Resource Partitioning and Power Management in the Exascale Era

ANL: Swann Perarnau, Idriss Daoudi, Kamil Iskra, Kazutomo Yoshii, John-Luke Navarro, Pete Beckman
LLNL: Tapasya Patki, Stephanie Brink, Aniruddha Marathe, Barry Rountree
University of Arizona: David Lowenthal and team

Improving all layers of the open-source resource management ecosystem

The goal of the Argo resource management effort is to provide user-facing advanced mechanisms to control and monitor resource usage across the system. This includes performance isolation, support for advanced workloads such as workflows and coupled-codes, and comprehensive power management.

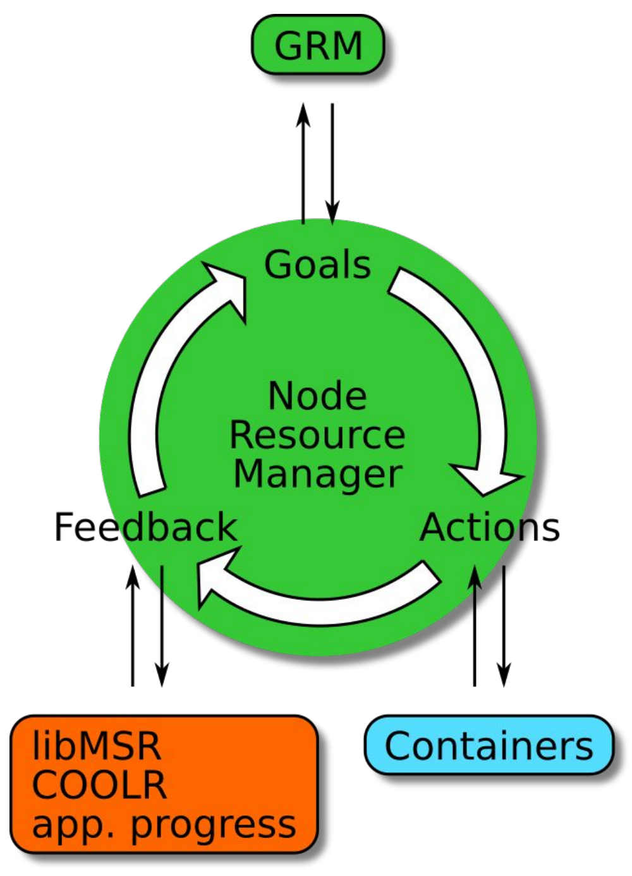
Local (Node) Resource Management

Overview

- Hierarchical resource partitioning
- Containers for intra-node resource partitioning
  - Using the cgroups mechanism of Linux
- More efficient “packing” of multi-component applications
- Arbitrate resources between applications and runtime services
- Reconfigurable, dynamically tracking resource changes
- Integration with batch schedulers, power management

Node Resource Manager

- Single API endpoint for all node resource management services
- Map containers to topology, interact with container runtime
- Pub/Sub API to control, measurements events
- RPC API: user access to container management, control loop configuration, actuators
- Goals: integration across hierarchy levels, collaboration with job schedulers, MPI

