LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-CONF-685097

Systemwide Power Management with Argo

D. Ellsworth, T. Patki, S. Perarnau, S. Seo, K. Yoshii, H. Hoffman, M. Schulz, P. Beckman

March 7, 2016

High Performance Power Aware Computing
Chicago, IL, United States
May 23, 2016 through May 27, 2016

Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.

Systemwide Power Management with Argo

Daniel Ellsworth*, Tapasya Patki†, Swann Perarnau‡, Sangmin Seo‡, Abdelhalim Amer‡,
Judicael Zounmevo‡, Rinku Gupta‡, Kazutomo Yoshii‡, Henry Hoffman§,
Allen Malony*, Martin Schulz†, and Pete Beckman‡
*University of Oregon
†Lawrence Livermore National Laboratory
‡Argonne National Laboratory
§The University of Chicago

***Abstract*—The Argo project is a DOE initiative for designing a modular operating system/runtime for the next generation of supercomputers. A key focus area in this project is power management, which is one of the main challenges on the path to exascale. In this paper, we discuss ideas for systemwide power management in the Argo project. We present a hierarchical and scalable approach to maintain a power bound at scale, and we highlight some early results.**

I. Introduction

Exascale supercomputing poses several new challenges. These include tuning for upcoming hardware technologies and changing application workflows, adhering to strict power constraints, and addressing resilience and system scalability concerns [4]. It is thus expected that exascale systems will have to manage and coordinate a massive number of resources with varying system-level constraints. The current system software stack thus needs to be redesigned in order to support future supercomputers and to address the aforementioned challenges. The Argo project [20], [21], funded under the DOE ExaOSR initiative, caters to this design issue and has the goal of providing a flexible and modular operating system and runtime for extreme-scale scientific computing.

Argo approaches the management of resources through recursive partitioning and facilitates both global, system-level optimizations and local control of resources at a fine granularity. The core of the project involves four focus areas for innovation: reconfiguring of node resources dynamically in response to a workload, support for massive concurrency, a hierarchical framework for power and fault management, and a communication infrastructure or *beacon* mechanism that allows resource managers and optimizers to share information and control the platform.

In this paper, we present ideas for scalable, systemwide power management from the Argo project. Section II introduces the power problem and presents related work. Section III describes the Argo model in detail, and Section IV discusses the power infrastructure. We highlight some early results in Section V and conclude in Section VI with a brief summary and a glance at future work.

II. Power and Energy

Future supercomputers are expected to be extremely power-constrained. Utilizing the available power efficiently and translating the limited amount of power into application performance and system throughput are thus important areas in supercomputing research. Traditionally, supercomputers are designed to be *worst-case power provisioned*, wherein all components of the system can be operated at peak power simultaneously. As we approach the power wall, however, this view is changing. Recent research has explored *hardware overprovisioning*, a technique where more hardware capacity is available in the system and performance optimizations can be accomplished by tuning and reconfiguring dynamically based on workload characteristics [18], [23], [26]. Power-aware resource management and job scheduling for higher throughput have also been active areas of research [7]–[12], [19], [22]. Additionally, runtime systems and other mechanisms for optimizing application performance under a power bound are being proposed [5], [6], [13], [14], [17], [27], [28]. While most of this research presents novel and relevant ideas, it does not take a holistic, systemwide approach to power management. In the Argo project, we believe that in order to improve application performance, throughput and overall power utilization, power management strategies need to exist at all levels in the system stack, ranging from the node level (hardware), to the job or application level, and further to the system-level. In the sections that follow, we describe a scalable approach to systemwide power management.

III. The Argo Model

In this section, we present the details of the Argo project that are relevant to the power infrastructure. In the Argo model, the system is managed hierarchically and recursively [20], [21]. Using a recursive model simplifies reasoning about the system and facilitates efficient management of resources at scale. The top level or root of the hierarchy is the global level. The leaves of the hierarchy are compute nodes. The internal nodes of the recursive hierarchy are enclaves. These levels, described below, are depicted in figure 1.

