Global Extensible Open Power Manager: A Vehicle for HPC Community Collaboration Toward Co-Designed Energy Management Solutions

Jonathan Eastep, Steve Sylvester, Christopher Cantalupo, Federico Ardanaz, Brad Geltz, Asma Al-Rawi, Fuat Keceli, and Kelly Livingston
Power Pathfinding to Product (P3) Team, Advanced Development Strategy and Execution, Data Center Group, Intel Corporation
{jonathan.m.eastep, steve.s.sylvester, christopher.m.cantalupo, federico.ardanaz, brad.geltz, asma.h.al-rawi, fuat.keceli, kelly.a.livingston}@intel.com

***Abstract*—Performance of future large-scale HPC systems will be limited by costs associated with scaling power. Some HPC centers are reaching the limits of their existent site power delivery infrastructure and are facing prohibitive upgrade costs. Others are reaching budgetary limits on their energy operating costs. Without a breakthrough in energy efficiency, the HPC industry may fail to maintain historical performance scaling rates and fall short of 2018-2020 Exascale performance goals by an estimated 2-3x margin. Overcoming this gap will require co-designed hardware and software system energy management solutions and increased collaboration between hardware vendors and the HPC software community. In this work, we introduce the Global Extensible Open Power Manager (GEOPM): a tree-hierarchical, plug-in extensible, open source runtime framework that we are contributing to the HPC community to accelerate collaboration and research toward co-designed energy management solutions. First results with an experimental power rebalancing optimization demonstrate up to 32% improvements in the runtime of CORAL system procurement benchmarks like miniFE and Nekbone in a power-limited Xeon Phi cluster. These promising initial results motivate further work with the community to extend GEOPM to new optimization strategies to achieve further speedups.**

Keywords—runtime systems; scalability; control systems; tuning; power management; RAPL; P-states; DVFS; resource management; power-aware scheduling; performance optimization

I. INTRODUCTION

Performance of future large-scale HPC systems will be limited by costs associated with scaling power. Some HPC centers are reaching the limits of their existent site power delivery infrastructure and are facing prohibitive upgrade costs. Others are reaching budgetary limits on their energy operating costs [14]. Without a breakthrough in energy efficiency, the HPC industry may fail to maintain historical performance scaling rates and fall short of 2018-2020 Exascale performance goals by an estimated 2-3x gap [15].

This estimate of a 2-3x gap assumes the continued success of existing scaling techniques like shrinking transistor features, improving hardware architecture, and increasing integration of system components. These are hardware-centric techniques for scaling performance. Even if existing techniques are fully successful, overcoming the remaining 2-3x gap will require going beyond hardware-centric techniques and co-designing next-generation solutions with the HPC software community.

In this work, we introduce the Global Extensible Open Power Manager (GEOPM): a tree-hierarchical, plug-in extensible, open source runtime framework that we are contributing to the HPC community to accelerate collaboration and research toward co-designed energy management solutions. This paper is the result of 4 years of research and development within the GEOPM project and a preceding project internal to Intel by a subset of the same authors. This work makes three primary contributions:

- *This work provides a scalable runtime for global optimization of power and performance control knobs across all compute nodes in a job.* GEOPM's tree-hierarchical design and control algorithms enable scalable coordination of energy management decisions across nodes for better results than prior node-local approaches have obtained. Furthermore, the flexibility of its tree-hierarchical design makes GEOPM suitable for a wide range of deployments, spanning from rack-scale to Exascale deployments.
- *This work contributes GEOPM to the community as an open source, extensible framework to accelerate research of new HPC energy management solutions.* GEOPM provides a plug-in architecture enabling researchers, developers, hardware vendors, or HPC site administrators to add new optimization strategies to GEOPM or enable it to target new control knobs in the software or hardware layers. Via plug-ins, GEOPM's objective can range from finding the knob settings that minimize job runtime within a job power cap to finding settings that reach a custom tradeoff between runtime and power. In future work, the authors of this paper will examine select community plug-ins and research ways to co-design Intel hardware features for even better results with those plug-ins.
- *This work explores a new energy management algorithm as a plug-in for GEOPM that guides the hardware to more effective use of limited power for significant performance improvements.* To obtain these improvements, the plug-in leverages application-awareness to identify compute nodes on the critical path in an MPI job then diverts power from nodes off of the critical path to accelerate the critical path nodes. Power and performance are adjusted via RAPL [13] hardware controls. To our knowledge, our power rebalancing techniques are the first to integrate hierarchical algorithms intended for deployment at scale in TOP500 systems.

