Usage Experience with DL_POLY and OpenCL on PRACE Prototypes

Mariusz Uchoński\textsuperscript{a}, Agnieszka Kwiecien\textsuperscript{a}, Marcin Gębarowski\textsuperscript{a}, Justyna Kozłowska\textsuperscript{a}

\textsuperscript{a}WCSS, Wrocław University of Technology, Wyb. Wyspiańskiego 27, 50-370 Wrocław, Poland

Abstract

The prototypes evaluated within PRACE-2IP project provide a number of different computing hardware, including general purpose Graphics Processing Units (GPUs) and accelerators like Intel Xeon Phi. In this work we evaluated the performance and energy consumption of two prototypes when used for a real case simulation. Due to the heterogeneity of the prototypes we decided to use the DL_POLY molecular simulation package and its OpenCL port for the tests. The DL_POLY OpenCL port implements one of the methods – the Constraints Shake (CS) component. SHAKE is a two stage algorithm based on the leapfrog Verlet integration scheme. We used four test cases for the evaluation, one from the DL_POLY application test-suite – H2O, and three real cases, provided by a user. We show the performance results and discuss the usage experience with prototypes in a context of ease of use, porting effort required, and energy consumption.

Introduction

Top-class computing systems introduce more and more hybrid technologies to the HPC landscape. An adoption of general-purpose Graphics Processing Units (GPUs), integrating them into Accelerated Processing Units (APUs) and introducing Intel Xeon Phi accelerators, adds new dimensions to the heterogeneity of multi-node and multi-core systems. These technologies combined into a single system are often followed by new programming models, to enable an efficient usage by parallel applications. Using such a hybrid system may not be trivial, due to a complexity of the software stack.

One of the scientifically important applications which are possible to run on hybrid environments is DL_POLY. DL_POLY is a general purpose parallel molecular dynamics simulation package developed at Daresbury Laboratory by W. Smith and I.T. Todorov \cite{1}. The code has been parallelized using MPI, OpenMP, CUDA (by the Irish Centre for High End Computing) \cite{2} and OpenCL (by the Wrocław Centre for Networking and Supercomputing) \cite{3}.

Due to the heterogeneity of the PRACE-2IP prototypes we decided to use the DL_POLY OpenCL code for testing. We focused on the performance, power consumption and user experience aspects of the prototypes, to evaluate their potential and usability for scientific cases for which the molecular dynamics simulations are used. We evaluated two of the three PRACE-2IP prototypes: Brassica at PSNC and Eurora at CINECA. The DL_POLY OpenCL port implements one of the methods – the Constraints Shake (CS) component. SHAKE is a two stage algorithm based on the Leapfrog Verlet integration scheme \cite{2}.

OpenCL \cite{5} is an open standard for parallel programming of heterogeneous computing systems. It provides an API and a standard language to write portable code for multi-core CPUs, GPUs, APUs.

\textsuperscript{*} Corresponding author email address: agnieszka.kwiecien@pwr.edu.pl
and other architectures, including latest Intel Xeon Phi accelerators. OpenCL kernels are written in a subset of the ISO C99 language that is compiled at runtime to target a particular computing device. An interesting feature of this technology is that the code prepared for Intel Xeon processors is also runnable on the Intel Xeon Phi with minimal changes [7], but it might result in sub-optimal performance. The latest version of the standard, OpenCL v2.0 [6], was announced on March 18, 2014.

Test cases

We used four test cases for the prototype evaluation, one from the DL_POLY application test-suite – H2O (containing 256 water molecules) [4], and three – Benchmark 1: Transferrin in water, Benchmark 9: Cu₃Au Alloy, and Benchmark 12: Liquid phase of CBr₄ – prepared by a researcher from Wroclaw University of Technology. The initial geometries of the systems analysed in Benchmark 1 and Benchmark 9 were prepared based on the database available at: http://ccpforge.cse.rl.ac.uk/gf [8].

