Predictive Modeling and Analysis of OP2 on Distributed Memory GPU Clusters[*]

G.R. Mudalige, M.B. Giles
Oxford e-Research Centre, University of Oxford
mike.giles@maths.ox.ac.uk,
gihan.mudalige@oerc.ox.ac.uk

C. Bertolli, P.H.J Kelly
Dept. of Computing,
Imperial College London
{c.bertolli, p.kelly}@imperial.ac.uk

ABSTRACT

OP2 is an "active" library framework for the development and solution of unstructured mesh-based applications. It aims to decouple the scientific specification of an application from its parallel implementation to achieve code longevity and near-optimal performance through re-targeting the back-end to different multi-core/many-core hardware. This paper presents a summary of a predictive performance analysis and benchmarking study of OP2 on heterogeneous cluster systems. In this work, an industrial representative CFD application written using the OP2 framework is benchmarked during the solution of an unstructured mesh of 1.5M and 26M edges. Benchmark systems include a large-scale Cray XE6 system and an Intel Westmere/InfiniBand cluster. Performance modeling is then used to predict the application's performance on an NVIDIA Tesla C2070-based GPU cluster, enabling the comparison of OP2's performance capabilities on emerging distributed memory heterogeneous systems. Results illustrate the performance benefits that can be gained through many-core solutions both on single-node and heterogeneous configurations in comparison to traditional homogeneous cluster systems for this class of application.

Categories and Subject Descriptors

D.4.8 [**Performance**]

Keywords

OP2, Unstructured mesh, GPU, Performance modeling

1. INTRODUCTION

With heterogeneous HPC systems such as Tianhe-1A, Tsubame and Nebulae gaining recognition as leading multi Peta-FLOP systems [1], there appears to be an increasing trend in using many-core processor architectures in a hybrid combination with traditional CPUs. On the other hand, homogeneous systems such as the K-Computer and Jaguar, based on traditional multi-core CPU hardware, appear to be defending their dominant positions as the top performing systems in the world. Other systems architectures, such as the IBM BlueGene range follow a different "many-core" path with massively parallel quantities of independently operating low-speed cores interconnected by high-speed networks. As the many-core vs multi-core debate rages on, technologies that enable users to efficiently exploit these systems appear to be in an ever increasing state of flux, with a range of competing programming languages and architectural optimizations/configurations. Application developers will need to constantly keep up an expert level of knowledge in the intricate details of new technologies and architectures in order to obtain the best performance from their codes.

The demand of maintaining such a programming skills-set is distracting domain application developers from investing their full intellectual efforts in the scientific/engineering problems they are solving. It is clear that a level of abstraction must be achieved so that computational scientists can increase their productivity by focusing on solving problems at a higher level, write code that remains unchanged for different underlying hardware and not worry about architecture specific optimizations. At the same time, a lower implementation level, maintained by HPC technology professionals and optimization experts, can focus on how a computation can be executed most efficiently on a given platform by carefully analyzing the computation and data access patterns. This paves the way for easily integrating support for any future novel hardware architecture and maintain near optimal performance.

OP2 aims to provide such an abstraction layer, by developing an "active" library framework for the solution of unstructured mesh applications. The "active" library approach uses program transformation tools, so that a single application code written using the OP2 API is transformed into the appropriate form that can be linked against a given parallel implementation (e.g. OpenMP, CUDA, OpenCL, SSE/AVX, MPI, etc.) enabling execution on different back-end hardware platforms. At the same time, OP2 attempts to maintain near-optimal performance by exploiting low-level optimizations and/or configurations on a target platform without the intervention of the application programmer.

OP2 currently enables application developers to write a single program (using the OP2 API, in either C/C++ or Fortran) which can then be transformed (using OP2 code transformation tools) into executables for three different platforms: (1) single-threaded on a CPU, (2) multi-threaded using OpenMP for execution on a single multi-core CPU (including on an SMP or a large shared-memory node) and (3) parallelized using CUDA for execution on a single NVIDIA GPU. In our previous work [4, 2] we have presented the design and implementation of OP2 for single node systems and

*This research is funded by the UK Technology Strategy Board and Rolls-Royce plc. through the Siloet project, and the UK Engineering and Physical Sciences Research Council projects EP/I006079/1, EP/I00677X/1 on Multi-layered Abstractions for PDEs.

Table 1: Airfoil run-times comparison

Node System	Cores /node (Clk/core)	Memory /node	Run-time (seconds)
2×Intel Xeon X5650 (Westmere)	12 (24 SMT) (2.67GHz)	24 GB	37.89 (24 OMP)
2×AMD Opteron 6128 (Magny Cours)	16 (2.0GHz)	12 GB	46.30 (16 OMP)
GeForce GTX560Ti	384 (1.6GHz)	1.0 GB	19.63
Tesla C2070	448 (1.15GHz)	6.0 GB	13.20
HECToR (Cray XE6)	72 cores		13.22
	480 cores		2.09
	960 cores		1.12
CX1 (Intel Westmere /InfiniBand)	36 cores		20.66
	60 cores		12.29
	120 cores		6.07

benchmarked an industrial representative application written using the OP2 library on a range of flagship single node systems, consisting of NVIDIA GPUs and x86 based multi-core CPUs. The next step of the OP2 development is to facilitate code generation and execution on heterogeneous systems such as clusters of single/multi-threaded CPUs or GPUs. The objective is to layer distributed memory parallelization in combination with any intra-node parallelization technologies (OpenMP, CUDA, etc.). This short paper summarizes results from an early performance evaluation of OP2's distributed memory capabilities for such systems; the full paper can be found in [5]. Utilizing a recently developed distributed memory back-end layer based on MPI, its performance on a number of homogeneous cluster systems together with performance modeling techniques we present an "ahead of implementation" predictive performance analysis for a cluster of GPUs. Our objective is to gain quantitative and qualitative insights into the achievable performance on such systems and contrast it with performance from single-node and traditional homogeneous cluster solutions for unstructured mesh based applications.

