A Framework for Scheduling Scientific Computing Tasks on Heterogeneous Clouds

Brian Peterson Gerald Baumgartner* Qingyang Wang
School of Electrical Engineering and Computer Science
Louisiana State University

Email: {bpete16, gb, qywang}@csc.lsu.edu

October 31, 2014

For many scientific applications, it is not necessary and cost-competitive anymore for universities to maintain their own supercomputers. However, for renting computing resources in the cloud, there is a bewildering variety of different pricing and quality options, ranging from little more than a cent per hour for older machines to a few dollars per hour for dedicated cluster resources. Which resources are most appropriate depends on the computational behavior of the application. While a collection of cheap individual virtual machines would be appropriate, e.g., for a map-reduce application, the communication latency would likely be too high for SIMD matrix multiplications. The computing demands may even vary within an application.

Through funding from the National Science Foundation, we have engaged in a collaborative project with quantum chemists on developing a program transformation system, the *Tensor Contraction Engine* (TCE), that automatically transforms a high-level specification of a computation (expressed as complex tensor contraction expressions) into an optimized parallel program [5, 2]. The application domain for the TCE, electronic structure simulation, has been highlighted in the report of the Department of Energy Workshop on the Role of Computing at the Extreme Scale as one of the important fields underlying advances in basic energy sciences.

Computationally, tensor contraction equations can be reduced to large numbers of matrix multiplications, matrix additions, and higher-dimensional transpositions. Depending on the size of the molecule that is simulated, computational demands range from minutes on a desktop machine to teraflop weeks or more on a supercomputer. Because of the structure of the equations and the symmetric properties of tensors, tensor contractions can be broken up into many smaller tasks of varying sizes, some requiring only an individual node, others performing a SIMD matrix multiplication.

For running such an application on cloud resources, it would be desirable to have run-time scheduling support that selects for each task the appropriate resources depending on the task's requirements. A centralized scheduler with perfect knowledge of each task's requirements and the state of all compute resources, however, is a potential bottleneck. This is due to the size of computational work in the cloud and the fine-grained nature of near-optimal scheduling decisions.

In prior work, we have developed a distributed scheduling approach for a desktop grid infrastructure, the Organic Grid, that measures the performance of the compute resources and dynamically restructures the overlay network for task distribution [3]. We have demonstrated that this approach is feasible for SIMD matrix multiplication.

Based on this work, we envision a cloud computing platform hosted on a set of both virtual and physical machines. Each node in the system periodically measures both its own performance and the communication bandwidth and latency to its neighbors. This information is then propagated to other nodes in the network. An application can then query the performance of the compute resources and decide which task to run on

*Corresponding author.

which (set of) nodes. Such a platform could either be provided as a service by a vendor or, for long-running computations such as quantum chemistry simulations for large molecules, be installed by a user on a rented cloud infrastructure.

The goal for our proposed cloud platform is to allow harnessing any compute resources from local desktop machines to different classes of cloud resources to physical clusters. By deploying the platform on a heterogeneous set of machines, the computational tasks can be run on the most appropriate (and cheapest) resources.

We are planning to implement this platform in Java. Using our implementation approach for strong mobility [4], this also allows us to experiment with scheduling algorithms that migrate tasks if compute resources go away or if the resource requirements of a task change. Each node will, therefore, run a Java agent environment together with code for benchmarking the performance of the node and the network. This performance information is propagated to other nodes in the network in a style similar to routing table update algorithms. An application is written to generate a list of tasks together with performance requirements for each task. Groups of tasks are then encapsulated in a mobile agent that queries the performance data for the network and runs the tasks on the most appropriate nodes. The criteria for selecting nodes can be application-specific, e.g., it could be to minimize completion time or to minimize cost.

Our experiments with the Organic Grid [3] have demonstrated that such an approach to dynamic scheduling is feasible. To make this usable as a cloud platform, however, it is necessary to let the nodes instead of the agents perform the performance measurements and to develop scheduling algorithms appropriate for cloud resources. Our approach has similar goals to Barsness et al.'s air traffic control scheduling algorithm [1]. However, instead of some of the nodes (air traffic controllers) guiding the tasks (aircraft) to the most appropriate resources (destinations), we let the agents encapsulating the tasks (pilots) participate in the scheduling decision, just like in real-world air traffic control.

Our experimental needs for this research are fairly modest. We need access to different classes of cloud resources as are already available in CloudLab and planned for Chameleon, access to storage for storing intermediate results of the computation, and a Java installation.

For scientific applications with complex communication patterns it is currently often necessary to rent dedicated cloud resources at relatively high cost. We have proposed a dynamic scheduling approach in which computational tasks are placed on the most appropriate nodes according to the application's criteria. We propose to use quantum chemistry simulations as driving application and to develop the distributed scheduling algorithms on Chameleon and CloudLab.

References

[1] E. Barsness, D. Darrington, R. Lucas, and J. Santosuosso. *Distributed job scheduling in a multi-nodal environment*. `http://www.google.com/patents/US8645745`, Feb. 4 2014. US Patent 8,645,745.

[2] G. Baumgartner, A. Auer, D. Bernholdt, A. Bibireata, V. Choppella, D. Cociorva, X. Gao, R. Harrison, S. Hirata, S. K. amoorthy, S. Krishnan, C. Lam, Q. Lu, M. Nooijen, R. Pitzer, J. . Ramanujam, P. Sadayappan, and A. Sibiryakov. Synthesis of high-performance parallel programs for a class of ab initio quantum chemistry models. *Proceedings of the IEEE*, 93(2):276–292, February 2005.

[3] A. J. Chakravarti, G. Baumgartner, and M. Lauria. Self-organizing scheduling on the organic grid. *International Journal on High-Performance Computing Applications*, 20(1):115–130, Jan. 2006.

[4] A. J. Chakravarti, X. Wang, J. O. Hallstrom, and G. Baumgartner. Implementation of strong mobility for multi-threaded agents in Java. In *Proceedings of the International Conference on Parallel Processing*, pages 321–330. IEEE Computer Society, Oct. 2003.

[5] A. Hartono, Q. Lu, T. Henretty, S. Krishnamoorthy, H. Zhang, G. Baumgartner, D. E. Bernholdt, M. Nooijen, R. M. Pitzer, J. Ramanujam, and P. Sadayappan. Performance optimization of tensor contraction expressions for many-body methods in quantum chemistry. *The Journal of Physical Chemistry*, 113(45):12715–12723, 2009.