H2O Benchmark

This system consists of 256 water molecules (768 atoms) at 300 K. The time step is 0.5 fs and the simulation is for the 20 steps in the NPT isotropic Berendsen ensemble. The electrostatic forces are handled by the Ewald summation method. The real space cutoff is 8 Å and the Van der Waals terms are truncated at 8 Å. The SHAKE algorithm is used for the constrained motion.

Benchmark 1: Transferrin in water

The enzyme transferrin is the major iron-binding protein that controls the level of free iron in biological fluids (Figure 1). Due to the biological significance several studies have been devoted toward understanding this unique protein [9][10][11][12]. Moreover the transferrin enzyme was also widely investigated for targeted drug delivery and other potential biomedical applications [13][14].

This simulation is of the transferrin protein in a water solution. The enzyme transferrin is structurally maintained by constraints with SHAKE. The solvent consist of 8102 TIP3P water molecules held rigid by bond constraints with SHAKE. The total number of atoms in the system is 2753. Electrostatic interactions are handled by neutral groups with the Coulomb potential truncated at 12 Å. The simulation is for 2500 steps with a time step of 0.1 fs in the NVE ensemble.

Figure 1. A schematic representation of the enzyme transferrin.

Benchmark 9: Cu₃Au Alloy

This example represents a straightforward simulation of a Copper-Gold (3/1) alloy with the NVE ensemble and SHAKE algorithm for the constrained motion. The whole system is relatively small and
comprised of 256 atoms. The potential model is Gupta potential with a cutoff at 6.5 Å. The simulation is for 5000 steps with a time step of 5 fs. No electrostatic forces are present in the system.

**Benchmark 12: Liquid phase of CBr₄**
Carbon tetrabromide (CBr₄) is one of the prototypical material in studies of disorder within crystalline media [15][16]. At ambient pressure CBr₄ has two solid modifications, as well as its liquid and gaseous phases. This simulation models the liquid phase of CBr₄. The whole system consists of 20000 atoms and is defined with orthorhombic boundary conditions. The simulation is for 1500 steps with atime step of 1 fs in the NPT Berendsen ensemble. The Ewald Sum is used to calculate the electrostatic interactions, with a real space cutoff of 12 Å. The Van der Waals interactions are truncated at 7 Å. The simulation makes use of the SHAKE algorithm for the constrained motion.

**Tested systems**
Both tested systems, Brassica and Eurora, are Linux clusters with Infiniband interconnect, with a login node and a number of computing nodes. The computing nodes of each cluster are equipped with a different hardware and software (Table 1).

<table>
<thead>
<tr>
<th>Prototype</th>
<th>CPU</th>
<th>CPU clock rate</th>
<th>GPU</th>
<th>Accelerator</th>
<th>OS</th>
<th>Queueing system</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brassica</td>
<td>2x Intel Xeon E5-2620 (6 cores)</td>
<td>2.00 GHz</td>
<td>2x AMD FirePro S9000</td>
<td>-</td>
<td>Ubuntu 12.04.4</td>
<td>SLURM 2.3.2</td>
</tr>
<tr>
<td>Eurora</td>
<td>2x Intel Xeon E5-2687W (8 cores)</td>
<td>3.10GHz</td>
<td>2x Nvidia K20</td>
<td>-</td>
<td>CentOS 6.5</td>
<td>PBSPro 12.2.1</td>
</tr>
<tr>
<td></td>
<td>2x Intel Xeon E5-2658 (8 cores)</td>
<td>2.10GHz</td>
<td>2x Intel Xeon Phi</td>
<td>-</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1 Comparison of the prototypes (computing nodes considered for tests)

**Brassica @PSNC**
Brassica is a PRACE prototype with Intel Xeon CPU E5-2620 (Ivy Bridge-EP), 2.00GHz 6-core, 12 threads and AMD FirePro S9000 GPUs, located at PSNC, Poland. The following software stack has been used for the compilation and runs:
- Operating system: Ubuntu 12.04.4 LTS 3.5.0-49-generic
- OpenCL version: OpenCL 1.2 AMD-APP (1113.2)
- MPI implementation: Open MPI 1.5.4
- Compiler: GNU Fortran (Ubuntu/Linux 4.6.3-1ubuntu5) 4.6.3
- Queueing system: SLURM 2.3.2

