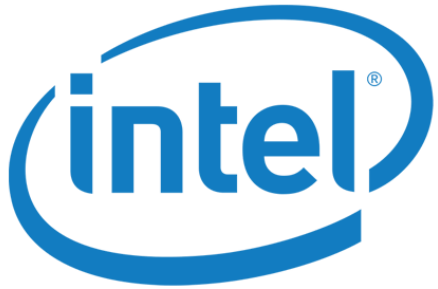
GEOPM Overview



https://geopm.github.io

GEOPM PI: Dr. Jonathan Eastep
Principal Engineer
Intel Corp., DCG

Outline

- GEOPM intro
- Use-cases
- Key results
- Traction
- Next steps
- Call to action

Introduction to GEOPM

- Global Extensible Open Power Manager
  - Open source runtime framework for profiling an application and optimizing its performance or energy efficiency
  - Leverages ML + control system techniques to analyze data and optimize HW control knob settings in real-time
  - Discovers patterns in application and coordinates global optimization of settings across nodes
  - Plugin architecture facilitates adding profiling features, optimization capabilities, and porting to different HW platforms
  - Open source project started and supported by Intel, but has grown into community collaborative project
  - Project page: https://geopm.github.io/

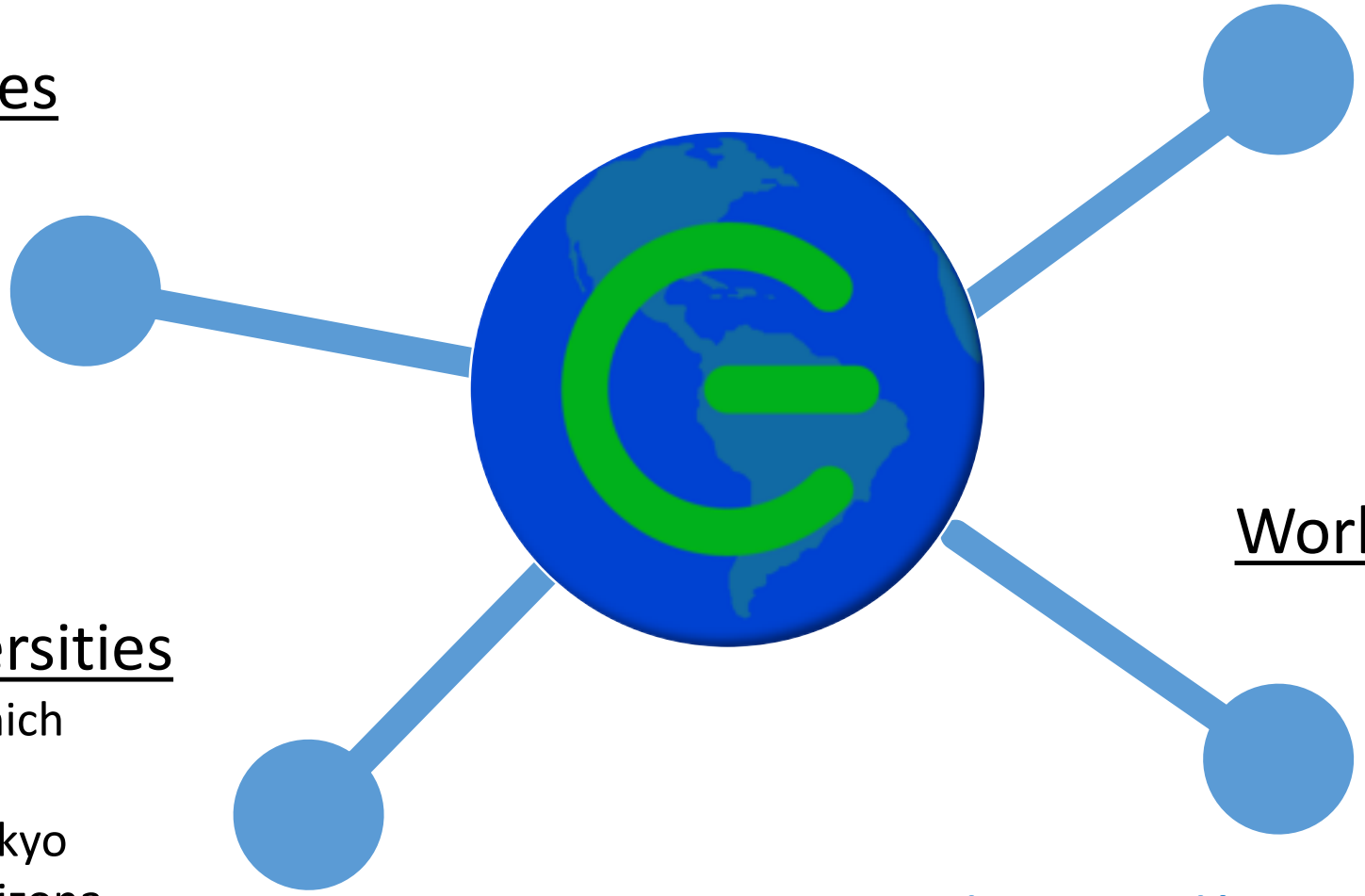
GEOPM Community

Companies

Intel
IBM
HPE
Cray
Marvell
Arm

HPC Centers

Argonne
LLNL
LRZ
CINECA
Sandia
Hartree



Universities

TU Munich
LMU
U. of Tokyo
U. of Arizona
U. of Bologna

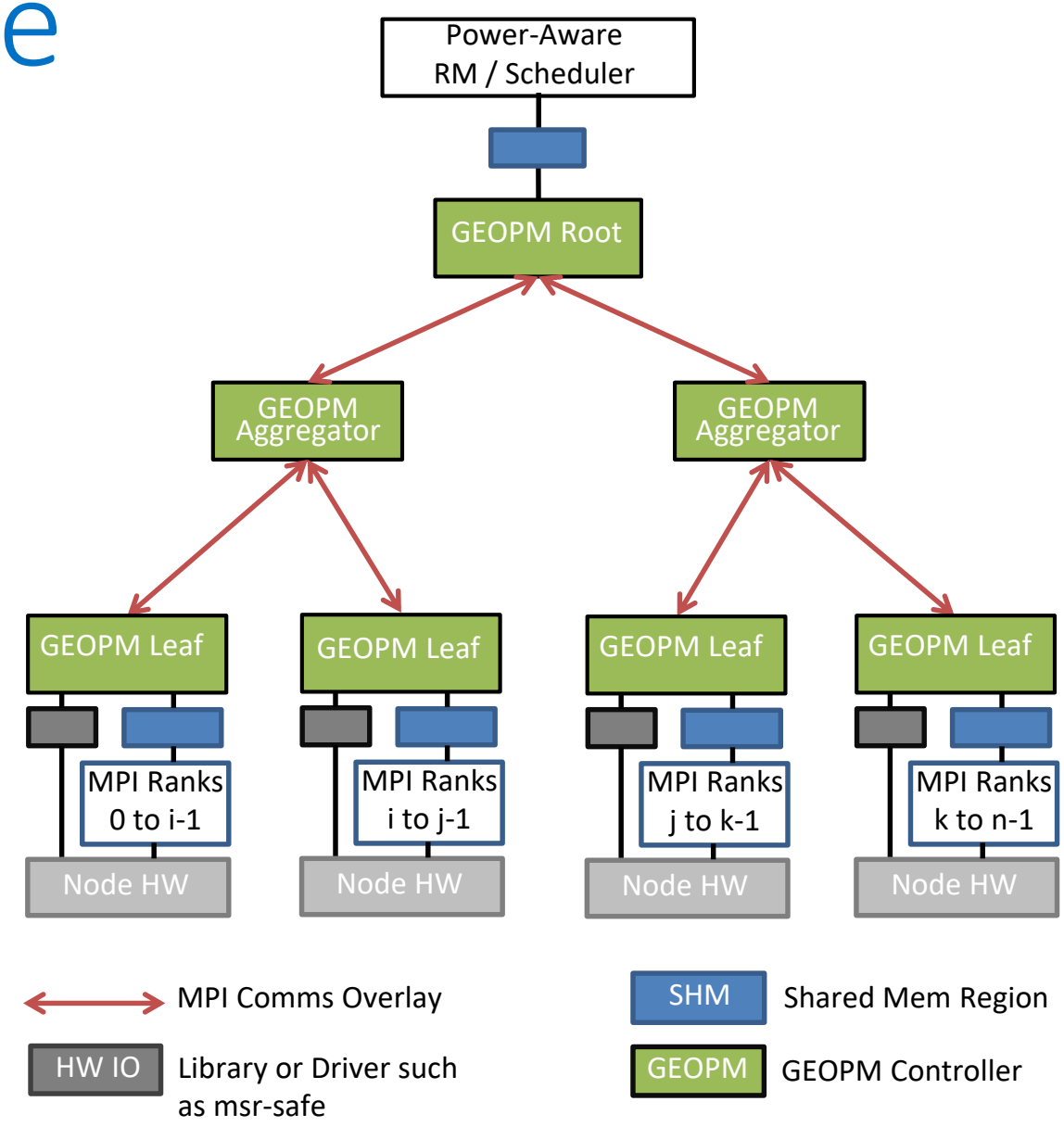
Working Groups

HPC PowerStack
Power API
EE HPC

And rapidly growing…

GEOPM Architecture

- Job-level runtime. Controlled by RM / scheduler. Provides profiling feedback to RM / scheduler
- Implemented as a tree-hierarchal distributed runtime comprising Root, Aggregator, and Leaf controllers
- Controllers coordinate hierarchical optimizations and tree-based aggregation of profile data
- Runs in-band under Linux. 1 GEOPM process per compute node in the job. Runs on the "OS core"
- Communication between GEOPM processes via MPI library over application network fabric
- Userspace runtime. Bypasses OS governors. Accesses low-level processor registers for power / performance control and monitoring via a driver such as msr-safe
- Obtains app-level profiling data via PMPI, OMPT, GEOPM profiling API. Comm mechanism is SHMEM
- GEOPM designed for low overhead to application, low memory footprint, and low communication BW



EOPM Use-Cases

- Designed for production deployment as well as research infrastructure
- Out-of-the-box: supports basic use-cases such as
  - Interfacing to processor power management features. Library of C interfaces for programmatic access
  - Characterizing variation in performance, power, temperature across nodes of the system
  - Profiling runtime and energy of MPI applications or application phases
- Out-of-the-box: supports advanced optimization use-cases such as
  - Optimizing job time-to-solution within a job power cap (critical for power-limited systems)
  - Optimizing job energy-to-solution within constraints on runtime impact (critical if Op-Ex limited)
