Contents lists available at ScienceDirect

Parallel Computing

journal homepage: www.elsevier.com/locate/parco

Design and initial performance of a high-level unstructured mesh framework on heterogeneous parallel systems

G.R. Mudalige [a,*], M.B. Giles [a], J. Thiyagalingam [a], I.Z. Reguly [a], C. Bertolli [b], P.H.J. Kelly [c], A.E. Trefethen [a]

[a] *Oxford e-Research Centre, University of Oxford, UK*
[b] *IBM T.J. Watson Research Center, New York, USA*
[c] *Dept. of Computing, Imperial College London, UK*

A R T I C L E I N F O

Article history:
Received 30 November 2012
Received in revised form 27 August 2013
Accepted 15 September 2013
Available online 25 September 2013

Keywords:
OP2
Domain specific language
Active library
Unstructured mesh
GPU
Heterogeneous systems

A B S T R A C T

OP2 is a high-level domain specific library framework for the solution of unstructured mesh-based applications. It utilizes source-to-source translation and compilation so that a single application code written using the OP2 API can be transformed into multiple parallel implementations for execution on a range of back-end hardware platforms. In this paper we present the design and performance of OP2's recent developments facilitating code generation and execution on distributed memory heterogeneous systems. OP2 targets the solution of numerical problems based on static unstructured meshes. We discuss the main design issues in parallelizing this class of applications. These include handling data dependencies in accessing indirectly referenced data and design considerations in generating code for execution on a cluster of multi-threaded CPUs and GPUs. Two representative CFD applications, written using the OP2 framework, are utilized to provide a contrasting benchmarking and performance analysis study on a number of heterogeneous systems including a large scale Cray XE6 system and a large GPU cluster. A range of performance metrics are benchmarked including runtime, scalability, achieved compute and bandwidth performance, runtime bottlenecks and systems energy consumption. We demonstrate that an application written once at a high-level using OP2 is easily portable across a wide range of contrasting platforms and is capable of achieving near-optimal performance without the intervention of the domain application programmer.

1. Introduction

Heterogeneous parallel computing systems are becoming an increasingly common architecture adopted by many High Performance Computing (HPC) systems designers and vendors. Based on the current generation of CMOS micro-processor technology, it is seen as a key path towards reaching exaFlop levels of performance, within a tractable energy envelope. In a heterogeneous system, traditional processors will be augmented by an attached co-processor (usually called an accelerator) which gives higher performance for solving certain types of computations. Current examples of heterogeneous combinations range from traditional x86 processors accelerated by discrete (e.g. NVIDIA GPUs [1], Intel MIC [2]) or integrated

* Corresponding author. Address: Oxford e-Research Centre, University of Oxford, 7 Keble Road, Oxford OX1 3QG, UK. Tel.: +44 (0)1865 610787; fax: +44 (0)1865 610612.

E-mail addresses: gihan.mudalige@oerc.ox.ac.uk, gihanmudalige@gmail.com (G.R. Mudalige), mike.giles@maths.ox.ac.uk (M.B. Giles), jeyarajan.thiyagalingam@oerc.ox.ac.uk (J. Thiyagalingam), istvan.reguly@oerc.ox.ac.uk (I.Z. Reguly), cbertol@us.ibm.com (C. Bertolli), p.kelly@imperial.ac.uk (P.H.J. Kelly), anne.trefethen@oerc.ox.ac.uk (A.E. Trefethen).

0167-8191/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.parco.2013.09.004

(e.g. AMD APUs [3]) SIMD type co-processors to more specialized DSP type compute engines [4] or FPGA based accelerators [5,6]. To gain good performance from such systems, in addition to the complexity in programming the accelerator devices, the applications developer needs to manage concurrency between the different processors/co-processors and manage data locality within a complicated memory hierarchy. Maintaining high performance becomes even more complicated on clusters of heterogeneous nodes, which are essential for production applications requiring significantly more compute resources and capabilities.