2. RESULTS

Table 1 presents best run-times of an industrial representative CFD application (Airfoil [3]), written using OP2 on an number of single node and distributed memory systems. The application in this case is solving a mesh with approximately 1.5 million edges. The GPUs execute NVIDIA CUDA code generated by the OP2 framework, while the Westmere and Opteron multi-core CPUs utilize OpenMP generated by OP2. The executables on the CrayXE6 and the Westmere/InfiniBand cluster run under MPI also generated by OP2. The best run-time with OpenMP (37.89 seconds) on the Westmere processor node (compiled with icc -O2 -xSSE4.2) was obtained by executing 24 OpenMP threads on 12 symmetric multi-threading (SMT) enabled cores.

On the Opteron processor node, the best run-time (46.30 seconds) was gained with 16 OpenMP threads (compiled with icc -O2 -ipo -xSSE2 -funroll-loops). The GPU results give a best runtime of 19.63 seconds and 13.20 seconds on the GTX560Ti and the Tesla C2070 respectively. These are speedups of 1.93× and 2.8× respectively, compared to the Intel Westmere processor node with 24 OpenMP threads on 12 (SMT) cores. In contrast we see that the performance of Airfoil on HECToR with 72 cores (3 nodes) and CX1 with 60 cores (5 nodes) is approximately equivalent to the performance of one C2070 GPU. Considerable performance gains (not to mention power and cooling savings) can be achieved even on a single consumer-grade GTX560Ti (equivalent to approximately 36 Westmere cores) for this application.

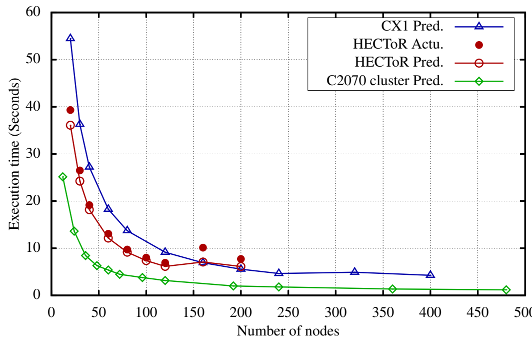

Figure 1: Airfoil - 26M edge mesh: HECToR - 24 cores/node, CX1 - 12 cores/node, C2070 cluster - 1 GPU/node

Figure 1 presents projected performance from a performance model of Airfoil, on CX1 and a hypothetical NVIDIA Tesla C2070 GPU cluster at scale. Actual run times from HECToR are also provided as a reference. The application in this case is solving a larger 26 Million edge mesh. We see a C2070 cluster to give the same performance that is equivalent to performance given by traditional homogeneous clusters that are more than three times its size.

3. CONCLUSION

The Airfoil unstructured mesh application benchmarked in this work shows up to 3× speedup on current flagship GPUs compared to their equivalent multi-core CPU counterparts. Our experiments show that this holds true compared to both single node CPUs utilizing thread-level parallelism (OpenMP) as well as traditional homogeneous distributed memory clusters (single threaded CPU clusters). On a heterogeneous cluster system, we expect such applications to exhibit similar performance gains given that the individual GPU nodes do not exhaust their resources during the solution of a given workload. On the other hand, our performance modeling predicts that the limiting factor in scalability is not primarily the PCIe or MPI/InfiniBand overheads, particularly when non-blocking operations are used to hide communication costs. Scalability is affected more by the amount of parallelism available per partition to be exploited by each GPU at scale. Thus a balance must be achieved to not overload the resources of individual GPUs but at the same time have enough computation that can be parallelized within a node to gain good performance. The full paper can be found in [5].

4. REFERENCES

[1] Top500 Systems, June 2011. `http://www.top500.org/list/2011/06/100`.

[2] Giles, M., *et. al* Performance analysis and optimization of the OP2 framework on many-core architectures. *The Computer Journal* (2011).

[3] Giles, M. B., Ghate, D., and Duta, M. C. Using automatic differentiation for adjoint CFD code development. *Computational Fluid Dynamics Journal 16*, 4 (2008), 434–443.

[4] Giles, M. B., Mudalige, G. R., Sharif, Z., Markall, G., and Kelly, P. H. Performance analysis of the OP2 framework on many-core architectures. *SIGMETRICS Performance Evaluation Review 38(4)* (2011), 9–15.

[5] Mudalige, G. R., Giles, M. B., Bertolli, C., and Kelly, P. H. Predictive Modeling and Analysis of OP2 on Distributed Memory GPU Clusters. *SIGMETRICS Performance Evaluation Review 40(2)* (2012)