Actionable Performance Modeling for Future Supercomputers

Laxmikant V. Kale and Osman Sarood‡
Eric Bohm, Nikhil Jain, Akhil Langer and Esteban Meneses
Department of Computer Science, University of Illinois at Urbana-Champaign
‡{kale,sarood1}@illinois.edu

1. INTRODUCTION

As we go beyond the current scale of computers to those with peak capabilities beyond an ExaFLOP/s, it is becoming clear that an introspective and adaptive runtime system (RTS) will be essential, to deal with the complexities generated by sophisticated applications and complex machines. The applications will incorporate adaptive numerical algorithms, such as dynamic adaptive mesh refinements, and multi-time-stepping. The machines will exhibit static and dynamic variability, including component failures/errors. The RTS will need to make quick decisions by adjusting machine configurations (e.g. processors used, power levels of each component, etc.), runtime strategies (e.g. changing scheduling strategy, selecting load balancers, or parameterizing strategies) and application. For making such decisions quickly, it needs simple but effective models of various subcomponents of the parallel machine and the application to predict how they will behave under a reconfiguration the RTS is considering. Such fast models, which may sacrifice some accuracy in return for extreme speed, are called *Actionable Models* in the rest of this paper.

The points we want to make in this position paper are: introspective and adaptive RTS are both feasible and necessary at exascale, and will be pursued by DOE's x-stack projects, exa-OS/R projects, as well other projects in the broad HPC community. To make them successful, they need to be complemented by development of fast modeling techniques for various system/application components. Figure 1 gives a high level view of the suggeted infrastructure. The instrumentor module will be responsible to interact with the machine hardware and runtime system to retrieve instantaneous hardware and application characteristics. The RTS will provide statistics about the execution time for the application whereas hardware will provide information such as the core temperatures, current frequency and power levels at which the processors are operating. The instrumentor will feed these characteristics to the appropriate actionable model. Each actionable model uses these *current* characteristics to make relevant predictions. Since multiple modules can be interdependent, the global control system in Figure 1 is responsible for receiving the predictions from each of the modules and deciding what are the best actions that can be taken to improve application performance. The global control system is responsible to do a cost-and-benefit analysis of the actions it decides to suggest. Based on these actions, the RTS chooses the appropriate strategy and mechanism to implement it. In the following section we lay out an agenda for each module.

2. INDIVIDUAL MODELS

Application Evolution: Systems such as Charm++ [1], and Trilinos [2], support dynamic load balancing even on current systems. However, they typically use very simple models: either average measurements of the past behavior of objects are used as predictions for future load, or some simple metrics (sometime supplied by the user) based on counts of application entities (e.g. number of particles inside a box) are used to predict performance. Further, decisions about when to balance load and which of the many possible strategies to use, each with its pros and cons, are typically left to the application developer. These need to be automated at extreme scale because leaving it to the application developer to handle the load balancing decisions for dynamic applications is inefficient. For this purpose, more sophisticated models of evolution of load of individual objects are needed. Also needed are models that can predict the load balancing cost which includes the cost of making load balancing decisions, as well as the cost for data migrations. An example of a preliminary attempt in this directions is the "meta-balancer" [3] developed recently, which uses linear models for load evolution, collects simple global metrics asynchronously, uses data from earlier iterations for estimating costs of balancing, and a simple and fast algebraic cost/benefit analysis to decide when to balance load. Such approach needs to be made more sophisticated, and combined with application behavior models to decide which types of load balancers are most suited for the application at that particular point in time.

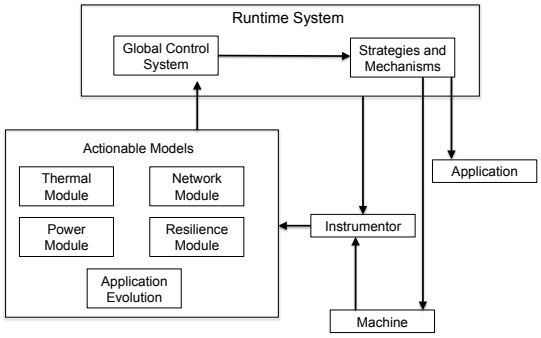

Figure 1: Interaction between actionable models, runtime system, application and machine.

Thermal: Recent work shows that restraining processor temperature is possible using Dynamic Voltage and Frequency Scaling (DVFS) [4]. The runtime system can now leverage the newly introduced power capping features that allow restricting the package and memory power. However, processor temperature is also affected by the level of cooling in the machine room. Due to differences amongst chips, the temperature-power relation for different chips might vary.

