BigDataX 2015: Parallel Programming System Projects

Justin Wozniak, University of Chicago

1 Parallel Programming Systems (Wozniak)

Runtime systems are critical in the support for big data applications. For example, task dependency management needs to be done in a distributed fashion (in order to offer scalable support for data-flow driven programming systems with massive dependency graphs). Scheduling needs to be done in a distributed fashion, as single gateway nodes will not be able to manage millions of jobs to be scheduled to billions of cores; it would be both computationally prohibitive (most scheduling algorithms are NP-complete), and communication prohibitive (maintaining the state of millions of active state-full network connections is not feasible with today's systems). In order to hide latency, we need to overlap computations with communication. In order to better support dynamic systems, run-time systems should also support dynamic provisioning, allowing the system to be easier to make resilient as well as making it usable in a cloud computing environment. We need the framework to support heterogeneous hardware (e.g. accelerators with many-core architectures) to ensure it is compatible with the growing segment of high-end computing. And run-time systems need HPC support, in order to have a broader appeal to traditional high-end computing systems and applications. This work will extend two existing research projects, namely Swift [2, 3, 5] and MATRIX [9] with the following enhancements.

Distributed Scheduling: In order to improve both performance and reliability, clients will be able to submit computational jobs into the execution fabric by submitting to any compute node. The fabric will guarantee jobs execution, and would re-execute failed or slow jobs. However, this distribution of the jobs/tasks and the state of the resources on which they run on, introduces significant challenges in using traditional scheduling algorithms, which rely on centralized state management. The PIs will leverage the distributed load balancing algorithms (e.g. work-stealing) towards decentralized scheduling algorithms.

Data-Aware Scheduling: The FusionFS file system (ongoing work funded by PI Raicu's NSF CAREER award) will have data locality information as part of the metadata information, which will be stored in ZHT. A data-aware scheduler could query ZHT for the metadata for a given set of files; the retrieved metadata will contain the location (e.g. node ids) of the files/blocks. The data-aware scheduler can then schedule tasks close to data to minimize data movement, or schedule tasks on the first free core/node to minimize wait time. The underlying storage system can also be monitored for hot spots, and trends can be used to attempt to predict future contention points, or even future I/O burst requests. The PI Raicu has significant experience data-aware scheduling (data diffusion [4] in Falkon [1] and Swift [6]). The data diffusion data-aware scheduler is centralized, and its scalability has only been shown to thousands of cores [4]. A distributed data-aware scheduler, in which every compute node keeps track of a local caches and can make autonomous decisions about data movement, will be needed in MATRIX to scale to exascales.

Distributed Task Dependency Management: MATRIX [9] will have many queues distributed across compute nodes/processors (e.g. 1:1 mapping of queues to nodes). There will be no ordering requirement on unrelated items in the queues, and therefore expensive synchronization overheads among queues can be avoided. However, there will support job dependencies, and MATRIX will guarantee that dependent jobs are executed in the correct order to satisfy the data-flow or task dependencies. Distributed key/value stores such as ZHT [8] should make a good building block to maintain globally accessible task lookup functionality; for reliability, ZHT replicates stored data to avoid loss in case of failures.

Dynamic Execution and Dataflow: MATRIX tasks may create new tasks and block on other tasks. We plan to enhance this model by enabling a task to block on data, thus building a highly dynamic execution model. Human-in-the-loop manipulation of data objects enables run time steering of the whole application. Combining dynamic steering of DAGs with support for ensemble MPI applications enables MATRIX to support dynamic steering of HPC workloads.

Heterogeneous Computing: Accelerators (e.g. GPUs, MIC, FPGAs) with 1000s of cores are already available today, and general purpose manycore processors with 100s to 1000s of cores will be common place by the end of the decade. The PI has started (with small seed funding from IIT) the GeMTC project [7] aiming to support accelerators, such as NVIDIA GPUs and Xeon Phi. The GeMTC framework allows MTC applications implemented through the Swift parallel programing system to run on accelerators

efficiently. A related project PI Raicu has is how a given application's task graph described by MTC applications, maps the tasks to the best cores in order to maximize performance while minimizing power consumption.

All of these projects are quite ambitious on their own, but with the right guidance from the mentor Wozniak and senior PhD students who have been working on topics for many years, we believe that students with a solid systems background could be productive on these topics over a summer 10-week program.

2 Mentor

Dr. **Justin M. Wozniak** (mentor) is an Assistant Computer Scientist at Argonne National Laboratory and a research staff member at the Computation Institute at the University of Chicago. He received his Ph.D. in 2008 from the University of Notre Dame. His research focuses on high-performance computing, language and runtime development, data management and storage, and scientific applications. He developed a novel implementation of the Swift language for use on high-performance computers such as the Cray XE6 and IBM Blue Gene/Q. He is co-chair and program committee chair of MTAGS, an SC workshop on many-task computing. He has informally guided the research work of many postdocs, graduate students, and undergraduates, and was a Google Summer of Code mentor and summit attendee.

3 References

[1] I. Raicu, Y. Zhao, C. Dumitrescu, I. Foster, M. Wilde. "Falkon: A Fast and Light-weight tasK executiON Framework," IEEE/ACM SC 2007

[2] Y. Zhao, M. Hategan, B. Clifford, I. Foster, G. von Laszewski, I. Raicu, T. Stef-Praun, M. Wilde. "Swift: Fast, Reliable, Loosely Coupled Parallel Computation," IEEE Workshop on Scientific Workflows 2007

[3] M. Wilde, I. Foster, K. Iskra, P. Beckman, Z. Zhang, A. Espinosa, M. Hategan, B. Clifford, I. Raicu. "Parallel Scripting for Applications at the Petascale and Beyond", IEEE Computer Nov. 2009 Special Issue on Extreme Scale Computing, 2009

[4] I. Raicu, I. Foster, Y. Zhao, P. Little, C. Moretti, A. Chaudhary, D. Thain. "The Quest for Scalable Support of Data Intensive Workloads in Distributed Systems", ACM HPDC09

[5] Y. Zhao, I. Raicu, I. Foster, M. Hategan, V. Nefedova, M. Wilde. "Realizing Fast, Scalable and Reliable Scientific Computations in Grid Environments", Grid Computing Research Progress, Nova Publisher 2008

[6] Swift, http://www.ci.uchicago.edu/swift/, 2010

[7] Scott J. Krieder, Justin M. Wozniak, Timothy Armstrong, Michael Wilde, Daniel S. Katz, Benjamin Grimmer, Ian T. Foster, Ioan Raicu. "Design and Evaluation of the GeMTC Framework for GPU-enabled Many-Task Computing", ACM HPDC 2014

[8] Tonglin Li, Xiaobing Zhou, Kevin Brandstatter, Dongfang Zhao, Ke Wang, Anupam Rajendran, Zhao Zhang, Ioan Raicu. "ZHT: A Light-weight Reliable Persistent Dynamic Scalable Zero-hop Distributed Hash Table", IEEE International Parallel & Distributed Processing Symposium (IPDPS) 2013

[9] Ke Wang, Anupam Rajendran, Ioan Raicu. MATRIX: Many-Task Computing Execution Fabric for Extreme Scales, Illinois Institute of Technology, Department of Computer Science, Technical Report, 2013.