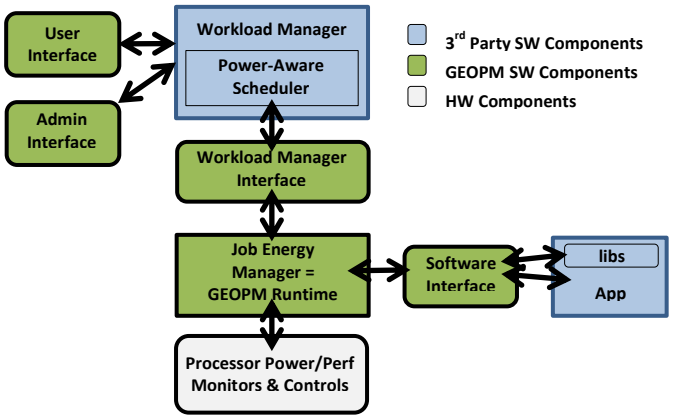

Figure 1: GEOPM Interfaces and HPC System Stack Integration

The GEOPM runtime framework is being developed with the intent of deployment on the Argonne CORAL Theta system (Knights Landing Xeon Phi) and other future Xeon and Xeon Phi systems. First results with the experimental power rebalancing optimization demonstrate up to 32% improvements in the runtime of CORAL system procurement benchmarks like miniFE and Nekbone in a power-limited Xeon Phi cluster. These promising initial results motivate further work with the community to extend GEOPM's optimizations for further speedups.

The GEOPM software package is available under the BSD three clause open source software license in the GEOPM source code repository on github (project page: geopm.github.io/geopm). The GEOPM runtime framework, test infrastructure, and Intel-developed power rebalancing plug-in are all open source. Over time, the code will be further hardened for deployment in production systems.

II. Related Work

To our knowledge, GEOPM is the first open extensible runtime framework to be contributed to the community by a hardware vendor with the intent of collaborative research toward software-hardware co-designed energy management solutions in future HPC systems. While there are parallel co-design efforts such as OpenPOWER [17] which seek to enable the community to customize systems based on the IBM POWER architecture, we are not aware of any specific efforts within the OpenPOWER project to research co-designed energy management solutions. While GEOPM only currently provides plug-ins supporting x86 systems, users can develop and add plug-ins supporting POWER or other system architectures.

To our knowledge, GEOPM is the first open source job-level energy management runtime for HPC systems to support extensible energy management control strategies through a plug-in architecture, making it suitable for the differing energy management needs of a wide range of HPC installations around the world. The Power API Specification from Sandia [16] is a synergistic effort, but it is an orthogonal effort because it emphasizes power interfaces rather than runtime techniques for optimizing energy. The Power API project is defining community-standard interfaces for power monitoring and control at various granularities throughout the HPC stack. Runtimes like GEOPM and other components can collectively target these interfaces to achieve interoperability. Sandia and the GEOPM team are collaborating to explore changes targeted at future releases of the specification to increase support for GEOPM and its interfaces.

In this work, we develop a plug-in for GEOPM for power rebalancing at the job level. Prior works such as Conductor [1], Adagio [2], and Jitter [3] have demonstrated effective algorithms for reallocating power between nodes to compensate for application load imbalance – whether for the purpose of increasing application performance under a job power cap by accelerating the critical path or improving application energy efficiency by reducing performance in nodes off of the critical path. While these algorithms are effective at smaller scales (i.e. less than a few thousand nodes), their centralized designs are not intended for large-scale deployments or extreme-scale deployments in future Exascale systems. The key difference is that the GEOPM power balancer plug-in has a flexible tree-hierarchical design suitable for deployments ranging in scale from rack-scale to extreme-scale deployments. We note, however, that we have a collaboration underway with LLNL and Argonne National Laboratory to compare approaches and meld together the best aspects of each approach in a future GEOPM plug-in and paper.

There is a parallel work to GEOPM called the Argo project [24] which is developing a task-based programming model and runtime for Exascale HPC systems. Its design includes a hierarchical power manager. Unlike GEOPM, the Argo power manager is not intended as a vehicle for the community and hardware vendors to collaborate on researching new energy management solutions. Furthermore, while the Argo project envisions this power manager performing automatic hierarchical power budgeting, that functionality is not complete to our knowledge. What has been demonstrated is hierarchical enforcement of power budgets that were adjusted manually at runtime. That said, the GEOPM team is interested in exploring if Argo's algorithms could be implemented as GEOPM plug-ins and brought to fruition in production deployments through the GEOPM framework.

