in Cell
PLASMA	Parallel Linear Software for Multicore Architectures
PLUTO	Astrophysical fluid dynamics code
PMPI	MPI standard profiling interface
PMU	Performance Monitoring Unit
PNNL	Pacific Northwest National Laboratory
POMP	OpenMP Performance Monitoring Interface
POSIX	Portable Operating System Interface
PPM	Piecewise-Parabolic Method
PRACE	Partnership for Advanced Computing in Europe; Project Acronym
PRACE-PP	PRACE Preparatory Phase

PSC	Pittsburgh Supercomputing Centre
PSHM	Process SHared Memory
PSNC	Poznan Supercomputing and Networking Center, Poland
PT-Scotch	Software package and libraries for parallel graph partitioning
PVODE	Parallel ODE solver
QE	Quantum ESPRESSO
RAMSES	Grid-based hydro solver with adaptive mesh refinement
RDMA	Remote Direct Memory Access
RISC	Reduced instruction set computing
RMA	Remote Memory Access
SARA	Amsterdam Foundation for Academic Computing, Netherlands
SDK	Software Development Kit
SHMEM	SHared MEMory
SIGMA-NTNU	Norwegian Metacenter for Compuational Science
SIMD	Single Instruction, Multiple Data
SLEPc	Scalable Library for Eigenvalue Problem Compuations
SMP	Symmetric MultiProcessing
SNIC	Swedish National Infrastructure for Computing
SoC	System on a chip
SOR	Symmetric Over Relaxation
SP	Single Precision, usually 32-bit floating point numbers
SPMD	Single Program Multiple Data
SpMV	Sparse Matrix-Vector Multiplication
SSD	Solid State Disk or Drive
SSE	Server Sent Events
SSOR	Symmetric Successive Overrelaxation Method
STFC	Science and Technology Facilities Council (represented in PRACE by EPSRC, United Kingdom)
TACC	Texas Advanced Computing Centre
TAU	Tuning and Analysis Utilities
TBB	Threading Building Blocks, C++ template library
TEXT	Towards Exaflop applications
Tier-0/ Tier-1	Denotes the apex of a conceptual pyramid of HPC systems. In this context the Supercomputing Research Infrastructure would host the Tier-0 systems; national or topical HPC centres would constitute Tier-1
TSQR	Tall and Skinny QR factorization
TUD	Technical Universität Dresden
TVD	Trusted Virtual Domains
UCLA	University of California, Los Angeles
UFL	Unified Form Language
UHeM	National Center for High Performance Computing, Istanbul Technical University (Turkey)
UNIX	Unplexed Information and Computing System
UPC	Unified Parallel C
UVAS	Unified Virtual Address Space

VexCL	Vector EXpression template library for openCL
VPIC	Vector Particle-In-Cell
VTF	Vampir Trace Format
WCSS	Wroclaw Centre for Networking and Supercomputing (Poland)
WP	Work Package
WRF	Weather Research and Forecasting Model
X10	Parallel asynchronous Partitioned Global Address Space language based on Java and developed by IBM
XIOS	Extended Input/output System
XML	Extensible Markup Language
XSEDE	Extreme Science and Engineering Discovery Environment
ZIH	Centre for Information Services and High Performance Computing, TU Dresden, Germany

Executive Summary

The objective of PRACE-3IP Work Package 7 (WP7) 'Application Enabling and Support' is to provide applications enabling support for HPC applications codes which are important for European researchers to ensure that these applications can effectively exploit multi-petaflop systems. This applications enabling activity will use the most promising tools, algorithms and standards for optimisation and parallel scaling that have recently been developed through research and experience in PRACE and other projects.

This deliverable contains a comprehensive survey of the research activity undertaken within PRACE to date so as to better understand what HPC tools and techniques have been developed that could be successfully applied to help other applications within WP7 effectively exploit multi-petaflop systems and so see how the various applications communities have progressed over the last few years within PRACE. As well as surveying the tools and techniques that have been employed in PRACE, this deliverable also reports on how tools and techniques are being used in various exascale projects and initiatives outside of PRACE. This perspective is presented so as to inspire new "forward looking" approaches to enable European applications on the road to exascale computing.

The survey covers four separate topics that we consider relevant to enable applications on current multi-petascale systems. We summarize our findings separately by topic: Programming Interfaces and Standards, Debuggers and Profilers, Scalable Libraries and Algorithms and I/O Management Techniques.

Programming Interfaces and Standards:

As part of this report we have surveyed thirteen individual programming languages and standards and report on how they have been used in PRACE to date. While we have found that, unsurprisingly, the MPI model still dominates within PRACE, evidence suggests that the most recent version of the standard provides features that are starting to confront the challenges of exascale computing and which have not yet been exploited within PRACE to any considerable extent. As a result, we recommend that the latest features of the standard be exploited during the enablement of applications on multi-petascale systems in WP7.

We have also assessed the wide range of programming models for exploiting heterogeneous architectures and conclude that the entry of new competitors to the many-core space has increased the relevance of open standards on the road to exascale (where many-core typically implies > 50 cores). Indeed, even for GPUs we have found considerable evidence that an open standards approach, of which OpenACC represents the strongest offering to date, is becoming more popular both within and outside PRACE and should be considered for enabling applications within WP7. In terms of more novel approaches to exploiting multi-petascale systems, we have drawn rich information from the European exascale projects as well as work being carried out in the US, which includes interesting findings on novel extensions to OpenMP and the use of Partitioned Global Address Space (PGAS) languages in real applications, which should inspire WP7.

Debuggers and Profilers

As part of this report, we have surveyed fourteen debugging and profiling tools. We have found that all of the European exascale projects are concentrating effort into tools for debugging and performance analyses. This is deemed a necessity for efficient use of multi-petascale and future exascale systems: If we are to enable applications on such systems, then we need to have as clear a view as possible of the barriers to achieving performance. In some

respect, we feel that the European exascale project, DEEP, provides a model for how the profiling tools should enable applications in WP7. It is worth noting that this type of rigorous assessment of debugging and profiling tools has rarely been seen in PRACE reports or whitepapers to date. A substantial effort of training on tools for debugging and performance analysis has been carried out within PRACE. However, very little is documented on how successfully these tools have been employed within enabling projects. One of our missions within WP7 is to fix this discrepancy and to work more closely with tool developers to understand the full benefits and limits of such tools in extreme cases.

Scalable Libraries and Algorithms

As part of this report we have surveyed a representative collection of libraries and techniques that currently garner much interest both within and outside PRACE. As a consequence of the move towards large multi-petascale heterogeneous systems, there is an increasing demand for new and improved scalable, efficient, and reliable numerical algorithms and libraries that confront existing and upcoming complexities associated with such systems, including complex memory hierarchies, the overhead of data movement and fault tolerance. In particular, we have surveyed very interesting exploratory work that has recently been carried out in WP12 PRACE-2IP on libraries and algorithms, which we feel should be exploited further on real applications within WP7. As well as surveying research within PRACE, we have also looked to the work being carried out in European exascale projects and further afield to find out more about how such projects are tackling the challenges confronting libraries and algorithms at extreme scales.

I/O Management Techniques

As part of this report we have surveyed five I/O management techniques. The increasing data needs of scientific and engineering applications mean that the problems associated with reading, writing, analysing, storing and sharing large amounts of data are becoming more relevant to a wider user community within PRACE. While the performance gap between file systems and compute systems is well known, during our surveying we have found that users within PRACE have in general not been able to squeeze as much performance from existing parallel file systems as they have from computational hardware, particularly for the case of high-level I/O libraries. Deeper investigations into extracting performance from parallel file systems (with WP7 applications) will be the main focus of our enablement work within WP7.

1 Introduction

1.1 Purpose of the document

The objective of PRACE-3IP Work Package 7 (WP7) 'Application Enabling and Support' is to provide applications enabling support for HPC applications codes which are important for European researchers to ensure that these applications can effectively exploit multi-petaflop systems. This applications enabling activity will use the most promising tools, algorithms and standards for optimisation and parallel scaling that have recently been developed through research and experience in PRACE and other projects.

There has been significant research activity undertaken both within PRACE and outside PRACE investigating novel techniques to enable applications on petascale and future exascale systems. Such activities include, for example, PRACE Work Packages [WP6 ('Software Enabling for Petaflop/s Systems') in PRACE-PP, WP7 ('Enabling Petascale Applications: Efficent Use of Tier-0 Systems') and WP9 ('Future Technologies') in PRACE-1IP, WP7 ('Scaling Applications for Tier-0 and Tier-1 Users'), WP8 ('Community Codes'), and WP12 ('Novel Programming Techniques') in PRACE-2IP], other EU-funded projects (European Exascale Software Initiative (EESI) [1], Towards Exaflop applications (TEXT) [2], Collaborative Research into Exascale Systemware, Tools and Applications (CRESTA) [3], Dynamical Exascale Entry Platform (DEEP) [4], Mont-Blanc [5]) and international collaborations such as the International Exascale Software Project (IESP) [6].

As stated in the Description of Work (DoW), the objective of this deliverable, D7.2.1, is to survey these activities so as to better understand what software tools, algorithms and standards have been developed within them that could be successfully applied to help other applications effectively exploit multi-petaflop systems and so see how the various applications communities have progressed over the last few years. These techniques will be used on some of the applications identified in Task 7.1. Through this task, WP7 will pursue effective engagement and dialogue with major exascale projects. The outputs from this task will be useful within PRACE, for European HPC users and also more generally.

The survey on HPC tools and techniques presented here reports on four separate topics that are important to enabling applications within WP7. These are: (1) Programming Interfaces and Standards, (2) Debuggers and Profilers, (3) Scalable Libraries and Algorithms and finally, (4) I/O Management Techniques. The report here represents the first phase of T7.2. It should be stressed that actual implementation (or exploitation) of the tools and techniques reported here will only occur during the next phase of the task. The purpose of this report is to give a global picture of how these tools and techniques fared during PRACE projects to date and to also bring the reader up to date on the latest state of each of the tools and techniques that we report on. In this way, we hope to provide, primarily WP7 partners, with information that should hopefully stimulate further interest when considering the tools and techniques for the exploitation phase of T7.2. We also hope that the report will be of interest to European HPC users and more generally.

1.2 On the road to exascale

All of the HPC tools and techniques that we survey here are considered to be the state-of-the-art for enabling applications on current multi-petascale systems. However, it is widely expected that the exascale systems of the future will be qualitatively different from current and past computer systems. They will be built using massive multi-core processors with hundreds of cores per chip, their performance will be driven by parallelism, constrained by energy, and with all of their parts, will be subject to frequent faults and failures [7]. While the

focus of WP7 is on enabling European applications for current multi-petascale systems, this is not to mean that WP7 should ignore the challenges that are expected to confront applications on the road to exascale. While there still may not be a general consensus on what an exascale machine in the future will look like, it is becoming increasingly likely that it will share some of the characteristics of the current No.1 systems in both the Top500 (Titan, Cray XK7) and Green500 lists (Beacon, Appro GreenBlade), indicating a possible convergence towards heterogeneous architectures as a means to reach exascale. With this view in mind, there are opportunities now for WP7 to anticipate and prepare for the challenges that will be faced as we advance from the petascale era to the exascale frontier.

For example, the main programming environment challenges on the road to exascale are expected to be within nodes rather than across nodes. The total number of nodes for any given machine is not changing dramatically, so current practices of using MPI between nodes, certainly up to the deep petascale, provides one option of utilizing future exascale systems. If MPI is to be used in the future however, it certainly cannot be used in its current form and, as is increasingly evidenced by practice on petascale systems both within and outside PRACE, not on its own. One big challenge for MPI on its own is fault tolerance or resilience. Currently, a standard MPI implementation will abort the entire computation if any of its ranks encounters a failure. The traditional handling of these failures is using checkpoint/restart techniques. However, as the overhead of these implementations grow with core count, such methods will become highly inefficient. A fault tolerant MPI would enable an application to recover from failures and continue execution although some parts of the system have been lost indefinitely. Fortunately, there is much active work in this area within the MPI research community, some of which has already been accepted as part of the most recent version of the MPI 3 standard and, as has been found during our surveying, is already being investigated by on-going projects within PRACE.

While there are a myriad of models to choose from when augmenting the MPI model for exploiting many-core heterogeneous petascale machines, many of which we characterise in this report, in terms of key metrics such as performance and productivity, there is as of yet, no single model that sticks out from above the rest. The fact that there are so many models to choose from, also begs the question of how best to maintain application robustness on the road to exascale within WP7. If there is a ‘one-size fits all’ programming model for the future then maybe it is best represented by a Partitioned Global Address Space (PGAS) language such as Chapel [8]. Now starting to confront the challenges of exploiting architectures with deeper memory hierarchies and hybrid CPU/accelerator/coprocessor architectures [9], the bigger challenge that face such models are in terms of fault tolerance of which we have found little evidence of initiatives in this area to date. While such problems are being posed to PGAS development teams, an on-going challenge for WP7 is how best to exploit these forward looking models within real community applications on the road to exascale.

In considering the best programming models to choose from when enabling applications, a key and new aspect for WP7 to consider on the road to exascale is the scope of what enablement experts will require from debugging and profiling tools. Instead of one or two simple measures of performance, exascale computing will see a broadening of scope as to what tools report on and what they address. Examples of interest include memory utilization, temperature, reliability, and power consumption. European HPC applications are large, often complex and push the limits of language features. Tools deployed must also be robust enough to handle these codes on heterogeneous systems, but at the same time must be highly usable if they are to deliver impact in increasing the performance of applications. Fortunately, there are several debugging and profiling tools that we report on here that are starting to address these issues. While some of the features might not be highly relevant to applications within WP7 today, they will become increasingly so as we advance to exascale. We feel that T7.2 offers

an opportunity to work more closely with debugging and profiling tool developers and the user community to find out how advanced features can properly be exploited to prepare European applications for multi-petascale and future exascale machines.

As well as programming languages and performance tools, algorithms and libraries must also be designed to match the complexity of future exascale systems. While there are major challenges such as energy consumption reduction that algorithms and libraries can significantly help with on the road to exascale, priorities for applications within WP7 still lie within the areas of performance and scalability. That is not to say that these problems are independent of each other. In fact, quite the opposite, in that some of the most interesting research that we report on here on so-called 'communication avoiding/hiding algorithms' will most likely be of benefit to both the scalability of an application and reduce energy consumption needs on the system as well. Within PRACE there have been several interesting "forward looking" initiatives in the area of libraries and algorithms, most recently within WP12 PRACE-2IP that have looked closely at such issues, as well as others, with exascale in mind. Beyond scalability and performance, there is the important issue of fault tolerance of libraries and algorithms that will also need to be addressed as a high priority. While we have not found much activity within this area in PRACE to date, we suspect that this will become a more pressing issue as PRACE moves into the deep petascale and future exascale eras.

The increasing data needs of scientific and engineering applications mean that the problems associated with reading, writing, analysing, storing and sharing large amounts of data are becoming more relevant to a wider user community within PRACE. This will become increasingly so as we advance to the exascale era. There is a general consensus that the current development model of the storage I/O stack of current petascale machines will not scale to the new levels of concurrency, storage hierarchy, and capacity that will be found on exascale machines. As the depth of the storage hierarchy increases one of the biggest concerns is the programmability and performance of the I/O software stack. I/O system optimizations are often applied independently at each system layer. However, this approach can cause mismatches between the requirements of different layers (for instance in terms of load, locality, consistency). Due to this uncoordinated development, understanding the relationships between optimizations at different layers has become challenging. In this report we characterise several I/O tools that may help to enable WP7 applications for multi-petascale sytems, but that will in general need significant improvements in terms of both performance and resilience in order to prepare European applications for the exascale era. As well as improving I/O performance for WP7 applications, we hope to provide insight into where improvements can be made.

In all of the areas we mention above, we intend to liaise closely with the European exascale projects (CRESTA, DEEP, Mont-Blanc and TEXT), where possible, during the exploitation phase of T7.2, so as to expose WP7 to some of the advanced tools and techniques that are being researched and implemented within these projects and in doing so, prepare WP7 applications for the challenges that will be confronted on the road to exascale.

1.3 Organization of work

In order to structure work in WP7, the first goal for all task leaders was to define work plans, including a proposal for several subtasks. Figure 1 gives a schematic overview of the subtasks in T7.2

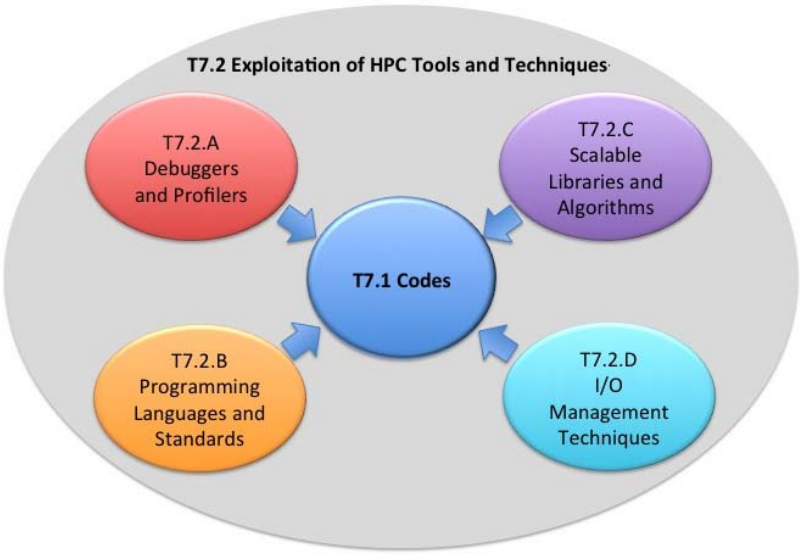

Figure 1: Schematic overview of subtasks in Task 7.2

During the PRACE-3IP kick-off meeting in Paris in September 2012, subtask leaders for T7.2 were identified. The subtask leaders are:

- Subtask 7.2.A 'Debuggers and Profilers': Bjorn Lindi (SIGMA-NTNU)
- Subtask 7.2.B: 'Programming Languages and Standards': Marc Tajchman (GENCI)
- Subtask 7.2.C: 'Scalable Libraries and Algorithms': Vit Vondrak (VSB)
- Subtask 7.2.D: 'I/O Management Techniques': John Donners (SURFSARA)

For deliverable D7.2.1, sources of input for the survey on HPC tools and techniques were identified clearly during and following the PRACE-3IP kick-off meeting. Activities within PRACE that have employed HPC tools and techniques have generally been well documented in PRACE deliverables and whitepapers which are available via the PRACE website and BSCW (The only exception to this rule is detailed reporting on debuggers and profilers, which has generally been more difficult to come by). Since a considerable amount of enabling work is still in progress within PRACE-2IP (mainly within WP8 and WP12), efforts were made to collaborate directly with relevant activities within PRACE-2IP via Face-to-Face (F2F) meetings and private communications.

While there has been a substantial amount of "exploratory" work on advanced disruptive technologies already carried out within PRACE (e.g., T7.5-1IP, WP12-2IP WP9-2IP), one of the objectives of T7.2 is to also survey how "future-looking" HPC tools and techniques have also been used outside PRACE, with particular emphasis on how such tools and techniques have been employed to enable "real" applications on multi-petascale systems and to also survey how European (and international) exascale projects are investigating tools and techniques for enabling applications on future exascale systems.

In order to identify and obtain relevant sources of input outside PRACE, direct contact was established with several European exascale projects including the TEXT, CRESTA, DEEP and Mont-Blanc projects to find out how HPC tools and techniques are currently being employed within these projects. (On the European level, we have also drawn from the European Exascale Software Initiative (EESI)). Activities within exascale initiatives outside Europe have also been surveyed here. Sources of input have included documentation from the IESP, and the US Department Of Energy (DOE)'s exascale workshops (e.g., the DOE sponsored Exascale Research Conference 2012 [10])

1.4 Structure of the document

The document presents four subsections which are aligned with the four subtasks within T7.2, and which will remain in place during the exploitation phase of the task. These are: Programming Interfaces and Standards, Debuggers and Profilers, Scalable Libraries and Algorithms and I/O Management Techniques. Within each section, a short introduction is provided which further details the structure of the individual section. This is followed by a survey of tools and techniques that are relevant to that particular section. Within each individual tool/technique assessment, we provide a brief overview of the current state of the tool, how the tool has been employed in PRACE to date, how it has been employed more widely (particularly on large-scale systems), a table listing the pros and cons of the tool/technique, which system/architecture the tool/technique is targeted at and finally a conclusion section which provides our own indication as to the applicability of the tool/technique during the T7.2 exploitation phase and beyond.

1.5 Intended Audience

Our objective in preparing this survey is to determine and document the characteristics of the most promising HPC tools and techniques that may have applicability for petascaling applications within WP7 PRACE-3IP. Targeted primarily at PRACE partners who will be involved in the exploitation phase of T7.2, it provides an up to date overview of a selection of HPC tools and techniques in order to allow for PRACE partners to determine if further investigation is warranted for the exploitation effort during the second phase of T7.2 and beyond. We also hope that the report here will be of interest to European HPC users and more generally.

2 Programming Interfaces and Standards

In this section, we characterize the following novel programming interfaces, languages, frameworks and standards (hereafter referred to as models) that are of interest to T7.2 in particular, and the European HPC community more widely, as we move towards the deep petascale and exascale eras:

- MPI (Message Passing Interface)
- OpenMP (Open Multi Processing)
- OpenCL (Open Computer Language)
- CUDA (Compute Unified Device architecture)
- OpenACC
- Threading Building Blocks (TBB)
- Cilk plus
- OmpSs
- Co-Array FORTRAN (CAF)
- Unified Parallel C (UPC)
- X10
- Chapel
- Global Arrays Toolkit (GA)

For each model, we provide an overview and discuss the model's present state, how it has been employed in PRACE to date, how it has been employed more widely, and our views on the suitability of the model for enabling PRACE application codes during the exploitation phase of T7.2.

At present, one model still dominates PRACE application codes more than any other, namely, Single Program Multiple Data (SPMD) message passing using MPI for internode communication, and increasingly, OpenMP for intra-node parallelism. MPI and OpenMP are mature standards and widespread expertise on their use can be found within PRACE. However, both standards are evolving and version 3.0 of MPI [11] and version 3.1 of OpenMP [12] have recently been ratified, with full implementations now being offered by several different initiatives. We report on several of the new features offered by both standards and point to the need to investigate these further as part of the exploitation phase of T7.2

While combining MPI and OpenMP is still considered to be the hybrid programming method of choice, the recent advent and rapid adoption of many-core coprocessors/accelerators in the design of Top500 supercomputers, including PRACE Tier-0 machines (as well as PRACE prototype architectures) has meant that additional models (albeit still hybrid ones) must increasingly be considered in order to exploit the full potential of the compute hardware space on upcoming European multi-petascale systems. To date, the challenge of exploiting such heterogeneous systems has typically been met within PRACE by augmenting the MPI/OpenMP hybrid model with an additional third model that targets the Single Instruction Multiple Data (SIMD) like architecture of GPUs (more accurately defined as Single Instruction Multiple Thread (SIMT)), thereby forcing the further extraction of hierarchical levels of parallelism in current PRACE applications. (It should also be mentioned that, as is the case for all accelerator/coprocessor-based systems, performance is limited by the high latency penalty of data transfers to and from the attached device, which is connected to the CPU host through the PCI bus. However, as the technology matures, accelerators/coprocessors are expected to integrate directly into the motherboard and this latency penalty will decrease.)

We have found that by far the most popular programming model for programming GPU architectures, both within and outside PRACE, is still CUDA [13]. While concerns are often voiced around the low-level, non-x86-based nature of CUDA, its ease of use and surrounding ecosystem is continually being improved upon by NVIDIA. While OpenCL [14] does offer an alternative open standard framework, adoption of the standard is slow, due possibly to its relatively poor ecosystem in comparison to CUDA, as well as lack of clarity on the level of long-term support for the standard. One possible solution to these issues is a directive-based open standards approach of which OpenACC [15] represents the strongest offering to date. Experience with the transition from vector codes to message passing codes twenty years ago proves the benefits of common, open standards for programming models. Much greater productivity was achieved following the widespread adoption of the MPI standard. An open standard for the utilization of GPUs holds a similar promise. It must be emphasized that OpenACC is a young standard and as a consequence has not featured much in PRACE activities to date. However, interest in it is growing and we are happy that we can report on several interesting cases where it has been employed most recently as part of on-going work within PRACE 2IP as well as within exascale projects outside PRACE.

With the arrival of Intel's Many Integrated Core (MIC) coprocessor to the market this year (2013), this is a particularly interesting time for heterogeneous systems. The Intel MIC is a component architecture in two of the PRACE prototypes and shares several similarities with GPUs. Some of the main differences compared to GPUs stem from the fact that the Intel MIC is an x86-based architecture and so familiar open standards such as OpenMP and MPI can be used to program the device, which is not the case for GPUs in general. While both of the aforementioned standards can be used to program the Intel Xeon Phi, it is still unclear which is the best model for extracting performance and whether different models are better for different problem cases. In fact, Intel supports several other open standards and libraries for programming the coprocessor including, TBB [16], Cilk Plus [17] and OpenCL [18], which are all reported on here.

With regards to the MPI plus X paradigm for programming heterogeneous systems, we also assess OmpSs [19], which is a programming model being developed at Barcelona Supercomputing Center (BSC) and is used in both the Mont-Blanc and DEEP exascale projects and in essence represents an effort to extend the OpenMP model with new directives to support asynchronous parallelism and heterogeneity (devices like GPUs). However, it can also be understood as new directives extending other accelerator based APIs, like CUDA or OpenCL.

Finally, we consider the Partitioned Global Address Space (PGAS) family of languages, (CAF [20], UPC [21], X10 [22], Chapel [8] and Global Arrays [23]). These models present an entirely different way of developing large-scale applications, which at the very least promises more concise and comprehensible code. Advantages of PGAS models pertaining to performance revolve around single-sided communications as opposed to two-sided MPI communications, which are prevalent in most applications within PRACE. While having featured within several PRACE activities, we have found that investigations into PGAS models have typically been exploratory in nature with no evidence of real applications being enabled with such models to date. (We have, however, found some exceptions to this within several of the exascale projects). Just as MPI plus X offers a solution to programming heterogeneous systems, several investigations are in progress to allow for PGAS languages to target GPU-based systems as well [9]. While still very much at an early stage of research, the benefit of a single language that can be used to efficiently target multi-petascale and future exascale heterogeneous systems in the entirety of their compute hardware space is a welcome prospect.

2.1 MPI

2.1.1 *Brief overview*

MPI [24] is still the most widely employed parallel programming model within PRACE and is generally well grasped within the PRACE community. For this reason we only report on some of the more novel features offered by the latest MPI 3.0 standard, which was published in September 2012 [11] and is widely viewed as a major update, positioning MPI for the deep petascale and future exascale era. We also briefly touch on some novel features not included in the standard, but are instead offered through various MPI implementations and have been shown to be particularly beneficial on heterogeneous systems. It should be born in mind that MPI attracts a huge amount of research interest, and so it would be impossible for us to cover all of the interesting work being carried out in its various implementations.

In brief, MPI 3.0 offers the following new features to MPI

- Non-blocking and neighbourhood collective operations
- Revamped remote memory access (RMA, a.k.a. “one-sided” operations)
- New Fortran 2008 bindings
- Richer external tool support
- Better support for large counts
- “Matched” probe support
- C const correctness
- Shared memory windows
- Non-blocking communicator creation / duplication
- Countless small grammar fixes, textual cleanups, and clarifications

To our knowledge, both the MPICH [25] MVAPICH [26] implementations of MPI offer the full set of MPI 3.0 features and OpenMPI [27] offers a subset of the features.

The MPI Forum [24] added support for one-sided communication (also known as remote memory access, or RMA) in version 2.0 of the MPI standard, to function alongside MPI’s traditional two-sided and collective communication models. While MPI 2 was effective for a variety of applications and systems, it has lacked various communication and synchronization features, and its conservative memory model has limited its ability to efficiently utilize hardware capabilities, such as cache coherence.

The MPI 3 standard now adds a variety of new atomic operations, synchronization primitives, window types, and a new memory model that better exposes the capabilities of architectures with coherent memory subsystems. It is believed that these features will address issues in the MPI 2 model and greatly improve the performance potential of MPI RMA.

Other important new functionality includes non-blocking collective communication and better handling of situations arising out of process failures, e.g., an application may choose to be notified when an error occurs anywhere in the system and an application may ignore failures that do not impact its MPI requests. There is also the capability to rebuild a communicator when a process fails or allowing it to continue in a degraded state. MPI processes may also ignore failures that do not impact its MPI requests. An important new feature is that an application that does not use collective operations will not require collective recovery.

One of the other interesting initiatives which is being investigated at the moment and which is not part of the standard, but rather features as part of the MVAPICH implementation, is MVAPICH2-X [28] which provides a unified high-performance runtime that supports both MPI and PGAS programming models on InfiniBand clusters. It enables developers to port parts of large MPI applications that are suited for PGAS programming model. This minimizes the development overheads that have been a huge deterrent in porting MPI applications to use PGAS models. The unified runtime also delivers superior performance compared to using separate MPI and PGAS libraries by optimizing use of network and memory resources.

MVAPICH2-X supports UPC and OpenSHMEM [29] as PGAS models. It can be used to run pure MPI, pure UPC, pure OpenSHMEM as well as hybrid MPI/OpenMP/PGAS applications. It takes advantage of the RDMA features offered by the InfiniBand interconnect to support UPC/OpenSHMEM data transfer and atomic operations. It also provides a high-performance shared memory channel for multi-core InfiniBand clusters.

Also not part of MPI standard, but which is proving very beneficial when enabling on heterogeneous systems, is the more optimized support being offered for coprocessors and accelerators including NVIDIA's Fermi and Kepler architectures [30] [31] as well as Intel's new MIC architecture [32].

Latest release/version: MPI 3.0 (standard now implemented)

2.1.2 *Evidence of use within PRACE*

MPI is the most widely used parallel programming method within PRACE to date and for this reason we have only selected a few examples of how it has been employed.

For an interesting in-depth analysis of MPI/OpenMP hybrid parallelisation, the reader is referred to section 2.4 within deliverable D6.4 in PRACE-PP [33], which focuses on the hybrid parallelisation of the materials science application, Quantum Espresso (QE). In this particular study, Fast Fourier Transform (FFT) and Linear Algebra subtasks within Quantum Espresso were hybridised using OpenMP in combination with MPI. The authors demonstrate that for large core counts and specific data sets,the hybridised version of QE allowed for scaling on up to 65,000 cores on a IBM BlueGene/P.

Another interesting investigation was also recently reported in the PRACE-1IP Preparatory Access whitepaper, **'**High resolution ocean simulations with NEMO modeling system' [34] where runs were carried out on the PRACE Tier-0 Bull supercomputer, CURIE, and where within the NEMO code all `MPI_Scatter` and `MPI_Gather` operations were replaced by `MPI_Send` and `MPI_Receive` calls which improved scalability on all platforms. In this study MPI persistent communication channels were also investigated but without much performance gain achieved.

More recently, within WP8 PRACE-2IP, PRACE partners have been working on introducing fault tolerance to the ocean modelling code, NEMO, where plans are afoot to interface with the new fault tolerant features of MPI 3.0 [35].

2.1.3 *Evidence of use outside PRACE*

MPI is used extensively on most petascale systems worldwide, so we have selected only a few examples of where it has demonstrated scalability on a large number of processes.

One of the codes using MPI that is already petascaling is the vector particle-in-cell, VPIC, code on Blue Waters, the Cray supercomputer at NCSA, Illinois [36], which integrates the relativistic Maxwell-Boltzmann system in a linear background medium for multiple particle

species. The science problem in question had a 3,072x3,072x2,464 cell domain with $7.44103x10^{12}$ particles and the code was run on 22,528 nodes of the Blue Waters system with 180,224 MPI processes (with 4 OMP threads/rank), and achieved 1.25 PFLOPS sustained.

Another example of an MPI-based code that has demonstrated excellent scalability on petascale systems is the PPM code [36], (the example of which we give here is also from runs on Blue Waters). PPM is a hydrodynamics code based on the Piecewise-Parabolic Method (PPM) Communication primarily involves halo exchanges on a 3D Cartesian mesh which are overlapped by computations, as is the I/O. In this work a rank re-ordering scheme was implemented to interleave the I/O server tasks in the MPI rank list. The test case uses a 10,5603 zone mesh and was run across 702,784 cores of Blue Waters system, with 681,472 worker threads organized into eight threads per MPI task. In total, 87,846 MPI ranks were running on 21,962 nodes with sustained 1.5 PF/s. More than 587 TB of data was saved with an aggregate of over 17 GB/sec I/O rate. In this example, communication and I/O were essentially 100% overlapped with computation.

With regards to MPI scalability, at the time of writing it was reported that researchers at Lawrence Livermore National Laboratory have performed record simulations using all 1,572,864 cores of Sequoia [37] based on IBM BlueGene/Q architecture, and is the first machine to exceed one million computational cores. The simulations are the largest particle-in-cell (PIC) code simulations by number of cores ever performed. The code used in these simulations was OSIRIS [38], an MPI-based PIC code that has also been run on PRACE systems. OSIRIS obtained 75% efficiency on the full machine.

All the six scientific applications in European exascale project, DEEP are parallelized using MPI. One of the requirements with respect to the software stack within DEEP is the need for an MPI layer running through the whole machine, from Cluster to Booster. (The general purpose Cluster consists of nodes with Intel Xeon processors and InfiniBand network whenas the Booster nodes contains Intel MIC processors and are connected with special 3D torus network are suited for highly scalable computation kernel.) This so-called ‘Global MPI’ implementation within DEEP is intended to realise the offload functionality that sends the highly scalable code parts of the applications and associated data from the Cluster to the Booster and receives results of these code parts the other way round. For this purpose `MPI_Comm_spawn()` will be implemented in ‘ParaStationMPI’ [39] (ParaStation MPI has been designed to select the most appropriate of all available interconnects at runtime), making it aware of the underlying resource management and allowing to start offloading-binaries on the Booster. The offload functionality is a collective operation of all MPI processes running on the Cluster side and initiating the offload. The MPI process space created on the Booster side is aware of and connected to the parent MPI processes running on the Cluster, and can communicate with them.