**Eurora @CINECA**
Eurora is a cluster with Intel Xeon processors E5-2687W (3.10GHz 8 cores, 16 threads) or E5-2658 (2.10GHz, 8 cores, 16 threads), and nodes equipped with Nvidia K20 GPUs or Intel Xeon Phi accelerators. The following software stack has been used for the compilation and runs:
- Operating system: RedHat CentOS release 6.5 (Final) 2.6.32-431.17.1.el6.x86_64
- CUDA and OpenCL version: CUDA 5.0 V0.2.1221, OpenCL 1.1
- MPI implementation: Open MPI 1.6.5
- Compiler: GNU Fortran (GCC) 4.6.3
- Queueing system: PBSPro 12.2.1
Ease of use

Ease of use is a factor difficult to measure, as it is highly dependent on a level of experience with both the application and the technology. Thus we decided to describe a set of steps we have performed on each prototype along with the lessons learned and problems identified, if any. This is to give an idea about the level of knowledge needed and issues possible to encounter while using a particular set of technologies.

Brassica @PSNC

There was no User Guide provided, nor Environment Modules system deployed. After some initial communication with the prototype owner, and with the SLURM queuing system being in place, it was quite easy to learn how to use the prototype’s features.

No significant porting was required, but it must be mentioned that the OpenCL code for DL_POLY was originally developed and tested on older, but similar architectures to those used in the PSNC prototype. During the test runs it happened a few times that some processes become defunct, and restarting of a node by a system administrator was required. Results from such runs were not included into the time summaries.

Eurora @CINECA

The best practices guides for GPGPU and Xeon Phi on Eurora are available on-line [17][18]. Along with the PBSPro queueing system the Environment Modules system is deployed, what altogether makes it easy to learn about the prototype features and usage.

No significant porting was required for Nvidia K20 GPUs. The code for the Xeon Phi was compiled with Intel SDK for OpenCL Applications 2013 Beta. It appeared there was a convergence issue for this compliant, which is not yet resolved. Usage of this part of the prototype required more effort to determine how the code should be compiled and run on the accelerators (native mode, off-load mode, MPI enabling, etc.).

Application runs

The tests performed on Brassica and Eurora with four inputs described above are presented in Table 2 and Table 3. The times include both the CPU and GPU compute times (an average value from 10 runs), giving the execution time of the whole application.

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>1 GPU</th>
<th>2 GPUs</th>
<th>4 GPUs</th>
<th>8 GPUs</th>
</tr>
</thead>
<tbody>
<tr>
<td>H2O</td>
<td>1.73</td>
<td>2.06</td>
<td>2.37</td>
<td>2.30</td>
</tr>
<tr>
<td>bench_1-transferin</td>
<td>1309.98</td>
<td>606.25</td>
<td>414.89</td>
<td>198.03</td>
</tr>
<tr>
<td>bench_9-Cu_Au</td>
<td>33.44</td>
<td>26.61</td>
<td>17.52</td>
<td>14.04</td>
</tr>
<tr>
<td>bench_12-cbr4</td>
<td>1131.86</td>
<td>642.17</td>
<td>540.74</td>
<td>364.61</td>
</tr>
</tbody>
</table>