We note that there have also been orthogonal efforts [25] to develop hierarchical power management frameworks for enterprise data centers. They employ significantly different energy management strategies suitable for enterprise workloads and virtualized environments. There have been other related works that focused on saving power given a time bound. Some have used linear programming to optimize energy savings with nearly no runtime increase [19]. Others have achieved bigger power savings in exchange for small performance degradations [20, 21, 22, 23].

Aside from prior works on saving energy while maintaining performance levels, hierarchical power capping, and rebalancing power across nodes to increase job performance under a power cap, there have also been prior works on power-aware scheduling algorithms for energy management at the system level [4, 5, 6]. These algorithms comprehend system-level power caps and assign a different power cap to each job based on its runtime and power characteristics with the goal of reducing job wait times or optimizing overall system throughput. GEOPM is synergistic with these works: the intent is for GEOPM to integrate with a power-aware scheduler in an extended energy management hierarchy. In particular, the scheduler can view GEOPM as a mechanism for optimizing the job's performance or energy efficiency within the scheduler-specified job power budget, and the scheduler can optimize system performance and efficiency by deciding the best allocation of the system budget among concurrent jobs. For maximum benefits, GEOPM supports dynamic adjustments to the job cap.

III. GEOPM Design Overview

This section provides an overview of the GEOPM design, beginning with discussion of how GEOPM integrates into the HPC system stack. We cover the components that GEOPM interacts with and the interfaces it provides for interaction with those components. Then, we describe GEOPM's scalable tree-hierarchical design. We cover GEOPM's hierarchical control algorithms and the mechanisms it uses for communication throughout its control hierarchy, communication with software layers, and communication with the hardware. Finally, we provide references to developer documentation describing how to use GEOPM's interfaces and write plug-ins for GEOPM.

A. *GEOPM Interfaces and Integration Architecture*

Figure 1 illustrates how the GEOPM runtime fits into the HPC system stack. GEOPM is a job-level energy manager. It runs in userspace. The GEOPM runtime interacts with the scheduling functions of the workload manager through the workload manager interface. This interface lets future power-aware schedulers assign a power cap for the job and configure which energy management plug-in GEOPM should use to manage the job. It also allows GEOPM to report back how much power the job consumed and statistics about the job that GEOPM has collected. There is an option for the interface to be used at job start and finish (statically) or periodically while the job is running (dynamically).

There is also an interface to the application software or libraries (shown at the middle right of the figure). This software interface allows the programmer to mark up their code and hint to GEOPM about global synchronization events in the application that could result in performance loss if some MPI ranks fall behind in the computation and reach the synchronization point late. The interface also enables

programmers to hint to GEOPM about regions (i.e. phases) in the application or library code between synchronization events and provide an application-level performance signal that GEOPM plug-ins can use to adapt their decisions as the application transitions between phases.

For example, a GEOPM plug-in may adjust the P-state of the cores as the application alternates between compute-intensive, memory-intensive, and communication-intensive phases, decreasing frequency in memory- or communication-intensive phases to save power. The software interface is lightweight and minimally invasive, but future work will explore methods of automatically inferring phase and performance information to enable GEOPM usages where applications require zero markup to use any plug-ins that make per-phase decisions.

Other future work on the software interfaces will explore extensions to enable application or library developers to express tunable knobs in the software layers of the system to GEOPM for dynamic tuning. Examples may include the choice of how many threads to employ in OpenMP parallel regions, the choice of heuristics employed by the MPI and OpenMP libraries, or traditional application-level auto-tuning knobs such as the choice of algorithm, cache blocking factor, etc.

The GEOPM team has also been exploring how administrators and users may want to interact with job-level power managers, and those interactions will be achieved through the user and admin interfaces. The user interface enables users to request that GEOPM use a specific energy management plug-in for the job. This user interface may be implemented using multiple methods. One method is to wrap the SLURM srun queue command (or equivalent queue commands provided by other workload managers). The admin interface enables the administrator to provide a default selection for the GEOPM plug-in.

A detailed description of GEOPM's interfaces is available but outside the scope of this paper. In addition to interface specifications for developers, we also provide tutorials and example MPI applications in the GEOPM source code repository illustrating how to use the interfaces. See [7] for the documentation and tutorials. Video walkthroughs of the tutorials are also available on YouTube [18].