2.1.4 *Pros and Cons*

Metric	Pros	Cons
Scalability	MPI can scale up-to several thousand cores	Memory consumption increases with cores. At large scale communication time becomes dominant
Performance	MPI libraries are tuned for host system and interconnect. Both inter-node and intra-node bandwidth/latency are close to hardware specifications.	At large scale MPI-only codes can suffer from network congestion
Productivity	MPI is still the favoured model for developing applications for distributed memory systems. Can be used with Fortran, C and C++.	Powerful one-sided communication features have been added to MPI 2, but these are typically underused, which begs the question as to their ease-of-use. Such issues may be rectified by the MPI 3 standard, but at the time of writing it is too difficult to say.
Sustainability	MPI standard has been available for last 20 years. It is supported by a large number of organizations.	-
Correctness	Debugging tools are widely available for MPI-based applications, but are mainly commercial (DDT and TotalView)	-
Portability	MPI codes are portable. In most cases one does not require any change in the source code when moving from one system to another.	
Availability	MPI is an open standard and is available on most modern supercomputers.	-
Resilience	MPI 3.0 standard provides better fault tolerance features	Application codes need to be adapted to fault features of MPI, e.g., if a process dies due to hardware failure an MPI program can still run provided the application program adjusts itself to the new situation.

Table 1 MPI - Pros and Cons

2.1.5 *Target systems/architectures*

X86	IBM	GPU	MIC	ARM	Heterogeneous
Yes	Yes	No	Yes	Yes	Yes (MPI+X)

Table 2 MPI - Target systems/architectures

2.1.6 *Conclusion*

Although all the projections of what an exascale machine will look like in the future show that the MPI-only programming model won't be sufficient, MPI+X will more than likely remain the dominant programming model for another few years. Certainly for the moment, there are few signs of it being usurped from this position on the road to exascale. It can be seen from our brief report that the MPI forum is starting to address the challenges that MPI will face on this road, and we believe that the exploitation phase of T7.2 provides PRACE partners with a very valuable opportunity to investigate new MPI 3.0 features. It is well appreciated that MPI has many shortcomings when considering new hybrid many-core systems, but it is our opinion that these shortcomings can only be addressed after properly considering the full scope of what the new standard has to offer.

2.2 OpenMP

2.2.1 *Brief overview*

OpenMP [40] is still the model of choice for 'X' in the MPI plus X paradigm and, like MPI, has been widely exploited within PRACE to date. Adding OpenMP threading to an MPI code is an efficient way to run on multicore processors and nodes like those currently found on all PRACE systems. Since OpenMP operates in a shared memory space, it is possible to reduce the memory overhead associated with MPI tasks and reduce the need for replicated data across tasks. While OpenMP is promoted as being easy to use and allows incremental parallelisation of codes, naive implementations frequently yield poor performance. In practice, as with MPI, the same care and attention should be exercised over algorithm and hardware details when programming with OpenMP.

The last major update to the OpenMP standard resulted in OpenMP 3.0 [12] where the major new feature was a model for task-based parallel programming. Evidence suggests that this is still a relatively underused method within PRACE to date. More recently (2011), the OpenMP ARB released version 3.1. Version 3.1 is a minor release that offers corrections of the 3.0 specification. Here, we mention some of the new features included in OpenMP 3.1 that we believe should be of interest during the exploitation phase of T7.2.

The OpenMP 3.1 Specification introduces predefined min and max operators for the C and C++ base languages. With these operators, programmers no longer have to work-around the lack of these operators and implement reductions themselves. Without lengthy coding patterns, it now becomes possible to directly use min and max in the reduction clause of a parallel region or work-sharing construct in C/C++.

The `atomic` construct has been extended. It now provides support to atomically capture the value of an updated variable for later reuse. This, for instance, enables programmers to efficiently implement dynamic scheduling approaches without expensive locking or critical regions. It is also now possible to read a memory location and to write to a memory location. The old behaviour of `atomic` is now made explicit through the introduction of the `update` clause for atomic updates of memory locations.

The tasking model has received an update to allow for more efficient coding of task-parallel applications. With the `taskyield` construct, OpenMP now supports user-defined scheduling points to indicate to the OpenMP runtime to suspend one task in favour of other tasks. As a means for performance optimization, the conditional `final` clause, the `mergeable` clause, and the corresponding calls in the runtime support routines, reduce overheads that are a result of task creation. When the `final` clause evaluates to true, a task may not be scheduled for deferred execution, but instead is immediately executed. This also applies to all child tasks that might be generated by this task. With the `mergeable` clause, programmers can avoid potentially expensive initialization of the task environment. With these additions, fine-grained task-based programming may benefit from less tasking overhead.

Version 3.1 also includes a variety of minor corrections and additions. The behaviour of `firstprivate` was corrected for types with constant qualification and the data environment of parallel regions in Fortran now support `intent(in)` for variables. The descriptions of the examples have been expanded to a large extent and clarifications have been added.

In the lead up to version 4.0 of OpenMP [41] the OpenMP consortium are working on added support for SIMD directives, significantly extended support for thread affinity, added UDRs, sequentially consistent atomics, atomic swap, and added initial support for Fortran 2003. Other key features that the OpenMP consortium is working on are support for accelerators (with a possible merge with the OpenACC standard) and improvements in error handling.

It is worth also mentioning that one of the more interesting target architectures for OpenMP is the new Intel MIC (Xeon Phi) coprocessor, where for the moment Intel recommends OpenMP (in combination with *Intel's Language Extensions for Offload (LEO)*) as a model for exploiting the many-core device. OpenMP parallelization on an Intel Xeon/MIC coprocessor platform can be applied in four different programming models that can be realized with different compiler options: native OpenMP on the Xeon host; serial Xeon host with OpenMP offload; OpenMP on the Xeon host with OpenMP offload and native OpenMP on the MIC coprocessor. For more information on how to program the Intel MIC coprocessor using OpenMP, the reader is referred to the recently produced PRACE 'Best Practice Guide for the Intel Xeon Phi', available on the PRACE website [42].

Latest version/release: v3.1

2.2.2 *Evidence of use within PRACE*

Since the standard is mature and provides a relatively low barrier to entry for hybrid programming on large-scale multicore systems, (as well as many-core coprocessors more recently), OpenMP has been widely used (in combination with MPI) within PRACE to date. Although easy to pick up quickly it is generally found to be quite difficult to extract the same level of intra-node performance from an OpenMP shared-memory implementation as that of an intra-node MPI implementation. Here, we provide only a few examples of how OpenMP is being and has been employed in PRACE to date.

In WP7 PRACE-1IP, the use of OpenMP in a hybrid scheme was reported in deliverable D7.5 [43] in the work entitled 'Hybridization of parallel sparse matrix vector multiplication - BiCGStab algorithm'. In that work, which was carried out and tested on JUGENE, it was found that adding OpenMP to an MPI-based sparse matrix vector multiplication algorithm gives better performance only when the granularity of each MPI task is large enough to exploit shared memory parallelism. Also within PRACE-1IP, the PRACE whitepaper, 'Performance Analysis and Petascaling Enabling of GROMACS' [44] reports on the hybridised version of GROMACS, which was tested on the IBM PLX cluster at CINECA.

The hybrid OpenMP/MPI mode of GROMACS was shown to significantly improve the scaling of large problem sizes on a large number of cores, typically over 200,000 particles on over 500 cores. On systems with very fast interconnects such as Cray XE6, LINDGREN machine at PDC, the hybrid mode was shown to not offer much advantage.

More recently, within WP8 PRACE-2IP, on-going work has focused on developing hybrid OpenMP/MPI parallelism for Fluidity-ICOM with tests being run on the Cray XE6, HECToR machine at EPCC [35]. As part of this work, all matrix assembly kernels in Fluidity-ICOM have now been successfully threaded using OpenMP, where memory bandwidth usage through NUMA optimisations (e.g., first touch, thread pinning) and using a NUMA aware heap memory manager have shown to achieve best performance using pure OpenMP within the NUMA node. In the matrix assembly kernels, the OpenMP parallel algorithm uses graph colouring to identify independent sets of elements that can be assembled simultaneously with no race conditions.

Also within WP8 PRACE-2IP, on-going work is being been carried out on hybridising the ABINIT code using OpenMP in combination with MPI, where the focus has been on the non-local operator within the many-body Hamiltonian [35]. Tests have been carried out on an Intel Sandy Bridge-based platform (TGCC-CURIE), where the input data test case was for 107 gold atoms. The calculations were run over 128 MPI processes and good performance gains were seen when using up to 8 OpenMP threads (85% parallel efficiency), but for more than 8 threads, thread synchronization issues were encountered and performance was seen to degrade.

With WP8 PRACE-2IP, the OpenMP model is also being implemented in the NEMO code using both loop-level and tiling/coarse-grained approaches [35]. The effect of array-index ordering has also been investigated. In NEMO, the 3D arrays have the level/depth index outermost. The outer loop for the vast majority of loop nests is therefore over this index. It is this loop that is parallelized in the loop-level approach to using OpenMP. The proposed approaches have been applied in two different forms of the tracer advection kernel (MUSCL and TVD) in NEMO and are being evaluated on an IBM Power6 cluster at CMCC, Italy, an IBM iDataPlex with Intel Westmere CPUs at CINECA, Italy, a dual-socket Intel Sandy Bridge system at STFC Daresbury, UK and on a Cray XE6 (HECToR), UK.

It should also be mentioned that within WP8 PRACE-2IP, there is some early-stage work being carried out at CINECA in porting the Quantum Espresso suite of codes to the Intel Xeon Phi using OpenMP, where early results are showing promise [35].

2.2.3 *Evidence of use outside PRACE*

On large-scale systems OpenMP is obviously used in a hybrid model with MPI. Interesting results on the improved scalability achieved by using OpenMP within the Fluidity code [45] Using large matrices generated by Fluidity, an open source CFD application code, which uses PETSc as its linear solver engine, the effect of explicit communication overlap using task-based parallelism was evaluated. The authors also show how to further improve performance by explicitly load balancing threads within MPI processes. A significant speedup over the pure-MPI mode (2x speedup on 32,768 cores) and efficient strong scaling of the sparse matrix-vector multiplication on a Fujitsu PRIMEHPC FX10 and a Cray XE6 (HECToR) was demonstrated.

Since the Intel MIC architecture is only at an early stage of release, there is not much publicly available data on how OpenMP is being used to exploit the new architecture. In a recent study on how OpenMP fares on the Intel MIC architecture, 'OpenMP Programming on Intel Xeon Phi: An Early Performance comparison' [46] the overhead of the standard OpenMP constructs

which use synchronization was shown to be smaller than on large Symmetric Multiprocessing (SMP) machines, which makes the approach very promising for many HPC applications using OpenMP. The overhead of the Intel LEO `offload` pragma was also shown to be quite low, demonstrating that it will not limit scalability. The bandwidth of one MIC coprocessor was shown to be up to 156 GB/s, which exceeds eight Intel Xeon X7550 processors. With the Roofline model the authors of this work have predicted a maximum performance of about 20 GFLOPS for the SMXV kernel they use as a benchmark. The authors claim that their early work on the Intel MIC architecture already demonstrates that scientific OpenMP applications can run efficiently on the upcoming Intel MIC coprocessor without requiring a major rewrite of code. It is worth bearing in mind that most information relating to the performance of the Intel MIC coprocessor has been released very recently after fairly lengthy NDAs, so recent publicly available results have often been obtained on test systems based on the early Knights Ferry architecture and are only representative of what can be achieved on the latest Knights Corner version of the coprocessor.

2.2.4 *Pros and Cons*

Metric	**Pros**	**Cons**
Scalability	Improves scalability of the code within a node. Good scalability also demonstrated on the Intel MIC architecture	Needs to be combined with other multi-node parallelization techniques.
Performance	Can improve performance of the application by alleviating MPI intra-node communication bottlenecks. However, OpenMP intra-node performance is often seen to be worse than intra-node MPI performance	cc-NUMA behaviour and False Sharing can have a negative impact on performance. Thread creation and context switching are severe overheads.
Productivity	Easy to incorporate in the code, the code may run on new hardware without a rewrite (e.g. Intel Xeon MIC coprocessor). Can be used with Fortan, C and C++	No major cons, other than refactoring that may be needed to extract good performance.
Sustainability	Proved sustainability over the years, support included for new platforms and tools.	-
Correctness	Debugging tools are improving (e.g Intel ThreadChecker). Multithreaded codes are notoriously difficult to debug	There is no support for error handling. Improvements in error handling are being proposed for OpenMP 4.0
Portability	OpenMP is an open standard and is portable across many different architectures.	Using OpenMP on large-scale systems is possible with message passing interface (MPI) – code needs to be redesigned and rewritten.
Availability	Open standard, well supported by compiler	-

	vendors (e.g. Intel, AMD, Cray, PGI, NAG). Also supported by GNU	
Resilience	-	-

Table 3 OpenMP - Pros and cons

2.2.5 *Target systems/architectures*

X86	IBM	GPU	MIC	ARM	Heterogeneous
Yes	Yes	No	Yes	Yes	Yes (MPI+X)

Table 4 OpenMP - Target systems/architecture

2.2.6 *Conclusion*

OpenMP offers the easiest means of hybridising existing MPI-based codes, a model that is becoming increasingly important as the core-count on nodes continues to increase. With the advent of Intel's Xeon Phi coprocessor, OpenMP is already finding new target architectures in the many-core space, and could become even more relevant as a standard if plans go ahead to merge OpenACC into OpenMP 4.0 in the near future. One particular feature supported since OpenMP 3.0 that we feel has been under-investigated in PRACE so far is task-based parallelism and we feel that T7.2 offers a valuable opportunity to investigate this new feature further.

2.3 OpenCL

2.3.1 *Brief overview*

OpenCL [14] is a parallel programming open standard intended for use with heterogeneous computing systems. It is similar to CUDA, [13] in that it is able to target graphics processing units (GPUs). However, OpenCL is more general-purpose than CUDA, with a goal to provide a standard language to write efficient, portable code for multi- core CPUs, GPUs, Cell-type architectures and other parallel processors. Programs that utilize OpenCL consist of two parts, the traditional code (C/C++), and the OpenCL API, which enables the setup and control of execution kernels performing the computationally intensive work requiring parallelization. Kernels are written in a subset of the ISO C99 language that is compiled to target a particular computing device.

OpenCL supports both task and data parallel execution models, while CUDA is primarily focused on data parallelism. A kernel applies a single stream of instructions to vast quantities of data. Each piece of data is known as a work-item, and kernels can have a practically unlimited number of work-items. Kernels form the parallel unit of OpenCL, and they can be composed into a task via asynchronous command queues.

The standard hasn't changed since the last release in 2011, but there are more architectures that it now targets and new tools supporting the development of the code (e.g. Intel SDK (Software Development Kit) for OpenCL Applications 2013 [18], ARM Mali OpenCL SDK [47]). For example, OpenCL has been adopted on ARM platforms, including the new ARM Mali-T6xx GPU series, and is the focus of much work within the European Mont-Blanc exascale project. During 2013 the Mont-Blanc project is developing the first Mont-Blanc prototype hardware and in parallel with hardware development, they plan to develop the required OpenCL support for the embedded GPU in the HPC system software stack, starting from the OpenCL runtime environment itself as well as higher-level scientific libraries [5].

Intel recently released the new Intel SDK for OpenCL applications [18], which provides a development environment for OpenCL 1.2 applications across both Intel Xeon processor and the Intel MIC coprocessor. One of the nice features of OpenCL to point out here is that the same OpenCL source code written for the Intel Xeon processor can be reused on the Intel MIC coprocessor with minimum modifications. NVIDIA has not announced any official statement on further support for OpenCL. However, since late 2010 active support from NVIDIA for OpenCL has decreased. OpenCL support is still included in the latest NVIDIA GPU drivers, but in 2012 the code samples were removed from the CUDA SDK, focusing instead on CUDA.

It is worth pointing out that since OpenCL is built on a subset of ISO C99 there are a number of restrictions such as no recursion and limited pointers. These restrictions in the kernels can be limiting. Expanding OpenCL to handle these advanced features (similar to the way CUDA functionality has grown) would be beneficial.

Latest version/release: v1.2

2.3.2 *Evidence of use within PRACE*

OpenCL has been investigated within several WP7 and WP9 PRACE-1IP projects. In the PRACE Whitepaper ‘‘Benchmarking and analysis of DL_POLY 4 on GPU clusters [48], comparisons were made between a CUDA (v4.0) and OpenCL (v1.1) port of DL_POLY to GPU architectures. In terms of productivity, it was concluded in that work that the OpenCL framework required much more effort than CUDA. The authors found that the lack of quality documentation, flexible debugging tools and the small number of libraries around the standard made the development process challenging. They also point out that C++ templates are not supported in OpenCL, which resulted in a far greater number of lines of code compared to the CUDA version. It should be pointed out that AMD's OpenCL SDK now supports *OpenCL Static C++ Programming Language* extensions, which allows for some C++ features for writing the kernels (inheritance and passing classes instances from the host to the device). However, documentation is still poor. According to the Khronos website C++ bindings don’t officially support OpenCL 1.2 entry points yet [14].

In deliverable, D9.2.2, in PRACE-1IP section 2 [49], reports on how the performance of OpenCL was evaluated by porting the EUROBEN mod2am, mod2as and mod2f kernels. Tests were carried out on several many-core architectures including NVIDIA GTX480, AMD/ATI Radeon HD 5970, AMD/ATI Radeon HD 5870, NVIDIA Tesla M2050, AMD Brazos platform (Zacate E-350: CPU 1.6 GHz 2 cores, with AMD Radeon HD 6310 492 MHz), Intel Core i7 CPU 3.20 GHz. Results of the mod2am and mod2f benchmarks, for both Double Precision (DP) and Single Precision (SP), have shown that OpenCL on the GPU was 2x better than on the CPU for large matrix sizes, but these results should be considered in the context of the time that they were obtained and the rapid progress that has been made in many-core technology since then.

OpenCL has also been employed more recently within WP12 PRACE-2IP, where in the PRACE whitepaper, “Optimization of SHAKE and RATTLE Algorithms” [50] further enabling and optimisation of DL_POLY algorithms on GPU-based systems was reported. Development and testing was carried on local WCSS GPU machines (2x GTX480, 2x AMD Radeon HD 6900 Series) and performance results for the DL_POLY H2O benchmark showed that the OpenCL implementation ran slower than the CUDA version for the same algorithms The biggest performance difference between the NVIDIA-CUDA and the NVIDIA-OpenCL implementations occurs for kernels: `k1_th` (OpenCL code is 10 times slower than CUDA

code), and `install_red_struct` (OpenCL code is 5.5 times slower than CUDA code). For other kernels OpenCL calls are 2 times slower than equivalent CUDA calls.

2.3.3 *Evidence of use outside PRACE*

Although several Mont-Blanc application codes are partially ported to OpenCL, these applications are at a very early stage of testing on the Mont-Blanc prototype system and in general we found it very difficult to find examples of OpenCL-based petascale applications running on large-scale systems outside PRACE. We do note that the popular Molecular Dynamics code, LAMMPS, has been ported to GPUs on the Cray Xk7, Titan machine at ORNL using both CUDA and OpenCL builds [51] but it is not clear whether performance results provided are from the CUDA or OpenCL builds of the code. In associated work [52] more detail is provided on how OpenCL and CUDA are exploited within LAMMPS using the GERYON library, which provides and API allowing a single code to compile with both CUDA and OpenCL [53]. Benchmarks were performed on the Keeneland system where at the time of publication each node contained two 2 Intel Westmere hex-core CPUs and 3 Tesla M2070 GPUs. For the OpenCL comparisons, device code for both CUDA and OpenCL was compiled with version 4 of the CUDA toolkit. Performance comparisons of CUDA and OpenCL were made for the same kernels using the 'rhodopsin' LAMMPS benchmark where it was shown the accumulated time for host-device data transfer and compute kernel time was demonstrated to be 24% larger for the OpenCL executable.

One initiative that we do think is worth pointing out is VexCL - vector expression template library for OpenCL[54]. It has been created for ease of C++ based OpenCL development. Multi-device and multi-platform computations are supported. Source code of the library is publicly available under MIT license. VexCL is integrated into odeint - a C++ library for numerical solution of ordinary differential equations.

2.3.4 *Pros and Cons*

Metric	**Pros**	**Cons**
Scalability	Strong scalability, up to thousands of concurrent threads.	Scalability on distributed memory systems can be achieved only when external libraries are used (e.g. MPI).
Performance	A fairly low-level API that consequently has the potential to extract performance from multi-/many-core architectures	Performance is implementation dependent. Best performance can only be achieved when targeting specific architectures Some evidence suggests that OpenCL implementations tend to underperform when compared to CUDA, for kernels solving the same problem.
Productivity	Based on C99. Many language bindings and wrappers exist. No direct support for Fortran.	Architecture based knowledge is essential.

	Ecosystem is quite poor in comparison to CUDA	
Sustainability	It seems to be on roadmaps of major new device and architectures providers.	Difficult to discern long term support from several major vendors
Correctness	Debuggers exist (AMD CodeXL, OpenCL Studio)	-
Portability	Many architectures supported. Portable code.	
Availability	Open and royalty-free standard. Support from several compiler vendors (e.g., Intel, AMD, CAPS, NVIDIA, IBM)	-
Resilience	-	-

Table 5 OpenCL - Pros and Cons

2.3.5 *Target systems/architectures*

X86	IBM	GPU	MIC	ARM	Heterogeneous
Yes	Yes	Yes	Yes	Yes	Yes

Table 6 OpenCL - Target systems/architectures

2.3.6 *Conclusion*

One of the main appeals of OpenCL is that it is an open standard intended for use with a wide variety of heterogeneous computing systems. Since PRACE uses diverse computing systems, OpenCL does seem to offer a solution in terms of portability. OpenCL is a shared-memory programming model, and therefore must be used in conjunction with another model such as MPI for inter-node parallelism.

Developing efficient OpenCL code is typically found to require more effort than other GPU frameworks. There are some efforts, besides the new SDKs and tools for code debugging and analysis, which try to address this issue. During the ADAPT workshop on January 22nd, 2013 ACL (Adaptive OpenCL) was presented [55]. It was proposed for real-time computing environments of mobile computing, but we feel that the core idea of the algorithm support for dynamically adapting data-model properties and runtime machine characteristics might be worth further investigation for heterogeneous and dynamic HPC systems in the future.

OpenCL lives something of a double life between the worlds of HPC and mobile, consumer computing. What drives both of these areas is that they are equally committed to solving core problems around heterogeneous computing as is exemplified by the use of OpenCL in the Mont-Blanc project. In terms of HPC, OpenCL lives somewhat in the shadow of CUDA, but its relevance may change with the release of the Intel MIC architecture and the support that Intel puts behind it. We feel that T7.2 should liaise closely with European exascale projects such as Mont-Blanc project to learn how OpenCL will be exploited during 2013.

2.4 OpenACC

2.4.1 *Brief overview*

OpenACC [15] is a directive-based open standard supported by NVIDIA, PGI, Cray and CAPS, designed to simplify parallel programming of heterogeneous CPU/GPU systems. As is the case for OpenMP, the programmer can annotate C, C++ and Fortran source code to identify the areas that should be accelerated using `#pragma` compiler directives and additional functions, where the the #pragma directive retains the compatibility of the code when compiling with non-OpenACC compilers. Unlike OpenMP, code can be started not only on the CPU, but also on the GPU. The latest public comment draft version of the standard, OpenACC 2.0 [56], was announced in November 12, 2012 at the SC12 Conference but is still under review. If ratified, version 2.0 will offer a series of new features and capabilities for better memory usage, data handling and more flexible programming.

OpenACC 2.0 intends to offer new features to deal with what can be a bottleneck in data management and to increase performance. This is provided through unstructured data lifetimes, which allow data to exist on the device beyond a kernel or parallel region. This allows for increased data locality and fewer barriers. There is also a `default(none)` clause that can be added to data directives optionally. This ensures that no data is handled automatically, which can help to isolate reasons for poor performance. Version 2.0 also intends to allow for nested parallelism, where kernels on the accelerator can generate another accelerator kernel within. If this takes place entirely on the accelerator, the number of data transfer events between host and device are reduced. Additionally, the improved readability added through the use of directives makes any additions to the code simpler. This will be helpful for implementing changes to code that may be required for exascale execution.

OpenACC runs on multiple hardware platforms and operating systems. It is fully compatible with the NVIDIA CUDA and GPU libraries. It is interoperable with MPI and OpenMP. Moreover, it is intended to merge with OpenMP to create a common specification for accelerators in the future [57].

2.4.2 *Evidence of use within PRACE*

OpenACC has not been employed in many PRACE projects to date, but indications are that it is becoming increasingly popular as a means of porting legacy codes to GPU-based systems. OpenACC has recently been reported on within WP12 PRACE-2IP. The PRACE whitepaper **'**Investigating Performance Benefits from OpenACC Kernel Directives' [58] reports on how the performance of matrix-matrix multiplication and classical Gram-Schmidt ortho-normalisation were analysed for different OpenACC gang and vector sizes by using PGI and CAPS compilers on different hardware architectures. The authors demonstrated that when gang and vector sizes are modified to match the architecture, the performance increased compared to the automatic scheduling parameters. In particular, for a basic hand-coded DGEMM algorithm the PGI and CAPS compiled versions of the OpenACC-based code show a speedup of 1.7x and 3.1x respectively. For both applications, the best results were obtained for different scheduling parameters selected for the PGI and CAPS compiled versions. PGI and CAPS compiled versions of the two applications were compared with the serial runtime and OpenMP implementation with four threads. These versions were shown to perform similarly with the optimal parameters and were also shown to perform better than the serial and quad-core OpenMP implementations. In particular, for Gram-Schmidt orthonormalisation, the PGI compiled version achieved a speedup of 1.9x over the corresponding OpenMP version. For the compiled version with CAPS, tests were also carried

out on the new NVIDIA K20 architecture. The best performance for this case was obtained using different gang and vector sizes than were used for NVIDIA M2090 based tests, demonstrating a clear link to scheduling and architecture. Moreover, a 1.3x improvement was achieved on the K20 when compared to the M2090.

Work with OpenACC within PRACE is also on-going within WP8 PRACE-2IP as part of optimisation of the ICON Non-Hydrostatic Solver [35] OpenACC was used in combination with MPI and performance results were compared between the application running on an Intel Sandy Bridge node versus the application running on a K20X GPU. The hybrid version of the code was found to be 2x faster for cases where memory is fully exploited. This work is at an early stage and the OpenACC version and the authors are currently working on further optimisations.

2.4.3 *Evidence of use outside PRACE*

Outside PRACE, OpenACC has also recently garnered much interest as a parallel programming model particularly with respect to porting large legacy codes to GPU-based systems. A hybrid MPI/OpenMP/OpenACC version of S3D (a parallel direct numerical simulation solver for turbulent reacting flows) was developed recently, targeting the Cray XK7, Titan system at ORNL, where the performance of the OpenACC port has recently been evaluated. As reported in [59] the performance of the application approximately doubled when moving from a hybrid OpenMP/MPI implementation to a hybrid OpenMP/MPI/OpenACC on a GPU-based system. When comparing the OpenMP/MPI/OpenACC 16 core performance to a 32 Core OpenMP/MPI hybrid implementation, the performance improved by roughly 1.4x on the GPU-based system, indicating that OpenACC is showing promise as a directive-based approach for enabling applications on peta/multi-petascale heterogeneous systems. It should also be pointed out that the refactoring involved in porting to OpenACC also improved the runtime of the OpenMP/MPI hybrid code by roughly 12% due to restructuring loops, combining computation, and restructuring computation.

OpenACC is also being actively investigated within the European CRESTA project where a recent report [60] demonstrates how OpenACC allowed for the performance of Nek5000 benchmarks to be improved by a factor of 5x-9x over hand optimized serial code using an NVIDIA C2050 GPU, and 6x-12x over naive code using only five directives. Auto tuning for optimal gang and vector sizes led to a further 2x improvement depending on matrix size and shape. While OpenACC is often promoted as an easy means of porting applications to GPUs, it is not always a trivial task to port legacy code to the GPU. In a separate project within CRESTA [61], hybridisation of the HemeLB application (similar to the Lattice Boltzman Ludwig application) was attempted by taking an MPI/OpenMP version of the code and replacing OpenMP directives with the analogous OpenACC “parallel loop” directive. This resulted in a number of compiler errors however (PGI compiler v12.3), revealing difficulties with pointers containing pointers/references to other structures (“deep data structures”), and difficulties with data management for virtual functions. It was pointed out in this work that these errors would have also occurred with CUDA, and would require a major code re-write.

2.4.4 *Pros and Cons*

Metric	Pros	Cons
Scalability	Good scalability can be obtained. Can be improved by specifying gang, worker	

	and vector clauses, and nested parallelism introduced in version 2.0	
Performance	Achieves good performance compared to OpenMP by enabling use of GPUs. Improved performance with `async` clause, new data clauses and directives in version 2.0	Requires data management and movement between the host and device memory. Relies on heavily on compiler.
Productivity	Based on compiler directives. Is intended to be as easy to use as OpenMP. Easy to maintain. Compatible with low-level GPU languages and libraries	For good performance, restructuring of code is generally needed.
Sustainability	Full support from PGI, Cray, CAPS and NVIDIA; Webinars and a user forum is available in the official web site; Integration to OpenMP with any type of accelerator support is an ongoing work	Many other competing emerging GPU programming models
Correctness	TotalView and Allinea DDT support for Cray CCE 8 compiled programs. Advantage of compiler support to detect errors; Better memory and data handling in version 2.0 can reduce errors caused by incorrect data	Requires effort to decide data dependencies, and errors with data can be hard to detect.
Portability	Portable across operating systems, and multi-core processors such as NVIDIA and AMD GPUs, and Intel Xeon Phi	Convergence needed from compiler vendors
Availability	Open standard; Supported by the latest versions of PGI, Cray and CAPS compilers; There is a free 30-day trial of the PGI Accelerator compiler	Compilers are not open source
Resilience		

Table 7 OpenACC - Pros and Cons

2.4.5 *Target systems/architectures*

X86	IBM	GPU	MIC	ARM	Heterogeneous
Yes	Unknown	Yes	Yes	Yes	Yes

Table 8 OpenACC - Target systems/architectures

2.4.6 *Conclusion*

Due to its relative ease of use in comparison to both CUDA and OpenCL, OpenACC is becoming an increasingly popular model for porting legacy applications to GPU-based systems. The standard is still in its infancy and there are many issues with regards to implementation that still need to be resolved, which also make it difficult to assess the performance of the model. Indications are that NVIDA is fully supportive of initiatives in this area and are also increasing the level of OpenACC material on their website regularly. While it is generally appreciated that CUDA offers the ability to perform lower-level optimization for the GPU, OpenACC may be increasingly used as a means of efficiently probing the potential benefits of porting to GPUs, with CUDA being used in an optional second optimization stage. In terms of code maintainability, OpenACC is a much more attractive option than CUDA for large-scale codes. If OpenACC can prove itself over the next year or two, it may become the model of choice for programming GPUs in the near future. Either way, it seems likely that OpenACC will be merged in some way with OpenMP. It is interesting to note that Intel's Language Extensions for Offloading share similarities with OpenACC offloading directives and more than likely some sort of convergence will be reflected in the new OpenMP 4.0 standard in 2013 [57].

2.5 TBB and Cilk Plus

2.5.1 *Brief overview*

While TBB [16] and Cilk Plus [17] are two very different models for programming x86-based multi-core (and more recently many-core) shared memory architectures, we group a report on the two together here for the reason that they are both strongly supported by Intel as models for programming Xeon- and Xeon Phi- based platforms. According to Intel, in many cases, the two models can also compliment each other in the same code. The models are based on the C++ language and do not contain constructs for multiple nodes such that an existing framework such as MPI would be required for communication across nodes of a platform. They have also not been widely used to date within and outside PRACE.

TBB:

Threading Building Blocks (TBB) is a C++ template library intended to support task parallelism without explicitly managing threads. TBB does not implement any vectorization, which is left instead for the programmer to manage, either with pragmas, compiler flags, platform-specific or vectorization API calls such as SSE.

There are substantial differences between TBB's scheduling implementation and Cilk Plus as Cilk Plus narrowly concerns itself with fork-join thread scheduling, while TBB supports a very broad range of scheduling constructs. TBB bases its scheduling on a task graph, which is a directed acyclic graph of tasks, ordered by their execution dependence on other tasks. The core implementation concept for TBB is that it implements a "work stealing" algorithm to traverse this task graph in both a breadth-first and depth-first manner simultaneously, responsively balancing workloads while deferring tasks as little as possible.

TBB contains lower-level and higher-level components. The lower-level components of TBB include: Task Scheduler, Thread Local Storage, Synchronization Primitives, Memory Allocation, Threads and others. The higher-level domain of TBB has two elements: *Generic Parallel Algorithms*, which are analogous to C++ STL algorithms, and *Concurrent Containers*, which are analogous to C++ STL containers.

The lower-level TBB constructs are aimed at allowing TBB programmers flexibility in making lower-level parallelism choices to optimize performance while preserving a serial coding style. The higher-level TBB constructs conveniently express and automate many common tasks. Parallel algorithms such as `parallel_for()` and `parallel_reduce()` allow programmers to express normal serial programming constructs in a natural way, yet still take advantage of the TBB core task stealing. Similarly, concurrent containers include C++ constructs, which present the proper interface for use with parallel algorithms, such as `concurrent_vector()`. The programmer thinks in terms of for loops and reductions, while TBB manages the load balancing and task graph.

Other examples of the high-level TBB toolkit include the `parallel_pipeline` template, which encourages data locality by passing tasks through a pipeline in chunks rather than executing each pipeline stage.

Latest version/release: v4.1.3

Cilk Plus:

Cilk Plus is fairly straightforward as an extension of C/C++. The language is meant to be processor oblivious, which provides the programmer scalability without the need to rewrite code in order to utilize new architectures with additional cores. As such it is targeted mainly at legacy code that may be easily parallelized through the aid of the Cilk Plus keywords and parallel constructs.

The main concept behind the Cilk Plus programming model is work-stealing which is handled by the Cilk Plus runtime system. Cilk Plus contains only a handful of keywords that control the bulk of the parallel-related tasks. These are the `cilk_spawn`, `cilk_sync`, and `cilk_for` keywords. The language provides additional features such as elemental functions that take advantage of vector operations available on the hardware. Definitions for reductions are present to simplify certain codes.

The most commonly used keyword in Cilk Plus is `cilk_spawn`, which is used to branch parallel sections of code. Each spawn creates a strand of work that is scheduled for a worker to process. Both the user thread and Cilk threads may be used to perform spawned work, and the default thread that runs the strand is the parent thread that spawned it. If and only if there is a larger quantity of work to do, multiple strands are broken up to be processed by other workers.