- Extensible to new use-cases via plugins
  - Write customized plugins tailored to needs of individual HPC centers or applications
  - Example: countdown plugin in development in collaboration with UniBo, CINECA, LRZ
  - Roadmap moves beyond tuning HW controls to jointly tuning params of SW layers like MPI, OpenMP

Results: Overview

- A recent focus area has been evaluation of GEOPM at scale on production systems
- Evaluation #1: Optimize job energy efficiency for Op-Ex limited systems
  - Targeted LLNL's Quartz system based on Broadwell Xeon Processors
  - LLNL doesn't ordinarily OpEx limit the Quartz system, but useful as a proxy for European systems
  - Demonstrated 8-47% energy savings with <10% runtime impact by tuning processor core frequencies
- Evaluation #2: Optimize job time-to-solution for power-limited systems
  - Targeted Argonne's Theta system based on Knights Landing Xeon Phi Processors
  - Argonne doesn't ordinarily cap Theta's power, but useful as a proxy for future system scenarios
  - First measured performance variation across nodes of the system under power caps
  - Demonstrated 5-30% improvements in time-to-solution by reallocating power to equalize performance
- Disclaimer: Your mileage may vary
  - Opportunities vary based on power management features available in the HW platform
  - Opportunities vary based on application characteristics, manufacturing variation, power cap assumptions, etc.

Results: Energy Optimization on Xeons (1)

- Developed energy optimization capabilities targeted to European centers
  - Studied FT, miniFE, Nekbone, and Gadget workloads on Quartz system at LLNL (60 node allocation, nodes are 2S with BDX processors)
  - GEOPM optimizes processor core frequency for a given application or phase
    - Programmer uses GEOPM profiling APIs to mark up phases of their application
    - GEOPM performs trial runs of an application to characterize it: sweep over frequencies and measure resulting energy and time-to-solution for each application phase
    - For each phase, determines frequency that minimizes energy subject to the constraint that time-to-solution is degraded less than 10% (% is configurable)
    - Those phase frequencies will be used in subsequent invocations of the application: user inputs phase frequency config to GEOPM when queuing up their job
    - GEOPM features involved: Python scripts to perform characterization, profiling + reporting features in GEOPM framework to obtain measurements, GEOPM Agent plugin to dynamically track phase changes and apply best frequency

Results: Energy Optimization on Xeons (2)

- We compared per-phase frequency tuning vs applying best-fit across all phases
- Note
  - For Gadget, observed 69.9% MPI time (surprising but maybe not wrong if very imbalanced workload – looking into it)