B. *GEOPM Scalable Tree-Hierarchical Design*

GEOPM is designed to scale for deployment on systems ranging in size from a single rack to an Exascale system with over 100,000 compute nodes. This is accomplished through a flexible tree-hierarchical design. As illustrated in Figure 2, the GEOPM runtime is implemented as a tree hierarchy of controllers. It is a feedback-guided tree-hierarchical control system. The energy management strategy employed by each controller in the tree is extensible through a plug-in architecture. The depth and fan-out of the tree are automatically adjusted by the GEOPM runtime to accommodate different job sizes.

Controllers in the tree (and therefore energy management plug-ins) take a recursive approach to coordinating energy and performance policy decisions globally, across all nodes in the job. The GEOPM root controller sets policy for its children, each of its children set policy for their children, and so on. Policies are defined hierarchically such that the parent constrains the space of policies that its children can select from and, in so doing, effects their decisions. Decisions at each level of the tree are based on feedback from each child. This feedback consists of a history of energy, performance, and other statistics collected over the last few control intervals. For scalability, the feedback is aggregated as it flows from the leaves toward the root. Thus, decisions at the root are informed by feedback from the leaves, and decisions flowing down the tree effect decisions made at each leaf.

To help controller plug-ins achieve control stability, the GEOPM plug-in API includes functions that enable plug-ins at different levels of the tree to synchronize using a handshake signal. I.e. the parent waits for its children to signal that they are ready for a new policy before

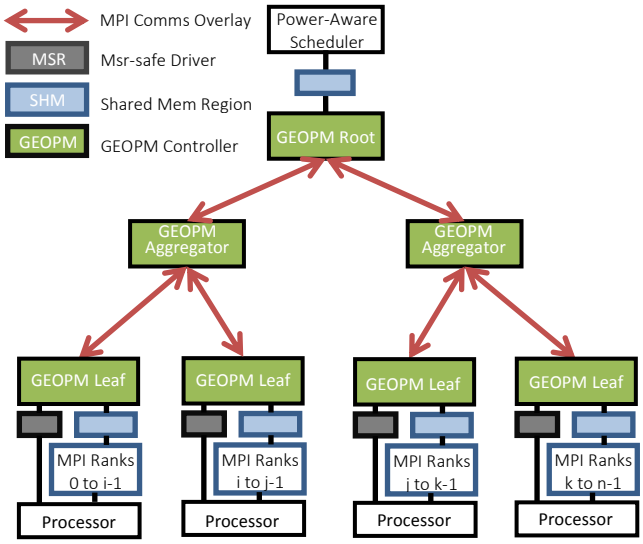

Figure 2: GEOPM Hierarchical Design and Communication Mechanisms

stepping its controller, and children wait until their controller has stabilized before providing the signal to their parents. At the leaf-level, GEOPM is designed for sub-millisecond controller reaction times.

The GEOPM github repository includes an Intel-developed controller plug-in as an example of tree-hierarchical control. This plug-in improves performance in power-capped systems by leveraging application-awareness to identify nodes on the critical path in a bulk-synchronous MPI application and accelerate them by diverting power from nodes off of the critical path. The algorithm identifies the critical path by tracking and comparing the runtime of the outer synchronization loop in each rank over a window of multiple iterations of the outer synchronization loop. At the leaf controllers, this information is obtained from the software interface described in Section A. Runtime is aggregated as it flows up the tree: each controller reports its runtime as the max runtime of its children. Each controller takes in a power budget from its parent and computes (based on the runtime reported by the children) how to divide that budget among its children such that all sub-trees will reach the end of the outer synchronization loop at the same time, avoiding wait times and associated performance loss. At the leaf controllers, power and performance are adjusted through the use of processor RAPL hardware controls. The overall power cap for the job is enforced recursively, with RAPL enforcing the cap at the compute node level.

All GEOPM controllers run in the compute nodes of the job. Each compute node runs one of GEOPM's leaf agents, some compute nodes also run an intermediate level of the tree, and one also runs the root of the tree. Dynamic communication between levels of the tree is currently achieved using MPI over the application's in-band network fabric. For scalability, we use MPI's Cartesian topology functionality to map GEOPM's control tree hierarchy to compute nodes in a way that makes the communication between levels of the tree efficient and balances the control tree across compute nodes. Our studies of the overhead introduced by GEOPM communications on OmniPath in-band network fabrics suggest that bandwidth use will be much less than 1% of the total bandwidth, but GEOPM may support out-of-band communication in the future nonetheless, if desired.