Following a `cilk_spawn` in which there are dependencies on the return from the spawn, the Cilk Plus program must provide a matching `cilk_sync` to ensure that the spawned threads have finished their strands of work before moving forward. This synchronization point is local, meaning a parent thread may continue once its children threads have all finished, as opposed to a global synchronization, where all threads must finish. This allows for more fluid workflow control in a program, especially with respect to task-based parallelism.

The `cilk_for` keyword is used primarily where there is a higher level of data parallelism, where the many iterations in a standard `for` loop are independent of one another, allowing for parallel execution. One of the key advantages to using the `cilk_for` keyword, as opposed to a standard loop with `cilk_spawn` inside the iteration, is that the `cilk_for`

splits up the work using a divide-and-conquer approach. This decreases the overall execution time because it counters some of the overhead associated with the otherwise serialized spawning.

Latest version/release: Offered as part of the latest Intel compilers. As of August 2011, Intel has announced that it is maintaining Cilk Plus as a branch of GCC 4.7. The runtime library is available dual-licenced, including BSD-3.

2.5.2 *Evidence of use within PRACE*

Cilk Plus has been reported on within PRACE-1IP, where it was used in a novel approach that saw it combined with UPC: In the PRACE-1IP deliverable, D7.5, 'HPC Programming Techniques' [43], the section on 'A cache-oblivious matrix transposition (FFTW)', describes how UPC/Cilk Plus was explored an alternative to the de-facto standard programming model for petascale systems (i.e, the mixed MPI/OpenMP model). UPC/Cilk was evaluated as an interoperable alternative hybrid model, because it offers a uniform shared-memory programming interface. UPC was used for the distributed memory parallelization across multiple nodes, while Cilk was used for the shared-memory parallelization inside the node. The evaluation revealed speedups of 4x compared to the proprietary Intel (MKL) implementation using MPI/OpenMP. In general, it was concluded that UPC presents an efficient, concise and expressive alternative to MPI and mixed UPC/Cilk programming is an abstract yet efficient tool for large parallel computations.

To our knowledge there has been little documented evidence of TBB being used in PRACE to date. In the PRACE-1IP deliverable, D9.2.1, 'First Report on Multi-Petascale to Exascale Software' [62], a molecular dynamics algorithm with van der Waals interactions was used for testing TBB, where tests were carried out on a 4-core Intel Corei7 920 with 8 SMT cores and 4x speedup was achieved for large problem sizes.

2.5.3 *Evidence of use outside PRACE*

We have found it quite difficult to find evidence of TBB and Cilk Plus being used on large petascale systems outside PRACE. The only reports that we have found that may be of interest are those concerning Cilk Plus and TBB being used on the new Intel MIC architecture.

In a paper entitled, 'Efficient Hybrid Execution of C++ Applications using Intel Xeon Phi Coprocessor' [63] TBB was used in combination with a C++ template library developed at the university of Vienna to port an SPH code to the Intel Xeon Phi architecture. Performance gains are demonstrated when using TBB on the Xeon Phi (~27x speedup for 2 Coprocesors in combination with one Host vs a single core on the Host), but performance gains are difficult to properly extract due the code not being a pure TBB implementation.

The C++ deal.II 'Differential Equations Analysis Library' [64] developed to enable rapid development of modern finite element codes makes use of TBB in combination with MPI based libraries such as PETSc and has been shown to scale to ~16,000 cores. The highly scalable Trilinos multi-physics library [65] also uses TBB, but detailed reporting on how TBB is implemented in both of these packages is diffult to come by.

In a paper recently written by the Ohio Supercomputer Center in collaboration with Intel [66] comparisons were made between Cilk Plus and OpenMP on two Intel Knights Ferry (Xeon Phi) cards where both models were used in dense linear algebra factorization algorithms. Task parallelism was used for both Cilk Plus and OpenMP and it was demonstrated that at low thread counts both models performed similarly, but that beyond 30 threads, neither scaled and

OpenMP performed somewhat better than Cilk Plus. It should be kept in mind that this was an early stage test on the KNF architecture and the authors indicated that further investigations were underway.

2.5.4 *Pros and Cons*

Metric	Pros	Cons
Scalability	Task based parallelism upon task-stealing scheduler is meant to be highly scalable on a growing number of cores.	Interoperability (calling from the inside of functors/lambdas) with classic paradigms (MPI) needs to be verified.
Performance	The overhead introduced by the TBB library is low since its scheduling mechanism is meant to mask stall latencies. Cilk Plus performance should be as good as OpenMP, but evidence is hard to come by.	Evidence suggests that OpenMP runtime outperforms Cilk Plus in certain situations.
Productivity	Development time is low for people that are not used to C++ template metaprogramming, STL algorithms and iterators.	Cannot be directly used from codes other than C++.
Sustainability	Supported by Intel and backed-up by a fairly large community of developers and users.	
Correctness	TBB masks all the complexity of multithreaded programming, helping write efficiently scalable, less error prone codes. Intel debugging tools are available. TBB is a library, so should be robust	
Portability	The current releases have been successfully ported and tested on x86, Power and SPARC architectures, several operating systems and compilers. Performance portability should be high due to the shared-memory parallelism paradigm common to all platforms. Supported by the GNU compiler	
Availability	The TBB library is open-source (GPL) and publicly available for download. As of	

	August 2011, Intel has announced that it is maintaining Cilk Plus as a branch of GCC 4.7. The runtime library is available dual-licenced, including BSD-3.	
Resilience		

Table 9 TBB & Cilk Plus - Pros and Cons

2.5.5 *Target systems/architectures*

X86	IBM	GPU	MIC	ARM	Heterogeneous
Yes	No	No	Yes	No	Yes

Table 10 TBB & Cilk Plus - Target systems/architectures

2.5.6 *Conclusion*

The high-level nature of TBB is probably not a feature that will attract PRACE WP7 partners looking to extend or improve existing codes. Most likely it is the lower-level ideas that might be important. While most reviews of TBB have generally been made in the context of Intel Xeon-based platforms, it might be worth considering the potential benefits of TBB for the new Xeon Phi coprocessor. An initial port from OpenMP to TBB might well be straightforward and worth investigating further, particularly on Xeon Phi-based systems.

Although interesting, particularly with the Xeon Phi architecture in mind, Cilk Plus does not seem to provide much advantage over OpenMP at the moment. In comparing the keywords in Cilk Plus and the directives in OpenMP, it is clear that the ease of programming is not a concern for either, with OpenMP providing additional options in scheduling and allowing for NUMA effects in some variations. In this sense, Cilk Plus is considerably more limited than OpenMP. It is, however, worth keeping in mind the success of the novel Cilk Plus/UPC combination that was reported on in PRACE-1IP.

2.6 CUDA

2.6.1 *Brief overview*

CUDA [13] is a programming model and instruction set architecture initially released in November 2006 by NVIDIA to allow for application developers to access GPUs without having to use graphics application programming interfaces. CUDA comes with a software environment that supports C and C++, along with Fortran, OpenCL, and DirectCompute.

The core concepts for CUDA revolve around three key abstractions: a hierarchy of thread groups, shared memories, and barrier synchronization. These abstractions are accessible to the programmer through a set of language constructs. The result is fine-grained data-parallelism/thread parallelism nested within coarse-grained data parallelism/task parallelism, where the coarse-grained data parallelism/task parallelism is solved independently by blocks of threads. At present, data cannot be shared between GPUs, so a task is limited in size by the amount of memory on a single GPU.

For well-behaved problems, CUDA performance is seen to be good and facilitates reasonably easy exploitation of the underlying GPU hardware. However many advanced optimizations

are required to obtain peak performance. The optimizations require knowledge of the underlying hardware being used for development and can differ with new hardware architectures.

The software system has reached a point of reasonable maturity and continues to be widely adopted. Version 4.0 included easier support for multiple GPUs, as well as a Unified Virtual Address space between CPUs and GPUs on the same node. CUDA 5.0 (the latest release) offers two new features that target the new NVIDIA Kepler K20 architecture:

- Hyper-Q is one of the new features of the Kepler architecture. It allows multiple processes to launch work on a single GPU simultaneously. Since this maximizes the GPU utilization while decreasing the CPU idle time, it plays a key role in increasing overall performance.
- CUDA 5.0 allows the K20 GPU to use its parallel processing capability efficiently by employing a new feature known as 'dynamic parallelism'. It allows GPU threads to dynamically spawn new threads without synchronizing with the CPU host.

As part of CUDA 5.0, RDMA (Remote Direct Memory Access) support is added to NVIDIA GPUDirect technology [31]. It enables direct communication between GPUs and third party devices such as SSDs, NICs and IB adapters, and allows these devices to directly access GPU memory without the involvement of the CPU. This reduces `MPI_SendRecv` latency between GPUs and demands on system memory bandwidth.

Latest version/release: v5.0

2.6.2 *Evidence of use within PRACE*

CUDA has been used in several projects within PRACE and in many cases has achieved significant performance gains over applications running on a single CPU and in some cases over an entire compute node.

In the PRACE-1IP whitepaper 'Extending the QUDA library for Domain Wall and Twisted Mass fermions' [67] two GPU-enabled operators were analysed. Performance tests were carried out on NVIDIA M2090 GPUs and a BlueGene/P cluster (IBM PowerPC450) as well as a M2070 GPU and CrayXE6 "Magny-Cours" 12-core socket. The GPU accelerated inverter (using mixed precision) on a single M2090 GPU was found to perform better than the inverter running (in double precision) on 64 BlueGene 4-way nodes for the same tolerance. The GPU accelerated CG Solver on a single M2070 GPU achieves a speed-up of 4.7x compared to a single 12-way Magny-Cours CPU socket.

Some of CUDA's more recent features were reported on in PRACE-1IP deliverable, D9.2.2, 'Final Software Evaluation Report' [49], where single GPU bandwidth measurements were made using the STREAM benchmark. Tests were carried out on 8 GPU-based nodes (2 NVIDIA M2070 GPUs per compute node). Each node was equipped with a two-socket mainboard, with Nehalem Xeon CPUs operating at 2.7GHz. The version of MVAPICH2 used (1.8a2) incorporates optimized support for GPU to GPU communications via the standard MPI interface. To our knowledge, the work made the first documented use of GPUDirect within PRACE to enable peer-to-peer transmission between "peer-accesible-devices" and CUDA Inter Process Communication (IPC), which allows for the passing of CUDA events between processes. Use was also made of Unified Virtual Addressing (UVAS).

2.6.3 *Evidence of use outside PRACE*

As is the case within PRACE, CUDA has been exploited in various codes running on large petascale machines, including the Cray XK7 Titan machine at ORNL. Unfortunately, we have not been able to find many examples of where the new features, namely 'Hyper-Q' and 'Dynamic Parallelism' offered by the new K20 architecture have been used in real applications running on petascale systems. One case that we did find was the evaluation of Hyper-Q in the materials science and computational chemistry code, CP2K [68]. CP2K is parallelised using MPI and OpenMP and in several components of the code CUDA is used for acceleration. In the investigation that we refer to [68], a data set of 864 water molecules was tested. Without Hyper-Q, only one MPI process runs on each GPU. In this case, the workload is too small for the GPU and there is not much performance improvement over the CPU-only case. With Hyper-Q, it is possible to use the same number of MPI processes per node as in the CPU-only case, which means 16 MPI processes per GPU in this instance. This unlocks the full benefit of the GPU, leading to a speedup of 2.5x. A nice feature of Hyper-Q technology is that it can be tested relatively quickly as no extra coding effort to enable it is needed. The only requirement is that you run on a K20 and have the CUDA 5.0 compiler and runtime.

We also mention work on porting Denovo, a code used for radiation transport modelling, to Titan, which was recently reported in a paper entitled 'High Performance Radiation Transport Simulations - Preparing for Titan' [69]. In this work the SWEEP kernel was rewritten in C++ and CUDA, where CUDA 4.1 was used for the port to GPUs. It was reported that the GPU accelerated program runs 40x faster than on a single CPU core. It was also pointed out that more than 50% of time was spent in coding GPU irrelevant parts of the code and the rest was spent on CUDA tuning. More recently, the same team reports that the outcomes of DENOVO sweep redesign have been positive: initial results showed CPU-only code is faster on Cray XT5 by 2x. At the time of publication, the performance of the code on the XK7 (Opteron + GPU) exceeded the performance on the XE6 (dual Opteron) by 3.3x.

2.6.4 *Pros and Cons*

Metric	**Pros**	**Cons**
Scalability	Strong scalability, up to thousands of concurrent threads.	Scalability on distributed memory systems can be achieved only when external libraries are used (e.g. MPI).
Performance	Fairly low-level API that can extract very good performance from NVIDIA hardware.	
Productivity	Reasonably high productivity, the CUDA compiler, math libraries, tools for debugging and optimizing, manuals and other documentation in a single SDK. Nsight Eclipse Edition IDE also helps. Accessible through many different programming languages. Direct support for	Low-level control of the GPU is often required for better performance

	C, C++ and Fortran (the latter via the PGI compiler)	
Sustainability	Strong support from NVIDIA.	Proprietary framework. How this fits within exascale roadmap is difficult to know
Correctness	Powerful tools for debugging and profiling, more efficient with the NVIDIA Nsight Eclipse Edition IDE, Kepler retains the full IEEE-754 compliant single and double precision arithmetic introduced in Fermi, in addition to ECC memory.	
Portability	Portable between all NVIDIA devices.	Only runs on NVIDIA GPUs.
Availability	Available free on the NVIDIA website.	Not Open Source
Resilience		

Table 11 CUDA - Pros and Cons

2.6.5 *Target systems/architectures*

X86	IBM	GPU	MIC	ARM	Heterogeneous
Yes	No	Yes	No	Yes	Yes

Table 12 CUDA - Target structures/architectures

2.6.6 *Conclusion*

CUDA has been in existence since 2006 and is supported by a single vendor, NVIDIA, so the system has had time to mature. Since some of the new PRACE prototypes will consist of the latest K20 GPUs, we see the exploitation phase of T7.2 as an opportunity to enable applications to exploit the full compute resources of these new platforms using some of the new features being offered by CUDA 5.0, including 'Hyper-Q' and 'Dynamic Parallelism'. It will be interesting to see what effect direct-based approaches will have on the continued use of CUDA within PRACE. While it is generally appreciated that CUDA offers the ability to perform lower-level optimizations for the GPU, OpenACC may be increasingly used as a means of efficiently probing the potential benefits of porting to GPUs, with CUDA being used in an optional second optimization stage. If a directive-based approach like OpenACC can prove itself over the next year or two, it may become the model of choice for programming GPUs in the near future.

2.7 OmpSs

2.7.1 *Brief overview*

OmpSs [19] is an effort to integrate features from the StarSs [70] programming model developed by BSC into a single programming model. In particular, the objective is to extend

OpenMP with new directives to support asynchronous parallelism and heterogeneity (devices like GPUs). However, it can also be understood as new directives extending other accelerator based APIs like CUDA or OpenCL. The OmpSs environment is built on top of the Mercurium compiler and Nanos++ runtime system [19]. OmpSs takes many of the ideas from the StarSs model (in particular a similar directive syntax for specifying data dependencies and MPI communication as tasks), but extends it to support accelerator devices as well as task-based parallelism across clusters of shared-memory / multi-core systems via a distributed runtime layer. Numerous other improvements e.g. data dependencies on array sections, tasks around arbitrary structured blocks, and CUDA device targets have been implemented.

OmpSs currently supports shared memory systems and systems with CUDA GPUs as well as clusters of these. Support for other accelerator devices (Intel MIC, OpenCL devices) is planned. The OmpSs runtime is built on top of Nanos++, which can handle communication between nodes directly (using GASNet [71] allowing task scheduling to take account of synchronization and communication between nodes, and thus enable efficient programming on clusters of multi-core (or accelerated) nodes. Alternatively, explicit mixed-mode programming with OmpSs and MPI is possible using the StarSs model of communication tasks.

OmpSs is still not widely deployed, but the published work to date indicates that it should be possible to install on many of the PRACE systems. Development of OmpSs is ongoing, supported by Mont-Blanc and DEEP and so any attempt to use it in PRACE should collaborate closely with these projects. (It is expected that work with OmpSs will feature heavily within the Mont-Blanc project this year.) OmpSs is designed to work with several tools – Scalasca or Paraver for profiling and performance analysis, Tareador for code analysis and parallelization, and Temanejo for parallel debugging.

Latest version/release: Regular snapshots of the development version of OmpSs are available from the OmpSs website. At time of writing the most recent was version '0.7a-2013-02-20' (for the Nanos++ runtime) and '1.99.0-2013-02-20' for the compiler. A separate download of a nightly build of the OmpSs Fortran compiler is also available.

2.7.2 *Evidence of use within PRACE*

OmpSs has not been widely used within PRACE to date. However, most recently in WP12 PRACE-2IP, an investigation into the performance of OmpSs has been reported in the whitepaper, 'Analysis and Optimization of a Hybrid Linear Equation Solver using Task-Based Parallel Programming Models' [72]. Investigations were carried out on the MinoTauro system hosted by BSC. The performance of OmpSs was reported, where running on 12 hardware threads was shown to greatly reduce the total execution time of the application (by up to 80%), in comparison with serial version of the application. Extensive experimentation using MPI/OmpSs, and final comparisons with original MPI/OpenMP implementations, are still underway and will be reported within PRACE soon.

The PRACE-1IP deliverable, D9.2.2 'Final Software Evaluation Report' [49] reports on how OmpSs was used to parallelise the application benchmark, HYDRO. Initially, the investigators took a C version of HYDRO in the PRACE_HYDRO_V1 package to port to OmpSs. However, loops proved to be too fine-grain leading to too much overhead. A more recent version of HYDRO takes a more coarse-grain approach and was selected instead. In this version, the main iteration loop of the Godunov scheme was parallelized using OmpSs and was tested on different configurations of numbers of nodes to evaluate the scalability of the code. Runtime results of the HYDRO with MPI/OmpSs were obtained as a function of the number of MPI processes. Two threads per MPI process were used in all cases and 6 MPI

processes were placed per node. It was found that HYDRO scales quite well with the number of MPI processes. (The execution time was 36 seconds at 35 MPI processes and decreases down to 7 seconds on 320 MPI processes.). Just as useful as the reporting of performance in this deliverable is the reporting of recommendations by the authors which are too extensive to be reported here, but should certainly be consulted during the exploitation phase of T7.2.

2.7.3 *Evidence of use outside PRACE*

As OmpSs is at an early stage of development, there are few publications yet available demonstrating the use of OmpSs with real applications. However, the DEEP and Mont-Blanc projects both propose using OmpSs as part of their prototype exascale platforms. In [73], the use of OmpSs to implement a number of synthetic benchmarks is reported where tests were carried out on up to 32 nodes (4 cores per node) on a PowerPC cluster and where overall performance was generally found to be on a par with MPI, but was significantly faster for the 'Sparse LU' problem "…due to the easier expression of complex data dependencies." The work in [74] extends the above investigation to a cluster with NVIDIA GPUs where OmpSs was found to give comparable performance to CUDA but with far less lines of code required.

The Mont-Blanc project is investigating the use of OmpSs to port existing HPC applications to a low-power exascale architecture based on ARM CPUs. While not yet at a petascale level, some work has already been done. Selected application kernels have been chosen for detailed porting and optimization work using OmpSs but results are not yet publicly available.

Finally, the DEEP project proposes a hybrid architecture of standard CPUs and a 'Booster' cluster comprising Intel MIC accelerators, using OmpSs as a programming model to manage task scheduling across these resources in an efficient manner. Within the DEEP project, OmpSs will serve as an offload abstraction layer to hide the complexity of the MPI offload functionality from the user. In OmpSs, the developer specifies the input and output data needed for each task. The execution runtime dynamically calculates the dependency-graph and schedules the tasks accordingly on the available resources. To offload MPI-tasks, the developer can also specify the target MPI process in which the task must run, providing an additional level of flexibility. In DEEP OmpSs will be extended to define the 'Booster' as a destination device in which a given number of nodes will be reserved for the offloaded highly scalable code parts of the applications. Thus, from the application programmer's point of view these offloaded code-parts can be seen as single OmpSs tasks, even if their internal structure is highly complex and might actually include internal MPI-based communication operations. The allocation of nodes for these highly scalable code parts will be done statically at first place and later on dynamically, as the DEEP software environment is further developed. With the extension OmpSs eases the distribution of work between Cluster and Booster. However, the application developer is free to use the MPI layer directly and to fully control the offload functionality by him/herself [75].

2.7.4 *Pros and Cons*

Metric	Pros	Cons
Scalability	OmpSs is designed specifically for programming at the Exascale and development is closely coupled to the DEEP and Mont-Blanc projects, so good scalability of the runtime	No results at large scale have yet been published

	should be expected.	
Performance	Performance is typically equal to or better than traditional programming models (e.g. MPI/OpenMP), especially where there are complex dependency patterns and communication and computation can be overlapped.	-
Productivity	Porting of real applications to OmpSs showed that fewer lines of code were needed than for CUDA and CUDA+MPI	-
Sustainability	At least in the medium-term development will be supported by DEEP and Mont-Blanc. There is an effort to include task dependencies in the OpenMP 4.0 standard, somewhat similar to what is implemented in OmpSs.	-
Correctness	-	-
Portability	Supports C, C++ and Fortran source code	Ports to MIC, ARM and OpenCL are ongoing but not yet available.
Availability	Nightly builds are available from BSC website	Currently at 'alpha' development stage
Resilience	-	-

Table 13 OmpSs - Pros and Cons

2.7.5 *Target systems/architectures*

X86	IBM	GPU	MIC	ARM	Heterogeneous
Yes	Yes	Yes	Planned	Ongoing	Yes

Table 14 OmpSs - Target systems/architectures

2.7.6 *Conclusion*

By extending the OpenMP directive syntax and implementing a new runtime, OmpSs is a significant improvement over StarSs, and will make porting of existing applications to this model much more practical. The planned support of Intel MIC and ARM CPUs are clearly important as these types of components are likely to feature in future exascale architectures.

Similarly to StarSs, the ability to express complex data dependency patterns in a concise matter is clearly advantageous for large-scale parallel programming, and the improvements to both the programming model and runtime in OmpSs make it easier to port real applications without substantial re-engineering. However, OmpSs is still at an early stage of development and no results showing petascale performance have yet been published. The ability to manage parallelism across heterogenous architectures consisting of CPUs and accelerators in a transparent fashion will be necessary on deep petascale and exascale systems. OmpSs is an integral part of the DEEP and Mont-Blanc architectures for a future exascale system and we suggest that if OmpSs is to be exploited in T7.2, partners should work closely with both of these projects.

2.8 Co-Array Fortran (CAF)

2.8.1 *Brief overview*

Co-Array Fortran (CAF) [20] [76] is a small set of extensions to the Fortran 90 standard and subsequently now part of the Fortran 2008 standard [76] for SPMD, parallel processing. CAF is an example of a PGAS language which supports access to non-local data using a subscripted array syntax, lightweight and flexible synchronization primitives, pointers, and dynamic allocation of shared data. An executing CAF program consists of a static collection of asynchronous process images. Like MPI programs, CAF programs explicitly manage locality, data and computation distribution. However, CAF is implicitly based on one-sided communication so that rather than explicitly coding message exchanges to obtain off-processor data, CAF programs can directly reference off-processor values using subscripted references. Since both remote data access and synchronization are expressed in the language, communication and synchronization are amenable to compiler-based optimizing transformations. CAF is aimed at multi-core multi-node system architecture.

Although now part of the Fortran 2008 standard, CAF compiler support is still lagging. So far the Cray compiler on Cray systems has the most optimized implementation of CAF. Other implementations either work on single node (for gfortran) or multiple nodes using MPI (Intel CAF) or using GASNet (Rice university CAF [77]). These latter two implementations are currently un-optimized and not ready for large-scale systems. Although the latest version of the Intel Fortran compiler has support for CAF it currently compiles source code into MPI executables, which subsequently have to be launched using the Intel MPI runtime. The Rice University version of CAF is an open-source implementation. However, it should be pointed out that the Rice University implementation of CAF does not follow the CAF standard exactly.

Latest version/release: CAF is now part of the Fortran 2008 standard.

2.8.2 *Evidence of use within PRACE*

Although several PRACE Tier-0 and Tier-1 (Cray-based) systems, including HERMIT, LINDGREN, MONTE ROSA and HECToR offer CAF supported compilers, CAF-based development work has not been widely reported within PRACE to date. In PRACE-1IP a parallel benchmark suite for CAF was developed from scratch and reported on in deliverable D7.5 ‘HPC Programming Techniques’ [43] in a section entitled, ‘Parallel Benchmark Suite for Fortran Coarrays’. This benchmark was tested on Cray architectures and a general-purpose Intel cluster, where the project aimed to see if bottlenecks caused by MPI performance at large-scale could be solved by using CAF instead. It was shown that under certain problem-defined conditions, CAF can outperform MPI on platforms with appropriate hardware support

such as the Cray XE6, which has native compiler support and a communications network (GEMINI) that is optimized for remote memory access. On Cray systems without hardware support (e.g. the XT4 with Seastar2+ interconnect) performance was seen to be poorer than MPI. Anecdotal evidence suggests that performance of CAF on general clusters with recent Intel compiler CAF support is still quite poor.

2.8.3 *Evidence of use outside PRACE*

The use of CAF on large real applications is not very common. However there are several projects in which CAF is being investigated and where some performance gain has been achieved. The CRESTA [3] project has a CAF co-design team consisting of many active developers. CRESTA is using global Numerical Weather Prediction software from ECMWF called the Integrated Forecast System or IFS for testing CAF. The CAF implementation of IFS was tested on HECToR using the Cray compiler where on 70,000 cores it was found to be 20% faster than the original MPI/OpenMP implementation [78]. The CAF implementation of IFS actually involves a small part of the code where it has been implemented as a mix of MPI, OpenMP and CAF. In particular it is, to our knowledge, the first time that coarrays have been used in a real production application within the context of OpenMP parallel regions. The purpose of these optimizations is primarily to allow the overlap of computation and communication, and to reduce the volume of data communicated. It should be pointed out that if these developments are successful then the IFS model may continue to use the spectral method to 2030 and beyond on an exascale sized system.

2.8.4 *Pros and Cons*

Metric	**Pros**	**Cons**
Scalability	Based on one-sided communication. Easy to overlap computation and communication	Actual scaling figures for large number of cores are not known
Performance	Cray CAF is optimized. In point to point benchmarks it is faster than MPI.	Other CAF implementations only provide functionality. Performance is often poor.
Productivity	CAF syntax is very easy-to-use. CAF programs can be written faster than MPI programs.	Not tested much on large codes.
Sustainability	CAF is now part of Fortran standard. So vendors are likely to support it.	-
Correctness	Commercial debugging tools such as DDT and TotalView support debugging of CAF applications	-
Portability	As CAF is part of Fortran standard it should be portable as the Fortran code itself	As not many compiler vendors are supporting full Fortran 2008 standard portability is currently an

		issue.
Availability	Available mostly on Cray systems.	
Resilience		

Table 15 CAF - Pros and Cons

2.8.5 *Target systems/architectures*

X86	IBM	GPU	MIC	ARM	Heterogeneous
Yes	Yes	No	No (in dev)	No	No

Table 16 CAF - Target systems/architectures

2.8.6 *Conclusion*

It is not clear if CAF alone can be used on a petascale or future exascale system. There are very few decent CAF installations on general-purpose clusters and the only performing CAF implementations tend to be found on Cray systems. It has been demonstrated in the CRESTA project that CAF can be used in hybrid programming together with MPI and OpenMP, which suggests that CAF is one of the few PGAS languages that can be exploited in an incremental fashion with legacy code, which is undoubtedly an attractive feature for the exploitation phase of T7.2. It should also be noted that Intel has indicated support for CAF as a model for programming the Intel MIC architecuture and is currently developing support.

2.9 Unified Parallel C (UPC)

2.9.1 *Brief overview*

UPC, or Unified Parallel C [79] is a parallel programming language that is an extension of ISO C [ISO99]. Additionally, UPC is a PGAS (Partitioned Global Address Space) language using a globally shared memory programming model which exploits data locality combined with a distributed memory model for its underlying implementation. The application developer is presented with a single shared, partitioned address space, where variables may be directly read and written by any processor, but each variable is physically associated with a single processor. UPC uses the SPMD model of computation in which the amount of parallelism is fixed at program startup time, typically with a single thread of execution per processor. Parallelism is achieved through the use of shared memory and work sharing across independent UPC threads of execution, hereafter referred to as simply “threads.” Each thread has its own private memory space, as well as an associated shared memory region of the global address space that can be accessed by other threads.

The implementation of UPC threads is not restricted to actual user-space threads and the two are distinct concepts. In the case of Berkeley UPC, UPC applications run on top of GASNet and GASNet determines the actual thread implementation. Possible GASNet layers include pthreads, PSHM (process shared memory), MPI (in which the UPC shared memory is actually implemented via MPI calls), and various network APIs such as Infiniband verbs.

The number of UPC ‘threads’ is fixed at program startup, and does not change during the code’s execution. This attribute of UPC makes it similar to MPI in that each process or ‘thread’ is alive from inception through exit.

Shared memory variables in UPC form the foundation of UPC's parallelism. Rather than exchanging data across threads through explicit communication as in MPI, information is exchanged primarily through the use of shared memory.

Shared memory variables are declared through the use of the shared qualifier. In UPC, shared variables are always of global scope and must be declared globally; there is no provision for local shared variable declarations.
UPC compilers are compliant to a UPC specification that is not part of the ANSI C standard.

Latest version/release: Berkeley UPC v2.16.0

2.9.2 *Evidence of use within PRACE*

UPC has not been widely used in PRACE to date. UPC was used in PRACE-1IP and was reported on in the PRACE-1IP whitepaper, 'Porting and Optimizing HYDRO to new platforms and programming paradigms - lessons learnt' [80]. In this project, HYDRO, a 2D Computational Fluid Dynamics benchmark code, was parallelised using UPC. Tests were carried out using the Cray UPC compiler on HERMIT and performance was shown to scale across a number of threads. However, significant degradation occurred in memory accesses outside the affinity of the threads, especially when accessing memory across nodes.

In the PRACE-1IP, deliverable, D7.5 [43] a report was conducted on 'A cache-oblivious matrix transposition (FFTW)' where the main goal of this work was to explore an alternative to the de-facto standard programming model for petascale systems, i.e., the mixed MPI/OpenMP model. The authors evaluated UPC-Cilk as an interoperable alternative hybrid model, because it offers a uniform shared memory programming interface. UPC was used for the distributed memory parallelization across multiple nodes, while Cilk was used for the shared-memory parallelization inside the node. The evaluation revealed speedups up to a factor of 4x compared to the proprietary Intel (MKL) implementation using MPI/OpenMP. In general, the authors concluded that UPC presents an efficient, concise and expressive alternative to MPI and mixed UPC/Cilk programming is an abstract yet efficient tool for large parallel computations.

2.9.3 *Evidence of use outside PRACE*

Just as in PRACE, example cases of where UPC has been employed to enable real codes on large petascale systems have been difficult to find.

There has been some interesting work carried out recently at Intel's Exascale Science Labs [81] where a 2D Electrostatic Particle-in-Cell algorithm was implemented in UPC with dynamic load-balancing [82]. The underlying algorithm that was implemented was the Conjugate Gradient method and the authors of the work were inspired by recent investigations showing that UPC can be used efficiently for the development of 1D PIC codes The overall conclusion made by the authors was that UPC allows for an efficient design and implementation of the required data structures, but the level of detail provided on the scalability of the UPC code is quite low, other than that weak scaling on up to 500 "threads" is shown to be good.

UPC functionality was added to the Ludwig Lattice Boltzmann application [84] and tested on the X2 component of HECToR on 122 vector processors. Although a full conversion was beyond the scope of the work, the ability of UPC and MPI to coexist allowed for the key section of the code to be converted to use UPC rather than MPI communications, and performance comparisons to be made. The use of UPC reduced complexity by allowing data structure halo cells and associated message passing halo-swap routines to be replaced by more

intuitive direct remote memory accesses. A straightforward adaptation involving direct use of UPC shared data structures was found to perform significantly worse than the MPI version, but it was found that this was not primarily attributable to communication performance degradation, but instead to overheads involving shared pointer operations. An optimised version using regular C pointers (obtained via casting) where possible was found to perform more comparably to, but still slightly worse than, the MPI version

Finally, in the paper, 'Hybrid Parallel Programming with MPI and Unified Parallel C' [83] a real hybrid of MPI+UPC application with good results was demonstrated. As the authors underline,

"…this model offers an incremental pathway that allows existing applications to take advantage of MPI's locality control and UPC's global address space. In addition, it can serve as a test-bed for developing new programming models that aim to combine these features. For memory-constrained MPI codes, the hybrid model enables the processing of larger problems by aggregating the memory of several nodes into a single, shared global address space. For locality-constrained UPC codes, the hybrid model can improve locality through the creation of UPC groups that are connected with MPI."

The model was evaluated on two benchmarks, a random access benchmark and the 'Barnes-Hut' n-body simulation. Compared against a baseline execution on 256 cores, it was found that, for groups that span two cluster nodes, the hybrid random access benchmark yields a 25% improvement in execution time and hybrid Barnes-Hut experiences a 2X speedup. In the case of Barnes-Hut the cost of hybridization was a 2% increase in code size.