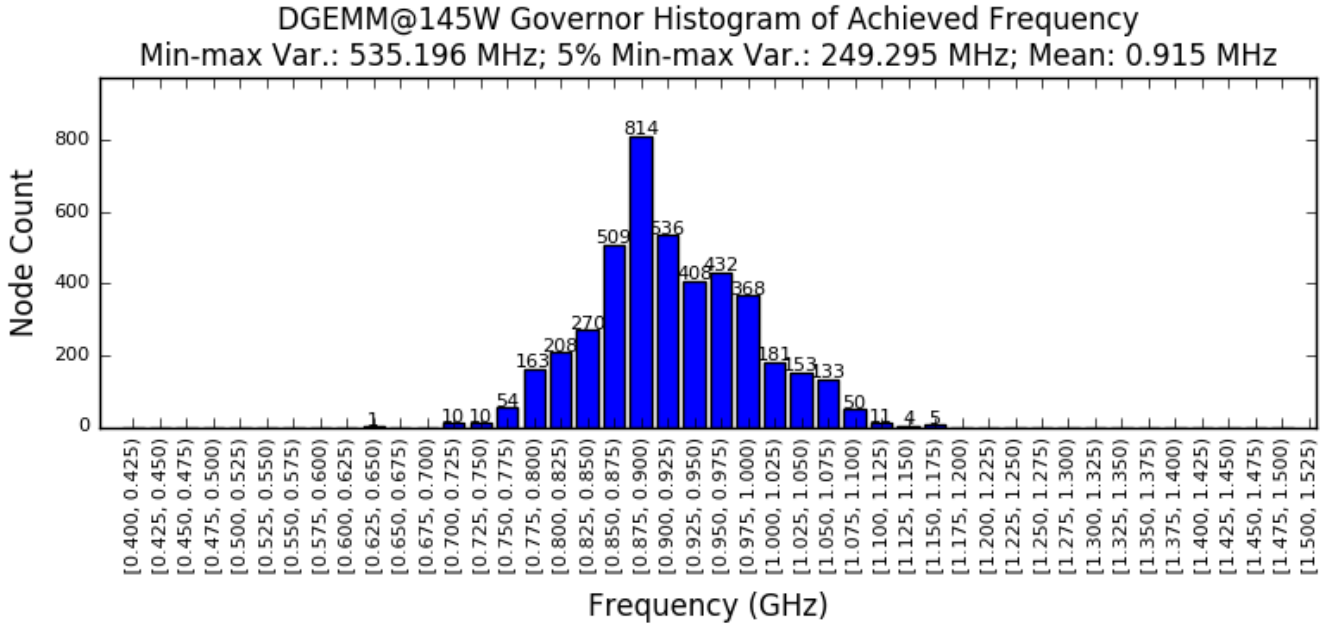
Energy-to-Solution and Time-to-Solution Comparison on Quartz				
	Offline Automatic ***Application*** Best-Fit		Offline Automatic ***Per-Phase*** Best-Fit	
Workload	EtS Decrease vs Sticker	TtS Increase vs Sticker	EtS Decrease vs Sticker	TtS Increase vs Sticker
FT	9.5%	6.8%	15.8%	4.8%
miniFE	8.5%	5.8%	Collecting data soon	Collecting data soon
Nekbone	7.9%	2.4%	Collecting data soon	Collecting data soon
Gadget	46.7%	2.9%	Collecting data soon	Collecting data soon

Results: Frequency Variation on Theta (1)

- Goal: understand how frequency varies across processors for various power caps
- Performed study on production system: Theta system at Argonne with ~4300 Intel Knights Landing processors and Cray Aries network fabric
- Performed study using GEOPM: its power capping capabilities, profiling capabilities, and analysis/plotting tools – **GREAT BASIC GEOPM USE-CASE!**
- Swept over a range of different node power caps and measured the achieved frequency on each node while running a DGEMM code
- Details:
  - For cleaner experimental analysis, we defined node power as socket power (did not include other node components, capped socket power only); held power cap constant over each run
  - Each node: 1 KNL processor w/ 64 cores and 16 GB MCDRAM; TDP = 215W; Turbo enabled
  - Local DGEMM computation on each node. Configured working set to fit in MCDRAM to minimize external DRAM access/power; this DGEMM consumes less than TDP even when not power capped.

Results: Frequency Variation on Theta (2)

- Significant frequency variation!
  - One example from sweep available now; will publish full set upon approval
- Example: DGEMM @145W cap
  - 535 MHz min-to-max variation
  - Still 249 MHz when discarding 5% least and 5% most efficient nodes
  - Data resembles log-normal distribution

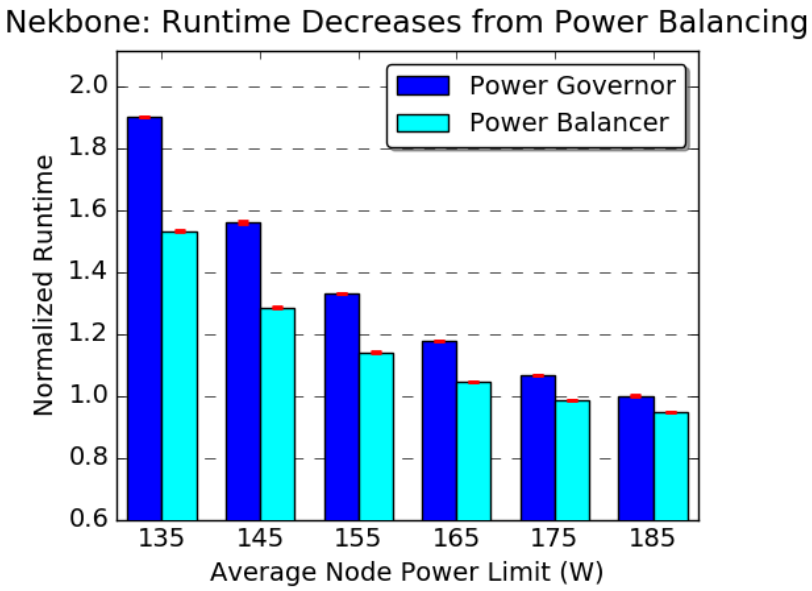


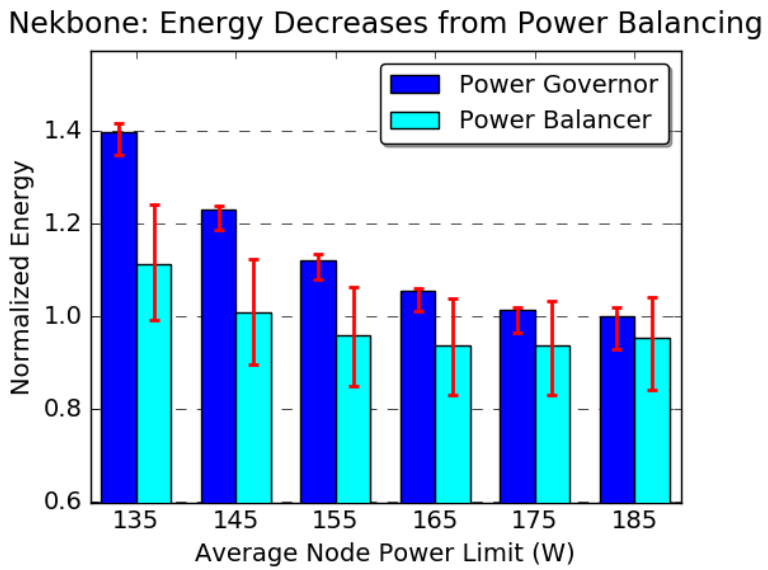
- What this means
  - Ratio of a node's achieved frequency to least efficient node's is the factor by which that node can be slowed down and still complete a given amount of DGEMM work in same time as least efficient node
  - If running bulk-synchronous workloads which tend to do frequent global synchronization across nodes, many of the nodes could be slowed down a lot without impact to time-to-solution
  - Alternatively, in a power-constrained scenario, time-to-solution could be reduced by diverting power from more to less efficient nodes
  - Data like this from Intel-internal studies in 2013 motivated us to develop GEOPM and an optimization plugin for GEOPM to capitalize on these opportunities

