Available online at www.prace-ri.eu

Partnership for Advanced Computing in Europe

Resource Scheduling Best Practice in Hybrid Clusters

C. Cavazzoni[a], A. Federico[b], D. Galetti[a], G. Morelli[b], A. Pieretti[b]

[a] CINECA, via Magnanelli 6/3, 40033 Casalecchio di Reno, Italy

[b] CINECA, via dei Tizii 6/b, 00185 Roma, Italy

Abstract

HPC green thinking implies the reduction of power consumption which is in contrast to the need of an ever growing demand in terms of computational power. To circumvent this dilemma, several types of "computing accelerators" have been adopted. Using an accelerator means partial if not total code rewriting, with the aim of achieving a speed-up which would be difficult to attain with present CPU and hardware evolution. After an initial period of concern about this new technology, programmers have shown a growing interest in this field and several of the most used scientific codes have undergone an intense software restyling.

Accelerators have introduced a new class of requests which need to be fulfilled by resource schedulers on hybrid clusters. Hence, exploring what schedulers can offer in terms of minimizing the effort and maximizing the resource exploitation has become a fundamental issue. As the CINECA supercomputing center runs a new generation hybrid cluster with two different accelerators, i.e. GPUs and MICs, it is involved in testing a resource scheduler, PBSPro in order to put its cluster to the best possible use.

Introduction

The average number of accelerator boards on a server is constantly increasing. Several HPC vendors now offer up to eight GPUs on a node equipped with sixteen CPU cores, a 1:2 ratio. The greater the number of accelerators on a node, the greater the importance of having exclusive access to them, given that even with a single accelerator card per node, whenever more than a job is being executed on a node, a protection mechanism should exist so as to guarantee that concurrent access to resources is prohibited.

Regardless of the resource types (i.e. CPUs, GPUs, MICs, etc.) the cluster scheduler should be able to identify, allocate and monitor these resources to fulfil the requests of the jobs and to maximize the cluster utilization. The more heterogeneous the cluster is, the more the work of the scheduler becomes hard. Furthermore, the nodes in a cluster might have different accelerator card types, forcing schedulers to provide finer selection mechanisms in order to ensure that complex requests will be correctly satisfied.

To the best of our knowledge, at present a great deal of supplementary work is needed for all the expected goals to be achieved.

In this paper we will discuss the features and limitations of Altair PBSPro in terms of scheduling GPU and MIC jobs on the CINECA new generation hybrid cluster, EURORA.

The hardware

EURORA is a single rack cluster built by Eurotech and co-funded by the PRACE-2IP project. It implements Intel Xeon Sandy Bridge CPUs in 64 double-socket nodes for a total of 1024 cores. The internal network is made of one FPGA (Altera Stratix V) per node, an IB QDR interconnect and a 3D torus interconnect. Thanks to the innovative cooling and system engineering EURORA was ranked in a top position of the Green 500 chart[1], with a sustained performance of 3,150 MFlops/W. Intel Xeon Phi nodes (MIC nodes) are equipped with Intel E5-2658 Xeon Sandy Bridge with a clock of 2.10GHz; nVIDIA K20 nodes (GPU nodes) are equipped with Intel E5-2687W Xeon Sandy Bridge with a clock of 3.10GHz. Both GPU and MIC nodes have 2 accelerators per node. The operating system running on EURORA is Centos 6.3.

The scheduler

PBS Professional (PBSPro)[2] is a resource management system developed by Altair. It provides a common way, using jobs and queues, to deliver computing power to applications. Users submit jobs in the form of shell scripts that will run on the allocated resources. PBSPro provides support for standard (CPU, memory, walltime, etc.) and generic resources as well. Administrators can define a new generic resource describing its type (integer, float, boolean, string), its kind (consumable/non-consumable, static/dynamic) and its context (server, queue or node). This approach makes it easy to define almost any type of resource but it lacks in their control and usage tracking. As opposed to known typed resources such as cputime or memory for which PBSPro can, for example, enforce resource usage limits, for non-typed resources this must be implemented with other tools and/or custom scripts written ad hoc by system administrators. In this paper we will discuss about the generic resource features for GPUs and MICs of version 12.2 of Altair PBSPro that is running on EURORA.

nVIDIA GPGPU scheduling

On EURORA, a GPU is a consumable integer resource defined at node level: each GPU node has a resource ngpus=2. For scheduling purpose this configuration is acceptable: a job J requesting a GPU will be scheduled to one of the GPU nodes, the node allocated to job J will have its value of ngpus reduced by 1 and will be considered for scheduling only to other jobs requesting a single GPU, until job J is completed. However, because PBSPro does not provide a way to control the usage of the GPUs of the node allocated to job J, some issues arise:

- λ job J can use any GPUs of the node (0, 1 or both);
- λ job J is not guaranteed the exclusive usage of the allocated GPU. Any other single GPU job scheduled to the same node of job J may use the same GPU of job J (or both the GPUs of the node).