2.9.4 *Pros and Cons*

Metric	**Pros**	**Cons**
Scalability	Good when thread has affinity to the memory it is accessing (up to 4K cores).	One may need to employ practices, which require explicit knowledge of the memory model to obtain good scaling. In such cases, the benefit of using UPC over MPI is not obvious.
Performance	In the serial sections UPC is identical to C with the well known benefits in performance associated.	
Productivity	For some simple parallel applications, implementing an algorithm in UPC can be much faster than in MPI.	For complex data accesses, where threads need to access non-local data, the developer may require to explicitly take care of data accesses, leading to considerably more complex code.
Sustainability	The first version of UPC was released in 1999 and development of UPC compilers is still ongoing.	Although versions have been released continuously for almost 15 years, UPC is still not considered a mature programming language for

		parallel architectures.
Correctness	C debugging tools can be used for the serial versions of the code. There are versions of the Totalview debugger (7.0.1 or greater) that will debug UPC programs on x86 architectures.	
Portability	Porting from C to UPC can be trivial for kernels where threads operate on local data.	For more complex kernels where threads operate on data local to other threads, one may require increased effort for porting as well as a detailed understanding of the application's details such as data paths.
Availability	A number of open-source and commercial implementations are available. The specification itself has reached a level of maturity to allow for several conforming compilers. Prominent compilers include the following: HP UPC (commercial), Cray UPC (commercial), GCC UPC (free), Berkeley UPC (free), Michigan Tech MuPC (free), IBM UPC Alpha Edition (commercial)	
Resilience		

Table 17 UPC - Pros and Cons

2.9.5 *Target systems/architectures*

X86	IBM	GPU	MIC	ARM	Heterogeneous
Yes	Yes	No	No	No	No

Table 18 UPC - Target systems/architectures

2.9.6 *Conclusion*

The present nature of the UPC language requires that an entire application be modified if one small region is to be parallelized. This implies that the time to parallelize a large code with UPC will always be longer than the time to parallelize with OpenMP, so for the moment we feel that UPC should be treated with curiosity by PRACE partners, but not as a candidate to incrementally add new levels of parallelisation to PRACE applications. For those interested in

investigating further, an excellent paper has been written outlining a successful process of MPI/UPC hybridisation [83], which was referenced and reported on above.

2.10 Chapel

2.10.1 *Brief overview*

Chapel (Cascade High-Productivity Language) [8] is a general-purpose parallel programming language being developed by Cray under the DARPA High Productivity Computing Systems (HPCS) program. Taken from its name, Chapel's purpose is to increase programmer productivity while enhancing code robustness. Chapel's aim is to support the expression of all parallelism in an application while targeting all available hardware parallelism with a single, unified set of language concepts.

Chapel also operates under a multi-resolution philosophy, meaning that programmers can initially write very abstract code and subsequently add more detail to tune for their target architecture. Object-oriented design, type inference, and other features allow for rapid prototyping and code reuse.

Chapel has several general and parallel language constructs that are meant to reduce the amount of code necessary to express a concept or perform work. Among many others, these include constructs to deal with the distributed nature of a global-view data structure, concurrency constructs, and data and task parallelism constructs.

- Data parallelism is generally invoked by the `forall` keyword where iterations of an otherwise serial loop may be calculated independently of other iterations. The number of threads that are used for this all- way parallelism depends on how many cores exist on a processor, but may be changed through configuration variables. Other methods for managing data parallelism include reductions, scans, and shorthand forms in dealing with arrays as a whole.
- Task parallelism is invoked using the `begin` and `cobegin` keywords for unstructured task parallelism and the `coforall` keyword for structured task parallelism. In general, there is a distinct thread for each task spawned through a task parallelism construct, and there may be many more threads than there are cores on a processor. In the case of structured task parallelism, each iteration is processed as a separate task from other iterations, and generally, the serial code inside an iteration is more complex than its data parallel counterpart. Because the threads of a task parallel region of code may exceed the number of cores in a processor, the kernel may switch among threads leading to possible issues with concurrency for poorly written codes.

All parallelism in Chapel is implemented using POSIX threads by default, and all communication is implemented using the portable GASNet communication library [71] supporting one-sided communication and active messages. As a result of this approach, Chapel runs on most parallel systems, whether custom or commodity.

The interoperability with other languages is presently not fully functional, and there are issues with calling Chapel generated code from an external language. Moreover, as Chapel does not support explicit pointers, it will require a great deal of effort to successfully glue Chapel together with languages such as C.

There is currently a lot of interest in how PGAS languages will confront the challenges associated with heterogeneous systems. It is generally appreciated that Chapel's current definition of locales is very adept at describing horizontal locality such as that which exists between nodes of a homogeneous cluster. However, once a system's compute nodes involve

NUMA domains or heterogeneous resources, Chapel programmers have no way to target code to specific processors or memories. To this end, the Chapel team is working on adding a concept of hierarchical locales to represent architectural substructures or realms of locality within a node (i.e. vertical locality) [85]. The idea is that a programmer could use Chapel's on-clauses to specify that a task should run on a specific processor type or instance, or to allocate a variable using a specific memory. Beyond this, the Chapel team and others are also carrying out very interesting work on allowing for Chapel to target GPUs [9].

Latest release/version: Version 1.6.0

2.10.2 *Evidence of use within PRACE*

Chapel has not been widely employed within PRACE to date. It was first reported within PRACE-PP in deliverable D6.6, 'Report on petascale software libraries and programming models' [87] where the EUROBEN kernels mod2am (dense matrix-matrix multiply), mod2as (sparse matrix-vector multiply) and mod2f (1D FFT) were ported to a Cray XT5 (AMD Barcelona processors at 2.3 GHz, 2 quad-cores per nodes). Version 0.9 of Chapel was used where performance results for the Chapel port were found to be very poor, however, it was explicitly noted by the authors of the report that the performance of the Chapel compiler was very poor and that the Chapel port should only be seen as a proof-of-concept and should not be used in performance studies at this premature stage.

Chapel has also been more recently reported on within PRACE-1IP in deliverable D9.2.1 [62] where it was noted that since the initial evaluation of Chapel during PRACE-PP the language and the compiler/runtime have evolved significantly. One of the most notable areas of improvement is interoperability with other languages. Chapel now supports calling C functions, converting to/from C data-types and using native C data-types. In principle, this permits linking C libraries such as LAPACK, BLAS, or MKL. However, it is unclear as of now, how regular C functions access distributed data. Chapel has not been designed to interoperate with other parallel programming models. However, as long as the model is implemented as a library, such as MPI, Chapel could be used in a hybrid setting. In terms of performance, the authors of the deliverable note that Remote Memory Accesses are implemented inefficiently, which is the primary reason why the EUROBEN kernels do not perform well. At the time of writing of D9.2.1 this situation had not changed since the initial evaluation. Tests with the Chapel compiler v1.2.1 on the Gemini interconnect of the Cray XE6 exhibit improved performance due to very low network latency. However, the benchmarks did still not scale beyond an unacceptably small number of nodes.

2.10.3 *Evidence of use outside PRACE*

As is the case within PRACE, Chapel has not been widely used on real applications on large – scale systems around the world to date. Performance measurements with respect to several of the HPC Challenge benchmarks have been performed. The most current published results from the challenge are from 2009 by the Chapel team; their entry for class 2 (most productivity) was awarded "most elegant implementation". Although the language has evolved since then, the performance results show that Chapel was comparable to an MPI version of the code for the Global and EP STREAM Triad benchmarks. Performance on the Random Access benchmark was not competitive with MPI. In terms of productivity, the paper noted that Chapel consistently beat MPI in terms of lines of source code (Chapel had far fewer lines for the same program) [88].

2.10.4 *Pros and Cons*

Metric	Pros	Cons
Scalability	Task spawning and synchronization across the machine seems to be reasonably efficient.	RMA transfer very inefficient; does not allow scaling of communication intensive applications. Generally, scaling has been seen to be poor to date, but is expected to improve.
Performance	Scalar performance has improved significantly.	-
Productivity	Very short, readable code. Easy to program and maintain. Clear and powerful concepts for parallel programming.	Practically no tools support.
Sustainability		Only supported by Cray
Correctness	-	No support on current debuggers
Portability	Intermediate C Code generated, can be ported onto Windows (Cygwin) or any Unix-like platform. Can be used with C language, but interopability at an early stage	-
Availability	Can be used under the terms of the BSD license and a user agreement.	-
Resilience	-	-

Table 19 Chapel - Pros and Cons

2.10.5 *Target systems/architectures*

X86	IBM	GPU	MIC	ARM	Heterogeneous
Yes	No	On-going work	No	No	Yes

Table 20 Chapel - Target systems/architectures

2.10.6 *Conclusion*

The primary advantages Chapel has over MPI are in programming ease and elegance. Because it is a multi-resolution language, it allows for quicker prototyping of algorithms, with architecture specific optimizations added in later. This provides greater code robustness due to a decreased chance for communication errors. Finally, because Chapel is multi-level, it is

able to provide homogeneous coding semantics, whereas one may have to add OpenMP to MPI code to fully utilize processor capabilities. With respect to performance, Chapel currently lags behind OpenMP and MPI in general, although it is comparable in some cases and is expected to improve overall as the language progresses. We feel that T7.2 were to explore the possibilities of exploiting Chapel (this would be difficult with a real application), it would be interesting to test its current capabilities on heterogeneous systems.

2.11 X10

2.11.1 *Brief overview*

X10 [22] is a parallel asynchronous Partitioned Global Address Space language based on Java and developed by IBM. Like Chapel, the X10 project started in 2006 as a response to the DARPA HPCS program. X10 is called an asynchronous PGAS language because of the ability to explicitly control concurrency through constructs such as `async` and `finish`.

The stated motivation behind X10 is to provide a language that addresses the inherent complexities of the increasingly popular many-core architectures (of which NVIDIA Kepler and the Intel MIC are examples) in a single, unified programming model. The goals of the X10 project are to create a language that is simple (hence its Java base), safe (from design errors, and through static checking), powerful (capable of expressing typical HPC codes), scalable, and universal (can be used and deployed on a host of architectures). Using the X10/CUDA backend, one can identify fragments of an X10 program to run on the GPU. For ideal workloads, this can give a speedup of up to 30x or more (claimed by X10 developer team on X10 website [22])..

X10's asynchronous PGAS framework provides the programmer with greater flexibility than a standard PGAS model by allowing threads (activities) to be created dynamically and by making it possible for dynamic load balancing to occur. This means heterogeneous systems and applications that require load balancing should both be supported, increasing the potential for adoption.

Like Chapel, X10 should greatly reduce the amount of code required for a particular parallel application, relative to the amount of code in an MPI-based or MPI/OpenMP code. This is due to the absence of message passing code, as global memory locations can be addressed as if they are local.

IBM developerWorks [89] provides an X10 debugger called 'The IBM Parallel Debugger for X10 Programming.' The X10 Parallel Debugger is still a fairly new product, so bugs are to be expected. However, the goal of the project is to provide a complete parallel debugger for X10 code, so in the long term, it should be robust.

Latest release/version: v2.3.1

2.11.2 *Evidence of use within PRACE*

X10 has not been widely employed within PRACE to date. It was first reported on within PRACE-1IP in deliverable D6.6, 'Report on petascale software libraries and programming models' [87], where the EUROBEN kernels mod2am (dense matrix-matrix multiply), mod2as (sparse matrix-vector multiply) and mod2f (1D FFT) were ported to an IBM pSeries 575 (108 nodes with 32 Power6 cores at 4.7 GHz by node) and where performance results for the X10 port were less than impressive. However, it was explicitly noted by the authors of the report that the performance of the X10 compiler (v1.7.5) was very poor and that the X10 port should

only be seen as a proof-of-concept and should not be used in performance studies at this premature stage.

A more recent investigation of X10 was reported in WP12 PRACE-2IP in the PRACE whitepaper, 'Parallelization Using a PGAS Language such as X10 in HYDRO and Triton' [90]. In that project X10 was used to parallelise two CFD codes, Hydro and Triton and was tested on the CEA Titane CCRT system (1068 nodes, 2 processors Intel Xeon 5570 by node, 4 cores by processor) Version 2.2.1 of X10 was used, but the test case was too small to obtain valuable conclusions. The measured compute times showed fair scalability behaviour (over 25 8-way compute nodes) but performance was far worse than the MPI/OpenMP versions of the codes.

2.11.3 *Evidence of use outside PRACE*

The X10 development team recently participated in the 2012 HPC Challenge contests [91]. In this contest, X10 received the "Best Performance" award. Benchmarks were run on up to 55K cores, and showed good scalability and performance (80%-100% scalability depending on the benchmark, 40%-80% relative performance vs. reference HPC Challenge implementation). On one of the benchmarks, X10 code performed significantly better due to the strongly unbalanced test behaviour [92].

Another interesting use of X10 can be found within Anuchem [93] a collection of computational chemistry codes written in X10. These codes are experimental in nature and are not guaranteed to run against the latest stable release of X10. However some components may be of interest to those considering X10 during the T7.2 exploitation phase. Examples of implemented algorithms are the parallel fast multi-pole method, or Hartree-Fock method. Performance studies have used up to 256 cores on the Watson 4P Blue Gene/P system (4 cores by node). The code exhibits close to linear scaling on up to 64 places (strong scaling experiment for 51000 particles). However, scaling is reduced for more than 64 places because of the relatively small problem size.

2.11.4 *Pros and Cons*

Metric	**Pros**	**Cons**
Scalability	Low level benchmarks scales up to ~50K cores, application tests up to 256 cores	
Performance	Performance improved a lot with newest X10 versions	Still behind MPI-based code
Productivity	Powerful concepts for parallel programming. X10 syntax was simplified in recent versions. Dedicated Eclipse framework version for X10 (with intelligent editor)	Experience needed in object-oriented programming. No clear indication that X10 can be used with other languages such as C, C++ and Fortran
Sustainability		Specification and development by IBM only.
Correctness	Dedicated parallel debugger	

	for X10 developed by IBM	
Portability	X10 binaries available on several platforms (Linux X86 – X86_64, Windows with Cygwin, MacOSX, BlueGene/P, AIX/Power), or can be compiled from source code	Even if X10 tools are compiled from source code, some external tools are available in binary form only
Availability	As X10 is managed by IBM only, availability of X10 follows the language specification versions	-
Resilience		

Table 21 X10 - Pros and Cons

2.11.5 *Target systems/architectures*

X86	IBM	GPU	MIC	ARM	Heterogeneous
Yes	Yes	Yes (using dedicated gateway)	No	No	Yes

Table 22 X10 - Target systems/architectures

2.11.6 *Conclusion*

Like Chapel, programming in X10 would be a radical shift for the majority of PRACE partners working within WP7. However, this is not necessarily a negative point if the shift provides benefit in the form of reduced time to code, large reductions in lines of code, and greater maintainability. Also like Chapel, at present, X10 cannot exist as a secondary language; while C or C++ object files can be linked into an X10 application, the reverse is not possible. Like Java or C++ there are characteristics of the language syntax that may make it more difficult for some programmers to maintain or work with. A solid understanding of existing classes would be necessary to optimally program and maintain application code, and in many cases there may be a nontrivial learning curve associated with acquiring this understanding, especially within the HPC community, where most code is written using C and Fortran. Beyond productivity considerations, the real issue at the moment when considering X10 is performance and scalability, which is yet to prove itself within PRACE or on a wider basis.

2.12 Global Arrays Toolkit

2.12.1 *Brief overview*

The Global Arrays (GA) Toolkit from Pacific Northwest National Laboratory (PNNL) [23] provides an efficient and portable "shared-memory" programming interface for distributed-memory computers. Each process in a MIMD parallel program can asynchronously access logical blocks of physically distributed dense multi-dimensional arrays, without need for explicit cooperation by other processes. Unlike other shared-memory environments, the GA model exposes to the programmer the non-uniform memory access (NUMA) characteristics

of the high performance computers and acknowledges that access to a remote portion of the shared data is slower than to the local portion. The locality information for the shared data is available, and a direct access to the local portions of shared data is provided.

Global Arrays has been designed to complement rather than substitute for the message-passing programming model. The programmer is free to use both the shared-memory and message-passing paradigms in the same program, and to take advantage of existing message-passing software libraries. Global Arrays is compatible with MPI and can be used with Fortran, C, C++ and Python- based source code. Fortran, C and C++ support is included by the toolkit natively

GA is aimed explicitly at distributed-memory architectures and its primary goal is to provide an efficient and portable "shared-memory" programming interface for such systems. GA does this by exposing a simple structure – a 'Global Array', whose storage can potentially span the whole memory of the distributed system. The programmer is given a uniform API to access array elements regardless of whether the element in question is stored locally or on another node. This greatly simplifies the programming model, but more importantly, it provides a way to work around memory-size bottlenecks in machines with hard memory limit. For example, an IBM Blue Gene/P system typically has 2 GB of RAM on every node, which might not be enough for certain tasks. GA allows the programmer to allocate/use arrays much larger than the 2 GB, where the programmer can, in fact, allocate an array that spans the entire RAM of the whole machine/partition.

The tool is based on an internal portability layer (ARMCI) [94] with support for a large selection of architectures, including IBM Blue Gene/P, IBM Blue Gene/L, Cray XT4/XT5, Infiniband/OpenIB, Myrinet, systems with Qsnet interconnect, IBM SP, etc. The ARMCI layer is built in the library and there are no additional steps needed in order to use the toolkit, apart from linking with the appropriate shared static library.

The ARMCI portability layer is specifically optimized for some of these architectures. For example, on an IBM Blue Gene/P it uses the low-level DCMF (Deep Computing Messaging Framework) layer to implement one-way data communication instead of relying on MPI to do this (in fact, MPI on IBM Blue Gene/P is also implemented using DCMF, so GA skips one layer of complexity and gains a lot of performance). On platforms for which no such low-level interface is available, the Global Arrays primitives are implemented on top of MPI. This effectively means that GA is available on every platform for which MPI is available.

The tool has a large selection of supported platforms and specific instructions for compiling on each of them. Basically, it relies on MPI compiler wrappers and the compiler suites native for each platform are usually supported.

Latest version/release: v5.1.1

2.12.2 *Evidence of use within PRACE*

Global Arrays has not been widely used to date within PRACE. However, it was employed in WP7 PRACE-1IP and reported on in the PRACE whitepaper, 'Data I/O Optimization in GROMACS using Global Arrays Toolkit' [95], where the authors describe how they used v5.0.2 of the tool to enable GROMACS to scale on an IBM BG/P. The GA toolkit helped in solving the memory-bottleneck in GROMACS on an IBM Blue Gene/P due to the hard memory limit of the Blue Gene/P compute nodes having only 2GB RAM (512MB in VN mode). By using GA the authors showed how GROMACS handled a system with over 2,000,000 atoms in VN mode, a problem that was otherwise not amenable to execution on such a system.

2.12.3 *Evidence of use outside PRACE*

A GA-based parallel implementation of NWChem coupled cluster calculations was performed at 1.39 PFLOPS using over 223,000 processors on ORNL's JAGUAR system, an achievement that won the Gordon Bell Finalist at SC09 [96].

While we found it quite difficult to find other examples of how Global Arrays is being exploited outside PRACE, we point the reader to an interesting article on how Global Arrays parallel programming model has recently been implemented using MPI's RMA functionality. This implementation was achieved by porting GA's low-level ARMCI PGAS runtime system to MPI's one-sided API where similar performance to the conventional ARMCI-based implementation was achieved [97].

2.12.4 *Pros and Cons*

Metric	**Pros**	**Cons**
Scalability	The tool mainly uses one-sided communication patterns, which show excellent scalability. On architectures that offer low-level access to internal messaging framework, it would surpass plain MPI.	-
Performance	Excellent performance – SC09 Gordon Bell finalist, achieving 1.39 petaFLOPS on Jaguar (Cray XT5)	In order to achieve transparent one-sided communication, a separate thread has to be used to advance the messaging state. On architectures that allow spawning limited number of threads (e.g. IBM Blue Gene), this can be viewed as one less worker thread and thus poorer performance.
Productivity	Offers shared-memory programming model in distributed-memory architectures, which greatly simplifies certain tasks and thus improves productivity. Excellent high-level abstraction leading to easy to use, but powerful API.	-
Sustainability	The project is actively being developed and supported with new version appearing every 6 months or so.	-
Correctness	The package comes with its	-

	own sanity-test routines that are performed during build time. The results usually show no major show-stopping bugs and raise the confidence level regarding the correctness of the package.	
Portability	The package is portable and supported on many high-performance architectures and clusters (Cray, IBM Blue Gene, special support for Infiniband and Myrinet, etc.) For those architectures that expose low-level messaging frameworks, such libraries are used as optimizations. On others, GA is built on top of one-sided MPI communication primitives, so the overall portability is not lesser than that of MPI. Supports Fortran, C/C++ and Python source code	-
Availability	Complete source code and documentation freely available at PNNL website.	-
Resilience	-	-

Table 23 Global Arrays Toolkit - Pros and Cons

2.12.5 *Target systems/architectures*

X86	IBM	GPU	MIC	ARM	Heterogeneous
Yes	Yes	No	No	Not sure	Yes

Table 24 Global Arrays Toolkit - Target systems/architectures

2.12.6 *Conclusion*

For distributed-memory systems the Global Arrays Toolkit demonstrates at least as good scalability and performance as plain MPI. Its added benefit is that it adds a simplified programming model and also been shown to solve memory bottleneck issues on large-scale systems (e.g. IBM Blue Gene). If however future exascale systems are built on top of coprocessors/acceletors, Global Arrays Toolkit may be less beneficial in its current state.

3 Debuggers and Profilers

A wide range of tools for debugging and performance analysis exists. A survey of system software stacks in the IESP [98] community lists approximately forty different tools for debugging and performance analyses. This survey is by no means exhaustive, as there are many additional tools both from vendors and Open Source initiatives that offer improved capabilities either for debugging or performance improvement.

The tools covered in this report are either on the IESP list or they have been covered by PRACE on an earlier occasion, especially in PRACE-PP as described in deliverable, D6.3.1. 'Report on available Performance Analysis and Benchmark Tools, Representative Benchmark' [99]. Several tools described in D6.3.1 (DewizPat, Allinea OPT, IBM VPA) are no longer available or do not exist as distinct products; some have been replaced by other products, others are merged into existing ones. The trend of merging of tools continues both in Open Source projects and commercial products. It is more and more challenging to maintain these tools due to the increasing complexity of the runtime environments.

The increasing complexity of the runtime environment is caused by the developments in hardware. Most current HPC-systems have CPUs with many cores and several CPUs on each node. This has led to hybrid codes where MPI is used for inter-node communication and OpenMP is used for intra-node communication and parallelism. A tool only addressing intra-node parallelism will not catch the whole picture; likewise will a tool only focusing on internode communication not get a complete view. Tools developed at HPC research centres have tried to accommodate for this, as these tools incorporate all levels of parallelism. Tools developed by hardware vendors are often to the contrary more intra-node focused.

It is more than likely that the next generation of HPC-systems will have coprocessors or accelerators within a node, like the newly installed STAMPEDE at TACC (Texas Advanced Computing Centre) and Titan at ORNL (Oak Ridge National Laboratory). This adds a new runtime environment to the node and yet another level of complexity. Very few tools can accommodate for this development. It is a daunting task to reliably analyse an execution environment with inter-node parallelism and two separate, but co-functioning intra-node threaded runtime environments. While the hardware vendors offer tools for analysing and debugging kernels executing on the coprocessor, it is in most cases the HPC research centres that develop tools, which can give a holistic view of the hierarchy of runtime environments.

All of the European exascale projects (TEXT, CRESTA, DEEP, Mont Blanc) are putting effort into tools for debugging and performance analyses. This is deemed a necessity for efficient use of upcoming exascale architectures. Ayudame/Temanejo has been developed in TEXT. Vampir is further developed as part of CRESTA. Support for OmpSs/ Intel Xeon Phi and OmpSs/ARM in Scalasca is developed as part of DEEP and Mont Blanc, respectively. In addition there is the Holistic Performance System Analysis project (HOPSA) [100]. The product of this project is the SCORE-P [101] measurement structure. This structure is shared by the tools Scalasca, Vampir, Tau and Periscope. A common measurement structure benefits both profiling tools developers and users. The developers can concentrate more of their resources on the analysis part of the specific tool instead of the measurement framework, while the users will experience increased operability between the tools as well as fewer learning curves.

We feel that the tools in the SCORE-P project are among the most natural to use in the exploitation phase, together with tools for analysing coprocessors and accelerators.

DEEP has analysed the space weather application iPIC3D. With the tool Scalasca, the behaviour of the different parts of the application has been identified. Some of these parts can

be accelerated by being partly moved to the "Booster" part of the DEEP architecture [102]. During the exploitation phase T7.2 should strive to establish similar insight as often as possible.

It is worth noting that this type of analysis has rarely been seen in PRACE reports or whitepapers to date. A substantial effort of training on tools for debugging and performance analysis has been carried out within PRACE. However, very little is documented on how successfully these tools have been employed within enabling projects. One of our missions for the exploitation phase is to fix this discrepancy and provide good examples of debugging and performance analyses with the different tools chosen.

In this section we report on the following debugging and profiling tools that we feel represent the state of the art and should be of interest during the exploitation phase of T7.2:

Tool	Category	Scalability	Sustainability	Portability	Availability	# of archs
Tau	Scalable OSS	131000	NSF, DOE, DOD, research centers	Large range of systems	Open source	5
Scalasca	Scalable OSS	274912	FZ Jülich, user groups	IBM BG, Cray XT, Linux	Open source	2
Vampir	Scalable OSS	200448	Dresden University, LLNL	Large range of systems	Open source	5
TotalView	Scalable ISV	786432	Wide use, vendor support	Linux, AIX, Solaris	Commercial	5
Allinea DDT	Scalable ISV	200000	ORNL, vendor support	Linux	Commercial	5
Intel tools	Workstation		Vendor support	x86 systems	Commercial	1
NVIDIA NSight	Workstation		Vendor support	CUDA systems	Free	3
IBM HPCT	Vendor tools		Vendor support	IBM systems	Commercial	
CrayPat/Apprentice2	Vendor tools		Vendor support	Cray systems	Commercial	
Paraver/Extrae	Other			Linux, GPU, Xeon Phi		
IPM	Other			Linux, GPU		
OpenSpeedshop	Other			Large range of systems		
PAPI	Other		Linux kernel	Linux	Open source	
Temanejo/Ayudame	Other		Mont Blanc project	StarSS/OmpSS programs	Open source	

Table 25 Debugging and Profiling Tools
(OSS: Open Source Software, ISV: Independent Software Vendor, archs: Hardware Architectures)

3.1 TAU

3.1.1 *Brief overview*

The development of Tuning and Analysis Utilities (TAU) [103] started in the nineties. The ambition for TAU has, from the very start, been to grow with the developments in hardware, making a tool that survived several hardware generations. Consequently, there is continuous on-going work to accommodate for the changes in large-scale system architectures [104].

TAU has developed a very rich set of features that can be utilized at varying degrees of invasiveness both for profiling and trace logging. Three methods for instrumentation are available to a user: library preloading, complier directives and source code transformation. Each method offers an increase in features at the expense of binary or source code modification [105]. Binary modification is done with recompilation through TAU supplied scripts. Source code is also modified with supplied scripts.

TAU can be used on applications developed with the programming languages C, C++, Fortran, Java and Python. The utilities offered are performance monitoring, performance data mining, parallel profile analysis and program analysis. All these utilities have different interfaces such that the analyses can be done with a graphical user interface or by scripts. Data mining can be done with R, Weka or Octave/Matlab. Measured data can be stored in a DBMS through the toolkit TAUdb, formerly known as Performance Data Management Framework (PerfDMF). Collected data can be processed with other analysis tools like Vampir, Scalasca or Paraver.

TAU is very portable and it is available on a range of platforms. It has been used on all generations of both Cray XT/XE and IBM BlueGene machines. It supports the latest generations of both platforms that include the Cray Cascade prototype with Intel Sandy Bridge and IBM Blue Gene/Q [103].

Here we list some of the features that we think should be of interest to PRACE partners during the exploitation phase of T7.2:

- Modules from different programming languages can be instrumented, i.e. a Python based program using FORTRAN made object modules.
- Both source (PDT) and binary rewriting capabilities, with support for logging time spent in application routines and outer-loops.
- I/O characterizations with peak read and write bandwidth as well as total volume. Time spent in I/O-phase can be measured
- Memory usage can be instrumented with detection of peak heap memory usage. Allocation and de-allocation of memory can be tracked.
- Support for debugging capabilities with callstack, memory leak detection, and runtime bounds checking
- Performance database technology to store performance data, cross experiment and data mining tool (PerfExplorer)
- ParaProf offers 3D visual browser. 3D communication pattern or 3D topology can be viewed
- Automatic performance measurement system on BG/P

TAU has been extended to support heterogeneous platforms [106]. It supports the Intel Xeon Phi in the native, offload and the host mode. This includes support for event based sampling, and the commonly used instrumentation techniques (PDT, MPI, and linker).

Measurements can be conducted both with profiling and trace logging of kernel invocations on an accelerator. Both PAPI [107] and CUPTI [108] can be utilized. CUPTI can be used with

library preloading. Calls to OpenCL can also be intercepted in this way. Accelerator code generation capabilities with PGI or HMPP are also supported.

As of version 2.21.2 TAU can track the memory usage for the lifetime of CUDA kernels (CUDA v4.1 and later). This involves the whole memory hierarchy of registers and both local and shared memory can be tracked.

TAU can also be combined with the auto-tuning framework Orio [109][110]. Orio can empirically execute thousands of accelerator kernels over a large set of different parameters. By instrumenting the kernels, TAU's data mining capabilities can be used for finding the most efficient set of parameters for a given kernel. An exploration of acceleration kernels carried out with Orio using PETSc has shown a 1.5 - 2x performance improvement for certain methods compared to implemented libraries like CUBLAS [111].

Latest version/release: v2.22.1.

3.1.2 *Evidence of use within PRACE*

Information on experience with TAU is generally quite scarce within PRACE. However, the survey carried out in PRACE-PP [99] shows that at the time TAU was installed on three PRACE partner sites. This number has undoubtedly increased significantly in the intervening period as TAU has become increasingly recognized as a tool that scales well and offers a large amount of versatility and functionality. In WP7 PRACE-1IP, TAU was reported on in the PRACE whitepaper, 'Optimizing GPAW' [112] where profiling and tracing on up to 1024 cores was carried out using TAU. Also, in WP7 PRACE-1IP, TAU was reported on in the PRACE whitepaper, 'High Resolution EC Earth porting and benchmarking on Curie' [113] where EC-EARTH was profiled on 394 cores using the TAU tool. In both cases, not a lot of detail was provided on how successfully TAU was employed.

3.1.3 *Evidence of use outside PRACE*

Just as is the case within PRACE we found it quite difficult to find reported examples of where TAU has been used to date with real applications on large-scale systems. TAU was used in a recent performance profiling of the S3D combustion modelling code on Intrepid (IBM BG/P) in the US. Data was collected for S3D on up to 12,000 cores using the C2H4 benchmark where the goal was to evaluate the scaling properties of code regions and the scalability of MPI operations with the S3D code. The full power of the TAU profiler is well demonstrated in the work where the bimodal behaviour in `MPI_Wait` calls with the code were discovered as well as other metrics of interest such as memory usage, flop/s and issues with PAPI counters on BG/P [114].

3.1.4 *Pros and Cons*

Metric	Pros	Cons
Scalability	TAU has been used to conduct a full profile of the application Pflotran executed on 131 000 cores [115].	
Performance	TAU has a throttling mechanism for managing the output stream. One can also choose a lightweight	

	management core which reduces the overhead by 50% compared to TAU's default core. Reducing overhead is a target: the MPI wrapper interposition library is now using a more efficient data structure for tracking asynchronous communication events.	
Productivity	Profiling can be done without any binary or source code modifications. As increased level of detail is needed, binaries or source code is modified with supplied scripts. Profiling level is controlled by the user in separate files. Different utilities may have varying degrees of threshold before they can be used usefully.	TAU is in essence a collection of tools more than a tool in itself. Experience is needed to gain results with the utilities available. It can produce a large volume of profiling and trace logging data.
Sustainability	The development of TAU has been funded by major US funding bodies (NSF, DOE, DOD). The project has a range of US and Europe research centres as partners.	
Correctness		
Portability	It is very portable and has been continuously ported to new platforms for more than twenty years. Can be used on Fortran, C and C++ applications	
Availability	Open Source BSD license	
Resilience		

Table 26 TAU - Pros and Cons

3.1.5 *Target systems/architectures*

X86	IBM	GPU	MIC	ARM	Heterogeneous
Yes	Yes	Yes	Yes	Yes	Yes

Table 27 TAU - Target systems/Architectures

3.1.6 *Conclusion*

TAU is a tool that has evolved for use on multi-petascale systems. It has been used on large core counts and has demonstrated an on-line analysis capability on large core counts. The visual tool ParaProf support 3D-views of torus and communication patterns for the major multi-petascale platforms. The TAU development team usually ensures that TAU is available at early stages of new platforms, as exemplified with the Cray Cascade prototype. TAU is being further developed to support new threading technologies, like the new generations of NVIDIA GPUs and Intel Xeon Phi. With its long traction, and continued support from its funding bodies, it is strongly expected that TAU will continue to be available for forthcoming multi-petascale systems. In this sense, it also shows potential for enabling applications on future exascale systems.

3.2 Scalasca

3.2.1 *Brief overview*

Scalasca [116] is, as the name suggests, a profiling tool for scalable performance analysis. It is specifically designed for large-scale high performance computing systems. Scalasca utilizes the inherent parallelism in HPC systems by letting each process maintain a trace log of logged events. The trace logs are written to a few files in a parallel file system. The analyses are done post-mortem, but with the same amount of processes as the instrumented simulation. This provides a good analysis capacity and ability to handle a large volume of trace logs. Scalasca is targeted at MPI applications written in C/C++ or Fortran. It uses PMPI, the profiling API of the MPI standard to profile MPI applications. OpenMP-based code can be analysed with a source pre-processor which instruments parallel regions in the application source code. It is also possible to profile hybrid codes.

Scalasca's distinctive features are:

- A summary-report, which identifies the most time-consuming call paths.
- Performance evaluation over the whole length of a simulation.
- Delay and wait state analysis. Wait states are often caused by load imbalance and cause delay. What causes wait stats and consequently delays can be identified [117].

It is available on Cray XT, IBM BlueGene and common Linux versions. Scalasca uses the format Open Trace Format 2 (OTF-2), which is a common trace log format also used by Vampir and Tau.

Latest version/release: v1.4.2.

3.2.2 *Evidence of use within PRACE*

At the time of writing the PRACE-1IP survey (D7.4.1) [118] reported that Scalasca was installed on six PRACE partner systems. As is the case for TAU, this number has most likely risen in the intervening period as Scalasca becomes recognized as a highly scalable tool with an easy to understand interface. Scalasca has been used in several PRACE projects including PRACE 'Type-C Preparatory Access projects' as well as petascaling community codes in PRACE-1IP and 2IP.

The project PRACE Preparatory Access Type C project "Shocks: Understanding Relativistic Plasmas Acceleration Systems", which was reported in the PRACE deliverable 'Applications Enabling for Capability Science' [119] used Scalasca with the PSC code on CURIE to

successfully highlight global communication hotspots, which were subsequently reduced leading to an overall performance improvement of 14.9%.