Inter-process shared memory is used both for dynamic communication between the root of the GEOPM control tree hierarchy

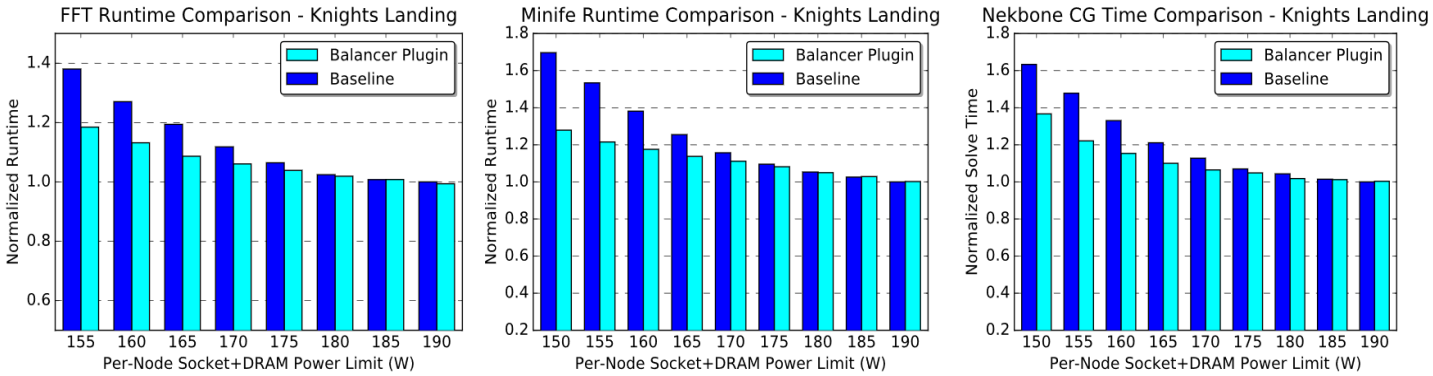

Figure 3: Runtime Improvements Obtained with GEOPM Power Balancing Plug-in on a 12-Node Knights Landing Cluster

and the power-aware scheduler and communication between the leaf controllers and the software running on the compute nodes. Over time, the interface to software will be hardened for security. Communication between the leaf controllers and the processors is achieved through access of Model Specific Registers (MSRs) [13] via the msr-safe Linux driver developed by LLNL [8].

While it is not GEOPM's only supported mode of operation, the preferred mode of operation is to run all controllers in one reserved core on each compute node. I.e. the application and GEOPM are affinitized such that the application does not run on GEOPM's core. The overhead of reserving a core is inexpensive relative to the benefits because HPC processors have many cores. The reserved core architecture affords GEOPM plug-ins the computational bandwidth of an entire core so that they can react quickly to phases and perform deep analysis of applications and tradeoffs.

C. *GEOPM Extensible Controller Plug-In Architecture*

A detailed description of GEOPM's plug-in architecture and a developer's guide for creating plug-ins is available but outside the scope of this paper. You can obtain the information in [7].

IV. First Results

We demonstrate the potential of the GEOPM framework by developing a power-balancing plug-in (described in prior sections) which improves performance in power-capped systems for bulk-synchronous MPI applications by accelerating their critical path. We present our first evaluations of this plug-in, beginning with a description of our experimental setup followed by initial results and analysis.

A. *Experimental Setup*

We performed our experiments on a Knights Landing cluster. See the table summarizing its configuration. When configuring workloads, we applied standard conventions. We sized the problem to use the majority of the MCDRAM (on-package memory) in each node. We used one hardware thread per core (no Simultaneous Multithreading). With the system not power-capped – i.e. with the processors allowed to run at Thermal Design Power (TDP) – a we swept over the different numbers of MPI ranks and OpenMP threads per rank using up all or almost all of the available cores; we then determined which configuration resulted in the best runtime and used this configuration in all evaluations of our power balancing plug-in.

To study how much application speedup our power-balancing plug-in provides in power-constrained systems, we swept over a range of job power caps and compared the workload runtime achieved while using our inter-node power-balancing plug-in versus a baseline. Our power-balancing plug-in dynamically reallocates the job power budget among nodes to mitigate load imbalance while the baseline applies a static uniform division of the job power budget among nodes. In the baseline, all controllers above the leaf level of the tree are inactive. However, both cases employ active controllers at the leaf level of the tree to enforce the node-level power budgets.

