Heterogeneous programming methodology based on OpenCL framework

O. Sudareva, A. Efremov, P. Bogdanov

Scientific Research Institute for System Studies, Russian Academy of Sciences,
Nakhimovskii pr. 36 1, Moscow, 117218 Russia

fioremay.olyas@gmail.com, antonyef@mail.ru, bogdanov@niisi.msk.ru

Abstract. *High-performance computing systems containing hybrid nodes with microprocessors like GPGPU or Intel MIC are widely spread nowadays. Many (though not all) computational algorithms can be effectively ported to hybrid systems. We have developed a multiparameter model of a hybrid computing system and a general method of evaluation the performance of algorithms on such systems in terms of the model parameters. In addition, we have implemented a system-level service, an Infrastructure of Heterogeneous Computing, which allows to program a whole distributed system within one programming model. In our work we discuss some applications of the Infrastructure to a wide class of tasks (HPL, NAS PB {FFT, MG, CG}, CFD), and provide a brief comparison with other heterogeneous approaches.*

Keywords

Parallel processing, heterogeneous programming, OpenCL programming.

1 Introduction

Nowadays hybrid computers based on massively-parallel accelerators are becoming more and more popular. The most common massively-parallel accelerators are GPGPUs (general purpose graphic processing units). At the same time, Intel MIC looks very perspective. In the last Top500 list (June 2013), the first place is Tianhe-2[1] with 50 PFlops, based on Intel MIC, and the second place is Cray Titan[2] with 30 PFlops, based on NVidia Tesla K20X.

However, there is still no common programming model which could be used to program all processors on one node. Therefore, there is no common model to program a whole distributed system with hybrid nodes too. There are some investigations of such models. In our presentation we plan to compare some relevant models briefly.

2 Related works

We provide a comparative review of several relevant heterogeneous programming models: StarPU[3], OpenACC[4], ompSS[5], OpenMP 4.0[6].

3 Main sections

We have to note that the developers of modern scientific and applicative software do not always calculate the computational capacity of used algorithms and evaluate the expected performance on the target hardware. We offer a way how to estimate the expected performance of a task before starting the actual code writing. Such estimates make it possible to decide whether the system is suitable for this task or not.

We propose a method of estimating the original problem, which relies on a formal OpenCL model for a compute node and a whole distributed system. We have constructed a general model for such system, defined by several parameters: the number of massively-parallel coprocessors, the memory size of each one, memory bandwidth, etc.. Each computational task can be analyzed in this model, and the expected performance can be deduced as a formula

depending on the parameters. We demonstrate that some well-known task classes can be effectively implemented on heterogeneous parallel systems.

Also, in our work we introduce the second version of our Infrastructure of Heterogeneous Computing. It is implemented as a command scheduler based on OpenCL model and MPI communication interface. The program for this scheduler is a dependency graph of commands executed on compute devices. The general way to program the whole distributed system using this infrastructure is the following. One device is controlled by one scheduler, the synchronization of schedulers on one node is done by adding dependencies between their commands, and the nodes communicate via MPI calls.

This Infrastructure was applied to the several tasks: the HPL test[7], NAS Parallel Benchmarks[8] (FFT, MG, CG) and the model CFD task (sphere in a supersonic flow). We implemented all necessary compute kernels and algorithms to launch these tasks on distrubuted system using all devices with OpenCL support. As a "side-effect" we obtained the heterogeneous BLAS version, which could accelerate an application on the OpenCL devices via simple re-link with the new library. The library can be scaled up to 8 accelerators on one node [9].

These tasks were launched on the wide range of processor architectires and node configurations: one "fat node" with 2 Intel Xeon E5-2670 CPUs and 8 accelerators (AMD Radeon 7970 GPU, NVidia TITAN GPU), supercomputer K100 [10](64 nodes with 2 Intel Xeon X5690 CPUs and 3 NVidia Tesla M2050 GPUs per node), supercomputer K10 (6 nodes with 2 Intel Xeon E5-2620 CPUs and 3 NVidia Tesla 2090 GPUs per node).

The implementation of all tasks consists of three parts: first, one has to investigate of the theoretical performance estimation, then write the compute kernels on the OpenCL C language and finally write the upper logic (dependency graph) as a scheduler program. We will not discuss the kernels, the attention will be focused on the dependency-graph programs. In the end we plan to discuss the achieved results and the prospects for further development.

4 Conclusion

The simplicity and transparency of the programming model, the ease of development a real-world scalable codes prove the viability of the approach. In addition, one of the major achievements is the code portability on all currently known hardware platforms. Of course, sometimes critical OpenCL C kernels require special porting to the particular accelerator. Thus, the upper level logic remains the same for all hardware platforms. In future, we plan to increase an amount of the Infrastructure of Heterogeneous Computing applications.

References

[1] J. J. Dongarra:Visit to the National University for Defense Technology Changsha, China. *Technical report, Oak Ridge National Laboratory*, 18p, 2013.
[2] http://www.olcf.ornl.gov/titan/
[3] http://runtime.bordeaux.inria.fr/StarPU/
[4] http://www.openacc-standard.org/
[5] http://pm.bsc.es/ompss
[6] http://openmp.org/wp/openmp-specifications/
[7] http://www.netlib.org/benchmark/hpl/
[8] http://www.nas.nasa.gov/publications/npb.html
[9] http://devgurus.amd.com/thread/159457
[10]http://www.kiam.ru/MVS/resourses/k100.html
[11]http://www.kiam.ru/MVS/resourses/#k10