From this point of view this configuration provides a very basic support of special resources such as GPUs or MICs.

An advanced configuration can be achieved using PBSPro virtual nodes (vnode). A vnode is an abstract object representing a set of resources which form a usable part of a machine. In a vnode based configuration each natural[3] GPU node (nodeX) has two vnodes (nodeX-gpu0 and nodeX-gpu1) with ngpus=1 each representing the GPU0 and the GPU1 of the natural node. Since each vnode can be managed and scheduled independently, this configuration has the advantage that make it possible to identify which GPUs are assigned to a given job. For example, if the scheduler allocates the vnode nodeX-gpu0 of the nodeX to job J of the previous example, job J should be using GPU0 of nodeX. However, because PBSPro does not provide a way to bind the job to the GPU(s) assigned by the scheduler this configuration still suffers from the issues described above.

---

1 http://www.green500.org/lists/green201306
2 http://www.pbsworks.com/
3 A “natural” node in PBSPro represents a “real” physical node.

Starting from CUDA version 3.1, nVIDIA provides a solution to address this scheduling issue. It is based on the environment variable CUDA_VISIBLE_DEVICE[4] used for restricting execution to a specific device. Setting this variable to a comma-separated sequence of integers only the devices whose index is present in the sequence are visible to CUDA applications and they are enumerated in the order of the sequence. Unfortunately this solution cannot be implemented on PBS-like resource management systems, such as PBSPro, because the environment of a PBS job is the same for all the processes launched within the job script. This means that a two processes job scheduled to use nodeX-gpu0 and nodeY-gpu1 could not have different values of this variable for processes running on nodeX and nodeY, where it should be CUDA_VISIBLE_DEVICE=0 and CUDA_VISIBLE_DEVICE=1, respectively. It would only work in the unlikely event where the same GPU index (e.g. GPU0) is selected on all of the nodes assigned to a job.

Not all resource management systems are affected by this issue. For example, the Simple Linux Utility for Resource Management (SLURM)[5] provides a specific generic resource plugin for GPUs[6] that sets the correct environment for each job process to determine which GPUs are available for its use on each node.An alternative to the variable CUDA_VISIBLE_DEVICE that may work with PBS-like systems was developed by the National Center for Supercomputing Applications (NCSA) in 2009. The CUDA wrapper library[7] is implemented as a preloaded library. As its name suggests, the library intercepts the device allocation calls to CUDA in order to virtualize the natural GPU devices and provide NUMA affinity mapping between CPU cores and GPU devices. By using an appropriate job prologue script, the administrator can set up the library to make only the allocated GPUs visible to the processes of a job. This solution can be implemented with PBSPro and it is in a testing phase on EURORA. However, it seems that this project is no more active or at least its last modification time dates back to 2012.Intel MICs scheduling

Considering the common features of almost any resource manager system, Intel MIC coprocessors are easier to manage than GPUs because they are based on x86-compatible multiprocessor architecture. The Intel Manycore Platform Software Stack (MPSS) provides the necessary software to configure and run the MIC cards. The MPSS command micctrl is used to configure the MIC kernel, filesystem and network as well as boot, reset, shutdown, and get the status of the MIC cards. By using this command in the prologue/epilogue script of the resource manager it is possible to switch on/off the MIC cards required to run a job.

On EURORA, each MIC natural node (nodeX) has two vnodes (nodeX-mic0 and nodeX-mic1) representing the two MIC cards. PBSPro is configured to assign a generic resource nmic=1 to each vnode nodeX-micY (Y=0,1), the corresponding natural node nodeX having nmics=2. Jobs requesting MICs are scheduled to MIC vnodes making it possible to identify the MIC cards assigned to a job on each allocated node. Using the micctrl command, it is possible to set up and boot the MIC cards on each node before job execution (prologue) and to reset them after job completion (epilogue).