The leaves enforce the budget as follows: they dynamically measure the power consumed in the external DRAM via the processor RAPL feature, they subtract this power from the node budget (obtained from their parents in the GEOPM control tree hierarchy), and then they set the RAPL socket power limit equal to the remaining power so that the sum of socket and external DRAM power matches the node power budget. Node power budgets are defined in terms of the dynamic power controllable via the processor RAPL feature. The remainder is not included but it is approximately static.

Knights Landing Xeon Phi Cluster	
Hardware Configuration	***Software Environment***
12x KNL-F nodes B0 Beta SKU	CentOS 7, 'performance' frequency governor
64x4=256 HW threads per node	NPB FT, NPB miniFE, CORAL Nekbone workloads
16GB MCDRAM, 256GB DRAM per node	Workloads instrumented with GEOPM APIs
Integrated OmniPath HFI, OmniPath Fabric	Workloads run on 56 cores
Turbo enabled, 1.3GHz sticker frequency	GEOPM and OS services run on 1 core
230W TDP processors	>10 runs performed per data point

The load imbalance exhibited in the workloads under study derives from the effects of hardware manufacturing variation. The workloads are configured with equal work per rank, so the load imbalance is not due to work imbalance. The effects of manufacturing variation on load imbalance are well known after 4 years of study by the authors of this paper internally at Intel and publications in the literature [9, 10]. When interpreting the results in this paper, it is important to note that the authors made no attempt to cherry-pick processors from extreme ends of the manufacturing quality distribution for a given bin. Therefore, it is not yet known if the processors in our cluster reflect the full potential for load imbalance (though we will explore this in future work). It may be possible for our power balancer plug-in to achieve higher speedups in systems with processors with higher manufacturing variation.

In our power cap sweep experiments, we set the max job power cap equal to the power each workload naturally consumes when processors are uncapped, and we set the min job power cap to the value at which performance scaling hit an inflection point; i.e.: where performance began degrading precipitously relative to the power reduction. Results at power caps below this inflection point may be meaningful in some research scenarios but they are omitted from this paper for brevity.

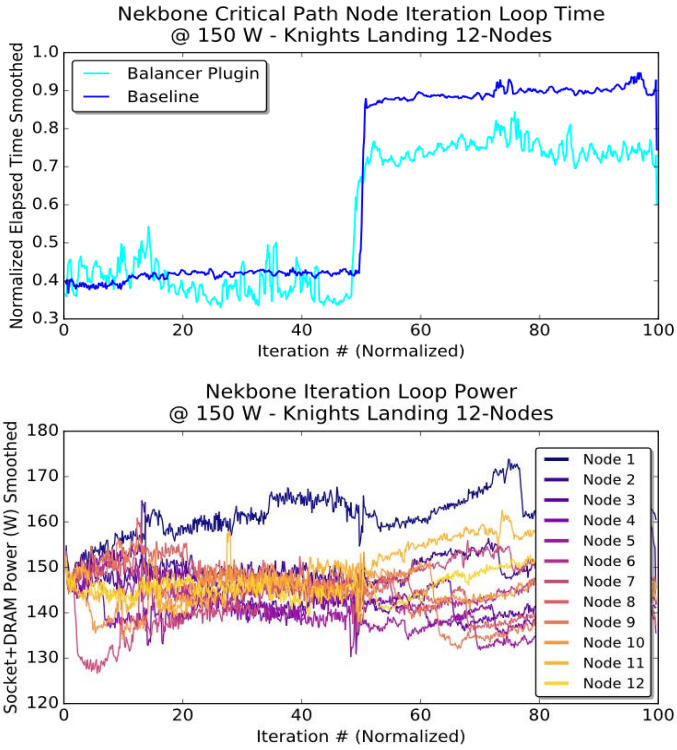

Figure 4: Traces of Outer Loop Runtime and Node Power Allocation

B. *Evaluation of the Power Balancing Plug-In*

Figure 3 shows the runtime improvements obtained by our power balancing plug-in over a range of job power caps for the NAS Parallel Benchmarks FFT and miniFE workloads [11] and the CORAL Nekbone workload. Nekbone is a thermal hydraulics code and a Tier 1 scalable science workload in the CORAL procurement benchmarks [12]. We added mark up to these workloads to enable tuning them with the plug-in. The modifications are available in [7]. In the figure, the lighter colored bars are the results with our power balancer plug-in, and lower values are better. Runtimes are normalized.