Results: Benchmarking GEOPM Power Balancer Agent Plugin (1)

- Purpose: correct for frequency variation under power caps due to manufacturing variation
- Approach:
  - Leverage programmer markup added near MPI collective in application to measure imbalance in runtime between global synchronizations on the nodes (i.e. 'epoch' markup)
  - Dynamically reconfigure node power caps via RAPL to balance times on each node while enforcing overall job power cap across all nodes
  - Measure and operate on node runtimes rather than frequencies to ensure proper handling of additional sources of imbalance beyond frequency variation
- Setup:
  - Ran on 256 nodes of Theta. Didn't pick any particular set of 256 nodes so they may not exhibit full extent of potential frequency variation on Theta
  - Swept over different job power caps comparing benchmark time-to-solution when using power balancer agent vs baseline. Baseline is static uniform division of job cap (like today's power capping frameworks do)
  - Targeted Nekbone benchmark. Ran it with 7 MPI ranks, 9 OpenMP threads per rank. Consulted Scott Parker to configure inputs to Nekbone. Increased iterations to 2000 to get longer runs (still < 6 mins)

Results: Benchmarking GEOPM Power Balancer Agent Plugin (2)





Notes:

- Left: runtime vs avg node power cap
- Right: energy vs avg node power cap
- Lower bars are better
- Results normalized to baseline at 185W (baseline in dark blue)
- Implemented baseline with another GEOPM plugin called 'Governor.' Sets a static uniform power cap via RAPL, RAPL enforces cap, plugin does no dynamic control during job

- Takeaways: Up to 19.4% reduction in time-to-solution with simultaneous 20.3% reduction in energy
  - Getting ~5% improvement in runtime and energy even at higher power caps where nodes are not very constrained
  - Our analysis shows up to 23.3% reduction in runtime is possible for longer workload runs (recall: these were < 6 mins)
  - Bigger benefits are possible if running on full Theta system and possibly even if running on a different subset of 256 nodes that exhibit more frequency variation than the 256 nodes we got did

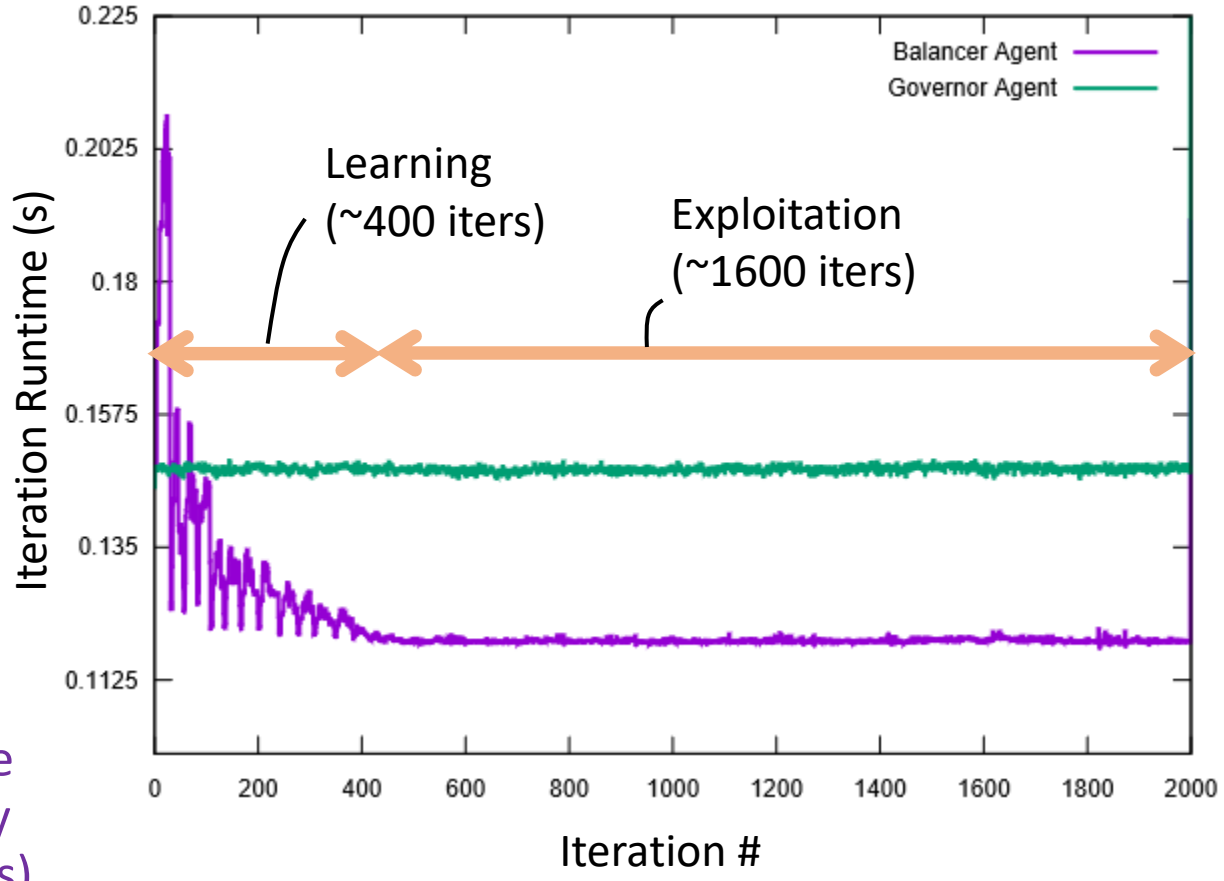
Results: Limit Study on Benefits of Power Balancer Agent Plugin (1)

- Purpose: estimate speedup and energy savings that current power balancing algorithm can provide if Nekbone runs longer. In our short runs, even small learning overheads eat into the speedup that's achievable
- Approach: leverage GEOPM to trace the Nekbone iteration time (max 'epoch' runtime across all nodes) while Nekbone runs. Look at iteration runtime after power balancer has learned / settled on steady-state power allocation. Project what runtime would be if power balancer instantly learned that allocation.
- Setup: same setup as Nekbone power balancer experiments on previous slides

Results: Limit Study (2)

	Governor	Power Balancer		Improvement	
Node Pwr Cap	Achieved Runtime (s)	Achieved Runtime (s)	Perfect Runtime (s)	Achieved (%)	Perfect (%)
135	361.01	291.47	276.66	19.27	23.37
145	296.37	244.26	238.90	17.60	19.37
155	252.69	216.54	213.99	14.27	15.33
165	223.72	198.70	197.73	11.20	11.63
175	202.68	187.15	186.72	7.63	7.90
185	189.87	180.09	179.94	5.17	5.23