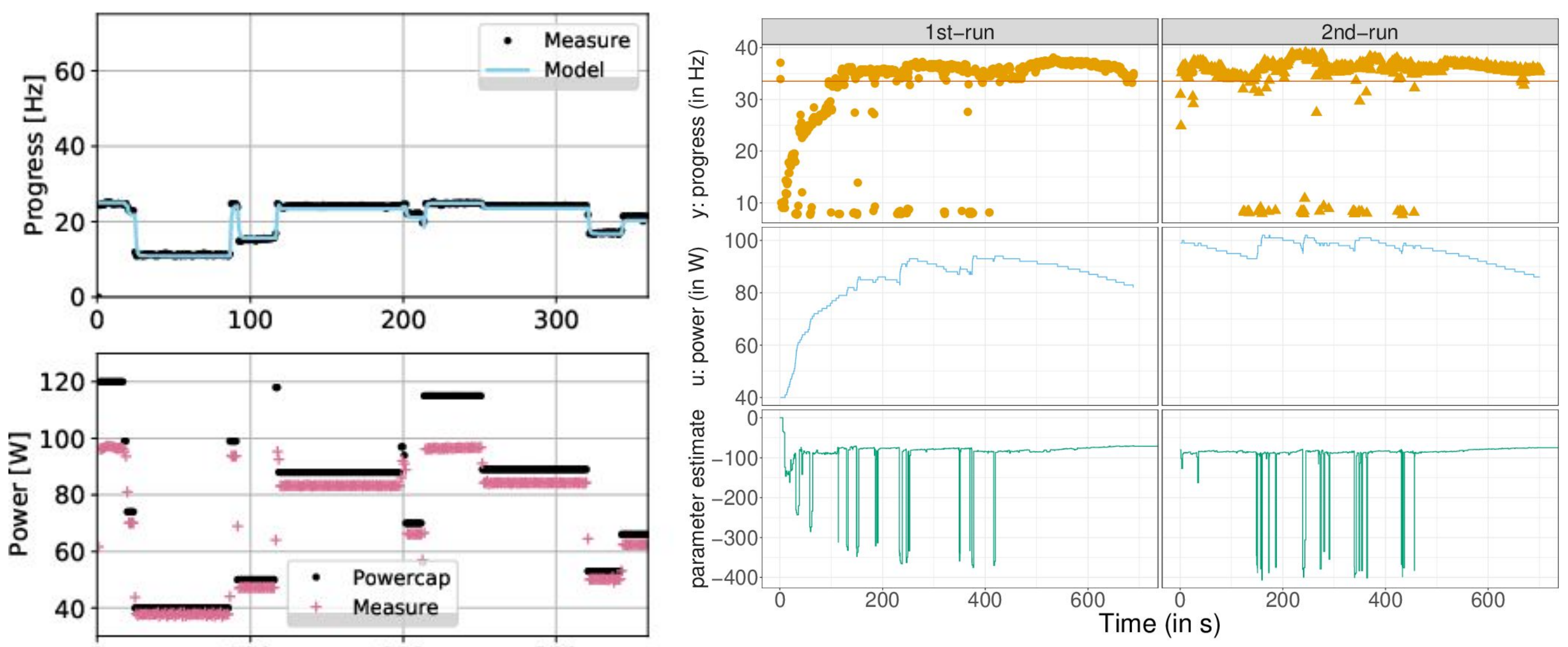


Sensors, Actuators, and Configurable Control

NRM performs abstracted resource accounting in the form of “sensors” and “actuators”, allowing for flexible control design.

- Monitor application and hardware through multiple APIs: self-reporting progress, PMPI, hardware performance counters
- Actuators act on hardware/application through resource arbitration layer: RAPL, control groups, signals
- Control loop is expressed as a function of available sensors, actuators, and user-defined goal.
- Specified through a Control Problem Description format
- Currently synthesizes a Multi-Armed Bandit controller
- ECP goals: improve control loop, integrate more sensor data into policies, better sensor data management

Reconfigurable, Application-Specific Control Loop



Modelization of application progress and power consumption, and adaptation of the controller parameters during the two-runs initialization phase.