A. Global

The global level of the hierarchy manages the system to support the objectives and constraints of the owning organization. Security concerns such as isolation, priority concerns such as job scheduling across users, and operational concerns such as the systemwide power limit are the primary concerns

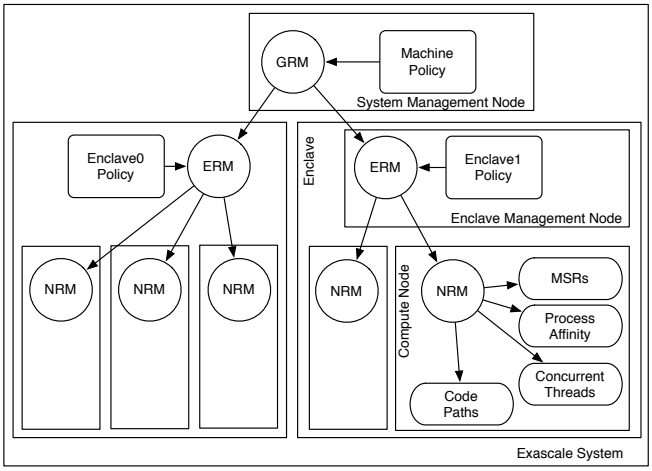

Fig. 1. Argo Model

of the global management system. To satisfy these objectives, managers at the global level receive telemetry from and issue commands to enclaves.

B. Enclave

The enclave levels exist between the root and the leaves in the system and represent sets of compute nodes. At least one enclave must exist on the path from the root to any leaf in the system. In the downward direction, enclaves receive constraints from above and apply settings below. In the upward direction, enclaves receive telemetry from below and produce digested aggregates that are relayed above.

C. Compute Node

The node level ends the recursion. Node managers are concerned with mapping the work and constraints received from the enclave to the actual hardware and use *compute containerization* techniques [29].

A *global information bus* (or *beacon* mechanism) allows for communication among the various levels and provides for a publish-subscribe mechanism for information exchange. This can be used to control and manage resources while adhering to constraints at the various levels in the hierarchy.

A runtime component called *Argobots* [24] has also been implemented within Argo. Argobots is a light-weight, low-level threading and tasking framework for massively concurrent systems that is based in user space. It gives users control over their resource utilization and provides a data movement infrastructure. Argobots can be used to support moldability and malleability in future applications seamlessly.

IV. Power Infrastructure in Argo

The power infrastructure in Argo is based on the model presented in Section III. Three major management tasks are involved. At the global level, a global resource manager (GRM) is responsible for allocating power across its children such that the global power bound is not exceeded. At the node level, a node resource manager (NRM) adjusts the local configuration and interacts with the hardware to adhere to a certain node-level power bound. Between the GRM and NRM, enclave resource managers (ERMs) are responsible for subdividing the power allocation received from the parent across the children and for maximizing enclavewide performance under the bound.

1) The GRM adheres to the system-level power bound set by the system administrators and uses the system-level policies for allocation of power to enclaves. The GRM is expected to give children suboptimal resource settings if needed to keep the machine within operational parameters. The PowSched component [7], [8] was designed and implemented to operate at this level.
2) The ERM responds to changes in power allocation from its parent and applies an enclave-level policy to allocate power in support of enclavewide performance goals. While the ERM must preserve the invariants expected by the GRM, the ERMs are given a broad license regarding how they may internally allocate resources across their children.
3) The NRM responds to changes in power allocation from the ERM and applies hardware and software configuration changes to remain within the allocation target. The NRM is the only subsystem with direct control over hardware configuration and is therefore the only subsystem that can actually effect resource consumption change. As part of this effort, the *msr-safe* kernel and *libmsr* library [25] were developed for power management and control on the Intel architectures that support the RAPL technology [16], [27]. A power monitor, PowMon, based on these low-level tools was also implemented.

For the power infrastructure to be effective, the NRM must have the ability to monitor and control the power on various components through mechanisms such as RAPL, DVFS, and Turbo Boost [15], [16], [27]. Similarly, the ERM should have access to strategies and runtime adjustable configuration settings supporting the resource constraints in order to tune for enclave-level/application-level performance. Argobots as well as other power-aware runtime systems provide such tuning opportunities at the ERM level. And, the GRM must have a global view of the entire system and the throughput across various applications. A power-aware job scheduler such as PowSched [7], [8] will need to operate at this level. Additionally, communication at all levels of the hierarchy must occur in order to convey power and performance requirements. In the current version, these have been implemented with a publish-subscribe global information bus, as described in Section III.