Table 2 Execution times of DL_POLY on Eurora

The tests with acceleration were run on both Eurora and Brassica with configurations: one node with 1x CPU+GPU, one node with 2x CPU+GPU, and two nodes - each with 2x CPU+GPU. The maximum job size tested was 4 nodes with 2x CPU+GPU each. In one case, bench_9-Cu_Au on Brassica, the data could not be produced for the biggest jobs, due to technical problems on the prototype. It may be seen that for a small problem size, like H2O, there is no reason to use more resources, as it only generates a bigger communication overhead, including copying the data to and from the GPU’s memory.
<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Execution time on Brassica [s]</th>
<th>1 GPU</th>
<th>2 GPUs</th>
<th>4 GPUs</th>
<th>8 GPUs</th>
</tr>
</thead>
<tbody>
<tr>
<td>H2O</td>
<td></td>
<td>6.62</td>
<td>22.33</td>
<td>34.28</td>
<td>40.65</td>
</tr>
<tr>
<td>bench_1-transferin</td>
<td></td>
<td>1681.75</td>
<td>960.77</td>
<td>610.33</td>
<td>416.01</td>
</tr>
<tr>
<td>bench_9-Cu_Au</td>
<td></td>
<td>77.70</td>
<td>50.90</td>
<td>72.60</td>
<td></td>
</tr>
<tr>
<td>bench_12-cbr4</td>
<td></td>
<td>3495.21</td>
<td>2259.88</td>
<td>1904.45</td>
<td>2405.95</td>
</tr>
</tbody>
</table>

Table 3 Execution times of DL_POLY on Brassica

The execution times on Brassica are longer than on Eurora, what is a consequence of the prototype architecture and hardware parameters, including the processor’s clock rate (2.0 on Brassica, 3.10 on Eurora). The results show that for the test case bench_12-cbr4 the optimal number of GPUs is 4 on Brassica, while on Eurora there is still speedup for 8 GPUs. Similar situation may be seen for the bench_9-Cu_Au test case, where the best execution time is reported for 2 GPUs on Brassica, while on Eurora the time is decreasing up to 8 GPUs (the maximum number tested).

Energy to solution

Energy to solution (EtS) is a metric proposed to evaluate a cost of computation in terms of energy consumed by a system [20][21]. The EtS value indicates the aggregated energy consumption of different system components used by an application during a single run (job). It includes computing nodes, part of a network and cooling sub-systems as well as infrastructure. The EtS formula (1) presented in [19] for a given job $J$ on a system $S$ is as follows:

$$
EtS(J,S) = \sum_{i=startIteration}^{endIteration} \Delta t_i \cdot P_i(J,S)
$$

where:
- $i$ is the iteration index
- $startIteration = min\{i|\text{startTime} \leq timestamp_i \leq \text{endTime}\}$
- $endIteration = max\{i|\text{startTime} \leq timestamp_i \leq \text{endTime}\}$
- $\text{startTime}$ and $\text{endTime}$ are start and end times of job $J$ respectively
- $\Delta t_i = timestamp_i - timestamp_{i-1}$
- $timestamp_i$ is the time and date of $i^{th}$ iteration
- $P_i$ is a power sum of all system $S$ components utilized by job $J$ at the $i^{th}$ iteration of monitoring

The detailed formula for calculating $P_i(J,S)$ is presented in [19]. Besides the power consumed by a job on computing nodes, it takes into account also part of power used by cooling and networking sub-systems”, what may be presented as:

$$
P_i(J,S) = P_i^j + P_i^{\text{cooling}}(J,S) + P_i^{\text{networking}}(J,S)
$$

where:
- $P_i^j$ is the power sum of all nodes utilized by job $J$ at the $i^{th}$ iteration of monitoring
- $P_i^{\text{cooling}}(J,S)$ is the part of cooling power value of system $S$ calculated for job $J$ at the $i^{th}$ iteration of monitoring
- $P_i^{\text{networking}}(J,S)$ is the part of networking power value of system $S$ calculated for job $J$ at the $i^{th}$ iteration of monitoring

If the cooling and networking power values are not provided, the $EtS(J,S)$ metric depends only on the time of calculation and the power of computing nodes utilized by the job.
Averaged energy consumption during the DL_POLY runs on Brassica and Eurora is presented in Table 4 and Table 5 accordingly.