As the figure indicates, our power balancing algorithm is able to provide substantial runtime improvements of up to 32%. The amount of improvement varies depending on the power cap and the workload, but it tends to increase as the job power is increasingly constrained since the critical path can be operated at a higher and higher frequency relative to the other nodes. At the right side of each graph, job power is not very constrained and the nodes have enough power to run at closer to full frequency, so the critical path cannot be accelerated. In all experiments, we note that we confirmed that the power balancing plug-in obtains its runtime improvements without going over the job power budget. We also note that, in other clusters, the improvements may vary if the processors exhibit differing amounts of manufacturing variation than seen in our cluster. As described above, we have not yet determined if the processors in our cluster exhibit the maximum range, so the maximum runtime improvements that a GEOPM user may obtain from the power balancing plug-in may exceed 32% on some clusters.

Next, we trace the action of the power balancing algorithm over the course of a run to show how the runtime improvements were obtained. In Figure 4, we show an example from a run of the Nekbone workload at one of the power caps studied in our sweeps. For brevity, we omit results collected for FFT and miniFE, but we note that we observed consistent trends in that data. The top plot shows the normalized runtime of each iteration of the Nekbone outer loop in the critical path node (i.e.: the node with lowest manufacturing quality). In the bottom plot, we plot the power allocated to each node for each iteration of the outer loop. These runtime and power traces were collected through GEOPM's tracing features. Together, the plots show that the power balancing plug-in minimizes the runtime of the outer loop by identifying the critical path node and allocating it a larger portion of the job power budget.

The data in the top plot exhibits two phases. In the first phase, the runtime of the outer loop is slightly better than the baseline runtime when using the power balancer plug-in, but in the second phase the power balancer significantly improves the runtime. The two phases can be explained by observing that the Nekbone benchmark executes two conjugate gradient computations of different problem sizes. The second one is more sensitive to manufacturing variation because it is more compute-intensive. Thus, it offers more opportunity for acceleration.

In the bottom plot, the trace confirms that the power balancer plug-in is responding to differences in the outer loop runtime across nodes. In particular, Node 1 is allocated more power. This is expected based on additional experiments we performed to confirm that Node 1 has the processor with the lowest manufacturing quality in our cluster. We also note that the data demonstrates that our plug-in adapts readily when Nekbone moves from the first conjugate gradient computation to the second. When the second begins, the plug-in realizes that the previous power allocation is no longer ideal and it learns a new power allocation.

V. Conclusions and Future Work

This paper introduced an open source, extensible, scalable runtime framework called GEOPM. GEOPM is being contributed to the community to accelerate collaboration and research toward software-hardware co-designed HPC energy management solutions. To demonstrate GEOPM's potential as a framework, this paper developed an experimental power balancing plug-in for GEOPM. We shared results from an initial evaluation of that plug-in which demonstrated substantial speedups for key CORAL system procurement benchmarks in power-capped systems.

In future work, the authors will expand upon their initial studies of the power balancer plug-in to a) determine bounds on how much benefit the plug-in will provide in systems containing processors representing all points in the manufacturing quality distribution for a given bin, b) evaluate benefits on additional benchmarks, and c) demonstrate that the plug-in's tree-hierarchical algorithm scales as well as expected in larger systems. In fact, the first scaling studies have already begun through a collaboration with LLNL.

Lastly, the promising initial results presented in this paper motivate future work to spin up additional collaborations with the community to research new energy optimization strategies through GEOPM's plug-in framework. It would be especially interesting to target optimizations that run in conjunction with power balancing optimizations to achieve speedups and energy efficiency improvements on top of the benefits of power balancing. The GEOPM team is also seeking collaborations to a) explore further integration of GEOPM with emerging power-aware scheduling functions in SLURM (or other workload managers) and b) explore tuning power-performance knobs in software libraries/runtimes like MPI or OpenMP or knobs in the application layer of the HPC stack.

VI. Acknowledgments

The authors would like to thank the following individuals for their input on this work: Vitali Morozov and Kumar Kalyan of Argonne; Barry Rountree and Martin Schulz of LLNL; James Laros and team from Sandia; and Richard Greco (retired), Tryggve Fossum (retired), David Lombard, Michael Patterson, and Alan Gara of Intel.

VII. References

[1] A. Marathe, P. E. Bailey, D. K. Lowenthal, B. Rountree, M. Schulz, B. de Supinski. A Run-Time System for Power-Constrained HPC Applications. In ISC, 2015.

[2] B. Rountree, D. K. Lowenthal, B. de Supinski, M. Schulz, and V. W. Freeh. Adagio: Making DVS Practical for Complex HPC Applications. In ICS, 2009.