Similarly, the temperature of the cool air/water cooling all the processors would not be the same. The job of the thermal module would be to predict processor temperature given a power level for the processor and the temperature of the cool air/water that is used to cool that processor.

Power: As we move towards the exascale era, the thrust is shifting from attaining high energy efficiency towards developing an exascale system with a power budget of 20MW. The latter has been a much harder challenge. Therefore, the focus now is to maximize performance under a given power budget. The thermal design power (TDP) of a processor die or a memory subsystem refers to the maximum amount of power that it can draw. Power requirements of data centers are calculated using the TDP of its subsystems. However, in practice, the power draw of subsystems very rarely reaches the TDP during the course of execution of an application. Nonetheless, respective TDP has to be kept aside for each of the subsystems. Recent microprocessor architectures such as IBM Power6 [5], Power7 [6], AMD Bulldozer [7], and Intel's Sandy Bridge [8] provide a very attractive option of limiting the power drawn by the processor die. Additionally, motherboards are also supporting limiting the memory power draw. These features have been studied recently [9] to propose what is called an over provisioned system in which the application's optimal configuration which includes the number of nodes, processor power, and memory power, is determined in such a way that the applications performance is maximized under a given power budget. Over-provisioning exploits the fact that applications do not yield a proportional improvement in performance as the processor/memory power is increased, and therefore the power can instead be used to add more nodes and strong scale the application [10].

Future supercomputers will exploit this idea to maximize their power efficiency i.e. flops/watt. Execution on a processor can be broken down into a series of sequential execution blocks (SEB), each with no remote dependency inside. Different SEBs will have different power characteristics. The Modeling system will profile the power characteristics of SEBs either during runtime or offline and generate a table of the execution time (and other characterizer's such as energy consumption) of these SEBs for different processor and memory powers. Runtime model development can leverage the availability of large number of processors by doing paratmetric runs of SEBs at different power levels, and use parallel implementations of machine learning models to build these tables. Different strategies can then be used to optimally allocate power to SEBs within an application. Additionally, job schedulers will use the application's power characteristics to schedule jobs and assign compute resources to them in a manner that maximizes the overall power efficiency of the data center while still maintaining fairness amongst jobs. This will require that the jobs have the runtime capability to *shrink* and *expand* to the scheduler specified number of nodes. Whenever a job terminates, scheduler will re-optimize resource allocation to the running and waiting jobs. Optimizing resource allocation will require solving linear programs which we have ascertained to be solvable, even for exascale resource optimization, within few seconds using the state-of-the-art linear program solvers [11, 12].

Shrinking and expanding a job at run-time is also useful in other contexts, such as HPC in cloud environments, or in Adaptive Mesh Refinement [13] applications, where the computational load changes drastically as the simulation progresses. Predictive power models are very useful in them.

Resilience: The mean-time-between-failures (MTBF) of current supercomputers can be anywhere from 2-10 hours [14, 15]. Optimistic estimations predict that the MTBF for an exascale machine will be between 35-40 minutes [16]. Machine reliability is fast becoming a limiting constraint for future machines. This implies that failures would become a norm for future large-scale machines and faulty environments should be embraced. Up to now, HPC researchers have mainly focussed on developing efficient fault tolerance protocols with little attention on improving the reliability of a machine. Failure rate of a compute node doubles with every 10°C increase in temperature [17, 18, 19, 20]. Given that applications consume different amounts of power, they end up heating the CPU and memory to different temperature levels. In our recent work [21], we use temperature capping to model the effect of core temperatures on the MTBF of a system. Restraining core temperatures can empower a data center operator to *choose* a reliability level [21]. This improvement in reliability resulting from temperature capping can significantly improve the execution time of an application at large scale [21].

The MTBF for a processor and hence for a system, depends on multiple factors [22]. Our existing model includes the effects of core temperatures. This can be extended by incorporating other variables into it. An extended model will take into account the instantaneous core temperatures and the thermal history of the machine in order to predict the MTBF of a system. Using this estimated MTBF, the runtime system can recalculate the optimum checkpointing period over the course of execution. The cost of incorporating such a model is minimal. The temperature readings for each core can be read very cheaply using the `coretemp` module installed in the kernel.

Network: Offline analysis of network behavior via complex models and simulation is a well researched topic [23, 24, 25]. Recent work has used offline analysis of network and application communication pattern to predict task mappings used during job startup resulting in improved performance [26, 27]. Further, simplified models, based on optimizating metrics such as hop-bytes and maximum dilation, have been deployed to reassign tasks during application execution [28, 29, 30]. As applications become more complex, and exhibit dynamic and irregular behavior, it is natural for runtime to monitor the application as well as the network, and take actions to maximize performance.