The project described in the PRACE whitepaper "Direct Numerical Simulation and Turbulence Modelling for Fluid Structure Interaction in Aerodynamics" [120] used Scalasca on CURIE and JADE to study three different load-balancing strategies for the application "Navier-Stokes Multi-Block" (NSMB). Scalasca was used by the authors to provide insight into the behaviour of the application as different strategies were investigated.

The PRACE whitepaper, "Semi-dilute polymer systems in shear flow - a particle based hydrodynamic approach" [121] reports on the use of Scalasca on the application MP2C on the systems JUGENE and JUROPA. The goal of the project was to reduce communication overhead on large core counts as scaling was deteriorating beyond 262,000 cores. Better scaling for lower core counts was achieved by hybridizing the code with OpenMP. Scalasca was successfully used to identify the proportion of communication vs. computation within the code.

3.2.3 *Evidence of use outside PRACE*

Scalasca is in use at several larger HPC-labs, including FZJ. The IESP survey [98] hows that it is installed in two sites in the US and 12 in Europe. In XSEDE it is available on KRAKEN, NCSA FORGE and BLACKLIGHT.

A trace analysis of a case executing on 294,912 processes on Blue Gene/Q is documented [122]. This was carried out with Scalasca 1.2 applied on the code Sweep3D. The analysis collected 790 GB of trace logs. To run the analysis with a rerun of the simulation took more than two hours. The instrumented Sweep3D application, which generated the trace logs, had an execution time of fifty minutes in comparison to the execution time for the un-instrumented application of ten minutes. This illustrates to some extent a contradiction with trace analysis of applications at scale. While the tool allow tracing of an application at large scale with real data sets running for hours, the time consumption and data volume produced can become impracticably large for a user to handle, however, this issue is currently being actively addressed by the Scalasca team [123].

In the course of the European exascale DEEP project, Scalasca was used to analyse the 6 prototype applications on a regular cluster to help to decide which kernels will be offloaded to the Booster Nodes. Further, Scalasca will be ported to the Intel Xeon Phi platform and, once available, to the DEEP System. In that way it can be used to monitor the application performance at all stages of the project [124].

3.2.4 *Pros and Cons*

Metric	**Pros**	**Cons**
Scalability	Excellent: scales to at least 300,000 cores	The volume of trace logs become large with a large number of processes
Performance	The tool has good performance as the analyses are done in parallel.	
Productivity	The tool is developed with large-scale simulations in mind. It makes use of the	Analyses are based on processing trace logs. Trace logs of large scale long

	system resources available on a HPC system for logging, storage and analyses.	running real applications can get so large that they influence node behaviour and the simulation under observation.
Sustainability	It is maintained by Forschüngszentrum Jülich and it is continuously improved with input from a broad user group both in Europe and USA.	
Correctness		
Portability	It is developed for use on IBM Blue Gene and Cray XT. It is reported used on a range of other platforms, especially medium sized HPC Linux clusters. Can be used on Fotran, C/C++ applications	
Availability	The software is available for free download under the BSD open-source.	
Resilience		

Table 28 Scalasca - Pros and Cons

3.2.5 *Target systems/architectures*

X86	IBM	GPU	MIC	ARM	Heterogeneous
Yes	Yes	Under development	Under development	Under development	Under developement

Table 29 Scalasca - Target systems/architectures

3.2.6 *Conclusion*

Scalasca has shown that it is an applicable profiling tool when considering the largest scales currently possible. It scales in its use from 1000 cores to close to 300,000 cores. Development of Scalasca continues with the purpose of meeting the needs of the HPC community as exascale technology make its inroads. Although the Scalasca project will specifically target large-scale parallel applications, the project recognizes that it faces several challenges, including the fact that collation time increases with increasing number of threads, the time needed for analysis is proportional to the number of threads and the behaviour of the application will become more dynamic and unpredictable as processes and threads are created and destroyed. One interesting initiative by the Scalasca team to note that may be of interest during the exploitation phase of T7.2 is the capability being introduced to profile one-sided communications [125].

3.3 Vampir

3.3.1 *Brief overview*

The Vampir tool-set is a set of programs to trace high performance computing workloads, and present the captured information in a graphical interface. It consists of an instrumentation component called VampirTrace [126], and two visualisation applications called Vampir [127] and VampirServer [127]. The former is freely available, while the latter two are commercially licensed from TU Dresden.

Vampir is aimed at highly parallel computing platforms, which manifests itself in three particular design objectives [128]. Exploiting distributed memory for analysis tasks, real-time processing of long-running workloads with high degrees of parallelism, limiting data processing at the client end to a volume which is independent of the amount of captured event trace data.

The software architecture of the Vampir tools separate the work of capturing traces of program event information and the work of visualizing it.

VampirTrace captures event logs in Open Trace Format (OTF), developed in a collaboration between TU Dresden, University of Oregon, and Lawrence Livermore National Labs. It is a successor to the previously used Vampir Trace Format (VTF). The visualization tools are also compatible with trace files captured by performance monitoring with Tau or KOJAK [129].

The visualization components of Vampir focus on presenting captured data as time line displays of events from large numbers of participating processes, with global, summary, counter and process time lines. In order to capture the event information required to produce these visualizations, VampirTrace caters to four methods of instrumentation, from compiler instrumentation, through source-to-source instrumentation, library instrumentation, and manual source instrumentation.

The scalability of the Vampir approach is connected to the management of the large volumes of data produced by these methods when applied on large-scale systems. The primary mechanisms for restricting event data is the capability of grouping and filtering functions according to their names or categories, allowing subsequent analysis to relate to aggregate statistics, or cross-sections of the available data with respect to a particular library. The categorization is configurable, but includes MPI calls, OpenMP calls and I/O functions by default, as these are tightly connected to the available methods for instrumentation. Exclusion lists for filtering are configurable by providing ignored functions in text format at the time of execution.

VampirTrace is capable of tracing GPU accelerated applications and generates exact time stamps for all GPU related events

Latest release/version: Vampir v8.0, VampirTrace v5.14.3

3.3.2 *Evidence of use within PRACE*

While Vampir is frequently presented at PRACE-training sessions and is also often mentioned in PRACE deliverables, there are very few cases where Vampir is reported as being used. In PRACE-1IP deliverable D6.3.1,'Report on available Performance Analysis and Benchmark Tools, Representative Benchmark' [99], both a LINPACK benchmark on 64 MPI processes the GYRE benchmark of NEMO were profiled with Vampir on HECTOR, but unfortunately very few details of how successfully Vampir was used are provided in the deliverable.

3.3.3 *Evidence of use outside PRACE*

The Survey of System Software Stacks in the IESP Community [98] presented at the IESP workshop in 2011 counts 7 installations of Vampir and 9 installations of VampirTrace, of which were 5 in Europe.

A prototype of Vampir using the I/O Forward Scalability Layer (IOFSL) library [130] has recently been used to trace over 200,000 processes on Titan[131].Without the IOFSL library, tracing produces too many files which prohibits scaling beyond 8000 processes.

The new exascale prototypes of the TU Dresden Vampir and VampirTrace performance monitoring and analysis tools have been released. The new features include the possibility of applying filtering techniques before loading performance data to drastically reduce memory needs during the performance analysis. The initial evaluation study of the development environment is targeted at the European CRESTA project applications to determine how the development environment could be coupled into a production suite for exascale computing [132].

3.3.4 *Pros and Cons*

Metric	**Pros**	**Cons**
Scalability	Distributed software architecture permits large-scale operation.	Dependency on availability of data distributed across entire platform affects possibility for offline/post-mortem analysis
Performance	Low overhead per-event facilitates real-time tracking	Buffering technique depends on surplus memory
Productivity	Black-box instrumentation available through automatic instrumentation	Large-scale workloads likely to require manual instrumentation to customize analysis for buffering constraints
Sustainability	Actively developed as Open Source and commercial projects	
Correctness		
Portability	It is ported to range of platforms: Linux (IA32, x86_64, IA64, PPC/32, PPC/64), Sun Solaris (SPARC/32, SPARC/64, x86_64), IBM AIX (PPC), SGI IRIX (MIPS), Mac OS X. Can be used on Fortran, C/C++ applications	
Availability	Instrumentation tool is open source and freely available	Visualization tools require commercial licenses
Resilience		

Table 30 Vampir - Pros and Cons

3.3.5 *Target systems/architectures*

X86	IBM	GPU	MIC	ARM	Heterogeneous
Yes	Yes	Yes. (Support for CUPTI under development)	Compatible at source level using OpenMP POMP interface	Not sure	Yes

Table 31 Vampir - Target systems/architectures

3.3.6 *Conclusion*

Vampir addresses the challenge of petascale profiling with hierarchal parallelism. Profiling tools addressing threads are often developed by vendors offering the threading technology. These do not show the total call stack and messaging that goes on in an MPI-application utilizing accelerator/coprocessors. It is the intent of the developers that VampirTrace will accommodate for threading technologies and address the hierarchy of parallelism in newer MPI-applications like MPI-applications with OpenACC enabling. While evidence of multi-petascale use of Vampir is hard to come by, at least in a PRACE context, the tool will be further developed. In the CRESTA-project, ZIH and other partners will jointly develop the scalable measurement environment used by Vampir as an Open Source project. Current research activities for performance analysis focus on pattern processing, GPGPU computing, scalability, and energy aware performance optimization. Hence, it is likely that it will feature as a tool on the road to exascale.

3.4 TotalView

3.4.1 *Brief overview*

Rogue Wave's TotalView [133] is a debugging tool for parallel computing. It supports scalar, multi-threaded and large-scale parallel applications. It supports debugging of memory errors, leaks and diagnosis of programs like deadlocks and race conditions. It contains a Replay Engine that allows stepping backwards through programs from the point where it crashed to the point where the problems started. TotalView supports the programming models: MPI, OpenMP, hybrid multi-threaded codes, CUDA, OpenACC, C, C++, Fortran. TotalView supports the platforms: Linux (including Blue Gene), Unix and Mac OS X. TotalView has specifically been designed for debugging on large-scale systems and Rogue Wave are currently working as co-design partners at various US DOE labs including with Livermore National Lab (LLNL) Sequoia team.

Using the RealplayEngine feature [133], TotalView can be made to store the history of a running program, so that it becomes possible to step backwards through a program from the point where a problem is detected to the point where the cause of the problem is. This can be important when searching for bugs that are difficult to reproduce, but it is highly unlikely that this feature will ever work on peta- or exascale systems, because of the amount of data that has to be stored for each process. TotalView also has some very nice batch scripting features designed for debugging in a batch environment, which allows users to define the events to act on and the actions to take when an event occurs.

3.4.2 *Evidence of use within PRACE*

TotalView is widely used within PRACE. It is for instance used at CINECA, CSC, EPCC, IDRIS, Jülich, LRZ, NTNU, and SARA. Unfortunately, we found it very difficult to find more information about how the debugger was used in practice on real applications on large-scale PRACE systems.

3.4.3 *Evidence of use outside PRACE*

Although it is widely appreciated that TotalView scales to many thousands of cores, we found it difficult to find concrete documentation on real cases of where the debugger has been used on real applications on multi-petascale machines. The Rogue Wave developers are involved in co-design partnerships with Lawrence Livermore National Lab (LLNL) who manage the million plus core IBM BG/Q Sequoia machine. Recently a hybrid MPI/OpenMP Jacobi benchmark code was debugged over 512 - 65,000 compute nodes (on up to 121,072 CPU cores) on the Sequoia system using TotalView [134]. In a separate announcement at SC12, Rogue Wave claim that TotalView has successfully scaled across 786,432 cores [135] on the Sequoia machine, but little detail can be found on which application it was used.

3.4.4 *Pros and Cons*

Metric	**Pros**	**Cons**
Scalability	Has debugged jobs as large as 786,432 processes. Petascale capability is, most likely, available today.	
Performance	Normal debugging has a low impact on application at runtime.	Reverse Debugging has a high negative impact.
Productivity	Has a built-in memory debugger. Supports a wide variety of programming models.	Reverse Debugging only works well with few processes.
Sustainability	Widely used and regularly updated. Commercial product with strong support	-
Correctness	-	-
Portability	Supports Linux, AIX, and SPARC Solaris. Can be used to debug Fortran, C/C++ code. Can also be used on PGAS languages, as well as accelerator-based applications	-
Availability	-	Commercial
Resilience	-	-

Table 32 TotalView - Pros and Cons

3.4.5 *Target systems/architectures*

X86	IBM	GPU	MIC	ARM	Heterogeneous
Yes	Yes	Yes	Yes	No	Yes

Table 33 TotalView - Target systems/architectures

3.4.6 *Conclusion*

TotalView is a professional debugging tool that specifically is aimed at the High Performance Computing market. It is designed for debugging programs running on very large supercomputers and has been successfully tested on 768,432 processes until now. TotalView state that they are working closely together with IBM to provide debugging facilities on IBM's Blue Gene systems, so petascale debugging is therefore, most likely, available today. However, the RealplayEngine is clearly aimed at small systems, for which it is practically possible to record the history for each process.

3.5 DDT

3.5.1 *Brief overview*

DDT [136] is a commercial debugger produced by Allinea Software, primarily for debugging parallel MPI or OpenMP programs, including those running on clusters of Linux machines, but also used by many for scalar code in C, C++ and Fortran 90. Allinea claims that it was the first debugger to be able to debug a petascale system - having debugged 220,000 processes, over 2 Petaflops, on a Cray XT5 at Oak Ridge National Laboratories [137]. The debugger has logarithmic performance for most collective debugging operations, due to using a tree architecture across the machine network to control the many single-process debuggers. It features a complete memory debugging tool, which can be used to detect memory leaks, or reading and writing beyond the bounds of arrays. Allinea DDT includes support for Intel Xeon Phi coprocessors and IBM Blue Gene. The debugger is also able to debug GPU software written for CUDA applications. Allinea has applied a co-design methodology, working closely with compiler vendors that support the languages for machines such as Titan and Blue Waters [138][139].

3.5.2 *Evidence of use within PRACE*

The use of Allinea DDT appears to be increasing within PRACE. According to the PRACE-1IP deliverable, D7.4.1 [118], from 2011 DDT was in use at 6 PRACE partners in 2011. However, at the time of writing, DDT appears to be in use at, at least, 10 locations: BSC, CEA, CINECA, EPCC, GENCI, IDRIS, ICHEC, Jülich Supercomputing Centre, NTNU, and PDC. Unfortunately, as is the case with most of the debugging and profiling tools used within PRACE to date, very little information is documented as to the successful use of DDT during development.

3.5.3 *Evidence of use outside PRACE*

By employing sophisticated tree topologies, Allinea, working along with ORNL, deployed the first petascale-level debugger, DDT, for Jaguar. Field-tested on development codes at ORNL, DDT has been shown to scale up to over 200,000 cores [137].

Within the European exascale project, CRESTA, the Allinea DDT debugger is being extended to provide a unified interface, to improve scalability, and to include a new disruptive technology based on statistical analysis of run-time behaviour of the application for anomalies detection [132].

3.5.4 *Pros and Cons*

Metric	Pros	Cons
Scalability	Delivers petascale debugging today. Has debugged jobs as large as 200,000 processes.	Exascale debugging is future work.
Performance	Logarithmic performance for most collective debugging operations.	-
Productivity	Has a built-in memory debugger. Supports a wide variety of programming models.	-
Sustainability	Widely used and regularly updated. Commercially supported.	-
Correctness	-	-
Portability	Supports Linux and has been demonstrated to work on a wide variety of large-scale platforms. Can be used to debug Fortran, C/C++, PGAS and accelerator-based applications	-
Availability		Commercial
Resilience	Part of on-going co-design research with exascale projects such as CRESTA.	

Table 34 DDT - Pros and Cons

3.5.5 *Target systems/architectures*

X86	IBM	GPU	MIC	ARM	Heterogeneous
Yes	Yes	Yes	Yes	Yes	Yes

Table 35 DDT - Target systems/architectures

3.5.6 *Conclusion*

DDT now has features that are specifically targeting debugging of petascale simulations. The response times of DDT are now short enough for making petascale debugging practically possible and the GUI has features that are specifically designed for giving an overview of

large amounts of data as well as the state of a large numbers of threads/processes. The fact that the developers of DTT have continued to show a quick response to the fast pace of changing hardware on large-scale heterogeneous systems, indicates that DDT will feature heavily as a debugging tool on the road to exascale. Although DDT's current feature-set will not be sufficient on an exascale system, DDT is actively working on new features such as fault-tolerance as part of co-design teams within the US DOE and European exascale projects.

3.6 Intel Debugging and Profiling Tools

3.6.1 *Brief overview*

In this subsection we provide an overview of debugging and profiling tools provided by Intel. In particular the tools of the Intel Cluster Studio XE 2013 [140] are discussed. With regard to multi-petascale (and future exascale systems), tools for debugging and profiling on the new Intel Xeon Phi coprocessors are of particular interest. As is to be expected, Intel tools already support Intel Xeon Phi (with some minor exceptions) [140]. However, since by design, the Intel Xeon Phi coprocessor more or less emulates a normal x86-based cluster node, porting other tools to run natively on Intel Xeon Phi is technically quite straight-forward. Moreover, this has already successfully been demonstrated in the case of Scalasca and Paraver, TotalView and DDT [141].

3.6.2 *Evidence of use within PRACE*

Currently no concrete applications of Intel tools within PRACE are documented. However this does not necessarily mean that they have not been used at all within PRACE. At least according to PRACE-1IP deliverable, D7.4.1 [118], it is known that Intel tools are installed on at least two PRACE systems.

3.6.3 *Evidence of use outside PRACE*

Unfortunately, we found it very difficult to find documental evidence of how Intel tools are being used on petascale systems. Intel tools have typically proven to be useful on smaller scales. For example they were successfully applied for optimizing a crash test simulation code running on 128 nodes with 1024 cores [142].

Intel Trace Analyzer and Collector (ITAC):

Intel Trace Analyzer and Collector (ITAC) is a pair of tools for profiling and correctness checking of MPI applications. The current product version is v8.1 and is included in the Cluster Studio product package. The Intel Trace Collector detects up to 50 different runtime errors (e.g. deadlocks, data corruption and errors with MPI parameters, data types, buffers, communicators, point-to-point messages and collective operations) and provides a full stack trace for those errors. Application trace data can be collected by, either relinking the application with the Intel Trace Collector Profiling library, or usage of special shared libraries (when using a dynamically linked MPI library.

The Intel Trace Analyzer is a customizable GUI for viewing generated trace files and provides different analysis views (‘event timeline’, ‘communication matrix’, ‘function profile’, etc.).

With ITAC it is neither sensible nor possible to show all events of a trace file on the timeline. Sampling resolution depends on the current timeline view range. Events can be aggregated by function groups (the coarsest level distinguishes between MPI and application functions) and

process groups (individual processes or grouped by node – further levels of hierarchies will be possible in future). Not all events can be displayed. However, interesting events can be tagged. In the timeline the regions containing tagged events are highlighted so that the user knows where to zoom. Trace data can be filtered by certain functions, messages and/or collective operations.

Intel VTune:

Intel VTune Amplifier [143] is a profiling tool where analyses are based either on event-based sampling or user-mode sampling. Event-based sampling collects system-wide profiling data through the 'Performance Monitoring Unit '(PMU) of Intel processors. User-mode sampling uses the software collector for gathering profiling data and is also available on Intel-compatible processors. In both cases, no recompilation of the application is necessary, but including debugging symbols is recommended. VTune Amplifier can be used to analyze intra-process performance of MPI applications. Typically, only a selected representative small subset of MPI processes (ranks) are run under control of the VTune Amplifier.

Hardware profiling is supported for the new Intel Xeon Phi and can be launched from the graphical user interface. It can collect lightweight hotspots and advanced event data and has time markers for correlation of data across multiple cards. Software collection (e.g., locks and waits analysis) is not supported on the Intel Xeon Phi. More details on MIC support for VTune Amplifier can be found in [144].

Intel Inspector:

Intel Inspector [145] is a dynamic memory and threading error checking tool for serial and multithreaded applications. It can also be used to visualize and manage static analysis results created by Intel compilers. Intel Inspector provides a GUI as well as a command line interface for build process integration and for tracing MPI applications.

3.6.4 *Pros and Cons*

Metric	**Pros**	**Cons**
Scalability	An application (128 nodes, 1024 cores) is known where Intel tools have successfully been used for optimization.	*ITAC:* Trace data has to be collected to one host for local analysis. *VTune, Inspector:* Only for intra node profiling; Dynamically spawned MPI processes currently not supported.
Performance	*VTune:* Little overhead (~2% for event-based sampling at 1ms, ~5% for user-mode sampling at 10ms). *Inspector:* Overhead ranges between 2 and 320 times depending on use-case and analysis depth and is well-adjustable.	

Productivity	Generally good usability and strong and helpful concepts for analysis of large applications.	
Sustainability	Currently there is no indication, that Intel will stop developing and supporting Intel Cluster Studio in future.	
Correctness	Intel tools are generally well tested. Issues and known limitations are well documented in the release notes.	
Portability	*VTune:* Can also be used on Intel-compatible processors (but only user-mode sampling available) *Inspector:* No particular constraints (available for Windows and Linux platforms).	*ITAC:* Trace collector runs only on Intel processor architectures
Availability	Commercial products.	
Resilience	*ITAC:* A failsafe version of the Trace Collector library is available, allowing to trace also failing MPI applications.	

Table 36 Intel Debugging and Profiling Tools - Pros and Cons

3.6.5 *Target systems/architectures*

X86	IBM	GPU	MIC	ARM	Heterogeneous
Yes	No	No	Yes	No	Yes

Table 37 Intel Debugging and Profiling Tools - Target systems/architecture

3.6.6 *Conclusion*

ITAC is a good tool for analysis of MPI applications running on clusters equipped with Intel-processors. Additionally to collecting and viewing trace data, the user can also greatly benefit from the MPI correctness checking features offered. Several HPC facilities and applications at smaller scales (not yet petascale, but some of them at least near to petascale) are known, where Trace Analyzer and Collector (and Intel tools in general) have been used and where they have proven to be beneficial. Currently, the fact that trace files need to be collected to the one host system for analysis is a limiting factor for scalability. Intel already has ideas to address this problem, but at the current stage they are very careful about making any disclosures. Also other ideas (like recognition for certain communication patterns) are under discussion for future program versions.

Intel VTune Amplifier is a mature tool for profiling applications on Intel (or Intel-compatible) architectures. Because Intel VTune was originally designed for SMP machines, it is an important challenge for Intel to further improve the tool's support for MPI application profiling. Recent developments have already addressed this issue and further improvements are forthcoming [147]. As well as optimizing the MPI-level of an HPC application, intra-process optimization is important for achieving best possible performance. Since this is also true for future multi-petascale and exascale systems, VTune Amplifier will also be of relevance to HPC applications at that scale.

Intel Inspector is a good tool for finding threading and memory errors on the intra-node level, which might otherwise be hard to find. Static code analysis can reveal bugs, which potentially can stay dormant for a long time until showing up the first time in a production run. Also dynamic code analysis can reveal more errors than those, which actually occurred during the test run (e.g. potential data races can be detected even if the test execution went well). As Intel is continuously developing their tools further, there is good reason for assuming that Intel Inspector and ITAC (as well as successors) will be relevant for future multi-petascale and exascale systems. Today, Intel Inspector in conjunction with ITAC has already shown potential for HPC applications on smaller scales [148].

3.7 NViDIA NSight

3.7.1 *Brief overview*

NVIDIA Nsight Eclipse Edition [149] is a full-featured IDE that provides an all-in-one integrated environment to edit, build, debug and profile CUDA-C applications. As it is based on Eclipse it supports standard IDE features, along with CUDA aware code completion and refactoring, and project templates. It supports simultaneous debugging of both CPU and GPU code and inspection of variables across CUDA threads. The profiler supports automated analysis of system optimization opportunities, highlights potential performance problems, and integrates the "nvprof" command-line profiler to enable visualization of profile data collected from headless compute nodes. As a development environment it is primarily aimed at x86 architecture workstations with CUDA GPUs.

The primary intent of this tool is to support development of CUDA applications in the code implementation, debugging and optimization phases, and it's meant to be used on developer's workstation with one or more CUDA cards. It has no use case for running on multiple compute nodes. However, "nvprof" can be used to collect profile information from compute nodes, and its profile outputs can be visualized by Nsight.

The Visual Studio plugin edition is at version 3.0 and Eclipse edition is version 5.0 at the time of writing. Both work with the CUDA 5.0 Toolkit, and Eclipse edition comes bundled in it by default. This tool was previously known as NVIDIA Parallel Nsight and it was available only on Windows platforms.

The profiling component can now collect events and metrics for all CUDA contexts in a multi-context application. In previous releases, the profiling component and the command-line profiler could collect events and metrics for only a single context per application. The latest version also allows profiling of concurrent kernel executions, which helps to examine issues regarding computation and communication overlapping, which is significant for achieving higher performance.

Latest release/version: v5.0 (Eclipse edition)

3.7.2 *Evidence of use within PRACE*

In PRACE-1IP Deliverable 9.2.2 'Final Software Evaluation Report' [150]. Nsight was used on an isolated Windows 7 node on the GPU cluster at PSNC. An older version of the Visual Studio plugin edition was used, and some limitations regarding kernel debugging were pointed out. The tool was used to test CUDA and OpenCL sections of code in NAMD, a parallel molecular dynamics code, which performs simulations of large biomolecular systems.

3.7.3 *Evidence of use outside PRACE*

The tool does not have features that aim at cluster deployment. It is relatively newly ported to platforms other than Windows, so its deployment is relatively limited. It is a tool targeted at workstations, which helps developers organize their projects, develop code, debug and profile so its use on multi-petascale systems is only indirect. 'ScalaLife' is a European FP7 project focused on software for life sciences, which collaborates with PRACE. Its Deliverable 6.1 mentions the use of Nsight in section 5.1.1 with regard to work carried out on extending ERGO for GPUs. It is interesting to note that in that work it was reported that the code was compiled using Visual Studio specifically for the reason that the Nsight profiler could be used [151].

3.7.4 *Pros and Cons*

Metric	**Pros**	**Cons**
Scalability	Can support multiple CUDA contexts, and can support profiling of overlapping computation and communication in the code.	Meant to work on single workstation. Although it is documented that parallel MPI jobs can be profiled using *nvprof* component, it is not documented that profiling information is aggregated automatically (each profile is contained in a separate file).
Performance	It introduces no significant profiling overhead.	-
Productivity	Greatly helps the developer with code completion, inline documentation, advanced code navigation, refactoring, syntax colouring, code folding, variable inspection features. Also profiling and debugging environment is fully integrated.	A limitation inherited from cuda-dbg is that it can debug kernels running on a single GPU system only if no X11 server is running.
Sustainability	NVIDIA directly supports the development of the tool.	-
Correctness	-	-
Portability	Visual Studio plugin edition runs on Windows, and	

	Eclipse edition runs on Linux and Mac OS X.	
Availability	It is freely downloadable from NVIDIA's Developer Zone site. Eclipse edition is included in Mac and Linux distributions of the CUDA Toolkit from version 5.	-
Resilience		

Table 38 NVIDIA NSight - Pros and Cons

3.7.5 *Target systems/architectures*

X86	IBM	GPU	MIC	ARM	Heterogeneous
Yes	No	Yes	No	No	Yes

Table 39 NVIDIA NSight - Target systems/architectures

3.7.6 *Conclusion*

NVIDIA's NSight covers most of the debugging and profiling tasks needed for CUDA development. It also gives great support to the developer in terms of code completion, inline documentation and other features of an advanced IDE, which makes it a good environment for development, debugging and profiling of CUDA code, and as such it can be used for applications targeted at multi-petascale and exascale systems, although for cluster-class debugging, NVIDIA points to Allinea's DDT or Rogue Wave's TotalView.

3.8 Other Tools

3.8.1 *Brief overview*

In this subsection we provide a very brief overview of other tools that we think are worth considering during the exploitation phase of T7.2. As is the case for the tools above, information about how PRACE partners have fared with the tools to date is quite sparse, but where possible we have included as much information as we could gather from the sources of input.

3.8.2 *CrayPat and Apprentice2*

The Cray Performance Analysis Tool, CrayPat [152] is the primary analysis tool on Cray supercomputers. It allows developers to perform trace experiments with function granularity. CrayPat supports programs written in Fortran, C and C++ with MPI, SHMEM, Pthreads, OpenACC and OpenMP.

Cray Apprentice2 is a GUI-based post-execution performance analysis tool that takes CrayPat performance data as input. Cray Apprentice2 provides call-graph-based profile information and timeline-based trace visualizations.

In PRACE D6.2.1 "Report on available Performance Analysis and Benchmark Tools, Representative Benchmark" [99] says about CrayPat: "It is versatile and is shown to be able

to analyze parallel software running on several thousands of CPUs. Because of this scalability, and the highly controllable degree of invasiveness, CrayPat is well suited for analysis of large parallel programs."

3.8.3 *IBM HPCT*

The IBM High Performance Computing Toolkit (HPCT) [153] is a suite of performance-related tools and libraries to assist in application tuning. This toolkit is an integrated environment for performance analysis of sequential and parallel applications using the MPI and OpenMP paradigms.

Currently, it is highly unlikely that HPCT will function properly when profiling a full petascale program. HPCT contains e.g. the tool *hpmcount*, which can provide hardware performance counters information and derived hardware metrics. This tool sends as default its output to standard output, but it is with the "-o" option possible to create a separate file for each process. However, this simple approach will most likely create a very high overhead, when profiling a petascale program running on e.g. 100,000 processes.

The HPCT has been mentioned in the PRACE D6.3.1 "Report on available Performance Analysis and Benchmark Tools, Representative Benchmark" [99]. However, it is unclear whether IBM's High Performance Computing Toolkit scales up to the number of processes that a full petascale simulation would use. There does not appear to be any on-going work on adapting IBM's High Performance Computing Toolkit to exascale systems.

Latest version/release:

3.8.4 *Paraver*

Paraver [154] is a very configurable visualization and analysis tool, which was developed to have a qualitative global perception of an application's behaviour by visual inspection. Expressive power, flexibility and the capability of efficiently handling large traces are key features addressed in the design of Paraver. Its power is based on two main pillars. Firstly, its trace format has no semantics, so extending the tool to support new performance data or new programming models requires no changes to the visualizer. Secondly, the metrics are not hardwired within the tool but are instead instrumented on the fly. To compute these, the tool offers a large set of time functions, a filter module, and a mechanism to combine two time lines. This approach allows for the displaying of a huge number of metrics. Paraver is aimed at any kind of cluster. The tool can analyse traces from very different kind of systems, including GPUs or Xeon Phi.

Extrae [155] is the package used to generate trace files, which are often subsequently analysed by Paraver. Extrae is a tool that uses different interposition mechanisms to inject probes into the target application so as to gather information regarding the application performance. The Extrae instrumentation package can instrument a wide range of parallel programming models, including MPI, OpenMP, CUDA, pthreads or OmpSs.

Paraver shows potential for getting real performance data from multi-petascale systems, as it has already been proved in large Tier-0 systems. Indeed, similar performance should be obtained in future exascale systems. These tools have been demonstrated to be very useful for performance analysis studies within the European exascale DEEP project, giving much more details about the applications behaviour than most performance tools. Within DEEP, Extrae has been ported to the Intel MIC, and both Extrae and Paraver have been used to understand the performance behaviour of the project applications [156].

Latest version/release: Paraver v4.4.0

3.8.5 *IPM*

The Integrated Performance Monitoring (IPM) tool [157] is a lightweight profiler. It is easy to use and it is not intrusive as some of its features are a small memory footprint and low CPU-usage. The low overall cost of using IPM can be illustrated by the fact that NERSC has collected more than 310K batch job performance profiles over a 6 years period by the use of IPM on their machines. IPM is open source, portable and it has been installed on several PRACE systems. Currently IPM is undergoing a redesign in the name of IPM2. The new version is enabled for OpenMP and file I/O. Support for MPI-IO, CUDA GPU and network interface counts for Infiniband hardware is under development.

IPM was used in PRACE PP on the application ECHAM5 as part of the investigation for the initial PRACE Application Benchmark Suite [99]. The profiling of ECHAM5 has the peculiar finding of showing faster computation when IPM is used compared to a computation where it is not used at low core count (2916 cores and 3600 cores). IPM has been installed on at least five sites in PRACE [99].

The ease of use and its portability make IPM an attractive starting point for any type of profiling. With the new features planned for IPM2, the tool shows potential for being applicable to applications on future multi-petascale and exascale systems [158].

Latest release/version: v0.983

3.8.6 *OpenSpeedShop*

OpenSpeedShop [159] is an open source multiplatform Linux performance tool targeted at performance analysis of applications running on both a single node and on large-scale IA64, IA32, EM64T, AMD64, IBM Power PC, Cray, and IBM Blue Gene platforms. OpenSpeedShop operates on existing application binaries, so there is no need to recompile the application being analysed. OpenSpeedShop uses both statistical sampling and traditional tracing techniques to record performance information. OpenSpeedShop is targeted at performance analysis of applications running on both a single node and on large-scale platforms [160].

According to information published by The Krell Institute, the “current version of OpenSpeedShop does not scale to the necessary level needed for BG/Q machines”. The Krell Institute is working on addressing the scalability of OpenSpeedShop [161].

OpenSpeedShop has been installed on supercomputers at a number of laboratories in USA, such ANL, LLNL, and Sandia NL. No evidence has been found for the use of OpenSpeedShop within PRACE, except in tutorials.

Latest release/version: v2.02

3.8.7 *PAPI*

PAPI provides the tool designer and application engineer with a consistent interface and methodology for use of the performance counter hardware found in most major microprocessors. PAPI enables software engineers to see, in near real time, the relation between software performance and processor events [162].

PAPI runs on most modern processors and operating systems of interest to HPC, including IBM POWER4-7, Cray XT{3-6}, XE{5,6}, IBM Blue Gene, x86_64 (Intel, AMD), ARM, MIPS, UltraSparc I, II & III, Intel Xeon Phi. PAPI is being used on all major supercomputers and by many performance tools.

PAPI was used by the authors of the PRACE-1IP report "CP2K – Scalable Atomistic Simulation for the PRACE Community", where they tested the CP2K code on the PRACE CURIE supercomputer [163].