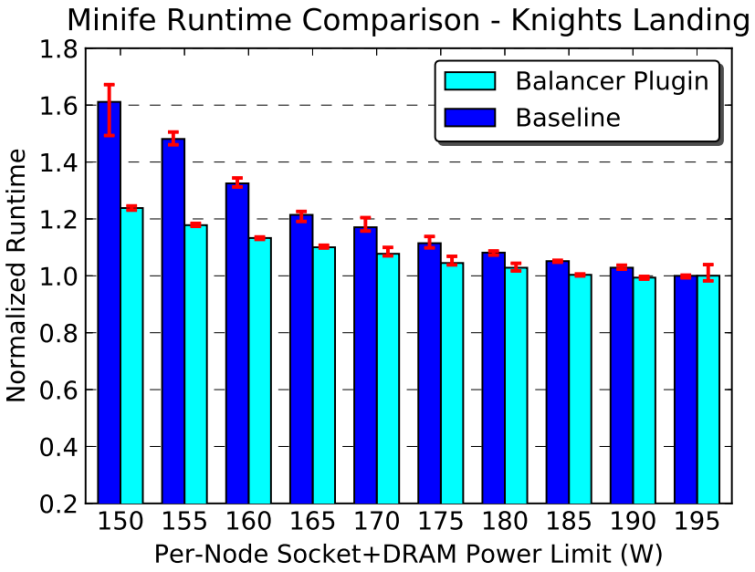
Nekbone: Trace of Iteration Runtimes @145W Node Power Cap

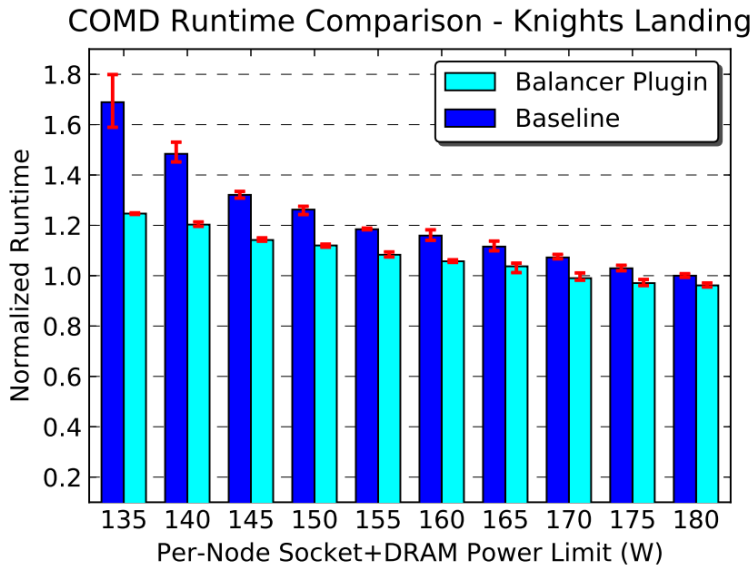


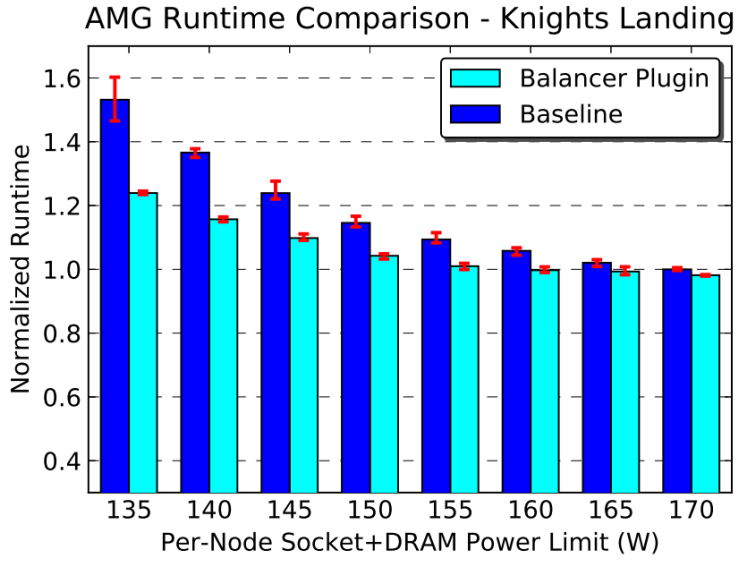
- Results: Up to 23.3% improvement is possible (in the limit) for longer runs of this Nekbone config. Already achieving much of it (especially at higher power caps)
- Note: Results collected under good “weather conditions” on Theta network fabric. High contention reduces opportunity for speedup as MPI calls become a bottleneck. However, bad “weather conditions” may increase energy savings opportunity as we can reduce processor core frequency with low runtime impact

More Results with an Intel KNL Cluster (1)

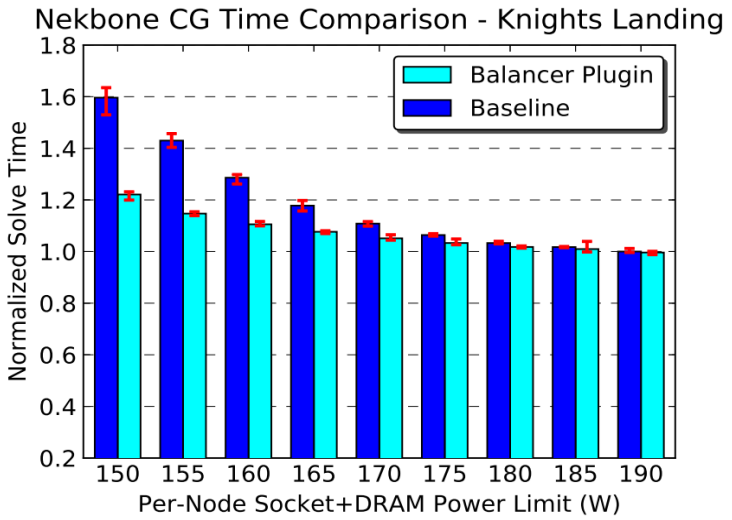
- These results from ISC paper by Eastep et al. published in June 2017
- Experiments are similar to the power sweeps on previous slide
  - Leverages GEOPM power balancing plugin to optimize time-to-solution within a job power bound
  - But, the experiments target a 12-node Intel KNL cluster not Theta and look at other workloads
- Main result: **up to 30% improvement** in time-to-solution at low end of caps (miniFE, CoMD, AMG), with **up to 9-23% for the rest.** Improvement generally increases as power is more constrained

