Exploring Asynchronous Many-Task Runtime Systems toward Extreme Scales

[Poster Summary]

Samuel Knight[§], Marc Gamell[†], Gavin Baker[¶], David Hollman[‡], Gregory Sjaardema[‡], Keita Teranishi[‡], Hemanth Kolla[‡], Jeremiah Wilke[‡], Nicole Slattengren[‡], and Janine Bennett[‡]

[§] Florida Institute of Technology, Melbourne, FL, USA
skingh@sandia.gov

[¶] California Polytechnic State Univerisity, San Luis Obispo, CA, USA
gmbaker@sandia.gov

[†]Rutgers Discovery Informatics Institute, Rutgers University, Piscataway, NJ, USA
mgamell@cac.rutgers.edu

[‡]Sandia National Laboratories, Livermore, CA, Albuquerque, NM, USA,
{*dshollm, gdsjaar, knteran, hnkolla, jjwilke,nlslatt, jcbenne*}*@sandia.gov*

ABSTRACT

Major exascale computing reports indicate a number of software challenges to meet the dramatic change of system architectures in near future. While several-orders-of-magnitude increase in parallelism is the most commonly cited of those, hurdles also include performance heterogeneity of compute nodes across the system, increased imbalance between computational capacity and I/O capabilities, frequent system interrupts, and complex hardware architectures. Asynchronous task-parallel programming models show a great promise in addressing these issues, but are not yet fully understood nor developed sufficiently for computational science and engineering application codes.

We address these knowledge gaps through quantitative and qualitative exploration of leading candidate solutions in the context of engineering applications at Sandia. In this poster, we evaluate MiniAero code ported to three leading candidate programming models (Charm++, Legion and UINTAH) to examine the feasibility of these models that permits insertion of new programming model elements into an existing code base.

1. INTRODUCTION

System architectures for extreme-scale computing are projected to be drastically different from current architectures which poses challenges in terms of performance, scalability and reliability. In particular, the several-orders-of-magnitude increase in parallelism with less proportional increase in the main memory capacity will pose a huge scalability challenge. This increase in parallelism is further complicated by increased imbalance between computational capacity and I/O capabilities, performance heterogeneity due to OS noise and thermal throttling, and complex hardware architectures. In response to these scalability and performance challenges, a number of asynchronous many-task (AMT) programming models [1–3, 7, 8] have emerged as alternatives to the single program multiple data model (SPMD) in which a sequential communicating "task" in each process is executed. In an AMT model, a program is viewed as a flow of data processed by many tasks, each of which executes a distinct kernel. During program execution, these tasks are launched in any order based on the availability of their input data, enabling multiple concurrent task execution on available computational resources, something which is difficult to optimize by hand for the SPMD model. The load imbalance is automatically managed by a task scheduler associated with the runtime, rather than explicit load re-balancing through data partitioning tools [4,5,9]. Despite these advantages in AMT models over SPMD, they are not yet fully understood nor developed sufficiently to support a variety of computational science and engineering applications. For example, one of large scale multi-physics engineering simulations require scalable implementations of several application components, including meshing, matrix assemblies, implicit solvers and preconditioners, particle interactions, and data (in-situ/in-transit) analyses. For production use, AMT programming models need to be capable of coupling these components in a seamless manner.

In the poster, we address our knowledge gap in AMT models through quantitative and qualitative exploration of the leading candidate solutions in the context of engineering applications at Sandia. For the initial step, we have evaluated MiniAero code, ported to three leading candidate programming models (Charm++, Legion an Uintah) [2, 3, 8]. Our evaluation includes (1) a qualitative ranking of each model based on our porting experience , (2) scalability test of Mini-

SC'15, November 15-20, 2015, Austin, TX, USA

DOI: 10.1145/1235

Aero including the original MPI version and (3) assessment of dynamic load balancing capability under performance heterogeneity.

2. EVALUATION

Asynchronous Many Task Runtime.