On EURORA, every MIC card is switched off by default. Those selected for a job are switched on when a job begins its execution and they are switched off again after the job is completed. For example, if a user U of group G submits a job requesting two MICs and the scheduler assigns vnodes nodeX-mic0 and nodeY-mic1 to the job. Before the job begins its execution the prologue script will run on nodes nodeX and nodeY with the following operations to set up the assigned MICs (MIC0/MIC1 on nodeX/nodeY):

1. it adds user/group credentials to the MICs filesystem (e.g. on nodeX[8], micctrl --useradd=U … mic0 & micctrl --groupadd=G … mic0)
2. it boots the MIC cards waiting for completion[9] (e.g. on nodeY, micctrl --boot --wait mic1)

---

4 http://www.nvidia.com/content/PDF/GDC2011/Dale_Southard_SC11.pdf
http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#env-vars
https://devblogs.nvidia.com/parallelforall/cuda-pro-tip-control-gpu-visibility-cuda_visible_devices/

5 https://computing.llnl.gov/linux/slurm/

6 https://computing.llnl.gov/linux/slurm/gres.html

7 http://sourceforge.net/projects/cudawrapper/
http://www.ncsa.illinois.edu/People/kindr/projects/hpca/files/hpca2010.pdf

8 The three dots (...) represents other arguments of the micctrl command used to set up all the fields of the passwd/group files (home directory, uid, gid, etc.).

Once the job is completed the epilogue script restores the default configuration:

1. it resets the MIC cards waiting for completion (e.g. on nodeY, micctrl --reset --wait mic1)
2. it removes user/group credentials from the MICs filesystem (e.g. on nodeX, micctrl --userdel=U mic0 & micctrl --groupdel=G mic0)

This approach ensures that only processes belonging to the job of user U can access the MIC cards assigned to that job. Moreover, it reduces the power consumption of a node by about 210 Watts, when both MIC cards are powered off. However, there is a major drawback. We have measured the time needed for booting (resetting) a MIC card to be approximately one minute. Since our hook is very simple, its Hook_execution_time depends on the MIC booting/resetting time only. In the presence of hooks, PBSPro schedules jobs in a serial way, which means that, given N pending jobs requesting MICs, that could be assigned free resources (nodes & MICs) at a given time, the Nth job will have to wait (N-1)*Hook_execution_time before being dispatched to the selected node/s.

This problem has been observed on a cluster with 64 nodes. The solution is, therefore, not viable for clusters with thousands of nodes.

Features and limitations summary

The features and limitations of PBSPro, with and without third party libraries and custom scripts, are summarized in the tables reported below where we use the following definitions:

- λ basic support: the possibility to ask for a given number of resources (e.g. 8 CPUs and 2 GPUs);
- λ advanced support: the possibility to identify a specific resource (e.g. MIC number 1);
- λ exclusive access: the absolute certainty that the given resources will not be usable by any other job;
- λ banned overuse: banning the possibility for a job to use resources not allocated to itself, whether free or already assigned to another job.

PBSPro

	basic support	**advanced support**	**exclusive access**	**banned overuse**
GPU	Y	Y (vnodes)	N	N
MIC	Y	Y (vnodes)	N	N

PBSPro + custom scripts/libraries

	basic support	**advanced support**	**exclusive access**	**banned overuse**
GPU	Y	Y (vnodes)	N/Y (CUDA wrapper?)	N/Y (CUDA wrapper?)
MIC	Y	Y (vnodes)	Y (prologue/epilogue)	Y (prologue/epilogue)

9 The --wait switch is needed to ensure that MIC cards have been booted before the job starts.

Conclusions

Unless users are forced to use all of the resources on a node as a single atomic unit in their job requests, whenever two or more jobs will run on the same node asking for accelerators, problems regarding correct placement of requests will arise. The new generation of schedulers must guarantee that special resources, such as accelerators, are correctly identified and assigned in an exclusive mode to the requesting jobs. At the present moment, our experience with the Altair PBSPro scheduler on CINECA's most recent hybrid cluster gives us an indication on how to proceed even though we are only halfway there. Custom tools/libraries must be put in place, in order to satisfy the job requests correctly.

Acknowledgements

This work was financially supported by the PRACE project funded in part by the EUs 7th Framework Programme (FP7/2007-2013) under grant agreement no. RI-283493.