ClusterSs: A Task-Based Programming Model for Clusters

Enric Tejedor
Barcelona Supercomputing Center, BSC-CNS
Universitat Politècnica de Catalunya, UPC
Barcelona, Spain
enric.tejedor@bsc.es

Montse Farreras
Barcelona Supercomputing Center, BSC-CNS
Universitat Politècnica de Catalunya, UPC
Barcelona, Spain
montse.farreras@bsc.es

David Grove
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598, USA
groved@us.ibm.com

Rosa M. Badia
Barcelona Supercomputing Center, BSC-CNS
Barcelona, Spain
Artificial Intelligence Research Institute (IIIA)
CSIC - Spanish National Research Council (CSIC)
rosa.m.badia@bsc.es

Gheorghe Almasi
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598, USA
gheorghe@us.ibm.com

Jesus Labarta
Barcelona Supercomputing Center, BSC-CNS
Universitat Politècnica de Catalunya, UPC
Barcelona, Spain
jesus.labarta@bsc.es

ABSTRACT

Programming for large-scale, multicore-based architectures requires adequate tools that offer ease of programming while not hindering application performance. StarSs is a family of parallel programming models based on automatic function level parallelism that targets productivity. StarSs deploys a data-flow model: it analyses dependencies between tasks and manages their execution, exploiting their concurrency as much as possible.

We introduce Cluster Superscalar (ClusterSs), a new StarSs member designed to execute on clusters of SMPs. ClusterSs tasks are asynchronously created and assigned to the available resources with the support of the IBM APGAS runtime, which provides an efficient and portable communication layer based on one-sided communication.

This short paper gives an overview of the ClusterSs design on top of APGAS, as well as the conclusions of a productivity study; in this study, ClusterSs was compared to the IBM X10 language, both in terms of programmability and performance. A technical report is available with the details.

Categories and Subject Descriptors

D.1 [**Programming Techniques**]: General

General Terms

Languages, Performance

Keywords

Parallel programming models, High-performance computing, Asynchronous execution, Productivity.

HPDC'11, June 8–11, 2011, San Jose, California, USA.
ACM 978-1-4503-0552-5/11/06.

1. INTRODUCTION

Next-generation architectures are increasing not only in size but also in complexity. In such a scenario, programming productivity is becoming crucial for software developers. Parallel languages and programming models need to provide simple means for developing applications that can run on parallel systems without sacrificing performance. In the last years, the research community has initiated different efforts to create a suitable and robust programming model for such architectures.

StarSs, a family of parallel programming models, represents one of these efforts. Each StarSs member targets a particular architecture [6] [4] [5]; nevertheless, all share the same philosophy: the user is only required to select a set of methods of a sequential application that will run as parallel tasks on the available resources. For each method parameter, its direction (input, output or in-out) needs to be specified; this information is used by StarSs to discover, at execution time, the data dependencies between tasks.

StarSs deploys a data-flow model: at execution time, the user-selected methods are automatically replaced by runtime calls that create the tasks. The runtime analyses the data dependencies between tasks, building a task dependency graph. The parallelism exhibited by the graph is exploited as much as possible, scheduling the dependency-free tasks on the available resources. This execution model helps reducing the critical path in irregular applications. The runtime also manages the data - performing data copies, transfers or renamings if necessary - and controls the completion of the tasks.

Here we introduce Cluster Superscalar (ClusterSs), a new StarSs member which is especially tailored for large-scale clusters. The ClusterSs tasks are asynchronously created and spawned to worker nodes as the application executes. For that purpose, the ClusterSs runtime is built on top of the IBM APGAS (Asynchronous Partitioned Global Address Space) runtime, which provides an efficient and portable layer based on one-sided communication.

The motivation to create ClusterSs was to fill an empty space in the StarSs family, which lacked a member capable of efficiently running applications on clusters. Some of the former StarSs models do not target distributed-memory infrastructures; others can spawn tasks to remote nodes but suffer from scalability issues, caused by factors like a communication system which does not exploit high-performance networks or a master-centric data exchange model.

For ClusterSs to be scalable, it needs a communication system that efficiently supports asynchronous execution and data transfers and allows overlap of communication and computation. On the one hand, asynchronous execution is the foundation for active message (AM) programming, which is the basis of the APGAS communication model. In this sense, the asynchrony of APGAS, already exploited by the X10 language [1, 8] to spawn computations, matches perfectly the StarSs execution pattern. On the other hand, the APGAS communication model is based on one-sided communications, as opposed to the two-sided communication model in MPI; the former has proven to achieve better overlap and scalability for large-scale clusters [2]. Finally, the IBM APGAS runtime implements a fast and portable communication layer that has widely proven its performance for other languages like UPC and X10 [3].