PAPI is a cross-platform interface to the hardware performance counters available on modern microprocessors and it does not contain the mechanisms for collecting this information across the many thousands of processes that a petascale simulation may run on. This task is left for profiling tools that make use of PAPI. PAPI is the underlying API for many open-source performance tools, such as TAU, Scalasca, and Vampir. It is has therefore been widely used on multi-petascale systems to date.

Latest release/version: v5.1.0

3.8.8 *Temanejo/Ayudame*

Temanejo is an extensible, scalable debugger for the StarSs/OmpSs programming model, whereas Ayudame is a library, which communicates between the StarSs/OmpSs runtime and Temanejo [164]. The tool is aimed at any kind of cluster. It is intended for exascaling applications, thus for large-scale systems. As of today, all the work related to Ayudame/Temanejo has only been implemented in the TEXT project and Mont-Blanc project [165].

Latest release/version: v5.1.0

4 Scalable Libraries and Algorithms

In this section, we characterize the following scalable libraries and algorithms (hereafter referred to as tools) that are of interest to T7.2 in particular, and the European HPC community more widely, as we move towards the deep petascale and exascale eras:

- Direct Solvers
- Iterative Solvers
- FFT Libraries
- PETSc
- Trilinos
- Zoltan
- ParMetis
- PT-Scotch
- NetGen

For each tool, we provide an overview and discuss the tool's present state, how it has been employed in PRACE to date, how it has been employed more widely, and our views on the suitability of the tool for enabling PRACE application codes during the exploitation phase of T7.2.

The list above is roughly divided into three areas of focus, namely scalable numerical algorithms/methods (Direct Solvers, Iterative Solvers and FFT Libraries), higher-level libraries and other, mesh/graph partitioning, tools NetGen). The list by no means represents an exhaustive survey of scalable libraries and algorithms, but rather tries to seek a balance between assessing what has been investigated in PRACE to date and what is currently being investigated elsewhere (specifically within exascale projects) with the same set of tools. In carrying out our assessments we have drawn valuable information from each of the European exascale projects, and in particular the CRESTA project [3], which has a significant co-design focus.

As a consequence of the move towards large multi-petascale heterogeneous systems, there is an increasing demand for new and improved scalable, efficient, and reliable numerical algorithms and libraries that confront existing and upcoming complexities associated with such systems, including complex memory hierarchies, the overhead of data movement and fault tolerance.

One such representative library that we would like to mention here, due to the fact that it confronts the challenges posed by heterogeneous systems from several different angles, is the dense linear algebra MAGMA library [166], the development of which is being led by the Innovative Computing Lab (ICL) at the University of Tennessee. During our survey we have found that several initiatives in both the development and exploitation of MAGMA have already been taken within PRACE, particularly most recently within PRACE 2IP (WP8 and WP12). The MAGMA library not only targets new heterogeneous architectures including GPUs (and most recently Intel's Xeon Phi coprocessor), but also employs innovative and forward looking methods for reducing the barriers that are faced when trying to scale both within and across nodes.

An interesting effort within MAGMA, which we believe PRACE enablement projects should take inspiration from is the substitution of the widely used fork–join model with a higher-level directed acyclic graph (DAG) model, which goes some way to reducing synchronization points, a well known barrier to the scalability of many algorithms. Indeed, as we have found from our survey, synchronization avoiding algorithms are being increasingly investigated

both within and outside PRACE and we are happy to report on several other initiatives that have recently been taken within PRACE in this area.

A closely related area of research is being actively pursued in the area of communication avoiding (CA) algorithms, where global communication, in particular, is known to be a severe barrier when trying to scale across large core counts (see reports on FFT libraries in section 4.3). We report here on how communication avoiding algorithms have been investigated within WP12 PRACE-2IP using the PETSc framework [167], which is increasingly being employed as a higher-level library for solving PDEs. Indeed, PETSc is representative of the type of high-level libraries that are being increasingly leveraged within PRACE applications due to the ease of use in harnessing underlying message-passing-based infrastructure.

Within this section we also report on graph partitioning and mesh generation libraries where the former are mostly used for domain decomposition, used by iterative or hybrid solvers, as well as for the computation of fill-reducing or block-preserving orderings required by direct or hybrid solvers.

Finally, there are also many issues regarding the reduced reliability of hardware expected to have an important impact on libraries and algorithms on the road to exascale. This is certainly the case for fault tolerance and resilience. However, these issues are still very new to the PRACE community and in general we have not found many examples of initiatives within PRACE in this area to date, an issue that we feel needs to be rectified preparing quickly if we are to prepare applications for future multi-petascale and exascale machines.

4.1 Direct Solvers

4.1.1 *Brief overview*

In this report we provide a brief overview of the state of the art in direct linear algebra methods, where we cover direct methods for solving both dense and sparse problems. We feel that applying a simple metrics table here, as was done in other reports, is a challenging and somewhat futile exercise as in most cases it is very hard to sum up metrics such as scalability when reporting on such a broad class of algorithms. Instead, where possible, we will provide details on metrics when discussing the individual libraries and algorithms that we report on throughout. We focus first on dense linear algebra solvers and then on sparse solvers.

Dense solvers:

As well as libraries targeted at multi-/many-core platforms, there have been other recent initiatives at improving the performance of distributed ScaLAPACK-like libraries, which include the ELPA library [168]. ELPA is a new efficient distributed parallel direct eigenvalue solver for symmetric matrices. It contains both an improved one-step ScaLAPACK type solver (ELPA1) and the two-step solver ELPA2. While ELPA uses the same matrix layout as ScaLAPACK [169] the actual parallel linear algebra routines are completely rewritten. ELPA1 implements the same linear algebra as traditional solutions (reduction to tridiagonal form by Householder transforms, divide & conquer solution, eigenvector backtransform). In ELPA2, the reduction to tri-diagonal form and the corresponding back-transformation are replaced by a two-step version, giving an additional significant performance improvement.

ELPA is a Fortran-based MPI-only implementation. Once compiled, ELPA library routines can be linked to from C, C++, Fortran. It will thus work both in a single-node, shared memory environment, as well as large clusters of separate nodes. ELPA has proven to be a scalable and well performing solver for all matrix sizes tested between 2,500 and 100,000 and tests have demonstrated good scaling on over 290,000 processor cores on JUGENE [170]. Generally ELPA2 is found to be superior to ELPA1 except for small matrix sizes [171].

According to the ELPA website, as of February 2013, a CUDA version of ELPA is in preparation in collaboration with NVIDIA [168].

The MAGMA library [166] is a dense linear algebra library that targets GPUs and more recently Intel's Xeon Phi architecture. It can be called from Fortran and C/C++ code. The libraries are a work in progress and not all features are available for all target architectures. The most mature version of the library is that targeted at NVIDIA GPUs (MAGMA 1.3), which is developed using CUDA. For this version, over 80 hybrid algorithms have been developed (a total of 320 routines), including one-sided factorizations, linear system solvers, as well as two-sided factorizations and eigenproblem solvers. This version of the library also features a subset of BLAS and auxiliary routines in CUDA. MAGMA uses a hybridization methodology where algorithms of interest are split into tasks of varying granularity and their execution scheduled over the available hardware components. Scheduling can be static or dynamic. In either case, small non-parallelizable tasks are scheduled on the CPU, and larger more parallelizable ones, often Level 3 BLAS, are scheduled on the GPU. For eigenproblem solvers in MAGMA, current results demonstrate that MAGMA with 1 GPU can be 12x faster than Intel's MKL on state-of-art multicore systems [172]. A scalable multi-GPU version of MAGMA is currently being developed using the StarPU runtime system and is showing very promising results [173]. More recently, MAGMA MIC [174] has been developed, which is targeted at shared memory systems featuring Intel Xeon Phi coprocessors and includes many one-sided factorizations. For a full set of performance results the reader is referred to the following report [174]. MAGMA MIC v1.0 (Beta) was released in March 2013, which is an update on v0.3 and includes several new functionalities including added multiple MIC factorization routines.

As is to be expected, both Intel and NVIDIA have implemented optimised versions of mathematical software, particularly the BLAS and LAPACK libraries within their own commercial libraries for their respective many-core offerings. In the case of Intel, all routines of interest are available as their libraries are based on the well-known MKL. There are three ways in which the MKL libraries may be used: 'Host' - running purely on the host, without reference to the Xeon Phi coprocessor, 'Auto Offload' - running on the host and automatically offloading data for execution on the Xeon Phi and 'Native' - running purely on the Xeon Phi. In the case of NVIDIA, there are NVIDIA-developed BLAS libraries such as CUBLAS, which have demonstrated impressive performance on the new K20 architecture as well third-party commercial libraries such as CULA [175] and Array Fire [176].

Sparse solvers:

There are in fact not many general purpose distributed memory sparse solvers. SuperLU [177] is one of the most widely used direct sparse solvers and performs a LU decomposition for large, sparse, symmetric/non-symmetric systems of linear equations on both shared memory architectures as well as distributed memory architectures. The library is written in C and is callable from either C or Fortran. It is freely available from Lawrence Berkeley National Laboratory. While `SuperLU_MT` is targeted at shared memory parallel machines, `SuperLU_DIST` is designed for large-scale distributed memory systems [178]. `SuperLU_MT` has three major steps including sparsity ordering, factorization that arranges partial pivoting, symbolic factorization and numerical factorization which are performed in an alternating fashion, and triangular solution. While `SuperLU_DIST` uses BLAS 3 for factorization, `SuperLU_MT` uses BLAS 2.5 with multiple matrix vector multiplications. `SuperLU_DIST` uses static pivoting instead of partial pivoting for numerical stability due to numerical pivoting being complicated on distributed memory architecture (this method is in contrast to that found in MUMPS [179]).

MUMPS [179] is a freely available package with Fortran and C interfaces for solving systems of linear equations of the form Ax = b, where A is a square sparse matrix that can be either unsymmetric, symmetric positive definite, or general symmetric on distributed memory computers. MUMPS implements a direct method based on a multifrontal approach, which performs a LU factorization in the unsymmetric case and a LDLT factorisation in the symmetric case. MUMPS exploits both parallelism arising from sparsity in the matrix A and from dense factorizations kernels. The main features of the MUMPS package include the solution of the transposed system, input of the matrix in assembled format (distributed or centralized) or elemental format, error analysis, iterative refinement, scaling of the original matrix, out-of-core capability, parallel analysis, detection of null pivots, basic estimate of rank deficiency and null space basis for symmetric matrices, and computation of a Schur complement matrix. MUMPS offers several built-in ordering algorithms, a tight interface to some external ordering packages such as METIS [180], and the possibility for the user to input a given ordering. Finally, MUMPS is available in various arithmetics (real or complex, single or double precision). A parallel analysis and an out-of-core functionality are also available. The parallel version of MUMPS requires MPI, BLAS, BLACS, and ScaLAPACK, but it has its limitations: MUMPS itself is parallelized only using the MPI model. Hybrid MPI with threading parallelization is available via shared memory BLAS implementations, but MUMPS does not currently include any explicit hybrid parallelization targeted at heterogeneous systems.

Latest release/version: LAPACK 3.4.2, ScaLAPACK 2.0.2, PLASMA 2.4.6, and MAGMA 1.3, MUMPS v4.10.0, SuperLU v4.3, ELPA v2013.02.BETA

4.1.2 *Evidence of use within PRACE*

In the PRACE-1IP whitepaper, 'Numerical Library Eigensolver Performance on PRACE Tier-0 Systems' [171], the parallel performance of several established as well as newly developed parallel dense symmetric eigensolver numerical library routines on PRACE Tier-0 systems are analysed. The performance results from the new ELPA software package are particularly impressive. Not only are the routines significantly faster than their ScaLAPACK counterparts, but also the scalability is better throughout. A two-fold to three-fold advantage in performance rendered by ELPA over ScaLAPACK was measured on both JUGENE and CURIE for a range of problem sizes, which would likely impact the performance of many PRACE applications significantly. Moreover, as the authors of the whitepaper point out, ELPA is also designed to be a 'drop-in' substitute for ScaLAPACK e.g., it uses the same block-cyclic data distribution and therefore applying it to application codes with existing ScaLAPACK interfaces should be straightforward. The authors also point out that at this stage it is not clear which of the two ELPA eigensolvers performs best, though a pattern evident in these benchmarks is that the one-stage solution performs best on smaller datasets whilst the two-stage approach scales better and is more efficient on larger test cases

In work produced recently in PRACE-2IP and reported on in D12.2 'Exploration of Scalable Numerical Algorithms' [181] and written in collaboration with the Innovative Computing Lab at the University of Tennessee 'A Hybrid Hermitian General Eigenvalue Solver' reports on the development of single node hybrid general eigenvalue solvers where development and testing was carried out on the Castor cluster at CSCS (Each node is a dual 6-core Intel Xeon 5650 with two NVIDIA M2090s). Results of the hybrid algorithms compared with a shared memory library (MKL) and a distributed memory library (ELPA) show that for the two algorithms investigated (one- and two-sided algorithms), the MAGMA implementations outperform both the MKL and ELPA libraries on a single node by factor of 2x-3x.

In WP12 PRACE-2IP there have been several projects to improve direct solvers, including SuperLU, which are reported on in deliverable D12.2 [181] in PRACE-2IP. Work on developing a hybrid SuperLU algorithm utilizing the MPI/OpenMP hybrid programming approach as well as investigations into porting SuperLU to GPUs are reported in PRACE-2IP deliverable 12.2. It is expected that the hybrid algorithm utilizing the MPI+X programming model within SuperLU_MCDT (Many Core Distributed) by ITU-UHeM [182] to solve large sparse linear systems will reduce the communication overhead associated with MPI so that better scalability can be achieved.

4.1.3 *Evidence of use outside PRACE*

The MAGMA libraries are increasingly being used within petascale codes both within and outside PRACE. A very large selection performance tests for various subroutines are available on the MAGMA websites [166].

Interesting work has recently been carried out at Lawrence Berkley Lab into developing an alternative hybrid solver to SuperLU (PDSLin) [183]. The developers of PDSLin point out that for small system sizes, a direct solver such as SuperLU can be employed to obtain an accurate solution as long as the condition number is bounded by the reciprocal of the floating-point machine precision. However, SuperLU scales effectively only to hundreds of processors or less. In PDSLin, the SuperLU_DIST 2.4 is used as a direct solver for interior subdomains and the Schur complement systems are solved using a preconditioned Krylov method in PETSc. It was shown in [183] that PDSLin significantly outperforms SuperLU_DIST on high core counts (2048 cores).

4.1.4 *Conclusion*

In terms of dense solvers, we feel that ELPA shows real promise and should be further investigated as an alternative to ScaLAPACK within PRACE applications. We also believe that MAGMA is one of the most promising libraries containing dense direct solvers with impressive performance and indications of long-term sustainability. The library targets all accelerators/coprocessor architectures and is also fully portable in its OpenCL form. Distributed-memory versions of the library are also currently in progress and should be investigated as alternatives to ScaLAPACK, possibly during the exploitation phase of T7.2

In terms of sparse solvers, MUMPS appears to be a very robust and efficient direct solver for medium-sized distributed or centralized sparse linear systems arising for instance from discretization of PDE problems. Regarding large-scale problems, MUMPS will not be usable as a standalone solver of the original linear system. However, MUMPS will still be a very important tool for the robust and efficient solution of auxiliary medium-sized distributed and centralized sparse linear systems arising in higher level methods like FETI domain decomposition methods, and will in turn extend their scalability. MUMPS has many unique features such as the detection of null pivots, rank deficiency, etc. that can be very helpful in higher level scalable methods. While both SuperLU_DIST and SuperLU_MCDT show promise as dense sparse solvers, like all the other libraries mentioned here, improvements, such as synchronization reduction, data movement minimization and fault tolerance need to be included in the library in order for SuperLU_MCDT to enable applications on future multi-petascale and exascale systems.

4.2 Iterative solvers

4.2.1 *Brief overview*

In this report we provide a brief overview of the state of the art in iterative methods. For the same reasons as were applied to direct solvers, we will only provide details on metrics when discussing the individual algorithms that we report on throughout.

Iterative algorithms have become the de-facto approach for the solution of sparse linear systems of equations on large-scale parallel systems due to their amenability to parallelization. All iterative algorithms used for the solution of linear systems require a number of global synchronization operations (e.g., `MPI_ALLREDUCE`) for computing global scalars as well as a number of local synchronization operations due to the point-to-point communications incurred by the sparse matrix-vector operations. These local and global synchronization operations create barriers beyond which computation cannot proceed until all participating processors have reached that point.

The most prominent iterative method for solving sparse systems of linear equations is the Krylov subspace method [184]. There are many different variants of the method (CG, Bi-CG, GMRES, etc), so that almost any system of linear equations can be approximately solved. However, in cases where the condition number of the matrix involved in the system of equations gets too large or the matrix becomes nearly ill conditioned, Krylov methods tend to be very slow in convergence. This lack of robustness makes it less likely that Krylov methods can be used in isolation within real applications. The common solution to this problem is the use of preconditioning. Unfortunately, finding a good preconditioner to solve a given sparse linear system is often viewed as a combination of art and science. There are many varieties of preconditioners to choose from including ILU, Jacobi, SOR, and SSOR precondtioners [184].

Libraries that provide robust collections of iterative methods as well as associated infrastructure (e.g., preconditioners) are becoming increasingly popular within and outside PRACE. PETSc [167], Trilinos [65] and Hypre [185] are particularly representative of such offerings. PETSc and Trilinos are high-level libraries for solving PDEs and multi-physics problems that consists of many different variants of Krylov subspace-based solvers. Due to them both being particularly high-level, we provide a separate report on each in section 4.4 [185] and section 4.5 respectively.

Hypre [185] is a library for solving large, sparse linear systems of equations on massively parallel computers. The main features of this library include scalable preconditioners (several families of preconditioned algorithms focused on the scalable solution of very large sparse linear systems) and the implementation of a suite of common iterative methods (Krylov-based). With exascale in mind, the developers of Hypre have begun to focus on a hybrid (MPI/OpenMP) programming model for BoomerAMG which is an Algebraic MultiGrid (AMG) method [186] within Hypre that has garnered a lot interest and has been investigated both within PRACE and within European exascale projects (CRESTA) recently [187].

Other high level libraries that are explicitly mentioned in the ESSI 'Working Group Report on Numerical Libraries and Algorithms' [188] that we would like to mention briefly here are:

- DUNE, the Distributed and Unified Numerics Environment [189] is a modular toolbox for solving partial differential equations (PDEs) with grid-based methods. It supports the easy implementation of methods like Finite Elements (FE), Finite Volumes (FV), and also Finite Differences (FD).
- FEAST, Finite Element Analysis and Solution Tools with significant exploitation of CPU/GPU computing [190].

- deal.II, a C++ program library targeted at the solution of PDE's using adaptive finite elements. Hybrid parallelization by MPI and Intel's Threading Build Blocks (TBB). For its creation, the principal authors of Deal.II received the 2007 J.H. Wilkinson Prize for Numerical Software [64].
- MLD2P4/PSBLAS. These two tightly related projects provide parallel sparse matrix tools, Krylov solvers for linear systems and algebraic multilevel preconditiones in Fortran 95/2003 [191].

As well as solving systems of linear equations it should also be pointed out that iterative solvers are used to solve large sparse eigenproblems where only a small number of eigensolutions are required with respect to the matrix size. There are several MPI-based iterative eigensolvers, including PARPACK [192] and more recently, SLEPc [193]. SLEPc is a particularly interesting example as it is built on top of PETSc and leverages much of its infrastructure. Indications are that in most cases SLEPc outperforms PARPACK and has a broader selection of algorithms to choose from due to its underlying structure being based on PETSc [194].

4.2.2 *Evidence of use within PRACE*

A significant amount of "forward-looking" investigations into iterative algorithms was carried out recently in WP12 PRACE-2IP within the task 'Exploration of Scalable Numerical Algorithms'. In the project, 'Asynchronous Algorithms for Large Sparse Linear Systems', reported on within PRACE-2IP Deliverable D12.2 [181], investigations were made into asynchronous implementations of the Jacobi method, where one traditional synchronous and two asynchronous variants of the method were implemented using three programming models: MPI, SHMEM and OpenMP, and where the performance of these implementations was investigated on HECToR. Results showed that SHMEM can provide a more efficient implementation of asynchronous message-passing than MPI, and that for problems that require high core counts, asynchronous algorithms can outperform their synchronous counterparts by 10%. The authors of the report point out that the OpenMP implementation was found to give good performance for asynchronous algorithms and was also very easy to program compared to MPI and SHMEM. The authors also suggest that OpenMP might be applicable in a hybrid model with MPI, particularly since they found that the asynchronous implementation of the Jacobi method in OpenMP to be 33% faster than the synchronous equivalent. The authors also point out that asynchronous algorithms are expected to be more tolerant to faults, which could be a major advantage when designing applications on the road to exascale.

In a separate project reported on in PRACE-2IP Deliverable D12.2 [181] 'Implementation and Performance Evaluation of the CA-CG Algorithm on Massively Parallel HPC Clusters' investigations were carried out into an emerging class of communication-avoiding Krylov subspace (Conjugate Gradient [CG]) methods which seek to reduce the amount of global synchronization points within iterative methods. The investigations were carried out by evaluating the feasibility of implementing the 'CA-CG' algorithm and testing its overall performance on a set of benchmark platforms. The framework used for the implementation of the algorithm was the one provided by PETSc (reported on section 4.4 PETSc) and performance comparisons were made between the standard CG algorithm and the communication-avoiding version of the algorithm within the PETSc library. The two algorithms were tested on a selected set of sparse positive definite matrices taken from the UFL [195] database, where calculations were carried out on the PLX and FERMI clusters at CINECA. The authors report that in many cases, non-negligible increases in performance were found.

Finally, in the PRACE-1IP whitepaper, 'Parallel Solvers for Incompressible Navier-Stokes Equations and Scalable Tools for FEM Applications' [196] the Hypre library was tested on variants of 1D, 2D and 3D domain partitioning for the 3D test problems of computational fluid dynamics (CFD) where the Algebraic MultiGrid (AMG) method BoomerAMG within Hyper was employed. For more details on the performance obtained, including scalability, the reader is referred to the whitepaper.

4.2.3 *Evidence of use outside PRACE*

Research into iterative methods is a huge area of activity and we do not in any way attempt to provide a global picture of all of the efforts being pursued within the confines of this report. Suffice to say that one increasing area of focus is on communication avoiding iterative algorithms as has been evidenced by the recent exploratory work carried out within WP12 PRACE-2IP. Such algorithms are the focus of recent investigations at Intel's ExaScience Lab [81] where very recently [197], a pipelined version of the GMRES algorithm was derived in which the communication cost can be hidden by using non-blocking all-reduce operations. At the time of publication (April 2012) it was noted by the authors that no complete implementation of the MPI 3.0 standard was available (the standard had not been published at that stage), so technical hurdles meant that the pipelined version of GMRES was only investigated by using an analytical performance model which predicted that speedups of 3.5X for the pipelined algorithm compared to the standard GMRES, for solving a problem with $N=2000^3$ unknowns on 200,000 nodes could be achieved. A detailed investigation of the impact of global communication latency on Krylov methods at extreme scales has also been reported by the same group [198].

We do not attempt to report here in any great detail on preconditioners (which are closely associated with iterative methods), but as pointed out in D4.1.1 of the CRESTA project [187] the new theory of hierarchical matrices (H-Matrix method [199]) promises to be one of the most interesting ways of finding effective preconditioners for iterative methods. There are several libraries for solving these systems of linear equations using 'H-Matrix' methods on shared memory systems (according to CRESTA developers, H-Libpro [200] is the one of the best implementations). Currently, work is underway to develop an H-matrix library that uses MPI. The CRESTA project, also wants to try to apply this new theory to solve systems of linear equations.

Work at the University of Utah demonstrated that the use of the direct solver library, Hypre, combined with the Uintah Software framework [201] to solve incompressible fluid flow problems resulted in "better than expected" weak scaling [202]. The tests were run on the Kraken Cray XT5 system made up of 112896 2.6 GHz AMD Opteron cores, and the Titan Cray XK6 system, with 299,008 cpu cores and 18,600 GPUs, as well as Titan's predecessor Jaguar (Cray XT5). On the Jaguar system, a Weak Scaling Taylor Green Vortex showed the time per iteration increased from 1s with 192 cores to 5s with 196k cores. The Titan system for the same test (CPU cores only) showed the time per iteration increased from 1s with 192 cores, to roughly 1.5s with 131k cores. In both cases, weak scaling was seen to be quite successful, especially when run on Titan. On the Kraken, a linear solver and red-black Gauss Seidel were run with a variety of hypre options for a Helium plume. These results showed that with the correct Hypre options, scalability on the order of log(p) was achievable where p is the number of cores.

4.2.4 *Conclusion*

Iterative methods, particularly Krylov subspace-based methods are fundamental algorithms to many PRACE applications and have therefore received a considerable amount of attention within PRACE projects to date. Very interesting exploratory work into communication avoiding Conjugate Gradient algorithms has recently been carried out in WP12 PRACE-2IP and is an active focus of research in many exascale projects in both Europe and the US. With new implementations of the MPI 3.0 standard available (and on the horizon), which now offer non-blocking communications, there is good reason for PRACE partners to investigate these forward looking algorithms in more detail during the enablement of applications on PRACE systems in T7.2.

4.3 FFT Libraries

4.3.1 *Brief overview*

The FFT method is fundamental to many PRACE applications. In practice most codes that perform FFTs perform transformations on large multi-dimensional datasets. In this case it is convenient to implement the overall transform as a series of data redistributions between different data decompositions with each of the active dimensions in turn being local to a node. Between each of the redistributions the local active dimension is transformed using a non-distributed FFT library. The convenience of this approach is that it allows distributed FFT implementations to be built out of optimized single node FFT libraries and highly optimized MPI collectives. However, in common with all implementations, the overall performance is largely limited by data movements. Many highly optimized node-local FFT libraries exist (see below). However, for distributed memory applications the performance of the inter-node data communications is far more significant for overall performance than the performance of the underlying node-local FFT library, so the choice of which underlying FFT library is largely irrelevant and the performance of the MPI collectives dominate the performance.

Although multi-node FFT libraries do exist (and have been investigated within PRACE), most applications do not employ them. Instead, each application implements multi-node FFTs out of a combination of node-local FFT libraries and MPI collectives. Most of the multi-node FFT libraries only support a limited range of input and output data decompositions that typically don't correspond to the data decompositions required by the real application (an exception to this rule is the DaFT library [203] within DL_POLY which has been investigated within PRACE-1P). Such implementations also typically have no performance advantage over what is generally obtained by application specific implementations built out of the same underlying libraries. Most applications use the collective call `MPI_ALLTOALL` due to the fact that this global communication pattern is known by all participating processors, allowing greater scope for optimisation. `The MPI_ALLTOALLV` collective allows greater flexibility in data decomposition and may be used by applications where load imbalance considerations are more important than the absolute performance of the FFT.

There are several popular FFT libraries that we report on briefly here:

FFTW [204] is a freely available library callable from C and Fortran codes for computing the discrete Fourier transform (DFT) in arbitrary dimension, of arbitrary input size, and of both real and complex data. It works best on arrays of sizes with small prime factors, with powers of 2 being optimal and large primes being worst case. FFTW, being free software, is the FFT library of choice for most applications, as its performance is claimed to be typically superior to that of other publicly available FFT software, and is even competitive with vendor-tuned codes (to justify its name "Fastest Fourier Transform in the West") but in contrast to the latter

FFTW is portable. FFTW makes use of several variants of the Cooley–Tukey FFT algorithm, Rader's, Bluestein's and prime-factor FFT algorithms. It supports SSE/SSE2/Altivec, since version 3.0. and since version 3.3.1 supports AVX and ARM Neon. In the investigated version (3.3.1), which is the first version to support parallel MPI 3D FFTs, only slab decompositions were supported.

FFTW is portable to any platform with a C compiler. It is not tuned to a fixed machine; instead, it uses a planner to adapt its algorithms to the hardware in order to maximize performance.

The recent upgrades of FFTW include:

- Adapting the number of cores involved in the DFT execution to the size of the index involved in the domain decomposition (especially useful in 2D DFT problems);
- Domain decomposition algorithm 2DDD instead of Slab (scaling N3 vs. N with the system size) for 3D arrays, thus providing a nearly ideal scalability at least up to several thousands of cores (also for relatively small data arrays);

FFTW has been shown to perform very well on 1024 cores within PRACE [205].

FFTE [206] is a freely available FORTRAN (77 and 90) subroutine library (callable from C and Fortran) for computing the FFT in one, two and three dimensions. It includes complex, mixed-radix and parallel transforms. FFTE is open source, highly portable, but comes with little documentation. The developers claim it is typically faster than other publically available FFT implementations, and is even competitive with vendor-tuned libraries. The assessed version (v5.0) supports both slab and pencil decompositions. FFTE also supports Intel's SSE2/SSE3 instructions.

FFTE is targeted at shared and distributed memory parallel computers. (OpenMP, MPI and OpenMP/MPI) but efforts on performance improvement are highly desirable due to the fact that, in general, FFT libraries do not scale well beyond a few hundred cores. Also, the index involved in the parallel domain decomposition may impose a limit on the maximum number of usable cores. FFTE does not scale as well as FFTW but outperforms FFTW in absolute computing time [205].

There are also several GPU-based FFT libraries worth mentioning, including NVIDIA's cuFTT. The NVIDIA CUDA Fast Fourier Transform library (cuFFT) provides a simple interface for computing FFTs up to 10x faster than the MKL as reported on the NVIDIA developer website [207] DiGPUFFT [208] adds cuFFT support inside of P3DFFT, for GPU-accelerated 3D FFT computations. It has only been tested with P3DFFT 2.4 and CUFFT from CUDA Toolkit 3.2 and is reported on within the PRACE-1IP whitepaper, 'An Analysis of FFT Performance in PRACE Application Codes' [205] ,which indicates that except for large sized dimensions the benefits of using a GPU-based distributed FFT implementation are currently negligible.

Latest release/version: FFTE: 5.0, FFTW: 3.3.3, cuFFT: available as part of the CUDA 5.0 Toolkit.

4.3.2 *Evidence of use within PRACE*

The PRACE-1IP whitepaper, "An Analysis of FFT Performance in PRACE Application Codes" [205] reports on the assessment of the suitability, performance and scalability of various implementations of FFT for large-scale PRACE applications including Quantum ESPRESSO and DL_POLY. The FFTs investigated are both in-code implementations

(typically distributed) as well as various third-party numerical libraries, where in both cases underlying algorithms were implemented in both serial and parallel form. The implementations of FFTs investigated range from pure MPI, OpenMP versions for multicore, hybrid (MPI/OpenMP) as well as GPU-based implementations. The overall conclusion is that the scalability of parallel 3D FFTs remains inherently limited, owing to the all-to-all communications involved. Likewise the variety of data decompositions supported by the available libraries is also limited.

The PRACE-1IP whitepaper, "Enabling FFTE Library and FFTW3 Threading in Quantum ESPRESSO" [209] reports on the work that was carried out on enabling support for the FFTE library for the main FFT operation in Quantum ESPRESSO, as well as enabling threading support for the FFTW3 library already supported in Quantum ESPRESSO. The work on enabling these libraries was motivated by the excellent performance results of the FFTE library described in [205] and by the expectation that the hybrid approach with FFTW3 in Quantum ESPRESSO would achieve better performance compared to the existing MPI implementation. However, this expectation could not be confirmed. In the case of the FFTE extension, a performance benefit may only be significant when a large charge density mesh is required by the physical system. The QE FFTW3 hybrid explicit and implicit extensions illustrated better performance compared to the internal QE FFTW hybrid approach, but were shown to be perform worse than the pure MPI version. It is suspected that the overhead related to thread management outweighs the benefits of reduced MPI communication, up to a certain number of MPI processes. However, this situation may change dependening on the configuration of the problems analysed.

More recently in WP12 PRACE-2IP, the whitepaper, 'Autotuning of the FFTW Library for Massively Parallel Supercomputers' [210], reports on the work on improving the performance of the FFTW library by refining its auto-tuning mechanisms. The major bottlenecks of the current FFTW implementation are identified, as well as the influence of the domain decomposition algorithms on the performance. An improved performance of the autotuning mechanism is achieved by a new parallel domain decomposition, which is detailed further in the whitepaper.

4.3.3 *Evidence of use outside PRACE*

The European exascale project, CRESTA, has carried out investigations into FFT libraries and suggests that the ultimate limiting factor in the performance of the distributed FFT operation is the performance of the `MPI_ALLTOALL` operation and this is in turn limited by non-pipelined message latencies. The CRESTA team point out that, in principle, it is possible to use single sided communications to overlap some of the data movement with the calculation of the local FFTs. However this requires the use of many small messages and would therefore also be very sensitive to communication latency. For the IFS and GROMACS co-design application, CRESTA has undertaken some detailed scalability analysis and these codes currently require MPI_ALLTOALL calls that consume an increasing fraction of time as the core count or model complexity increases. The non-blocking collectives that will be developed and implemented in later phases of the CRESTA project are likely to help with this issue. One particular approach that is currently under investigation is changing algorithms to use single-sided non-blocking communication that CRESTA expect will give opportunities for increased scalability [187].

4.3.4 *Pros and Cons*

		Pros	Cons
Scalability	FFTW	Better scalability with MPI-ony implementation than FFTE	Only scales to a few thousand cores at most
	FFTE	In all hybrid combinations scales better than in pure MPI (CURIE)	Only scales to a few thousand cores at most
Performance	FFTW	Shows excellent scaling on JUGENE	Hybrid performance is still considerably poorer than MPI-only version
	FFTE	Implemented in QE, FFTE library slightly outperforms FFTW3 for different number of MPI processes, the difference diminishing for higher number of MPI processes; The FFTE serial mode performance benefit could be significant for large charge density meshes.	
Productivity	FFTW	Library calls that are easy to implement. Callable from Fortran and C code. Wrappers and bindings exist for many other languages	
	FFTE	Library calls that are easy to implement. Callable from Fortran and C code	
Sustainability	FFTW	On-going further development (new versions) with new features. Active community	-
	FFTE		Last release is from Nov. 2011. Long term support for library is unclear
Correctness	FFTW		-
	FFTE		-
Portability	FFTW	Any platform with a C compiler; both C and FORTRAN interfaces	-

	FFTE	Highly portable	
Availability	FFTW	Free software. Extensive documentation	-
	FFTE	Open source	Almost no documentation
Resilience		-	-

4.3.5 *Target systems/architectures*

X86	IBM	GPU	MIC	ARM	Heterogeneous
Yes	Yes	Yes	Yes	Yes	Yes

4.3.6 *Conclusion*

As the CRESTA investigations into FFT methods point out, the major bottleneck in distributed FFT implementations is the `MPI_ALLTOALL` communication overhead, which becomes particularly problematic on large core counts. This problem can be alleviated somewhat by using hybrid MPI+OpenMP methods, but further optimisations are badly needed, as FFTs form the basis of many popular applications running on large-scale PRACE systems. In order to push FFT methods further along the road to exascale, we agree with CRESTA that further investigations should be made into non-blocking collective communications within in-code FFT implementations, possibly via the new MPI 3.0 standard features now becoming available. If FFT methods are to be investigated further in T7.2, we recommend that PRACE partners work closely with CRESTA and other European exascale projects to learn more about how their efforts in this area are progressing and if any non-blocking implementations are ready to be exploited in real applications.