V. Results

To evaluate the power infrastructure presented in Section IV, we implemented a live demo tool. This tool integrates all the components described in Section IV, which include *msr-safe*, *libmsr*, PowMon, PowSched, the Argobots runtime, and the communication infrastructure *Beacon*. We ported a standard

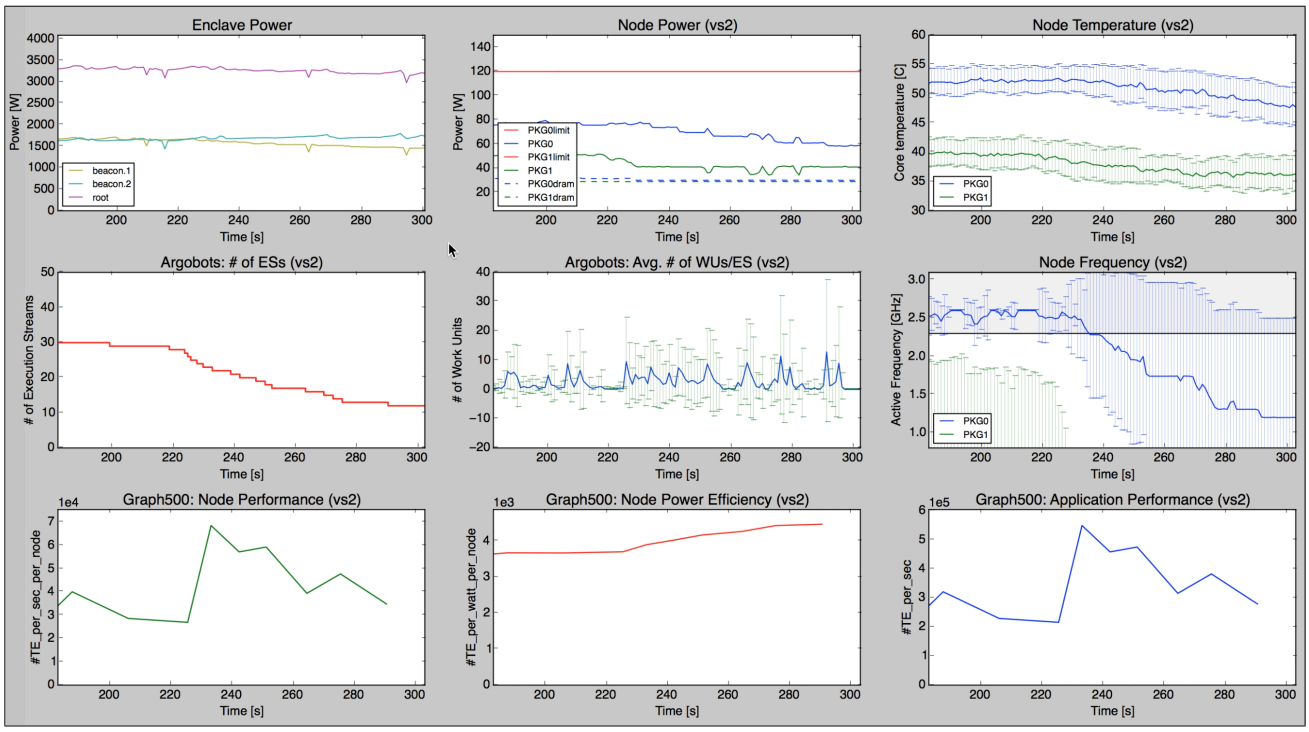

Fig. 2. Snapshot of Graph500 Run

data-intensive MPI application, Graph500 [3], to Argobots for the evaluation.

We evaluated the power infrastructure on a 20-node bare-metal setup on the Chameleon cluster [1]. Each node in this cluster had two Intel Xeon CPU E5-2670 v3 processors rated at 2.30 GHz. Each CPU had 12 cores (24 hardware threads). As a demo example, the Graph500 application run comprised of two enclaves, where each enclave had eight compute nodes. Figure 2 shows a snapshot from this demo example. The application's runtime is represented on the x-axis in all cases. The nine panels in the graph represent enclave power, node power, node temperature, node frequency, the number of execution streams from Argobots, average work units completed per execution stream in Argobots, node performance (number of traversed edges per second), node efficiency, and application performance (overall number of traversed edges per second) respectively.

The same Graph500 job was launched on each enclave. The initial power cap was set to 1400 W for both the enclaves, and the power setting was communicated from the GRM to the ERM and further to the NRM. The NRM was given the ability to make the job *malleable* by reducing or increasing the number of execution streams with the help of Argobots. During the application run, the power caps for the two enclaves were changed dynamically to 1000 W and 1600 W, respectively. This change triggered the NRMs in the first enclave to drop the number of execution streams and the NRMs in the second enclave to increase the available parallelism. Both enclaves adjusted to the new power cap. Note that this resulted in reduced performance in the first enclave because of the lowered power budget incurred by the changes in power allocation. As can be seen from this data, an integrated system that tunes for power has been successfully implemented in the Argo framework.