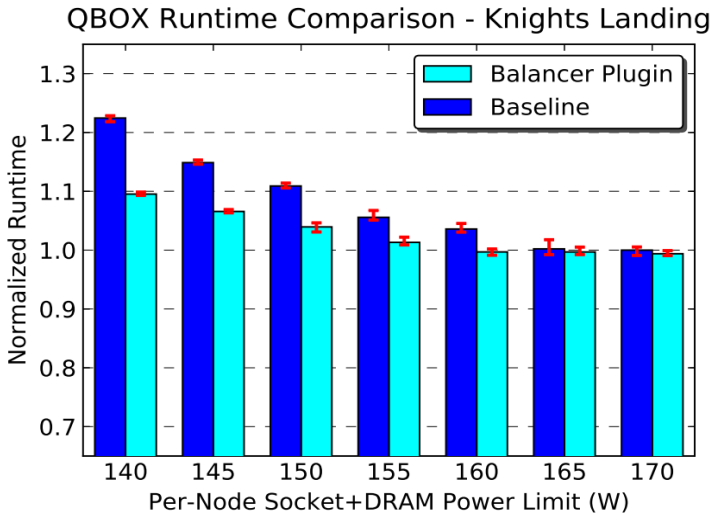


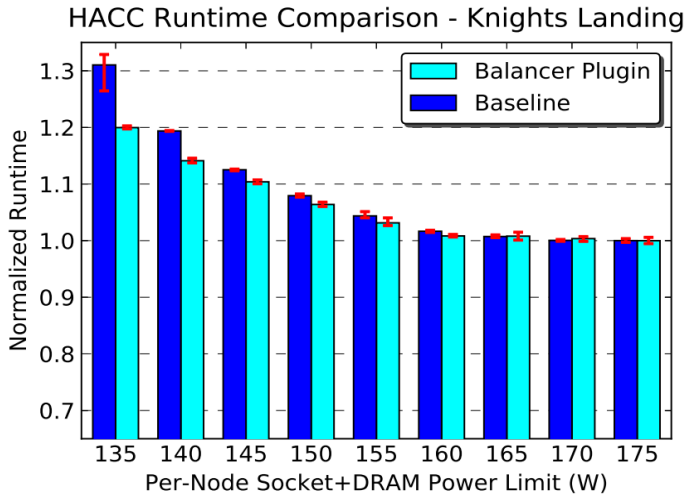


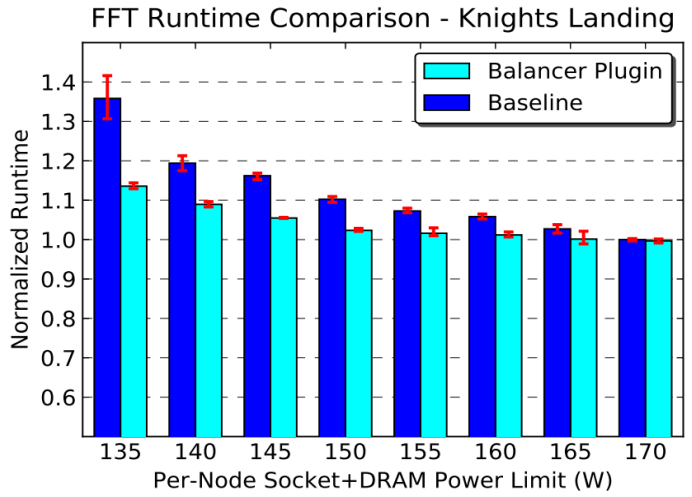


More Results with an Intel KNL Cluster (2)







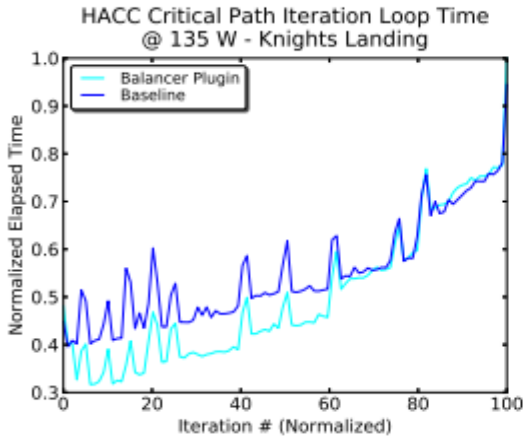


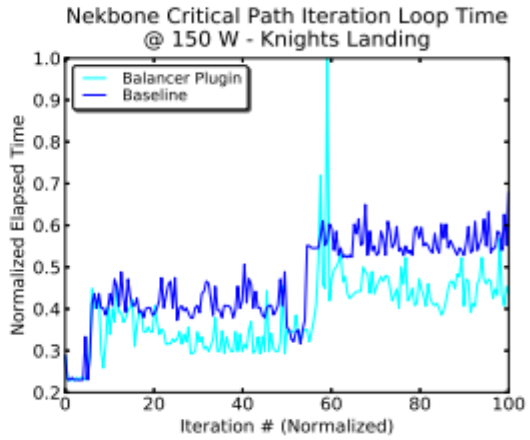
GEOPM Speedup Analysis

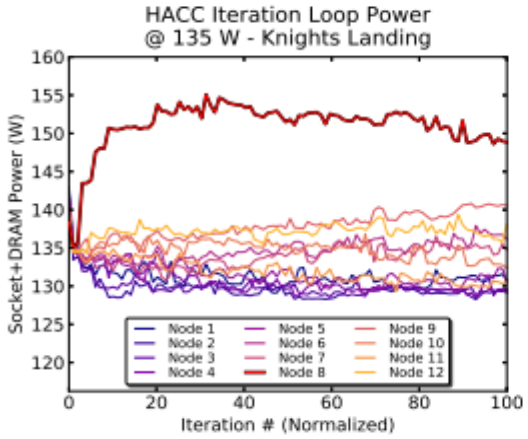
(using included GEOPM Trace and Python Visualization Tools)

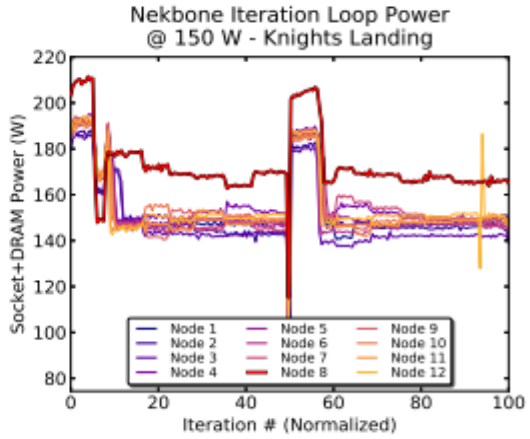
Take-away points:

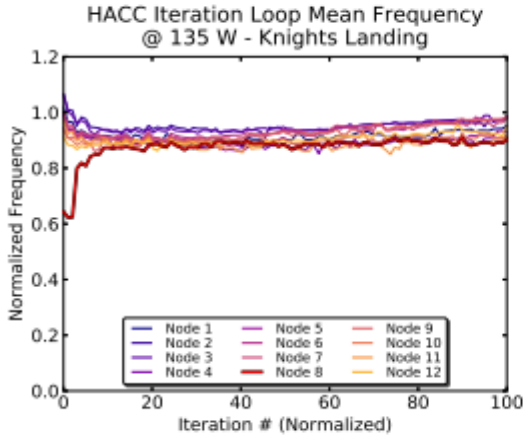
- Results demonstrate robustness of power balancing algorithm against time-varying amounts of work in the outer loop and sharp shifts in computational-intensity (top graphs)
- Node 8, with lowest power efficiency in our KNL cluster, is allocated more power (middle graphs)
- Power balancing algorithm improves critical path loop time by finding the power allocation that roughly equalizes the frequencies of all nodes (bottom graphs)

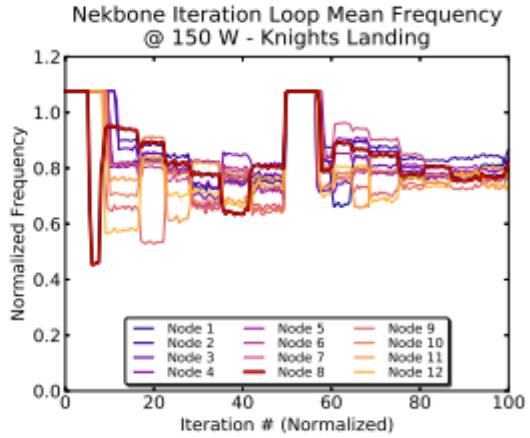












GEOPM Traction

- Project milestones we have reached
  - GEOPM v1.0.0 release now available!
  - Adopted into OpenHPC as of v1.3.6, ECP Extreme-Scale Scientific SW Stack (E4S) as of v0.2, and PowerStack
  - Deployed on Argonne Theta system. Used to characterize power/perf/thermal variation across nodes
  - Aiming for more deployments in 2019: collabs with LRZ, CINECA, Sandia, LLNL, Hartree, JCAHPC, …
  - Integrated into system power management solution of future Intel systems (including Exascale system)
  - Included in Intel and HPE bids for future HPC systems. Pillar of system power management solution
  - Receiving code contributions from IBM, Hartree, and LLNL. GEOPM port for POWER9 + Nvidia HW platforms
  - Leveraged internally at Intel as an energy modeling and profiling tool
  - Drove HW/FW features for GEOPM into broad-market Xeon server products
- Impact
  - Emerging as the standard in scalable power and performance management runtimes for HPC
  - Setting new directions in approach to power management throughout the industry and community
    - Established the notions of job-level power management agent and hierarchical power management
    - One of the first production in-band PM runtimes leveraging app-awareness + profile-guided optimization
    - One of the first production HPC PM frameworks to support extensive customization + portability via plugins

Next Steps

‼ roadmap subject to change

- GEOPM deployments anticipated on additional production systems in ‘19
  - Already deployed on Theta KNL system at Argonne in ’18. Used for system characterization
  - Collaborating toward deployments at LRZ, CINECA, Sandia, LLNL, Hartree, JCAHPC, …
- GEOPM ports for additional architectures anticipated in ‘19
  - Will support new Intel Xeon processors coming out soon
  - POWER9 + Nvidia port of GEOPM already well underway, driven by IBM/LLNL with Intel support
  - AMD or ARM port. HPE, AMD, ARM, LLNL, Intel already talking about collaboration on that
  - Intel CPU + AMD/Nvidia GPU port of GEOPM? Seeking collaborations on this too
- GEOPM optimization plugins may be expanded in ‘19
  - Possibility #1: phase-adaptive optimization of processor core frequency for energy efficiency
  - Possibility #2: extensions for tuning parameters in SW layers (e.g. tune thread concurrency and MPI/OpenMP runtime parameters). Move beyond PM and add perf optimization / auto-tuning
- GEOPM will be integrated with resource manager / schedulers / profilers in ‘19
  - Enable scalable in-band global monitoring, system power capping, power-aware scheduling, etc.

Join the Rapidly Growing Community!

- See our website for updates:
  - https://geopm.github.io
- Download source and try tutorials:
  - https://github.com/geopm/geopm
  - https://github.com/geopm/geopm/tree/dev/tutorial
- Collaborate on new features or bug fixes:
  - https://github.com/geopm/geopm/issues/
- Provide feedback or have Q&A via slack channel:
  - https://geopm.slack.com
- Reach out to me personally with collab proposals:
  - jonathan.eastep@gmail.com