4.4 PETSc

4.4.1 *Brief overview*

PETSc [167] the Portable, Extensible Toolkit for Scientific computation, provides sets of tools for the parallel (as well as serial), numerical solution of PDEs that require solving large-scale, sparse nonlinear systems of equations. PETSc includes nonlinear and linear equation solvers that employ a variety of Newton techniques and Krylov subspace methods. PETSc provides several parallel sparse matrix formats, including compressed row, block compressed row, and block diagonal storage. The table below gives an overview of the main numerical components of the PETSc library. PETSc is designed to facilitate extensibility. Thus, users can incorporate customized solvers and data structures when using the package. PETSc also provides an interface to several external software packages including BlockSolve95, ESSL, Matlab, ParMeTis, PVODE, and SPAI. PETSc is fully usable from Fortran, C and C++, and runs on most UNIX based-systems. PETSc has several features that make it very convenient for the application programmer. Users can create complete application programs for the parallel solution of nonlinear PDEs without writing much explicit message-passing code

themselves. Parallel vectors and sparse matrices can be easily and efficiently assembled through the mechanisms provided by PETSc. Furthermore, PETSc enables a great deal of runtime control for the user without any additional coding cost. The runtime options include control over the choice of solvers, preconditioners and problem parameters as well as the generation of performance logs. PETSc supports MPI, shared memory pthreads, and NVIDIA GPUs.

PETSc consists of a series of libraries that implement the high-level components required for linear algebra in separate classes: 'Index Sets', 'Vectors' and 'Matrices'; 'Krylov Subspace Methods' and 'Pre-conditioners'; and 'Non-linear Solvers' and 'Time Steppers'. The Vector and Matrix classes represent the lowest level of abstraction and are the core building blocks of most of the functionalities. PETSc uses a MPI/OpenMP/Pthreads model where the Krylov subspace methods and the pre-conditioners have not been threaded explicitly, but are instead threaded implicitly through the 'Mat' and 'Vec' classes. Other frequently used pre-conditioners, such as Symmetric Over Relaxation (SOR) or Incomplete LU-decomposition (ILU), have not been threaded yet due to their complex data dependencies. These may require a redesign of the algorithms to improve parallel efficiency.

PETSc is part of many high-level application codes like FLLOP, libMesh, Deal.II, PETScFEM, OpenFVM, OOFEM, etc.

Latest release/version: v3.3

4.4.2 *Evidence of use within PRACE*

As well as being leveraged by many PRACE application codes, PETSc is also becoming increasingly popular as an enablement tool.

In the PRACE whitepaper, 'Hybrid Total FETI Method' [211], an implementation of the PETSc library was shown to scale on up to 7700 cores with 75% efficiency. Furthermore, within this project, a linear elasticity problem with more than 200 million degrees of freedom was solved in 126 seconds. Also, in the whitepaper 'FETI Coarse Problem Parallelization Strategies and Their Comparison' [212], the PETSc library was used for different strategies for the coarse problem solution of an engineering problem with approximately 100 million degrees of freedom. The algorithm was tested on up to 5000 cores.

More recently in WP8 PRACE-2IP, on-going work on the enablement of Fluidity-ICOM, a Finite Element ocean modeling software framework, is using the PETSc 'OpenMP development branch', where the Krylov-based algorithm was tested on up to 32K cores. In this work it was found that the sparse matrix vector multiply kernels of the algorithm can scale up to 32K cores with up to 85% efficiency on HECToR. This is on-going work and further details will become available towards the end of PRACE-2IP.

4.4.3 *Evidence of use outside PRACE*

The PETSc webite [167] provides reasonably good updates on where and how PETSc is being used as well as how it is scaling on large-scale machines worldwide. In the CRESTA project, PETSc has been studied in detail as part of co-design efforts on the ELMFIRE application. The CRESTA developers have analyzed benchmark results on the examples for a matrix size of 10^8x10^8 to determine the most important challenges of an exascale Conjugate-Gradient-like calculation (Bone Matrix was used for the tests). Runtime measurements were carried out on the Cray XE6 system at HLRS and calculations were initially done in double precision with 64-bit representation for the matrix indices. Close to ideal scaling was demonstrated on up to 9280 cores, but the achieved performance was no more 1.5% of the theoretical peak

performance. This was explained by the matrix not consisting of blocks as well as the CRS format was not being optimal for the matrix vector multiplication. The increased overhead of MPI_ALLREDUCE calls within the iterative algorithm in PETSc is clearly identified as a bottleneck as the core count is increased and further work is being currently carried out into alleviating such global communication bottlenecks via non-blocking one-sided communications [187].

4.4.4 *Pros and Cons*

Metric	**Pros**	**Cons**
Scalability	Scalability of PETSc library was tested on very large problems and the library scaled up to ten thousands cores (processes) with negligible lost of efficiency.	
Performance	Performance depends on the algorithm used. But generally it is observed that performance of algorithms implemented in PETSc is very good.	
Productivity	PETSc includes a large suite of parallel linear, nonlinear equation solvers and ODE integrators that are easily used in application codes written in C, C++, Fortran and now Python. PETSc is relatively well documented and easy to learn. It uses only essential level of object orientation so that it is understandable for programmers not used to the object-oriented paradigm.	Documentation of more advanced features is sometimes very minimal and one has to browse through the code.
Sustainability	Long-term project with stable codebase and rapidly evolving development branch. Since 1990 developed by Argonne NL. PETSc is part of number of SW projects and many of projects interface PETSc.	
Correctness	Good error checking system, large test suite	
Portability	All significant platforms – see above	No information about Intel MIC portability

Availability	Open source, directly downloadable software, often preinstalled on HPC systems	
Resilience	Application-specific	

4.4.5 *Target systems/architectures*

X86	IBM	GPU	MIC	ARM	Heterogeneous
Yes	Yes	Yes	Development	Unknown	Yes

4.4.6 *Conclusion*

PETSc has strong potential for enabling applications on future multi-petascale and exascale systems, but several features expected from exascale libraries need to be implemented in the library (communication reducing algorithms, error resilience, fault tolerance etc.) As PETSc interfaces with several widely used numerical libraries (MUMPS, Hypre, SuperLU, METIS), its scalability is strongly dependent on the scalability of these libraries also. It is interesting to note that CRESTA chose the PETSc library as being representative of iterative methods and was chosen for further research on exascale co-design efforts. Indications are that PETSc is well supported by the DOE in the US and will continue to make inroads on building in features that will be important on the road to exascale.

4.5 Trilinos

4.5.1 *Brief overview*

Trilinos [65] is a collection of packages intended for a solution of large-scale complex multiphysics engineering and scientific problems. It is based mainly on C++ and the focus is given on modern object-oriented design, modularity and extensibility. Trilinos provides packages with basic linear algebra objects and routines, packages for iterative and direct solvers; preconditioners; nonlinear, transient and optimization solvers; eigensolvers; discretization and mesh generation tools; load balancing tools etc. Some packages also provide users with basic tools like I/O support, performance measurement or BLAS/LAPACK wrappers. Overall, there are more than 50 packages in Trilinos. Algorithms implemented in Trilinos can run in both serial and parallel manner. Although some of the packages lack extensive documentation, the core packages are relatively well documented as well as a shared memory parallelization using pthreads, Intel TBB and NVIDIA/CUDA. A hybrid MPI-shared memory and MPI-GPU parallelization is supported as well. Trilinos has been selected by Cray as a part of its 'Application Developer's Environment'. The Cray version of Trilinos also includes a set of Cray Adaptive Sparse Kernels (CASK) that performs SpMV and includes optimized versions of single- and multiple-vector matrix vector multiplies. Modern algorithms such as communication avoiding GMRES and communication avoiding hybrid-parallel orthogonalization TSQR have been implemented within the latest versions of the library (namely to the Belos packages). Fault-tolerant solvers are also in development [213].

Latest release/version: v11.0.3

4.5.2 *Evidence of use within PRACE*

Although there have been training events on Trilinos within PRACE, we have found no evidence of Trilinos being used in PRACE to date

4.5.3 *Evidence of use outside PRACE*

The Trilinos website [65] provides up-to-date information on where and how the library is being used. However, we have found it very difficult to find Trilinos being used in any real applications to date. In [214] Trilinos was used for parallel modeling of bone structure on a Cray XT system and scaled on up to 4000 cores, with tests being carried out for over one billion degrees of freedom.

4.5.4 *Pros and Cons*

Metric	**Pros**	**Cons**
Scalability	The scalability of some Trilinos based codes has been shown up to thousands of cores.	
Performance	The performance depends on the used architecture as well as the underlying low level LA routines. Generally the performance of the Trilinos based codes is considered to be good.	
Productivity	The object-oriented design and the modularity of the code allow a relatively easy code development. The most important packages are well documented. There are numerous tutorials on the project webpage and European Trilinos User Groups Meetings are organized annualy. ForTrilinos provides object-oriented Fortran interfaces to Trilinos C++ packages	Some packages have poor or confusing documentation. The high number of packages can be confusing. Another confusing factor for some people may be the object-oriented design.
Sustainability	Trilinos has been developed since 1998 under Sandia National Laboratory and is one of the most used scientific libraries nowadays. The changes in API between releases are usually not very significant and the code is	

	relatively sustainable (when compared e.g. to PETSc)	
Correctness	-	
Portability	Portable to UNIX and Windows systems, hybrid architectures, NVIDIA GPUs.	
Availability	Downloadable from project site. Most packages under BSD and LGPL licences.	
Resilience	-	

4.5.5 *Target systems/architectures*

X86	IBM	GPU	MIC	ARM	Heterogeneous
Yes	Yes	Yes	Unknown	Unknown	Yes

4.5.6 *Conclusion*

Although we could not find any evidence of Trilinos being used in PRACE to date, it appears to be a very powerful high-level library, which should be of interest to the PRACE community. The object oriented and templated design of the packages within Trilinos allows relatively easy development of codes for various types of architectures. Many modern techniques for improving scalability on future multi-petascale and exascale machines, such as communication reducing algorithms, have already been implemented into the framework which gives more reason to investigate the library further, possibly during the exploitation phase of T7.2.

4.6 Zoltan

4.6.1 *Brief overview*

Zoltan is a collection of data management services for unstructured, adaptive and dynamic applications. It includes a suite of parallel partitioning algorithms, data migration tools, parallel graph colouring tools, distributed data directories, unstructured communication services, and dynamic memory management tools [215]. The Zoltan Library contains a number of tools as listed below:

- Dynamic load balancing and parallel repartitioning algorithms, including geometric, hypergraph and graph partitioning methods.
- Data migration tools for moving data from old partitions to new one.
- Parallel graph colouring tools with both distance-1 and distance-2 colouring.
- Distributed data directories: scalable (in memory and computation) algorithms for locating needed off-processor data.

Zoltan runs on distributed-memory architectures. Zoltan requires MPI libraries and an ANSI C compiler. It has been tested on multiple architectures and is known to scale up to thousands of cores. Zoltan colouring algorithms are extended with new recoloring capabilities, providing lower numbers of colours at small additional cost. Also Zoltan's hierarchical partitioning is improved for better efficiency. Zoltan has support for 64-bit identifiers. Zoltan also can make use of ParMeTiS and PT-Scotch. If one chooses to do so, these libraries must be complied beforehand and their paths should be supplied to Zoltan during compilation. This feature of Zoltan enables a fair and fast comparison of ParMeTiS, PT-Scotch and Zoltan hypergraph partitioning tool (PHG). One can also use Zoltan with Fortran applications via the supplied interfaces.

Latest release/version: v3.6

4.6.2 *Evidence of use within PRACE*

Zoltan is already installed as a loadable module on HECToR and has been tested on JUGENE and CURIE [216] as well other supercomputing systems [217]. Among the PRACE supported applications, Vlasiator [217] uses Zoltan for distributing 3-D spatial grid elements as the communication pattern in Vlasiator is best modelled via hypergraph models.

4.6.3 *Evidence of use outside PRACE*

SuperLU_DIST: (Sparse Direct Solver and Preconditioner Distributed memory version) will be using Zoltan to perform parallel symbolic analysis to determine the nonzero structures of L and U factors [218]].

As PageRank [219] [220] is computed via iterative sparse-matrix-vector-multiplication operations and since these kinds of operations are best modelled via hypergraphs, Zoltan is used in parallel partitioning/distribution/redistribution of PageRank computations.

In flow simulations [221], as the simulation proceeds, spatial distribution of the computational load changes in a transient manner following the change in chemical state, and this change is unpredictable over a long interval. Therefore, adaptive schemes are devised by repeated partitioning and migration of data. Zoltan is used in these kinds of applications for its repartitioning and migration routines.

4.6.4 *Pros and Cons*

Metric	**Pros**	**Cons**
Scalability	Zoltan contains routines for partitioning with fixed vertices.	Zoltan data migration codes are written in a generic mode but thus not optimized for very large core counts. Data migration times in Zoltan can be very high.
Performance	Zoltan supports parallel hypergraph partitioning thus enabling correct communication modelling for various communication patterns. For these kinds of applications, the	Zoltan has a data-structure neutral design and an object-based interface, which causes extra data structure conversion costs prior to partitioning,

	performance obtained from the application after partitioning is better with Zoltan when compared with ParMeTiS or PT-Scotch.	ordering, colouring operations.
Productivity	Zoltan's data-structure neutral design allows it to be used by a variety of applications without imposing restrictions on application data structures. Its object-based interface provides a simple way for application developers to use the library and researchers to make new capabilities available under a common interface. Can be called from Fortran and C code	
Sustainability	Zoltan is maintained within the Trilinos package of Sandia National Labs. The tool is upgraded periodically and there is a strong community behind it. Any questions about the tool are directly and immediately answered by the developers.	
Correctness	-	-
Portability		Ports to MIC, ARM, and GPU are not expected to be available anytime soon.
Availability	Zoltan source code and binaries are publicly available from http://www.cs.sandia.gov/Zoltan/	
Resilience	-	-

Table 40 Zoltan - Pros and Cons

4.6.5 *Target systems/architectures*

X86	IBM	GPU	MIC	ARM	Heterogeneous
Yes	Yes	No	Unknown	Unknown	No

Table 41 Zoltan - Target systems/architectures

4.6.6 *Conclusion*

In conclusion, Zoltan with its ease of programming, support for hypergraph partitioning, fixed vertices, and colouring, shows potential for enabling applications on multi-petascale systems, especially if your applications communication pattern is best modelled via hypergraphs. Unfortunately, Zoltan has higher memory usage than other partitioning tools. It has relatively high data conversion preprocessing overheads. Its data migration routines seem to be not very efficient and they are amenable to improvement. Zoltan lacks support for hybrid systems and topology-aware routines. Due to these reasons, the current version of Zoltan seems to be not ready for enabling applications on future exascale systems. However, a new release of Zoltan is reported to be under development and many of these issues are expected to be addressed in this new version.

4.7 ParMeTiS

4.7.1 *Brief overview*

ParMeTiS [180] is an MPI-based parallel library that implements the multi-level paradigm for partitioning and repartitioning unstructured graphs and for computing fill-reducing orderings of sparse matrices. ParMeTiS is particularly suited for parallel numerical simulations involving large unstructured meshes. In this type of computation, ParMeTiS reduces the time spent in communication by computing mesh decompositions such that the numbers of interface elements are minimized [222].

In particular, ParMeTiS provides the following functionality:

- Partition unstructured graphs and meshes.
- Repartition graphs that correspond to adaptively refined meshes.
- Partition graphs for multi-phase and multi-physics simulations.
- Improve the quality of existing partitionings.
- Compute fill-reducing orderings for sparse direct factorization.
- Construct the dual graphs of meshes.

ParMeTiS is copyrighted by the Regents of the University of Minnesota. It can be freely used for educational and research purposes by non-profit institutions and US government agencies only. Other organizations are allowed to use ParMeTiS only for evaluation purposes, and any further uses will require prior approval [222].

ParMeTiS runs on distributed-memory architectures. It is only dependent on MPI libraries and should be able to run on any parallel architecture without much trouble. It has been tested on multiple architectures and is known to scale up to thousands of cores. The v4.0 release allows full support of 64 bit architectures that enables large-scale partitioning. It utilizes the latest version of MeTiS (v5.0), which allows for better support of multi-constraint partitioning. The v4.0 release has much lower memory requirements when compared with older releases. Multiple vertex weights/balance constraints are supported for most of the routines. This allows ParMeTiS to be used to partition graphs for multi-phase and multi-physics simulations. Support for 64 bit architectures by explicitly defining the width of the scalar “integer” data type used to store the adjacency structure of the graph. There has recently been complete re-write of its internal memory management, which has resulted in lower memory requirements.

ParMeTiS compiles with any C compiler (including Intel compilers). It also has bindings for FORTRAN so can be integrated into FORTRAN codes easily. It can be run independently so

that it can be called easily from other programs by means of `system()` or `popen()` system calls, or be piped together on a single shell command line.

Latest release/version: v4.02

4.7.2 *Evidence of use within PRACE*

Both MeTiS and ParMeTiS are already available as loadable modules in all HPC platforms at LRZ. Par-MeTiS has been tested and is known to perform well on PRACE systems such as CURIE, JUGENE and Hector as well other supercomputing systems. The previous versions (3.X) of ParMeTiS partitioning tool is successfully used in applications such as Code Saturne [223], Telemac-2D [224] and Elmer [225].

4.7.3 *Evidence of use outside PRACE*

MeTiS/ParMeTiS have been used in the parallelization of large number of applications including mesh partitioning and repartitioning [226] for CFD computations [227] and ocean simulations [228] and in generic libraries such as PETSc [229]. Some of the large-scale applications using ParMeTiS include PetFMM, A dynamically load-balancing parallel fast multipole library [230], uses ParMeTiS for mesh partitioning. SuperLU_Dist uses ParMeTiS to perform parallel symbolic analysis to determine the nonzero structures of L and U [218].

4.7.4 *Pros and Cons*

Metric	**Pros**	**Cons**
Scalability	ParMeTiS is significantly faster than the other successful graph partitioning tools that adopt multi-level paradigm. ParMeTiS can generate partitions with prescribed uneven part weights and thus can specify the target sub-domain weights for each of the sub-domains and for each balance constraint. So ParMeTiS is very suitable for load-balancing on heterogeneous systems.	When the number of parts increases over 32K parts, ParMeTiS is reported to produce empty parts. Currently ParMeTiS does not have threading support. However, this is expected to change in the next release.
Performance	Partitioning quality in terms of edge-cutsize minimization is very good.	Partitioning quality in terms of load balancing is generally inferior to PT-Scotch Currently does not support topology-aware routines for partitioning.
Productivity	Developing codes with MeTiS/ParMeTiS is quite simple.	

	ParMeTiS contains many beneficial distributed graph handling tools such as dual graphs constructing routines.	
Sustainability	Maintained at Karypis Lab of Minnesota University, the tool is upgraded periodically and there seems to be a significant community behind it. Can be called from Fortran and C code.	
Correctness	-	-
Portability		Ports to MIC, ARM, and GPU are not expected to be available anytime soon.
Availability	ParMeTiS source code and binaries are publicly available from http://glaros.dtc.umn.edu/gkhome/	
Resilience	-	-

Table 42 ParMeTiS - Pros and Cons

4.7.5 *Target systems/architectures*

X86	IBM	GPU	MIC	ARM	Heterogeneous
Yes	Yes	No	Unknown	Unknown	No

Table 43 ParMeTis - Target systems/architectures

4.7.6 *Conclusion*

In conclusion, ParMeTiS with its good edge-cutsize reduction, fast partitioning time, multi-constraint support that enables partitioning of graphs for multi-phase and multi-physics computations, uneven partitioning capability that enables load-balancing on heterogeneous systems, and repartitioning and topology-aware routines, shows potential for enabling applications on multi-petascale and future exascale systems.

4.8 PT-Scotch

4.8.1 *Brief overview*

PT-Scotch [231] is a software package that implements the multi-level paradigm to compute parallel static mappings/partitions and parallel sparse matrix block orderings of distributed graphs. (PT-Scotch also contains repartitioning routines, topology-aware partitioning heuristics and supports fixed vertices). PT-Scotch is distributed as free software. PT-Scotch is mainly used for workload distribution in the parallelization of applications on distributed memory architectures [232].

The sequential version of PT-Scotch (called Scotch), is known to scale partition counts of 128K and beyond on a single fat node (128GB RAM) [233]. For graph/mesh sizes of billions of cells, sequential partitioning with Scotch becomes infeasible. PT-Scotch (Parallel Threaded Scotch) developed at LaBRI of INRIA, provides efficient parallel tools to partition graphs with sizes up to a billion vertices, distributed over a thousand processors [234]. PT-Scotch is known to bipartition, in 76 seconds, a 3D graph of more than 2.4 billion vertices and 7.3 billion edges, distributed across 2048 processors [235].

Distinctive features of PT-Scotch as a partitioning tool are as follows: It uses low-memory and provides well-balanced partitions for large number of partition counts. It makes use of both thread parallelism and process parallelism. However, the edge-cutsize quality of PT-Scotch can sometimes be inferior to other parallel partitioning tools such as ParMeTiS. PT-Scotch is runs on shared-memory as well as distributed-memory architectures. It is only dependent on MPI libraries and should be able to run on any parallel architecture without much trouble. PT-Scotch requires MPI and POSIX libraries. It can make use of multiple cores in a node of the parallel system via effective threading. It has been tested on multiple architectures and is known to scale up to thousands of cores.

The new release (v6.0) of Scotch contains multi-threaded, shared memory algorithms in the (formerly) sequential parts of the library. New features such as topology-aware partitioning options are also added into the library in this release. Also PT-Scotch API now exposes many distributed graph handling routines with this release.

In the new release (v6.0) of Scotch, topology-aware partitioning options are embedded into the library. This option is a key feature of success for exascale computing. Unfortunately, to our knowledge, there is still no application that tests and evaluates this feature of PT-Scotch even under petascale settings. This can be considered for a possible enabling work for future T7.2c studies.

PT-Scotch compiles best with GCC compiler but can be compiled with other C compilers. The various routines implemented in ParMeTiS can be accessed from a C, C++, or Fortran program by using the supplied library. It can be run independently so it can be called easily from other programs by means of `system()` or `popen()` system calls, or be piped together on a single shell command line.

Latest release/version: v6.0

4.8.2 *Evidence of use within PRACE*

Scotch and PT-Scotch are already available as loadable modules in CURIE. PT-Scotch has been tested and is known to perform well on PRACE systems such as CURIE, JUGENE and HECToR as well other supercomputing systems such as Argonne‘s IBM Blue Gene/P or IBM POWER 7 systems. The previous versions (5.X) of PT-Scotch partitioning tool has been successfully used in applications such as Code Saturne [223], Vlasiator [217], and Telemac-2D [224]. In most of these applications, both ParMeTiS and PT-Scotch are tried and compared, and the superior load-balancing and inferior edge-cut minimization properties of PT-Scotch when compared with Par-MeTiS have been reported.

4.8.3 *Evidence of use outside PRACE*

PT-Scotch/Scotch has been used in the parallelization of many applications in diverse areas including CFD, Seismology and FEM [235] (in mesh partitioning). Some applications that use PT-Scotch are: CABARET (Compact Accurately Boundary Adjusting high-Resolution Technique) finite volume code that is used for accurately resolving turbulent flow structures

in high-fidelity CFD simulations [236] is tested on HECToR. Scotch was reported to lead to slightly better parallel performance than that of MeTiS. SPECFEM3D is used for simulating forward and adjoint seismic wave propagation on fully unstructured hexahedral meshes of arbitrary shaped model domains [237]. Scotch is used in load balancing parallel simulations in SPECFEM3D.

4.8.4 *Pros and Cons*

Metric	**Pros**	**Cons**
Scalability	Good scalability in terms of parallel partitioning: Can partition on thousands of nodes. Good scalability in terms of graph size: Can partition graphs with billions of vertices and edges. Good scalability in terms of the number of final parts: Can partition into hundred thousands of parts. Good scalability in terms of memory usage when the number of parts increases.	PT-Scotch usually runs slower than ParMeTiS PT-Scotch is not suitable for generating prescribed imbalanced partitions. So PT-Scotch is not very suitable for load-balancing on heteregoneous systems.
Performance	Partitioning quality in terms of load-balancing is very good. Supports topology-aware routines for partitioning.	Partitioning quality in terms of edge cutsize is generally inferior to ParMeTiS
Productivity	Developing codes with PT-Scotch is simple.	Distributed graph handling routines of PT-Scotch are not as versatile as those of ParMeTiS. However, the new release of PT-Scotch contains a number of beneficial routines for this purpose.
Sustainability	Maintained at LaBRI of INRI, the tool is upgraded periodically and there seems to be a significant community behind it.	
Correctness	-	-
Portability		Ports to MIC, ARM, and GPU are not expected to be available anytime soon.

Availability	PT-Scotch source code and binaries are publicly available from http://www.labri.fr/perso/pelegrin/scotch/	
Resilience	-	-

Table 44 PT SCOTCH - Pros and Cons

4.8.5 *Target systems/architectures*

X86	IBM	GPU	MIC	ARM	Heterogeneous
Yes	Yes	No	Unknown	Unknown	Support for hybrid systems

Table 45 PT SCOTCH - Target systems/architectures

4.8.6 *Conclusion*

In conclusion, PT-Scotch with its good load balancing, low-memory usage and scalability properties shows great potential for enabling applications on multi-petascale systems. Similarly, due to its topology-aware routines, PT-Scotch shows potential for enabling applications on future exascale systems, as well.

4.9 NetGen

4.9.1 *Brief overview*

NetGen is a sequential and automatic 3D tetrahedral mesh generator. It can generate volume meshes as well as surface meshes. It is available as LGPL open source software at [238] and has a wide user base. It employs the advancing front technique and also implements mesh optimization to improve mesh quality. A parallelized version of NetGen [239] was developed independently recently by Bogazici University on top of the sequential NetGen using MPI libraries and the C++ language.

Latest release/version: v5.0

4.9.2 *Evidence of use within PRACE*

In the PRACE whitepaper, 'Parallel Mesh Generation, Migration and Partitioning for the Elmer Application' [240] two kinds of methods were implemented in the parallelised NETGEN: (i) Geometry decomposition based method and (ii) Refinement based method. Geometry decomposition methods that use decoupled sequential mesh generators may not be suitable when run on large numbers of processors. This is mainly due to the difficulty of automating geometry decomposition. On the other hand, if a proper geometric decomposition is made, then a high quality mesh generation can be done. Therefore, automatic geometry decomposition can be used on a small number of processors and perhaps with a user assisted geometry decomposition in a semi-automatic manner on larger numbers of cores. Refinement based methods allow for the fast generation of billions of elements on distributed machines. In the investigation reported in the whitepaper an example was run on the CURIE system,

where 1.4 billion elements were generated in approximately 50 seconds on 1000 compute cores. Mesh migration routines were implemented to redistribute mesh data structures.

4.9.3 *Evidence of use outside PRACE*

We have found no evidence of NetGen being employed outside PRACE to date.

4.9.4 *Pros and Cons*

Metric	Pros	Cons
Scalability	Refinement based methods are scalable. Mesh migration routines implemented also scaled. Tests using up to 1K cores were performed.	Geometry decomposition based methods are not scalable
Performance	There is inverse relationship between mesh quality and parallel runtime/scalability performance.	
Productivity	Callable from Fortran and C	Installation of NETGEN on supercomputer systems requires a lot of time mainly due to problems related to installation of specific X libraries.
Sustainability	Sequential NETGEN was first released in 2003. It has been improved over the years. The latest version 5.0 was released in Nov. 2012.	Parallel NETGEN was developed on top of sequential NETGEN during PRACE-1IP WP7.6 Elmer application support activity in a short period of time. It has not been worked on since the end of WP7.6 task.
Correctness	Sequential version works robustly and is used worldwide by many users.	Parallelized version has not been tested exhaustively.
Portability	UNIX systems	Sequential NETGEN uses some specific X libraries which caused porting problems to IBM Blue Gene/P JUGENE system
Availability	Both sequential and parallelized versions are open and available under LGPL license	
Resilience	-	

Table 46 NETGEN - Pros and Cons

4.9.5 *Target systems/architectures*

X86	IBM	GPU	MIC	ARM	Heterogeneous
Yes	No	No	No	No	No

Table 47 NETGEN - Target systems/architectures

4.9.6 *Conclusion*

Parallel mesh generation offers a solution for cases where mesh problems with greater than 10^9 elements will not fit the memory of a single node and the time to generate the mesh becomes significant. The parallelized version of NETGEN implements an "owner updates"-based mesh migration algorithm [239], which has demonstrated good parallel performance within PRACE [240]. The usual features expected from exascale libraries such as communication reduction, error resilience and fault tolerance must be addressed in the future. A possible suggestion is that such issues can start to be addressed by developing a non-MPI, workflow version of Bogazici University's NETGEN-based parallelized mesh generation routines. A workflow system like Falkon [241] can be used for this purpose.

5 I/O Management Techniques

In this section, we characterize the following I/O Management tools that are of interest to T7.2 in particular, and the European HPC community more widely, as we move towards the deep petascale and exascale eras:

- HDF5
- PNetCDF
- XIOS
- ADIOS
- SIONlib
- Darshan

While the first five tools listed above can be classified as high-level I/O libraries, the final tool in the list, Darshan, is an I/O profiling tool. For each tool, we provide an overview and discuss the tool's present state, how it has been employed in PRACE to date, how it has been employed more widely, and our views on the suitability of the tool for enabling PRACE application codes during the exploitation phase of T7.2.

The increasing data needs of scientific and engineering applications mean that the problems associated with storing, reading, analysing and sharing large amounts of information are becoming more relevant to a wider user community within PRACE and will become even more so on the road to exascale [242]. While the performance gap between file systems and compute systems is well known, during our surveying we have found that users within PRACE have in general not been able to squeeze as much performance from existing parallel file systems as they have from computational hardware. For example, in real applications, the use of high-level, parallel I/O libraries such HDF5 [243] and PnetCDF [244], which we report on here, often only reach write speeds that are not much higher than the performance of a single hard disk. It is becoming increasingly recognised that in many cases, this problem has less to do with the capabilities of the software and hardware, and more to do with the myriad of software layers, tunable parameters and different file system settings that users are confronted with.

While several very promising parallel tools have emerged within scientific communities that traditionally deal with post-processing of huge datasets (e.g. NCO [245] CDO [246], FastBit [247]) these tools are quite domain-specific and are consequently not discussed in more detail here. However, at the same time, we feel that PRACE should be inspired by such initiatives within these domains and aim to become an infrastructure that supports the complete workflow from numerical experiment to publication, where I/O tools are an integral part of that workflow. To reinforce this message, we feel it appropriate to quote from the IESP roadmap [248] on this topic:

"Traditionally, I/O has been considered as a separate activity that is performed before or after the main simulation or analysis computation, or periodically for activities such as check-pointing, but still as separate overhead. File systems, which have mainly been adapted from the legacy (sequential) file systems with overly constraining semantics, are not scalable. I/O should be considered an integral activity to be optimized while architecting the system and the underlying software."

Several tools that we mention here are available on current PRACE systems, while some are still under development and aim to overcome some of the bottlenecks that are related to I/O on multi-petascale and future exascale systems. While we have naturally included ease of use for post processing and visualization as important characteristics in considering the tool's exploitation within T7.2, the most important prerequisites that we consider are the potential for scalability and performance on current PRACE Tier-0 systems as well as future multi-petascale/exascale systems. With the IESP view in mind, deeper investigations into extracting performance (with real applications) from parallel file systems will be the main focus of this subtask during the exploitation phase of T7.2.

5.1 HDF5

5.1.1 *Brief overview*

HDF5 [243] is a library used to read and write platform-independent files. The generic file format includes meta-data for all variables to produce self-describing files. The metadata can describe different aspects of the variables: authorship, type, shape, creation date, etc. This metadata functionality is shared with other, high-level file formats like NetCDF and BP. It supports large-scale parallel file systems (e.g., Lustre and GPFS) by using MPI-I/O calls for parallel file access, but can also be used without MPI-I/O as a serial library. The library is well maintained and there is an on-going research effort to prepare HDF5 for exascale [249]. The library is freely available as open source and runs on all PRACE platforms. The HDF5 library contains a nice suite for conversion, and compression and it is a suitable file format to be post-processed by e.g. visualization tools.

Latest version/release: 1.8.10-patch1

5.1.2 *Evidence of use within PRACE*

The HDF5 library is part of the PRACE 'Common Production Environment' and is therefore available across all PRACE Tier-0 systems. The library is used across a wide range of scientific domains (astrophysics, multi-physics, CFD) and has been used widely within PRACE to date:

The HDF5 library was implemented for the static grid version of the PLUTO astrophysical fluid dynamics code [250]. Unfortunately, it was found to be significantly slower than synchronous I/O on JUGENE, which was probably due to overhead or the incompatibility

between GPFS and HDF5. The performance is approximately 0.5GB/s for 500-10k cores. However, the introduction of the HDF5 file format for a static grid represents an improvement for PLUTO both in term of portability and also for post-processing and visualization.

In other work in PRACE-1P, a parallel implementation of HDF5 for the computational mechanics code, ALYA [251], was compared to the original, serial I/O method on both CURIE and JUGENE. Results showed an improvement in performance of over a factor of 3x in some cases.

Another interesting study within PRACE-1IP [252] compared the performance of different I/O methods and found that POSIX I/O (one file per process) was in almost all cases by far the fastest method on CURIE, with MPI-I/O catching up at higher core counts (4096 cores), while HDF5 and pNetCDF performed much more poorly in comparison.