We also ported and evaluated a Charm++ application, LeanMD [2], within our power infrastructure. The results of this evaluation can be found at https://anl.box.com/s/dy84ly376sy0jjz3ukaxjbgkvf3k0u5q.

VI. Conclusions and Future Work

In this paper, we presented a holistic, recursive approach for systemwide power management in the Argo framework. We presented the details of the hierarchical power infrastructure and depicted some initial integration results. Having various components for management and control in place for an integrated, hierarchical system was the first step in achieving a scalable solution for power management. We note that the current setup does not optimize for performance of applications or system throughput under a power constraint. Additionally, the power and performance tradeoffs that occur as a result of added parallelism and supporting application-level moldability and malleability have not been explored yet. Future work for the power management team in Argo involves several directions, such as developing algorithms that maximize performance, algorithms that minimize the amount of wasted power, and algorithms that maximize throughput, and exploring the potential for malleable applications for power management and resilience.

Additional Authors and The Argo Team

Argonne National Laboratory: Kenneth Raffenetti, Pavan Balaji, Franck Cappello, Kamil Iskra, Rajeev Thakur, Marc Snir.

University of Illinois at Urbana-Champaign: Cyril Bordage, Laxmikant Kale, Yanhua Sun, Jonathan Lifflander.

University of Tennessee: George Bosilca, Jack Dongarra, Damien Genet, Thomas Herault.

University of Oregon: Sameer Shende, Xuechen Zheng, Wyatt Spear.

Lawrence Livermore National Laboratory: Maya Gokhale, Barry Rountree, Brian Van Essen, Edgar Leon.

University of Chicago: Nikita Mishra, Huazhe Zhang.

Pacific Northwest National Laboratory: Sriram Krishnamoorthy, Roberto Gioiosa, David Callahan, Gokcen Kestor.

Acknowledgements

Results presented in this paper were obtained using the Chameleon testbed supported by the National Science Foundation.

This work was supported by the U.S. Dept. of Energy, Office of Science, Advanced Scientific Computing Research Program, under Contract DE-AC02-06CH11357.

Part of this work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 (LLNL-CONF-685097).

References

[1] Chameleon Cloud, 2015. https://www.chameleoncloud.org/.

[2] LeanMD Benchmark, 2015. http://charmplusplus.org/benchmarks.

[3] A. Amer, H. Lu, P. Balaji, and S. Matsuoka. Characterizing MPI and Hybrid MPI+ Threads Applications at Scale: Case Study with BFS. In *Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on*, pages 1075–1083. IEEE, 2015.

[4] S. Ashby, P. Beckman, J. Chen, P. Colella, B. Collins, D. Crawford, J. Dongarra, D. Kothe, R. Lusk, P. Messina, T. Mezzacappa, P. Moin, M. Norman, R. Rosner, V. Sarkar, A. Siegel, F. Streitz, A. White, and M. Wright. The Opportunities and Challenges of Exascale Computing. *Summary Report of the Advanced Scientific Computing Advisory Committee (ASCAC) Subcommittee*, 2010.

[5] P. Bailey, D. Lowenthal, V. Ravi, B. Rountree, M. Schulz, and B. de Supinski. Adaptive Configuration Selection for Power-Constrained Heterogeneous Systems. In *International Conference on Parallel Processing*, ICPP '14, 2014.

[6] P. Bailey, A. Marathe, D. Lowenthal, B. Rountree, and M. Schulz. Finding the Limits of Power-constrained Application Performance. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '15, 2015.

[7] Daniel Ellsworth and Allen Malony and Barry Rountree and Martin Schulz. Dynamic Power Sharing for Higher Job Throughput. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '15, 2015.

[8] D. A. Ellsworth, A. D. Malony, B. Rountree, and M. Schulz. POW: System-wide Dynamic Reallocation of Limited Power in HPC. In *High Performance Parallel and Distributed Computing (HPDC)*, June 2015.

[9] M. Etinski, J. Corbalan, J. Labarta, and M. Valero. Optimizing Job Performance Under a Given Power Constraint in HPC Centers. In *Green Computing Conference*, pages 257–267, 2010.

[10] M. Etinski, J. Corbalan, J. Labarta, and M. Valero. Utilization Driven Power-aware Parallel Job Scheduling. *Computer Science - R&D*, 25(3-4):207–216, 2010.