[3] N. Kappiah, V. W. Freeh, and D. K. Lowenthal. Just in Time Dynamic Voltage Scaling: Exploiting Inter-Node Slack to Save Energy in MPI Programs. In Supercomputing, Nov. 2005.

[4] M. Etinski, J. Corbalan, J. Labarta, and M. Valero. Optimizing Job Performance Under a Given Power Constraint in HPC Centers. In IGCC, 2010.

[5] M. Etinski, J. Corbalan, J. Labarta, and M. Valero. Linear Programming Based Parallel Job Scheduling for Power Constrained Systems. In HPCS, 2011.

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

[7] Global Extensible Open Power Manager project url, http://geopm.github.io/geopm/. Hillsboro, OR: Intel Corporation, September 2016.

[8] K. Shoga, B. Rountree, M. Schulz, and J. Shafer. Whitelisting MSRs with msr-safe, in 3rd Workshop on Exascale Systems Programming Tools, in conjunction with SC14, 2014.

[9] B. Rountree, D. H. Ahn, B. R. de Supinski, D. K. Lowenthal, and M. Schulz. Beyond DVFS: A first Look at Performance Under a Hardware-Enforced Power Bound. In HPPAC, 2012.

[10] Y. Inadomi, T. Patki, K. Inoue, M. Aoyagi, B. Rountree, M. Schulz, D. K. Lowenthal, Y. Wada, K. Fukazawa, M. Ueda, M. Kondo, and I. Miyoshi. Analyzing and Mitigating the Impact of Manufacturing Variability in Power-Constrained Supercomputing. In Supercomputing, 2015.

[11] D. Bailey, E. Barszcz, J. Barton, D. Browning, R. Carter, L. Dagum, R. Fatoohi, P. Frederickson, T. Lasinski, R. Schreiber, et al. The NAS Parallel Benchmarks Summary and Preliminary Results. In Supercomputing, pages 158–165, 1991.

[12] CORAL Procurement Benchmarks. https://asc.llnl.gov/CORAL-benchmarks/CORALBenchmarksProcedure-v26.pdf, September 2016.

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

[14] J. Shalf, S. Dosanjh, and J. Morrison. Exascale Computing Technology Challenges. In VECPAR, 2010.

[15] A. Sodani. Race to Exascale: Opportunities and Challenges. MICRO 2011 Keynote address.

[16] J. Laros, D. DeBonis, R. Grant, S. Kelly, M. Levenhagen, S. Olivier, and K. Pedretti. High Performance Computing - Power Application Programming Interface Specification, Version 1.0, Sandia National Laboratories, Technical Report SAND2014-17061, 2014.

[17] M. Gschwind. OpenPOWER: Reengineering a Server Ecosystem for Large-Scale Data Centers. In Hot Chips Symposium (HCS), pp. 1-28, 2014.

[18] GEOPM Video Tutorials, https://www.youtube.com/playlist?list=PLwm-z8c2AbIBU-T7HnMi_Pux7iO3gQQnz. Hillsboro, OR: Intel Corporation, September 2016.

[19] B. Rountree, D. K. Lowenthal, S. Funk, V. W. Freeh, B. de Supinski, and M. Schulz. Bounding Energy Consumption in Large-Scale MPI Programs. In Supercomputing, Nov. 2007

[20] K. W. Cameron, X. Feng, and R. Ge. Performance-Constrained Distributed DVS Scheduling for Scientific Applications on Power-Aware Clusters. In Supercomputing, 2005.

[21] R. Ge, X. Feng, W. Feng, and K. W. Cameron. CPU Miser: A Performance-Directed, Run-Time System for Power-Aware Clusters. In ICPP, 2007.

[22] C.-H. Hsu and W.-C. Feng. A Power-Aware Run-Time System for High-Performance Computing. In Supercomputing, Nov. 2005.

[23] D. Li, B. de Supinski, M. Schulz, K. Cameron, and D. Nikolopoulos. Hybrid MPI/OpenMP Power-Aware Computing. In IPDPS, 2010.

[24] D. Ellsworth, T. Patki, S. Perarnau, S. Seo, A. Amer, J. Zounmevo, R. Gupta, K. Yoshii, H. Hoffman, A. Malony, M. Schulz, P. Beckman. Systemwide Power Management with Argo. In Parallel and Distributed Processing Symposium Workshops, May 2016.

[25] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, X. Zhu. No "Power" Struggles: Coordinated Multi-level Power Management for the Data Center. In ASPLOS, 2008.