Recent efforts in utilizing heterogeneous parallel systems in HPC [7–9] have mostly discussed hand-porting CPU based code (often single-threaded sequential CPU code) to accelerator based systems. However this process is time-consuming, error-prone and takes a significant amount of software development effort by programmers who need to be experts in the new heterogeneous architectures. Emerging programming languages and extensions such as OpenACC [10] attempt to alleviate some of the burden in programming these systems, but the programmer still has to "instruct" the compiler where and how to identify and exploit parallel regions in the code for the accelerator. This may require significant effort, particularly for large production HPC applications. Furthermore, such directive based approaches are yet to provide evidence of achieving good performance for some classes of applications in comparison to hand-ported implementations that utilize specific many-core programming languages such as CUDA or OpenCL.

As the parallel processor landscape changes rapidly, there is a need to constantly maintain an expert level of knowledge in the intricate details of new technologies and heterogeneous architectures in order to obtain the best performance from applications. Application developers would like to benefit from the performance gains promised by these new systems, but are concerned about the associated costs of software development. One solution is to utilize domain-specific knowledge about applications in developing a suitable abstraction to increase productivity and maintain performance. This allows for scientists and engineers to develop code based on a higher level, writing applications that remain unchanged for different underlying hardware. At the same time, a lower implementation level provides opportunity for parallel programming experts to apply radically aggressive and platform specific optimizations when implementing the required solution on various hardware. The correct abstraction will pave the way for easy maintenance of a higher-level application source with near optimal performance for various platforms and make it possible to easily integrate support for any future novel hardware.

OP2 aims to provide such an abstraction layer for the solution of unstructured mesh-based applications. OP2 uses an active library approach [11,12] where a single application code written using the OP2 API can be transformed into multiple parallel implementations which can then be linked against the appropriate parallel library (e.g. OpenMP, CUDA, MPI, OpenCL, etc.) enabling execution on different back-end hardware platforms. At the same time, the generated code from OP2 and the platform specific back-end libraries are highly optimized utilizing the best low-level features of a target architecture to make an OP2 application achieve near-optimal performance including high computational efficiency and minimized memory traffic.

In previous papers we presented OP2's API and its back-end design facilitating the development and execution of unstructured mesh applications on single-node CPU and GPU systems [13–17] and performance on large CPU clusters [18]. These back-end designs enable the OP2 framework to generate code targeting: (1) single-threaded CPUs, (2) multi-threaded CPUs/SMPs, (3) single NVIDIA GPUs and (4) distributed memory clusters of single-threaded CPUs. In this paper we present the next stage of the OP2 framework: the design facilitating the code generation and execution on multiple heterogeneous parallel systems, particularly incorporating distributed memory and many-core parallelism, and its performance. The contributions of this paper are twofold:

1. Design: We present the design of OP2's heterogeneous back-end which allows an application written using OP2 to be executed on a cluster of GPUs (using MPI and NVIDIA CUDA) and a cluster of multi-threaded CPUs (using MPI and OpenMP). This work specifically focuses on the combination of distributed memory and intra-node or many-core/multi-core parallelism. Key design issues such as (1) handling race-conditions due to data dependencies in indirectly accessed data, and (2) optimizing communications for the target heterogeneous/hybrid parallelization strategies are explored.
2. Performance: A range of performance metrics are benchmarked to explore the performance of OP2's heterogeneous back-end design, including runtime, scalability, achieved compute and bandwidth performance and system energy consumption. Two industrial-representative unstructured mesh application benchmarks (Airfoil and Aero) written using OP2 are used for this study. Benchmarked systems include three large-scale distributed memory systems (a Cray XE6, an Intel Westmere/InfiniBand cluster and a distributed memory Tesla M2090 GPU cluster interconnected by QDR InfiniBand). The OP2 design choices are explored with quantitative insights into their contributions to performance on these systems. We also isolate performance bottlenecks of the application by breaking down the runtime and analyzing the factors constraining total performance. The energy performance measurements and analysis investigate whether the speedups (if any) gained through OP2 on the heterogeneous platforms are achieved within a proportionate energy envelope.

We believe that the design of OP2 for heterogeneous clusters and its performance based on standard benchmarks, forms an important step forward with respect to our previous work [13–16,18,17]. With the inclusion of the heterogeneous back-ends, the range of target platforms supported by OP2 gives us a unique opportunity to carry out an extensive study into the comparative performance of modern systems. As such, this paper details the most comprehensive platform comparison