[11] M. Etinski, J. Corbalan, J. Labarta, and M. Valero. Linear Programming Based Parallel Job Scheduling for Power Constrained Systems. In *International Conference on High Performance Computing and Simulation*, pages 72–80, 2011.

[12] M. Etinski, J. Corbalan, J. Labarta, and M. Valero. Parallel Job Scheduling for Power Constrained HPC Systems. *Parallel Computing*, 38(12):615–630, Dec. 2012.

[13] C. Hankendi, A. K. Coskun, and H. Hoffmann. Adapt&Cap: Coordinating System- and Application-Level Adaptation for Power-Constrained Systems. *IEEE Design & Test*, 33(1):68–76, 2016.

[14] Y. Inadomi, T. Patki, K. Inoue, M. Aoyagi, B. Rountree, and M. Schulz. Analyzing and Mitigating the Impact of Manufacturing Variability in Power-Constrained Supercomputing. *Supercomputing*, November 2015.

[15] Intel. Intel Turbo Boost Technology 2.0, 2008. http://www.intel.com/content/www/us/en/architecture-and-technology/turbo-boost/turbo-boost-technology.html.

[16] Intel. Intel-64 and IA-32 Architectures Software Developer's Manual, Volumes 3A and 3B: System Programming Guide, 2011.

[17] A. Marathe, P. Bailey, D. K. Lowenthal, B. Rountree, M. Schulz, and B. R. de Supinski. A Run-Time System for Power-Constrained HPC Applications. In *In International Supercomputing Conference (ISC)*, July 2015.

[18] T. Patki, D. K. Lowenthal, B. Rountree, M. Schulz, and B. R. de Supinski. Exploring Hardware Overprovisioning in Power-constrained, High Performance Computing. In *International Conference on Supercomputing*, June 2013.

[19] T. Patki, A. Sasidharan, M. Maiterth, D. Lowenthal, B. Rountree, M. Schulz, and B. de Supinski. Practical Resource Management in Power-Constrained, High Performance Computing. In *High Performance Parallel and Distributed Computing (HPDC)*, June 2015.

[20] S. Perarnau, R. Gupta, and P. B. et al. Argo: An Exascale Operating System and Runtime. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2015.

[21] S. Perarnau, R. Thakur, K. Iskra, K. Raffenetti, F. Cappello, R. Gupta, P. Beckman, M. Snir, H. Hoffmann, M. Schulz, and B. Rountree. Distributed Monitoring and Management of Exascale Systems in the Argo Project. In *IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS)*, 2015.

[22] O. Sarood, A. Langer, A. Gupta, and L. V. Kale. Maximizing Throughput of Overprovisioned HPC Data Centers Under a Strict Power Budget. In *Supercomputing*, Nov. 2014.

[23] O. Sarood, A. Langer, L. V. Kale, B. Rountree, and B. R. de Supinski. Optimizing Power Allocation to CPU and Memory Subsystems in Overprovisioned HPC Systems. In *IEEE International Conference on Cluster Computing*, pages 1–8, Sept 2013.

[24] S. Seo, A. Amer, P. Balaji, P. Beckman, C. Bordage, G. Bosilca, A. Brooks, A. Castelló, D. Genet, T. Herault, P. Jindal, L. V. Kale, S. Krishnamoorthy, J. Lifflander, H. Lu, E. Meneses, M. Snir, and Y. Sun. Argobots: A lightweight low-level threading/tasking framework, 2015. https://collab.cels.anl.gov/display/ARGOBOTS.

[25] K. Shoga, B. Rountree, and M. Schulz. Whitelisting MSRs with msr-safe. *In Third Workshop on Extreme-Scale Programming Tools, held with SC 14*, November 2014.

[26] E. Totoni, A. Langer, J. Torrellas, and L. Kale. Scheduling for HPC Systems with Process Variation Heterogeneity. *Technical Report, Parallel Programming Laboratory, University of Illinois Urbana-Champaign*, January 2015.

[27] H. Zhang and H. Hoffman. A Quantitative Evaluation of the RAPL Power Control System. In *Feedback Computing*, 2015.

[28] H. Zhang and H. Hoffman. Maximizing Performance Under a Power Cap: A Comparison of Hardware, Software, and Hybrid Techniques. In *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2016.

[29] J. A. Zounmevo, S. Perarnau, K. Iskra, K. Yoshii, R. Gioiosa, B. C. Van Essen, M. B. Gokhale, and E. A. Leon. A Container-Based Approach to OS Specialization for Exascale Computing. In *Cloud Engineering (IC2E), 2015 IEEE International Conference on*, pages 359–364. IEEE, 2015.