The design of ClusterSs is based on a master-worker model, where nodes can either generate tasks (main node) or execute them (worker nodes). Such design was conceived with scalability in mind. The fact that workers can exchange data, bypassing the main node, helps the latter not becoming a bottleneck. In addition, the data cache maintained by every worker makes possible to exploit data locality and to avoid unnecessary transfers. Regarding the data layout, ClusterSs allows to allocate data directly on the worker nodes via tasks; this prevents the overall memory from being limited to that of the main node, and also removes the need to transfer all the data to the workers at the beginning of the application, which would slow down the execution.

A productivity study of ClusterSs has been performed, comparing it to the X10 language. This study has shown that ClusterSs can bring ease of programming while still providing good performance.

Regarding programmability, a ClusterSs programmer does not have to worry about explicitly spawning asynchronous computations, controlling their dependencies and synchronizing them. Similarly, data transfer and caching and locality-aware scheduling of such computations is also automatically managed by the runtime.

Concerning performance, we considered three applications with different characteristics: (i) *Matrix Multiplication*, (ii) *K-means* and (iii) *Sparse LU*. Our study shows that ClusterSs is able to keep up with the X10 results in (i) and (ii), while being clearly better in (iii). We conclude that ClusterSs performs better than X10 in applications where some data is both read and written, and often a given piece of data updated on one node needs to be read by another node. Such applications usually have complex data dependencies which are hard to control manually in the code. ClusterSs handles those cases with no burden for the programmer, creating a task dependency graph, defining renamings of written data, transferring them if necessary to balance the load and caching them for later use. On the contrary, when the user can partition the application data in such a way that every node only accesses its own fragment or few transfers are required, X10 can manage it more efficiently. Also the possibility of spawning asynchronous computations between any pair of nodes in X10 helps distribute the load.

An extended description of the ClusterSs design, programming model, implementation and productivity study can be found in [7].

Acknowledgments

The researchers at BSC-UPC are supported by the Spanish Ministry of Science and Innovation (contract no. TIN2007-60625 and CSD2007-00050), the European Commission in the context of the HiPEAC Network of Excellence (contract no. IST-004408) and the HPC-Europa2 Research Infrastructure (contract no. 222398), and the MareIncognito project under the BSC-IBM collaboration agreement. This work has also been supported by the Spanish Ministry of Science and Innovation through the Programa Nacional de Movilidad de Recursos Humanos del Plan Nacional de I-D+i 2008-2011, and by the Universitat Politècnica de Catalunya with a UPC Recerca predoctoral grant.

2. REFERENCES

[1] P. Charles, C. Grothoff, V. Saraswat, C. Donawa, A. Kielstra, K. Ebcioglu, C. von Praun, and V. Sarkar. X10: an object-oriented approach to non-uniform cluster computing. In *OOPSLA '05: Proceedings of the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, pages 519–538, New York, NY, USA, 2005. ACM.

[2] M. Farreras. *Optimizing programming models for massively parallel computers*. PhD thesis, Universitat Politecnica de Catalunya, 2008. Advisor-Toni Cortes.

[3] George Almasi, Ganesh Bikshandi, Calin Cascaval, David Cunningham, Gabor Dozsa, Montse Farreras, David P. Grove, Sreedhar B. Kodali, Nathaniel Nystrom, Igor Peshansky, Vijay Saraswat, Sayantan Sur, Olivier Tardieu, Ettore Tiotto. HPC Challenge 2009 Awards Competition: UPC and X10, 2009.

[4] J. M. Perez, R. M. Badia, and J. Labarta. A dependency-aware task-based programming environment for multi-core architectures. In *Proceedings of the 2008 IEEE International Conference on Cluster Computing*, pages 142–151, 2008.

[5] J. M. Perez, P. Bellens, R. M. Badia, and J. Labarta. CellSs: Making it easier to program the cell broadband engine processor. *IBM Journal of Research and Development*, 51(5), August 2007.

[6] E. Tejedor and R. Badia. COMP Superscalar: Bringing GRID superscalar and GCM Together. In *8th IEEE International Symposium on Cluster Computing and the Grid*, May 2008.

[7] E. Tejedor, M. Farreras, D. Grove, R. M. Badia, G. Almasi, and J. Labarta. Clusterss: A task-based programming model for clusters. Technical Report UPC-DAC-RR-2011-14, Universitat Politecnica de Catalunya, 2011.

[8] The X10 project. http://x10-lang.org.