In our studies, we have selected three AMT programming models (Charm++ [8], Legion [2], and Uintah [3]) based on the following criteria:

1. Performance demonstrated at scale
2. Maturity of runtime
3. Different methodologies in implementation and abstractions of data and tasks
4. Accessibility of developers

The goal of our studies is to identify the characteristics in programmability, relevance to out target multi-physics applications (Computational Fluid Dynamics and Particles in Cell), and performance at extreme scale.

MiniAero.

MiniAero is designed to represent parallel CFD simulations, which employs finite volume method for 3 dimensional unstructured mesh. For the numerical solution, explicit Runga-Kutta 4th order (RK4) is implemented for time marching, and inviscid Roe Flux and three types of boundary conditions are supported. The code is written in C++ and MPI-2.2 (3800+ lines total) with extensive use of C++ templates and Kokkos [6] for the parallelizing the loops in single node level.

Qualitative Evaluation.

Our porting experience reveals several feasibility issues for the development of Sandia's multi-phyiscs simulation code. We have evaluated the three AMT programming models through the metrics listed in Table 1 to identify the requirements of future AMT programming models. Note that these metrics are inherently subjective, but still serve our purpose. In summary, we found that none of these models meet our requirements as indicated by Figure 1.

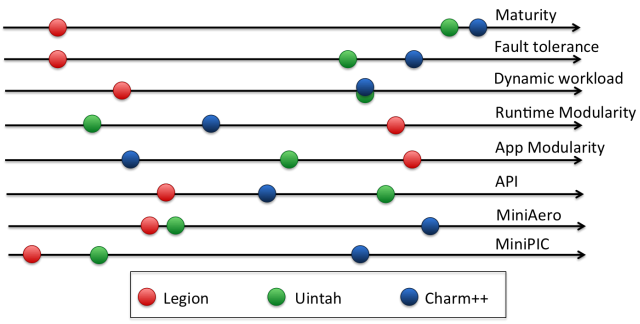

Figure 1: Qualitative Evaluation of Three AMT models based on the porting experience of MiniAero.

Quantitative Evaluation.

We made two different performance evaluations: (1) scalability test and (2) load balancing studies. For scalability test, we execute three versions of MiniAero implementations (MPI, Charm++ and UINTAH) on Cielo (Cray XE6) system at Sandia and Los Alamos National Laboratory. For the initial load balancing study, we execute Charm++ version of MiniAero on 16 nodes of a PC cluster with power throttling applied to several nodes to artificially create performance heterogeneity across the nodes in the range of 1.2 to 2.3GHz. Figure 2 highlights the load-balancing capability of Charm++, which can mitigate the worst scenario when over-decomposition is enabled with its load-balancer.

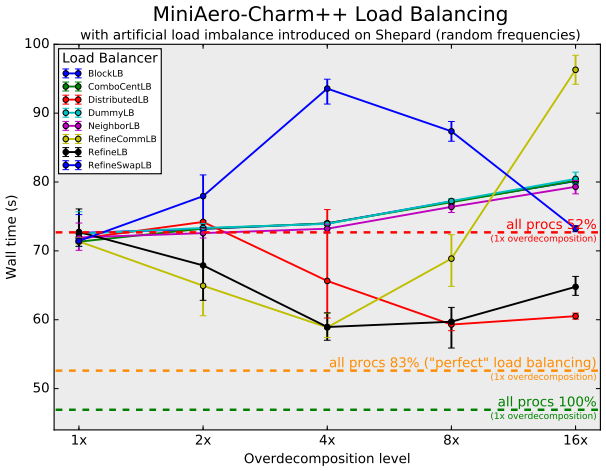

Figure 2: Performance of Charm++ version of MiniAero. The power of individual nodes are set to between 52% and 100% (83% in average).