With the help of runtime, application's communication graph can be captured and modeled. Different communication phases and patterns can be identified, and actions can be taken to improve performance. Based on information on application's behavior and the topology of allocated nodes, one can predict performance implication of parametrization and use of various communication algorithms, optimized communication libraries, and special hardware. In addition, the congestion traits of a network can be captured with help from the runtime, e.g. by running periodic ping-pong in the background or using feedback from the hardware, to capture any variation in network's behavior due to the application or other jobs. Based on this information, a runtime is capable of using basic modelling to great effect. A runtime may choose to change the overlaying tree used for a collective, select a different scheme for the same collective, temporarily discard certain nodes, reroute messages etc.

Effort: A significant portion of the effort required to bring this vision to fruition will be in the RTS, presumably being developed as a part of the x-stack effort and OS/R effort. Developing actionable models requires expertise in

multiple domains, and so we estimate 3 FTEs and 4 graduate assistants as the effort level needed.

3. REFERENCES

[1] Laxmikant Kale, Anshu Arya, Nikhil Jain, Akhil Langer, Jonathan Lifflander, Harshitha Menon, Xiang Ni, Yanhua Sun, Ehsan Totoni, Ramprasad Venkataraman, and Lukasz Wesolowski. Migratable Objects + Active Messages + Adaptive Runtime = Productivity + Performance A Submission to 2012 HPC Class II Challenge. Technical Report 12-47, Parallel Programming Laboratory, November 2012.

[2] Michael A. Heroux, Roscoe A. Bartlett, Vicki E. Howle, Robert J. Hoekstra, Jonathan J. Hu, Tamara G. Kolda, Richard B. Lehoucq, Kevin R. Long, Roger P. Pawlowski, Eric T. Phipps, Andrew G. Salinger, Heidi K. Thornquist, Ray S. Tuminaro, James M. Willenbring, Alan Williams, and Kendall S. Stanley. An overview of the Trilinos project. *ACM Trans. Math. Softw.*, 31(3):397–423, 2005.

[3] Harshitha Menon, Nikhil Jain, Gengbin Zheng, and Laxmikant V. Kalé. Automated load balancing invocation based on application characteristics. In *IEEE Cluster 12*, Beijing, China, September 2012.

[4] Osman Sarood Phil Miller Ehsan Totoni and Laxmikant Kale. 'cool' load balancing for hpc data centers. *IEEE transactions on computers special issue on energy efficient computing*, 2012.

[5] Brad Behle, Nick Bofferding, Martha Broyles, Curtis Eide, Michael Floyd, Chris Francois, Andrew Geissler, Michael Hollinger, Hye-Young McCreary, Cale Rath, et al. IBM Energyscale for POWER6 Processor-based Systems. *IBM White Paper*, 2009.

[6] Martha Broyles, Chris Francois, Andrew Geissler, Michael Hollinger, Todd Rosedahl, Guillermo Silva, Jeff Van Heuklon, and Brian Veale. IBM Energyscale for POWER7 Processor-based Systems. *white paper, IBM*, 2010.

[7] Advanced Micro Devices. BIOS and Kernel DeveloperâĂŹs guide (BKDG) for AMD Family 15h Models 00h-0fh Processors. January 2012.

[8] Barry Rountree, Dong H Ahn, Bronis R de Supinski, David K Lowenthal, and Martin Schulz. Beyond DVFS: A First Look at Performance Under a Hardware-enforced Power Bound. In *IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW)*, 2012.

[9] T. Patki, D. Lowenthal, B. Rountree, M. Schulz, and B. Supinski. Exploring Hardware Overprovisioning in Power-Constrained, High Performance Computing. In *Proceedings of the International Conference on Supercomputing*, pages 222–231, 2013.

[10] Osman Sarood, Akhil Langer, Laxmikant Kalé, Barry Rountree, and Bronis de Supinski. Optimizing Power Allocation to CPU and Memory Subsystems in Overprovisioned HPC Systems. In *Cluster Computing (CLUSTER), 2013 IEEE International Conference on*, pages 1–8. IEEE, 2013.

[11] Gurobi Optimization Inc. Software, 2012. `http://www.gurobi.com/`.

[12] IBM CPLEX Optimization Studio. Software, 2012. `http://www-01.ibm.com/software/integration/optimization/cplex-optimization-studio/`.

[13] Akhil Langer, Jonathan Lifflander, Phil Miller, Kuo-Chuan Pan, Laxmikant V Kale, and Paul Ricker. Scalable Algorithms for Distributed-Memory Adaptive Mesh Refinement. In *Computer Architecture and High Performance Computing (SBAC-PAD), 2012 IEEE 24th International Symposium on*, pages 100–107. IEEE, 2012.