The HDF5 library has also been adopted in the ENZO [253] and LB3D codes [254]. The LB3D code developers reported that they can write a multi-gigabyte dataset in the order of a second, which is negligible to the computational effort. Data can be compressed using internal filters, so one can still access the data without uncompressing, transparently.

5.1.3 *Evidence of use outside PRACE*

HDF5 is widely used outside PRACE and is deployed on multi-petascale systems throughout the world. HDF5 is the most commonly used parallel I/O library in both DOE SC and DOE SciDAC applications and there is a very active research effort in the US focused on preparing HDF5 for future exascale systems. The ExaHDF5 [249] project aims to enhance HDF5 for Exascale platforms. Its goals are aimed at improving HDF5 performance on existing platforms by removing collective restrictions for metadata modifications, adding metadata and raw data indexing, adding support for asynchronous parallel I/O, designing and implementing file system autotuning mechanisms and supporting “ordered updates” in parallel. Early efforts have already demonstrated 8x-10x improvement in speedup and scaling to 32,000 processors [255].

5.1.4 *Pros and Cons*

Metric	**Pros**	**Cons**
Scalability	Better performance than master-I/O because of less communication.	Scalability is usually poor without a good understanding of the HDF5 structure and the underlying file system.
Performance	Performance can be up to 27GB/s (using 120k cores), or 90% of the theoretical peak.	Can be as low as 200MB/s (using a few thousand cores without further scalability)
Productivity	The library is platform-independent and the files contain metadata to describe them. Many advanced analysis programs support reading and writing HDF5.	
Sustainability	HDF5 is well supported and widely used. Several projects	

	are underway to improve HDF5.	
Correctness	Release versions are available on the web and are often included in linux distros.	
Portability	Available on all PRACE platforms.	
Availability	Open and royalty-free.	
Resilience	Work is ongoing to improve fault-tolerance by changing the software and the file format.	

Table 48 HDF5 - Pros and Cons

5.1.5 *Target systems/architectures*

X86	IBM	GPU	MIC	ARM	Heterogeneous
Yes	Yes	N/A	N/A	N/A	N/A

Table 49 HDF5 - Target systems/architectures

5.1.6 *Conclusion*

Several efforts by the US DOE are on-going to improve the performance of HDF5 on multi-petascale machines and to adapt the library for exascale. The HDF5 library has shown the ability to scale and perform well enough for multi-petascale simulations. The advantages of HDF5 are that files in the HDF5 format are platform-independent and suitable for high-quality visualizations using the VisIt and ParaView tools. It is the only file format that allows for complex data types like structures and a hierarchical organization of variables. The major downside is the disparity between the presented performance results for some applications and the achieved results in some of the PRACE projects. We believe that one interesting area of focus during the exploitation phase of T7.2 could be on the tuning of applications with HDF5 on parallel file systems of specific PRACE platforms.

5.2 PNetCDF

5.2.1 *Brief description*

Parallel NetCDF (PNetCDF) [244] is a library that implements the Unidata NetCDF3 file format, but provides parallel I/O capabilities using MPI-I/O. The API is an extended version of the NetCDF3 [256] API that supports C and Fortran. The NetCDF3 standard defines a file format for array-oriented data that is platform-independent and self-describing. A file contains dimensions, variables and attributes. Files can be appended efficiently with the use of an unlimited axis, for example to add another snapshot periodically. Many tools can read the NetCDF3 file format and can often interpret the data if the metadata for the variables uses the right naming and attribute conventions (e.g. wind speed on a longitude-latitude grid). The format provides enough flexibility for most scientific applications. Due to this combination of

functionality and simplicity, it is very popular in the climate community. It is not compatible with the NetCDF4 standard. The library is freely available as open source.

Latest version/release: v1.3.1

5.2.2 *Evidence of use within PRACE*

The PnetCDF library has primarily been used in US-based weather (WRF) and climate (CCSM/CESM) models, but only received minimal adoption in European models. The European exception is the RAMSES astrophysics code, which implemented several different output options, including PnetCDF. The molecular dynamics package AMBER also included an output option for PnetCDF, but this option seems to be little used. The PnetCDF library is probably mostly used by the PRACE allocations that use the US-based weather and climate models for their simulations.

The IO performance of PnetCDF was tested on the CURIE system using both the IOR benchmark suite and the RAMSES astrophysics code [252]. For the RAMSES code, the performance is usually well below 1 GB/s. Only for read operations with the `romio_cb_read` (flag to control collective buffering) MPI-I/O file hint disabled, does the read performance reach almost 2 GB/s. The read performance with IOR is also optimal when using this hint for HDF5, MPI-IO and PnetCDF, reaching more than 12GB/s for PnetCDF. This shows that the right settings are critical to reach a good performance, which is probably very system-dependent.

5.2.3 *Evidence of use outside PRACE*

Since the PnetCDF library was developed in the US, most of the use cases are also for US-based models.

In a study for the optimization of I/O for the FLASH code [257]. MPI derived datatypes were used to write non-contiguous parts of the memory to a contiguous PnetCDF file using only one call to the PnetCDF library. The routine `MPI_Type_create_subarray` was used to exclude guardcells (or “ghost” or “halo” cells), while the routines `MPI_Type_indexed` and `MPI_Type_create_resized` were used to combine mesh variables at different memory locations into one variable for writing. Furthermore, they implemented “non-blocking” I/O [257], which combines writes from different calls to the PnetCDF library.

The I/O in the GCRM (global cloud-resolving model) application was optimized using PnetCDF on the Cray XE6 system “Hopper” using Lustre at NERSC [258]. By saving all variables into one file, a maximum performance of 24GB/s was reached using 40,960 MPI processes.

5.2.4 *Pros and Cons*

Metric	Pros	Cons
Scalability	Excellent scaling up to 40k cores for some applications	Scalability is usually poor without a good understanding of the PnetCDF structure, the MPI-IO library and the underlying parallel file system.

Performance	Performance can be up to 24GB/s,	Can be as low as 200MB/s.
Productivity	The library is platform-independent and the files contain metadata to describe them. Many advanced analysis programs support reading and writing the NetCDF-3 format.	
Sustainability	PnetCDF is developed at Argonne National Laboratory (ANL) and is supported software.	
Correctness	Release versions are available on the internet. Development branches can be checked out with svn.	
Portability	Available on some PRACE platforms (e.g. HERMIT and MareNostrum).	
Availability	Open and royalty-free.	
Resilience		

Table 50 PNetCDF - Pros and Cons

5.2.5 *Target systems/architectures*

X86	IBM	GPU	MIC	ARM	Heterogeneous
Yes	Yes	N/A	N/A	N/A	N/A

Table 51 PNetCDF - Target systems/architectures

5.2.6 *Conclusion*

It has been demonstrated that the PnetCDF library can reach a performance that is close to the theoretical peak of the file system on petascale systems, using tens of thousands of MPI tasks. It gives the user a range of tuning options to reach a better performance. The flat file format is one of its strengths, but also a major weakness. Models that use regular, structured grids could benefit from this library, and a large collection of analysis tools can readily read and interpret the output files. The NetCDF format is less well suited for use with VisIt and ParaView. It is less well suited to modern numerical methods like non-structured grids and adaptive mesh refinement.

5.3 XIOS

5.3.1 *Brief description*

XIOS [259] is an I/O library with integrated processing in development by the NEMO ocean model team. It is a library that allows flexible, parallel I/O by using XML files to define the variables that should be written to file. In the XML file, operations (e.g. time averaging and slicing) can be defined for variables. Output variables are processed locally and then sent to separate I/O tasks to be written to disk. The extra library in-between the user and the disk system is an extra complication in case of performance problems. The library is written in C++ and provides an API for Fortran-90. XIOS uses the NetCDF-4 format and library, which is in turn built on top of the HDF5 library and its dependencies. Although the NetCDF-4 format supports hierarchical data structures and complex variables, it is not clear if these are supported by XIOS. Other file formats could be used, but are not yet implemented (XIOS has recently been implemented in the ocean model NEMO 3.4). Different processing capabilities are in development, e.g. spatial operations and adding or multiplying two variables. The long-term plans for XIOS include the possibility for input e.g. to manage asynchronous reading. The library is freely available as open source and runs on several PRACE platforms.

Latest version/release: v1.0 (subversion)

5.3.2 *Evidence of use within PRACE*

The XIOS library has not been used widely to date within PRACE, but interest in the library is increasing and it is currently being used within the WP8 PRACE-2IP work package. XIOS is under development by ICHEC where a "memory proxy" XIOS server is being developed to enable XIOS performance on nodes with low memory, by using extra nodes as buffers between the compute and I/O nodes [35].

5.3.3 *Evidence of use outside PRACE*

The Pulsation project runs a high-resolution configuration of the NEMO ocean model in combination with the WRF atmospheric model on the CURIE system. The XIOS library has been used for output with 128 XIOS servers and reached a writing performance of about 3.5GB/s when writing in parallel to a single file [260].

5.3.4 *Pros and Cons*

Metric	Pros	Cons
Scalability	The number of parallel I/O tasks can be freely chosen to balance performance, communication and node usage. Data movement is reduced through on-line, client-side data processing	
Performance	Performance is proven up to 3.5GB/s on the CURIE PRACE platform.	Options to tweak performance of the underlying HDF5-library are not directly available.

Productivity	I/O can be easily adjusted through an XML-file. Output is in NetCDF4, which is widely supported.	Library is now targeted to climate and weather codes.
Sustainability	XIOS is developed and adopted by the NEMO team. Plans for wider adoption in climate and weather codes improve sustainability.	Code is mostly commented in French.
Correctness	Output files are well-structured and contain the necessary metadata.	
Portability		For now only tested on the CURIE system with Lustre.
Availability	Open and royalty-free. Code is available through an svn repository.	
Resilience		

Table 52 XIOS - Pros and Cons

5.3.5 *Target systems/architectures*

X86	IBM	GPU	MIC	ARM	Heterogeneous
Yes	Yes	N/A	N/A	N/A	N/A

Table 53 XIOS - Target systems/architectures

5.3.6 *Conclusion*

One of the greatest advantages of XIOS is its flexibility to change the output of variables of a simulation without recompiling: writing only sub-regions, changing the frequency of output per variable or not writing a variable at all can be done through editing an XML file. This seems to be especially valuable for models with many variables and many users, so output can be filtered for each purpose. Most exascale reports mention the need for data reduction through online post-processing, which is one of the main advantages of this library. The XIOS library uses the NetCDF4 layer for its I/O and it therefore inherits its strong and weak points. However, unlike NetCDF4, it allows a user to easily vary the number of processes for I/O, circumventing the I/O scalability problems when using master I/O or I/O from every task.

5.4 ADIOS

5.4.1 *Brief description*

Essentially, ADIOS (Adaptable I/O System) [261] is componentization of I/O transport methods. It uses an XML-file to describe the data and the layout of variables in the code. ADIOS then translates this XML-file into include files for the application. The I/O can be changed by modification of the XML-files and recompiling the application with the newly translated 'include' files. The approach is less flexible than XIOS, due to the needed

recompilation. Different I/O formats and libraries can be used: parallel HDF5 or NetCDF4, one file per process, MPI-IO, the own BP format or the data can be directly connected to advanced visualization software. The own BP file format is specially optimized for HPC workloads and is resilient to failures in compute nodes and the file system. The library contains routines to give hints that can be used to optimize asynchronous I/O operations. I/O is throttled to be finished in time for the next phase of I/O, but doesn't need to be faster. This can optimize the overlap of computation, communication and I/O. Furthermore, on most systems the file system is shared between many jobs and it is important to use the resource as efficiently as possible, and still keep an interactive response.

Latest version/release: v1.4.1

5.4.2 *Evidence of use within PRACE*

We found no evidence of ADIOS being used within PRACE to date

5.4.3 *Evidence of use outside PRACE*

The combustion code, S3D, uses ADIOS [262]. The fusion particle code GTC [263] runs on more than 120,000 cores on the Jaguar system. The implementation of the ADIOS library doubled the performance of the GTC model with a write performance of 80GB/s and a read performance close to the peak of the I/O system. The developers of the SPECFEM3D GLOBE model are busy implementing ADIOS [264] to reduce the writing of 40 files per process to 1 file per process, to reduce the burden of metadata operations

5.4.4 *Pros and Cons*

Metric	Pros	Cons
Scalability	Used for simulations on 120,000 cores.	No online post-processing capabilities to reduce data.
Performance	Great read and write performance (80GB/s on Jaguar XT5) is easy for the internal BP format,	Problematic for all other formats (NetCDF4, HDF5).
Productivity	Easy to implement. BP files can be read by VisIt (from v2.0). Fortran90, Java and NumPy bindings. Told to convert to HDF5 and NetCDF4.	Besides VisIt, no tools can natively read BP files.
Sustainability		
Correctness		
Portability	Fully supported on IBM BG/P, Cray XT, Linux clusters and Mac OSX.	
Availability	Open and royalty-free. Source code is available online. Installed on e.g. the	

	CURIE system.	
Resilience	ADIOS's BP file format is resilient to failures in the compute nodes and the file system.	

Table 54 ADIOS - Pros and Cons

5.4.5 *Target systems/architectures*

X86	IBM	GPU	MIC	ARM	Heterogeneous
Yes	Unknown	N/A	N/A	N/A	N/A

Table 55 ADIOS - Target systems/architectures

5.4.6 *Conclusion*

The ADIOS library has been shown to reach excellent performance for petascale applications when the internal BP format is being used. Other file format backends will have different performance and scalability characteristics that are also described in this report. If the limited support by other tools of the BP file format is no problem, the ADIOS library could be a good candidate. The ADIOS library is the only library that has a file format with resilience features built into production software. As with most other software except XIOS, data reduction by online post-processing is missing from the ADIOS library, which for the moment seems to be an essential part of exascale I/O.

5.5 SIONlib

5.5.1 *Brief description*

SIONlib [265] is a scalable I/O library for the parallel access to files local to every process. The library not only supports writing and reading binary data to or from several thousands of processors into a single or a small number of physical files but also provides for global open and close functions to access SIONlib files in parallel. SIONlib provides different interfaces: parallel access using MPI, OpenMP, or their combination and sequential access for post-processing utilities. Each process involved in I/O gets assigned a number of file blocks to which it has exclusive access. The explicit allocation of file blocks to processes eliminates performance problems due to file locking. Another advantage of the SIONlib library is that POSIX I/O calls can be used as-is and the code only needs to be changed to open and close files and to ensure that enough blocks are available for the I/O that each process wants to write. Only the open and close calls are collective operations, while all other operations can be done independently or even asynchronously.

Latest version/release: v1.3p5

5.5.2 *Evidence of use within PRACE*

The SIONlib library has been used in a PRACE project with the MP2C code [266] although there is no further discussion why this library was chosen, or what its performance is like.

5.5.3 *Evidence of use outside PRACE*

An in-depth discussion of MP2C and SIONlib can be found in [267]. The original application used master I/O, which is inherently unscalable. The advantages of SIONlib were the relatively small code changes needed and its resulting performance. The SIONlib library has been implemented in several other models mainly for checkpoint-restart files and sometimes also for result files and post-processing [268]

5.5.4 *Pros and Cons*

Metric	**Pros**	**Cons**
Scalability	Expected good scaling. Reduction of excessive file metadata operations.	
Performance	Read performance of 35GB/s, write performance of over 25GB/s on JUGENE.	
Productivity	Applications using POSIX or ANSI-C I/O don't need to be rewritten drastically to profit from parallel I/O systems. Both Fortran and C supported.	Files are not platform-independent, no variable metadata, not structured and not supported by any analysis tools.
Sustainability	FZJ and German Research School for Simulation Sciences (GRS) maintain this library.	
Correctness	Release versions are available on the internet.	
Portability	Can be used on block-based file systems, e.g. Lustre and GPFS.	Limited availability. Available on JUQUEEN and JUROPA.
Availability	Open and royalty-free.	
Resilience	Usually less relevant, files are written once, but not updated.	

Table 56 SIONlib - Pros and Cons

5.5.5 *Target platforms/architectures*

X86	IBM	GPU	MIC	ARM	Heterogeneous
Yes	Yes	N/A	N/A	N/A	N/A

Table 57 SIONlib - Pros and Cons

5.5.6 *Conclusion*

The SIONlib library avoids potential performance bottlenecks by explicating the blocked nature of the filesystem. The SIONlib library is well suited for temporary files and situations without an established tool-chain for post-processing that expects certain file formats. Another advantage is that applications already using POSIX I/O calls don't need to be rewritten radically to parallelise their I/O. As noted in the introduction, the IESP Roadmap expects check-pointing as a technique likely to continue on exascale systems. The SIONlib library seems ideally suited for this purpose. Checkpoints are usually written in one go and resilience of the library is therefore less critical. As soon as a checkpoint is written successfully, older checkpoints can be removed.

5.6 Darshan

5.6.1 *Brief description*

Darshan is a light-weight profiling tool that can be used to characterize I/O-load at petascale [269]. It gives an accurate picture of the I/O-access pattern, how read/write and metadata operations are performed by the application. Further, it characterizes the access pattern within file and files, describing whether the access is MPI-based or POSIX. All this is done with a minimum of overhead. Reduction, compression and storage are performed at the moment when `MPI_Finalize` is called. Darshan is designed to reflect application I/O behaviour while being transparent to users. Since it has the ambition to be a Petascale characterization tool, it also must have strong scaling properties. It is implemented as set of user space libraries. These are linked into the application during the linking phase. The application's I/O-calls are substituted with calls to the darshan libraries. No application source code modification is necessary. During execution of the application, Darshan collects statistics that are stored in a file record per process. As the application ends its computation by a call to `MPI_Finalize`, a shutdown routine is executed to gather all the Darshan file records. The overhead introduce is negligible as profile data produced is small on smaller computations. On larger computations (petascale), the time used for handling Darshan profile data is small compared to the overall shutdown time of the computation. Accordingly, Darshan does not influence the computation. The profile achieved will be representative for the application's I/O behaviour. Darshan is supported on the IBM Blue Gene and the Cray XE6 platforms, but it works on most Linux platforms. It supports the PGI, Cray, Intel and GNU compilers as well as static and dynamic linking [270]. This tool does not have an API but depends on MPI. It therefore works with C, C++ and all Fortran flavors.

Latest version/release: 2.2.4

5.6.2 *Evidence of use within PRACE*

Darshan has been used in several studies within PRACE to date. Darshan was used to better understand the I/O of the OpenFOAM application [271] and the I/O of the EC-EARTH climate model has also been analysed with Darshan [272]. A climate model often consists of multiple components (e.g. atmosphere and ocean) that are each developed independently. Each component can consist of hundreds or thousands of subroutines and it is not easy to find the I/O strategy that has been used in each component. Darshan was proven to be a valuable tool for finding the non-trivial pattern of I/O in this project [272].

5.6.3 *Evidence of use outside PRACE*

As Darshan was specifically designed with petascale systems in mind, it has been used on several large-scale systems throughout the world.

In a study for the optimization of I/O for the FLASH code [270] the Darshan tool was used to measure the access size when using the PnetCDF library. The results show that there is a shift from writes of 4MB when using the standard file layout to writes of 16MB when using an experimental file layout. Larger disk writes are more efficient for the file system. The authors suggested that the Darshan tool could give more information about the two-phase collective I/O optimisations specifically the time spent in I/O versus the amount of time spent re-arranging the data. The Darshan report gives an overview of the files that are created or read, the amount and size of data and metadata operations during the simulation and also which tasks are involved.

5.6.4 *Pros and Cons*

Metric	Pros	Cons
Scalability	Negligible overhead for I/O intensive jobs with 65,536 processes.	
Performance	Negligible overhead, even for very large file counts.	
Productivity	Quick overview of the amount of I/O, files and the processes involved.	Little information on the I/O behaviour in time. Only for MPI applications.
Sustainability		Developed at ANL, Financed by a DOE project. Not known what happens when the project ends.
Correctness	A release version is available, but also the latest unstable branch can be checked out with svn.	
Portability	Supported on BlueGene and Cray platforms. Intel linux clusters usually work as well.	
Availability	Source downloadable from internet.	Compilation needed.
Resilience		The application needs to complete successfully to be able to generate a report.

Table 58 Darshan - Pros and Cons

5.6.5 *Target platforms/architectures*

X86	IBM	GPU	MIC	ARM	Heterogeneous
Yes	Yes	N/A	N/A	N/A	N/A

Table 59 Darshan - Target systems/architectures

5.6.6 *Conclusion*

Darshan has been developed with petascale systems in mind and its performance and scalability is therefore excellent. Its statistical analysis is also useful at scale. It is useful for applications with an unknown I/O profile, to get a quick overview of when, where and how much I/O is done. Another use case is the investigation of parallel I/O, to see how well the I/O is coalesced into larger chunks. Resilience could be improved by relaxing the condition that the application needs to complete successfully. One useful and relatively easy improvement could be some subroutine calls that can be added to the application by the developer to start the analysis and generate the report at arbitrary points in the code.

6 Summary

The survey has covered four separate topics that we consider relevant to enable applications on current multi-petascale systems. We summarize our findings separately by topic: programming languages and standards, debuggers and profilers, scalable libraries and algorithms and I/O management techniques. Conclusions for each individual tool can be found in the invidual reports, so here we list only what we think are the most salient points when considering the tools for enablement.

Programming Languages and Standards:

As part of this report we have surveyed thirteen individual programming languages and standards and report on how they have been used in PRACE to date.

- It can be seen from our brief report that the MPI forum is starting to address the challenges that MPI will face on this road, and we believe that the exploitation phase of T7.2 provides PRACE partners with a very valuable opportunity to investigate new MPI 3.0 features.
- OpenMP offers the easiest means of hybridising existing MPI-based codes, a model that is becoming increasingly important as the core-count on nodes continues to increase. With the advent of Intel's Xeon Phi coprocessor, OpenMP is already finding new target architectures in the many-core space, and could become even more relevant as a standard if plans go ahead to merge OpenACC into OpenMP 4.0 in the near future.
- Since some of the new PRACE prototypes will consist of the latest K20 GPUs we see the exploitation phase of T7.2 as a great opportunity to enable applications to exploit the full compute resources of these new platforms using some of the new features being offered by NVIDIA CUDA 5.0, including 'Hyper-Q' and 'Dynamic Parallelism'
- Developing efficient OpenCL code is typically found to require more effort than other GPU frameworks. There are some efforts, besides the new SDKs and tools for code debugging and analysis, which try to address this issue. Its relevance may change with the release of the Xeon Phi architecture and the support that Intel puts behind it and for this reason we feel that T7.2 should liaise closely with European exascale projects such as Mont-Blanc project to learn how OpenCL will be exploited there during 2013.
- The high-level nature of TBB is probably not a feature that will attract WP7 partners looking to extend or improve existing codes. Most likely it is the lower-level ideas that might be important. While most reviews of TBB have generally been made in the context of Intel Xeon-based platforms, it might be worth considering the potential benefits of TBB for the new Xeon Phi coprocessor. An initial port from OpenMP to TBB might well be straightforward and worth investigating further, particularly on Xeon Phi-based systems.
- Although interesting, particularly with the Xeon Phi architecture in mind, Cilk Plus does not seem to provide much advantage over OpenMP at the moment. In comparing the keywords in Cilk Plus and the directives in OpenMP, it is clear that the ease of programming is not a concern for either, with OpenMP providing additional options in scheduling and allowing for NUMA effects in some variations. In this sense, Cilk Plus is considerably more limited than OpenMP. It is, however, worth keeping in mind the success of the novel Cilk Plus/UPC combination that was reported on in PRACE-1IP.
- Due to its relative ease of use in comparison to both CUDA and OpenCL, OpenACC is becoming an increasingly popular model for porting legacy applications to GPU-based systems. The standard is still in its infancy and there are many issues with

regards to implementation that still need to be resolved, which also make it difficult to assess the performance of the model. While it is generally appreciated that CUDA offers the ability to perform lower-level optimization for the GPU, OpenACC may be increasingly used as a means of efficiently probing the potential benefits of porting to GPUs, with CUDA being used in an optional second optimization stage.

- The improvements to both the programming model and runtime in OmpSs make it easier to port real applications without substantial re-engineering. However, OmpSs is still at an early stage of development and no results showing petascale performance have yet been published. The ability to manage parallelism across heterogenous architectures consisting of CPUs and accelerators in a transparent fashion will be necessary on deep petascale and exascale systems. OmpSs is an integral part of the DEEP and Mont-Blanc architectures for a future exascale system and we suggest that if OmpSs is to be exploited in T7.2, partners should work closely with both of these projects.
- While we have seen evidence of the exploitation of MPI/PGAS hybrid models, we have found it quite difficult to ascertain whether there has been any genuine benefit to such approaches over the more conventional MPI/OpenMP approach. If such an approach is to be pursued during the exploitation phase of the T7.2, then we recommend that WP7 partners work closely with European exascale projects, where deeper investigations into such hybrid models are already underway as part co-design initiatives on production codes.

Debuggers and Profilers:

As part of this report, we have surveyed 14 debugging and profiling tools. We have found that all of the European exascale projects are concentrating effort into tools for debugging and performance analyses.

In some respect, we feel that the DEEP project provides a model for how the exploitation phase should be conducted within T7.2. DEEP has analysed the space weather application iPIC3D. With the tool Scalasca, the behaviour of the different parts of the application has been identified. Some of these parts can be accelerated by being partly moved to the DEEP-architectures “Booster” part.

We also feel that T7.2 should work closely with tool developers to learn how they can be used more effectively to enable applications within WP7, particulary in extreme cases.

- The TAU development team usually ensures that TAU is available at early stages of new platforms, as exemplified with the Cray Cascade prototype. TAU is being further developed to support new threading technologies, like the new generations of NVIDIA GPUs and Intel Xeon Phi. With its long traction, and continued support from its funding bodies, it is strongly expected that TAU will continue to be available for forthcoming multi-petascale systems. In this sense it also shows potent
- Scalasca has shown that it is an applicable profiling tool when considering the largest scales currently possible. It scales in its use from 1000 cores to close to 300,000 cores. Development of Scalasca continues with the purpose of meeting the needs of the HPC community as exascale technology make its inroads.
- While evidence of multi-petascale use of Vampir is hard to come by, at least in a PRACE context, the tool will be further developed. In the CRESTA-project, ZIH and other partners will jointly develop the scalable measurement environment used by Vampir as an Open Source project.
- TotalView is a professional debugging tool that specifically is aimed at the High Performance Computing market. It is designed for debugging programs running on very large supercomputers and has been successfully tested on 768,432 processes until

now. TotalView state that they are working closely together with IBM to provide debugging facilities on IBM's Blue Gene systems, so petascale debugging is available today. Although TotalView's current feature-set will not be sufficient on an exascale system, RogueWave is actively working on new features such as fault-tolerance as part of co-design teams within the US DOE.

- DDT now has features that are specifically targeting debugging of petascale simulations. The response times of DDT are now short enough for making petascale debugging practically possible and the GUI has features that are specifically designed for giving an overview of large amounts of data as well as the state of a large numbers of threads/processes. The fact that the developers of DTT have continued to show a quick response to the fast pace of changing hardware on large-scale heterogeneous systems, indicates that DDT will feature heavily as a debugging tool on the road to exascale. Although DDT's current feature-set will not be sufficient on an exascale system, Allinea is actively working on new features such as fault-tolerance as part of co-design teams within the US DOE and European exascale projects.
- Intel tools are excellent for debugging and profiling Intel Xeon and more recently Xeon Phi platforms on small scale, but there is very little evidence of their use on large mult-petascale systems and what Intel's long term aims are for their tool sets on such systems.

Scalable Libraries and Algorithms:

As part of this report we have surveyed a representative collection of libraries and techniques that currently garner much interest both within and outside PRACE. As a consequence of the move towards large multi-petascale heterogeneous systems, there is an increasing demand for new and improved scalable, efficient, and reliable numerical algorithms and libraries that confront existing and upcoming complexities associated with such systems, including complex memory hierarchies, the overhead of data movement and fault tolerance.

- In terms of dense solvers, we feel that ELPA shows real promise and should be investigated further as an alternative to ScaLAPACK within PRACE applications. We also believe that MAGMA is one of the most promising libraries containing dense direct solvers with impressive performance and indications of long-term sustainability. The library targets all accelerators/coprocessor architectures and is also fully portable in its OpenCL form Distributed-memory versions of the library are also currently in progress and should be investigated as alternatives to ScaLAPACK, possibly during the exploitation phase of T7.2
- In terms of sparse solvers, MUMPS appears to be a very robust and efficient direct solver for medium-sized distributed or centralized sparse linear systems arising for instance from discretization of PDE problems. Regarding large-scale problems, MUMPS will not be usable as a standalone solver of the original linear system. However, MUMPS will still be a very important tool for the robust and efficient solution of auxiliary medium-sized distributed and centralized sparse linear systems arising in higher level methods like FETI domain decomposition methods, and will in turn extend their scalability. MUMPS has many unique features such as the detection of null pivots, rank deficiency, etc. that can be very helpful in higher-level scalable methods.
- While both SuperLU_DIST and SuperLU_MCDT show promise as dense sparse solvers, like all the other libraries mentioned here, improvements, such as synchronization reduction, data movement minimization and fault tolerance need to be included in the library in order for SuperLU_MCDT to enable applications on future multi-petascale and exascale systems. An interesting alternative to investigate is the PDSLin library.

- Iterative methods, particularly Krylov subspace-based methods are fundamental algorithms to many PRACE applications and have therefore received a considerable amount of attention within PRACE projects to date. Very interesting exploratory work into communication avoiding Conjugate Gradient algorithms has recently been carried out in PRACE-2IP (WP12) and is an active focus of research in many exascale projects in both Europe and the US. With new implementations of the MPI 3.0 standard available (and on the horizon), which now offer non-blocking communications, there is good reason for PRACE partners to investigate these forward looking algorithms in more detail during the enablement of applications on PRACE systems in T7.2.
- As the CRESTA investigations into FFT methods point out, the major bottleneck in distributed FFT implementations is the `MPI_ALLTOALL` communication overhead, which becomes particularly problematic on large core counts. This problem can be alleviated somewhat by using hybrid MPI+OpenMP methods, but further optimisations are badly needed, as FFTs form the basis of many popular applications running on large-scale PRACE systems. In order to push FFT methods further along the road to exascale, we agree with CRESTA that further investigations should be made into non-blocking collective communications within in-code FFT implementations, possibly via the new MPI 3.0 standard features now becoming available. If FFT methods are to be investigated further in T7.2, we recommend that PRACE partners work closely with CRESTA and other European exascale projects to learn more about how their efforts in this area are progressing and if any non-blocking implementations are ready to be exploited in real applications.
- PETSc has strong potential for enabling applications on future multi-petascale and exascale systems, but several features expected from exascale libraries need to be implemented in the library (communication reducing algorithms, error resilience, fault tolerance etc.) As PETSc interfaces with several widely used numerical libraries (MUMPS, Hypre, SuperLU, METIS), its scalability is strongly dependent on the scalability of these libraries also. It is interesting to note that CRESTA chose the PETSc library as being representative of iterative methods and was chosen for further research on exascale co-design efforts. Indications are that PETSc is well supported by the DOE in the US and will continue to make inroads on building in features that will be important on the road to exascale.
- Although we could not find any evidence of Trilinos being used in PRACE to date, it appears to be a very powerful high-level library which should be of interest to the PRACE community. The object oriented and templated design of the packages within Trilinos allows relatively easy development of codes for various types of architectures (mainly because of the Kokkos core kernels package. Many modern techniques for improving scalability on future multi-petascale and exascale machines, such as communication reducing algorithms, have already been implemented into the framework which gives more reason to investigate the library further, possibly during the exploitation phase of T7.2.
- Several partitioning and mesh generation methods have also been reported on here, all of which show promise for high scalability on the road to exascale.

I/O Management Techniques:

As part of this report we have surveyed five I/O management techniques. The increasing data needs of scientific and engineering applications mean that the problems associated with reading, writing, analysing, storing and sharing large amounts of data are becoming more relevant to a wider user community within PRACE.

- Several efforts by the US DOE are on-going to improve the performance of HDF5 on multi-petascale machines and to adapt the library for exascale. The HDF5 library has shown the ability to scale and perform well enough for multi-petascale simulations. The advantages of HDF5 are that files in the HDF5 format are platform-independent and suitable for high-quality visualizations using the VisIt and ParaView tools. It is the only file format that allows for complex data types like structures and a hierarchical organization of variables. The major downside is the disparity between the presented performance results for some applications and the achieved results in some of the PRACE projects. We believe that one interesting area of focus during the exploitation phase of T7.2 could be on the tuning of applications with HDF5 on parallel file systems of specific PRACE platforms.
- It has been demonstrated that the PnetCDF library can reach a performance that is close to the theoretical peak of the file system on petascale systems, using tens of thousands of MPI tasks. It gives the user a range of tuning options to reach a better performance. The flat file format is one of its strengths, but also a major weakness. Models that use regular, structured grids could benefit from this library, and a large collection of analysis tools can readily read and interpret the output files. The NetCDF format is less well suited for use with VisIt and ParaView. It is less well suited to modern numerical methods like non-structured grids and adaptive mesh refinement.
- The ADIOS library has been shown to reach excellent performance for petascale applications when the internal formats are used. Other file format back-ends will have different performance and scalability characteristics that are also described in this report. If the limited support by other tools of the BP file format is no problem, the ADIOS library could be a great candidate. The ADIOS library is the only library that has a file format with resilience features built into production software. Data reduction by online post-processing is missing from the ADIOS library, which seems to be an essential part of exascale I/O.
- The SIONlib library avoids potential performance bottlenecks by explicating the blocked nature of the filesystem. The SIONlib library is well suited for temporary files and situations without an established tool-chain for post-processing that expects certain file formats. Another advantage is that applications already using POSIX I/O calls don't need to be rewritten radically to parallelise their I/O. As noted in the introduction, the IESP Roadmap expects check-pointing as a technique likely to continue on exascale systems. The SIONlib library seems ideally suited for this purpose. Checkpoints are usually written in one go and resilience of the library is therefore less critical. As soon as a checkpoint is written successfully, older checkpoints can be removed.
- Darshan has been developed with petascale systems in mind and its performance and scalability is therefore excellent. Its statistical analysis is also useful at scale. It is useful for applications with an unknown I/O profile, to get a quick overview of when, where and how much I/O is done. Resilience could be improved by relaxing the condition that the application needs to complete successfully. One useful and relatively easy improvement could be some subroutine calls that can be added to the application by the developer to start the analysis and generate the report at arbitrary points in the code.