On Brassica the power data are collected per single node and stored in an SQL database. The average level of power consumption for one node was between 73,14 and 103 W (DC), and individual measurements’ range was between 55 and 107 W. The measures were performed with 2 minutes interval for each node. For smaller problems, like bench_9-Cu_Au and H2O, the calculations were repeated 10 to 100 times accordingly within one job, to make sure the monitoring system will measure and report the power usage.

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Average energy consumption on Brassica per job [Wh]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 GPU</td>
</tr>
<tr>
<td>H2O</td>
<td>0.68</td>
</tr>
<tr>
<td>bench_1-transferin</td>
<td>39.60</td>
</tr>
<tr>
<td>bench_9-Cu_Au</td>
<td>1.98</td>
</tr>
<tr>
<td>bench_12-cbr4</td>
<td>70.82</td>
</tr>
</tbody>
</table>

Table 4 Energy consumption during DL_POLY runs on Brassica

*value includes estimation

The energy consumption of the biggest jobs (8 GPUs) on Brassica for H2O and bench_9-Cu_Au is missing, as the monitoring system didn’t report the power usage values for the first runs. Due to time limitations and lack of sufficient computing resources during the testing period (the system was shared with other users) we were not able to repeat all the runs. The values for bench_1-transferin and bench_12-cbr4 have been estimated, as the power usage for one of the four nodes assigned to the jobs was not produced. The missing power usage values has been calculated as an average of the energy consumption of the rest of the nodes assigned to the particular job.

On Eurora the PowerDAM [19] tool is deployed to provide a user with the energy consumption data. The tool was not in full production at the beginning of the tests, so no data has been collected for the first runs, and some tests needed to be repeated. As on Brassica, the calculations of the smaller problem, bench_9-Cu_Au and H2O, have been repeated 10 to 100 times within one job, to make sure the monitoring system will measure and report the power usage. Although, for the last jobs, bench_1-transferin runs on 8 GPUs, the monitoring tool has not delivered any report. The EtS data for Eurora are presented in Table 5.

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Average energy consumption on Eurora per job [Wh]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 GPU</td>
</tr>
<tr>
<td>H2O</td>
<td>0.47</td>
</tr>
<tr>
<td>bench_1-transferin</td>
<td>107.00</td>
</tr>
<tr>
<td>bench_9-Cu_Au</td>
<td>1.68</td>
</tr>
<tr>
<td>bench_12-cbr4</td>
<td>46.04</td>
</tr>
</tbody>
</table>

Table 5 Energy consumption during DL_POLY runs on Eurora

The PowerDAM tool presents the power usage per single computing job giving also detailed information about the usage structure, according to the EtS formula (1). An example report for one of the jobs is as follows:

```
$ ets --system=Eurora --job=812989.node129

EtS is: 0.0194452 kWh
Computation: 99 %
Networking: 0 %
Cooling: 0 %
Infrastructure: 0 %
```
Reports for other jobs have been similar, showing 99% of power used by computations and 0% by other factors. It is possible that the power usages of networking, cooling and infrastructure have not been monitored, or just the values have been so small for every job, that resulted in reporting 0%.

Conclusions

We were able to compile and run the application on both prototypes without many difficulties. The compilation for Intel Xeon Phi required usage of Intel compilers, and only for this accelerator no satisfactory results were achieved. The best results in terms of performance and scalability were obtained on Eurora. If a user considers only the computation time, the best choice to run the jobs on Eurora, would be, with a few exceptions, on 8 GPUs – the maximum tested configuration. However, looking at the power consumption of jobs, changes the perspective, giving the optimal resources as 2-4 GPUs. Providing users with the EiS values may let them to use the computing systems in more conscious and reasonable way, also from the “green” perspective.

References

[8] CCPForge: http://ccpforge.cse.rl.ac.uk/gf/
Acknowledgements

This work was financially supported by the PRACE project funded in part by the EU’s 7th Framework Programme (FP7/2007-2013) under grant agreement no. RI-283493. The calculations were carried out on computing resources at CINECA (Eurora) and PSNC (Brassica).