[14] E. N. Elnozahy, R. Bianchini, T. El-Ghazawi, A. Fox, F. Godfrey, A. Hoisie, K. McKinley, R. Melhem, J. S. Plank, P. Ranganathan and J. Simons. System resilience at extreme scale. Defense Advanced Research Project Agency (DARPA), Tech. Rep., 2008.

[15] Marc Snir, William Gropp, and Peter Kogge. Exascale Research: Preparing for the Post Moore Era. `https://www.ideals.illinois.edu/bitstream/handle/2142/25468/Exascale%20Research.pdf`, 2011.

[16] Peter Kogge, Keren Bergman, Shekhar Borkar, Dan Campbell, William Carlson, William Dally, Monty Denneau, Paul Franzon, William Harrod, Jon Hiller, Sherman Karp, Stephen Keckler, Dean Klein, Robert Lucas, Mark Richards, Al Scarpelli, Steven Scott, Allan Snavely, Thomas Sterling, R. Stanley Williams, and Katherine Yelick. Exascale computing study: Technology challenges in achieving exascale systems, 2008.

[17] Chung hsing Hsu, Wu chun Feng, and Jeremy S. Archuleta. Towards efficient supercomputing: A quest for the right metric. In *In Proceedings of the HighPerformance Power-Aware Computing Workshop*, 2005.

[18] Wu-chun Feng. Making a case for efficient supercomputing. volume 1, pages 54–64, New York, NY, USA, October 2003. ACM.

[19] Wu-chun Feng. The Importance of Being Low Power in High-Performance Computing. *Cyberinfrastructure Technology Watch Quarterly (CTWatch Quarterly)*, 1(3), August 2005.

[20] Ericsson. Reliability Aspects on Power Supplies. *Technical ReportDesign Note 002, Ericsson Microelectronics, April 2000.*

[21] O. Sarood, E. Meneses, and L. Kale. A Cool Way of Improving the Reliability of HPC Machines. In *Supercomputing 2013 In Submission*, 2013.

[22] J. Srinivasan, S.V. Adve, P. Bose, and J.A. Rivers. Lifetime reliability: toward an architectural solution. *Micro, IEEE*, 25(3):70–80, 2005.

[23] Gengbin Zheng, Gunavardhan Kakulapati, and Laxmikant V. Kalé. Bigsim: A parallel simulator for performance prediction of extremely large parallel machines. In *18th International Parallel and Distributed Processing Symposium (IPDPS)*, page 78, Santa Fe, New Mexico, April 2004.

[24] Kathryn Berkbigler, Graham Booker, Brian Bush, Kei Davis, and Nicholas Moss. Simulating the Quadrics Interconnection Network. In *High Performance Computing Symposium 2003, Advance Simulation Technologies Conference 2003*, Orlando, Florida, April 2003.

[25] K.D. Underwood, M. Levenhagen, and A. Rodrigues. Simulating red storm: Challenges and successes in building a system simulation. In *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, pages 1 –10, 2007.

[26] Abhinav Bhatele, Nikhil Jain, William D. Gropp, and Laxmikant V. Kale. Avoiding hot-spots on two-level

direct networks. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '11, pages 76:1–76:11, New York, NY, USA, 2011. ACM.

[27] Venkatesan T Chakaravarthy, Naga Praveen Kumar Katta, Monu Kedia, Yogish Sabharwal, Aruna Ramanan, and Ramakrishnan Rajamony. Mapping Strategies for the PERCS Architecture. In *19th annual IEEE International Conference on High Performance Computing (HiPC 2012)*, December 2012.

[28] Abhinav Bhatele, Todd Gamblin, Steven H. Langer, Peer-Timo Bremer, Erik W. Draeger, Bernd Hamann, Katherine E. Isaacs, Aaditya G. Landge, Joshua A. Levine, Valerio Pascucci, Martin Schulz, and Charles H. Still. Mapping applications with collectives over sub-communicators on torus networks. In *Proceedings of the ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '12. IEEE Computer Society, November 2012 (to appear). LLNL-CONF-556491.

[29] Torsten Hoefler and Marc Snir. Generic topology mapping strategies for large-scale parallel architectures. In *Proceedings of the international conference on Supercomputing*, ICS '11, pages 75–84, New York, NY, USA, 2011. ACM.

[30] Abhinav Bhatele. Topology Aware Task Mapping. In D. Padua, editor, *Encyclopedia of Parallel Computing (to appear)*. Springer Verlag, 2011.