3. CONCLUSION

Our preliminary qualitative study indicates that all leading AMT programming models and runtime systems can meet the requirement for a limited set of applications, and none of them are production ready for broad classes of applications. However, these AMT models show tremendous potential for addressing extreme scale challenges; in particular, load-balancing capability of Charm++ can handle the performance heterogeneity anticipated in future extreme-scale computing systems. Future work includes the completion of the performance study on Legion and Uintah on the scalability and load-balancing capability and more detailed qualitative study with other Sandia's applications (multi-physics simulations and Particles-in-Cell code) in order to identify the requirements for general-purpose scalable AMT runtime and software.

Acknowledgments

The authors would like to thank National Energy Research Scientific Computing Center and Karen Pao at DOE Office of Science for allocating a large amount their MPP system resources. The authors also thank Nikhil Jain, Laxmikant Kale and Eric Mikida at University of Illinois, Martin Berzins, Todd Harman, and Alan Humphrey at University of Utah,

Categories	Description
Maturity	How stable is the API? Likelihood of encountering bugs?
Fault tolerance support	What fault models/recovery mechanisms are supported? How easy is the API?
Dynamic workload support	What load-balancing/work stealing mechanisms are supported? How easy is the API?
Modularity (Runtime and Application)	How reusable are components? How extensible is the framework?
APIs	What do developers like/dislike regarding the interface?
MiniAero	How easy is it to express MiniAero?
MiniPIC	How easy is it to express Mini version of Particles in Cell code?

Table 1: Metrics of qualitative evaluation. Value and relevance measures may differ across the users.

and Alex Aiken, Michael Bauer, Wonchan Lee, Elliot Slaughter and Sean Treichier at Stanford University for the consistent code development assistance. Finally, the authors thank Robert Clay, David DeBonis, Ryan Grant, Simon Hammond, Victor Kuhns and Stephen Olivier for interesting discussions and infrastructure support. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

4. REFERENCES

[1] C. Augonnet, S. Thibault, R. Namyst, and P.-A. Wacrenier. StarPU: A unified platform for task scheduling on heterogeneous multicore architectures. *Concurr. Comput. : Pract. Exper.*, 23(2):187–198, Feb. 2011.

[2] M. Bauer, S. Treichler, E. Slaughter, and A. Aiken. Legion: Expressing locality and independence with logical regions. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '12, pages 66:1–66:11, Los Alamitos, CA, USA, 2012. IEEE Computer Society Press.

[3] M. Berzins, J. Schmidt, Q. Meng, and A. Humphrey. Past, present and future scalability of the uintah software. In *Proceedings of the Extreme Scaling Workshop*, BW-XSEDE '12, pages 6:1–6:6, Champaign, IL, USA, 2012. University of Illinois at Urbana-Champaign.

[4] E. G. Boman, U. V. Çatalyürek, C. Chevalier, and K. D. Devine. The zoltan and isorropia parallel toolkits for combinatorial scientific computing: Partitioning, ordering and coloring. *Sci. Program.*, 20(2):129–150, Apr. 2012.

[5] C. Chevalier and F. Pellegrini. PT-Scotch: A tool for efficient parallel graph ordering. *Parallel Comput.*, 34(6-8):318–331, July 2008.

[6] H. C. Edwards and C. R. Trott. Kokkos: Enabling performance portability across manycore architectures. In *Proceedings of the 2013 Extreme Scaling Workshop (Xsw 2013)*, XSW '13, pages 18–24, Washington, DC, USA, 2013. IEEE Computer Society.

[7] C. Huang, C. W. Lee, and L. Kale. Support for adaptivity in armci using migratable objects. In *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, pages 9 pp.–, April 2006.

[8] L. V. Kale and A. Bhatele, editors. *Parallel Science and Engineering Applications: The Charm++ Approach.* Taylor & Francis Group, CRC Press, Nov. 2013.

[9] J. D. Teresco, J. E. Flaherty, S. B. Baden, J. Faik, S. Lacour, M. Parashar, V. E. Taylor, and C. A. Varela. *Parallel Processing for Scientific Computing*, chapter Approaches to Architecture-Aware Parallel Scientific Computation, pages 33–58. SIAM, 2006.