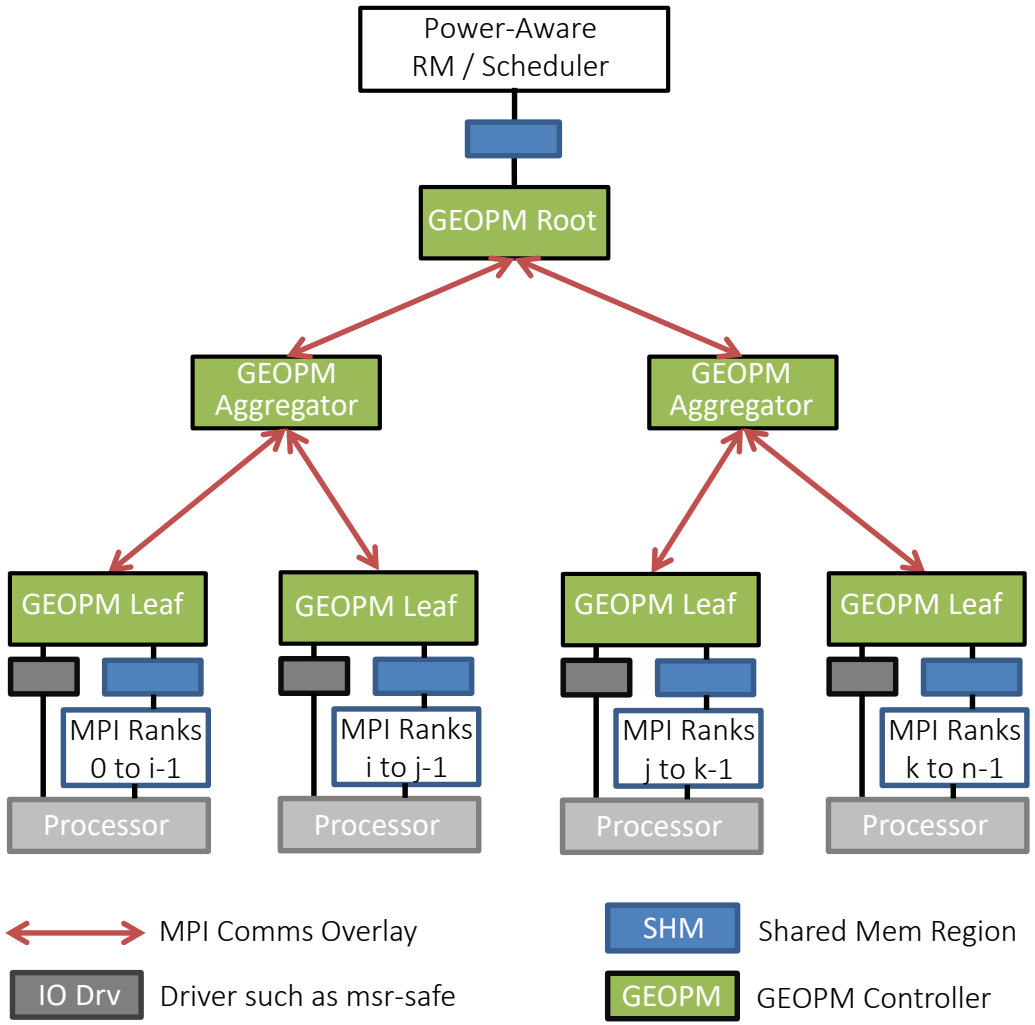
Intel GEOPM Software Team

Jonathan Eastep (Project PI)
Chris Cantalupo (Dev Lead)
Brandon Baker (Dev)
Diana Guttman (Dev)
Brad Geltz (Dev)
Ali Mohammad (Research)
Siddhartha Jana (Research)
Matthias Maiterth (Research)

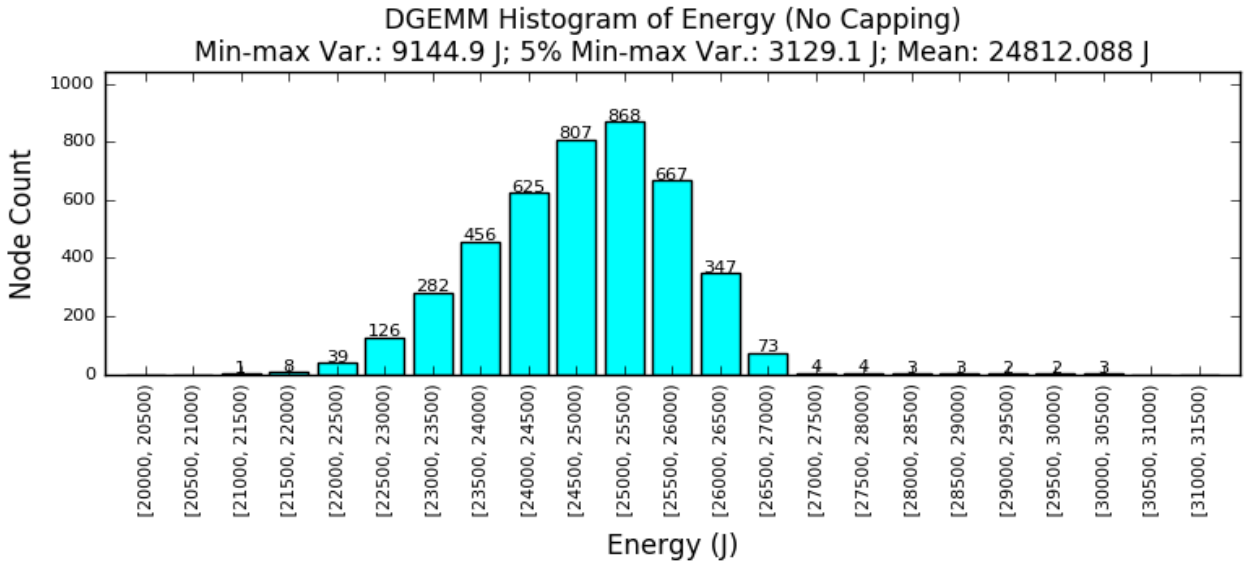
Backup

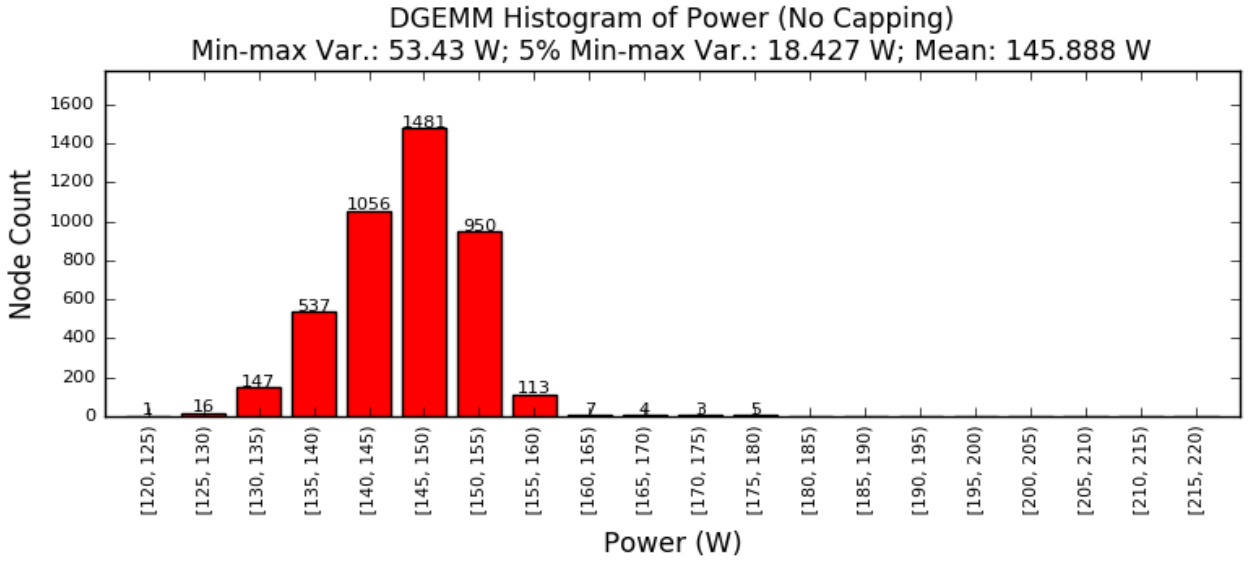
GEOPM: Design and Communications

- Scalable global tuning is achieved via tree-hierarchical design and decomp
  - Tree hierarchy of agents
  - Recursive control and feedback
  - Strategy is extensible via 'Agent' plugins
- Implementation info
  - Agents run in job compute nodes, one GEOPM process per node, in a core reserved for OS
  - Tree comms go over in-band network, use MPI
  - Application profile data collected via PMPI, OPMT, and optional provided programmer API; data exchange via inter-process shared mem
  - Access to HW controls / monitors achieved via driver (e.g. power caps, DVFS, energy counter)



Results: Energy/Power Variation on Theta





- Goal: understand variation in power / energy with no processor power caps
- Performed study leveraging GEOPM features, targeted DGEMM again
- Takeaways
  - Substantial variation in energy and power consumption across processors (~53 Watts)
  - Distributions resemble log-normal distributions
  - Ratio of energy consumption between any node and lowest energy node is the factor by which that node is less energy-efficient
  - Not all processors reach TDP (215W). Not all reached max 1400 MHz Turbo. We think some reached electrical limits before TDP
  - Variation in energy consumption implies an opportunity for scheduling, load-balancing, and further GEOPM energy mgmt research