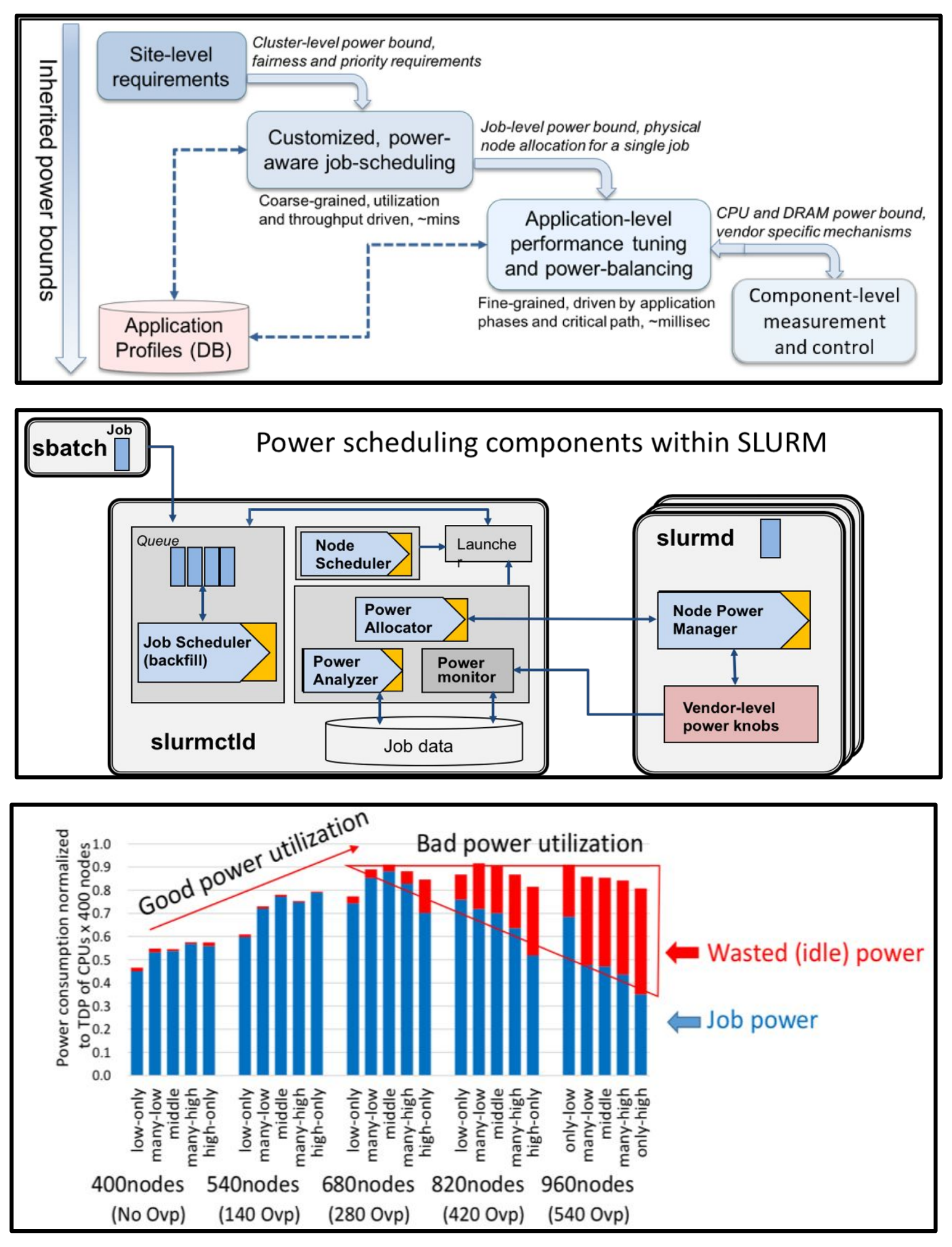
NRM Latest Features

- https://nrm.readthedocs.io
- Added scope (topology information) to sensor data, to map monitoring position to actuators granularity
- Support for OMPT, variorum, PAPI as sensor providers

Global Power Management

Scheduler-Aware Hierarchical Control

- Integrate job scheduler, enclaves to control power across jobs
- Use NRM data to monitor power and application performance
- Steer power where it can most advance the application’s progress
- ECP goals: production-ready GRM, integration across different hierarchy levels
- PowerStack: Developed first prototype with SLURM, GEOPM and Variorum to lead community effort toward capturing power management details from the microarchitecture-level up to the site-level.
- PowerStack continues to bring together industrial partners as part of CRADA effort and active collaborations with vendors (HPE, ARM, Intel, AMD, IBM and NVIDIA).


Inherited power bounds
Site-level requirements
Cluster-level power bound, fairness and priority requirements
Customized, power-aware job-scheduling
Job-level power bound, physical node allocation for a single job
Coarse-grained, utilization and throughput driven, ~mins
Application-level performance tuning and power-balancing
CPU and DRAM power bound, vendor specific mechanisms
Fine-grained, driven by application phases and critical path, ~millisec
Component-level measurement and control
Application Profiles (DB)
Power scheduling components within SLURM
sbatch
Job
Queue
Job Scheduler (backfill)
Node Scheduler
Launcher
Power Allocator
Power Analyzer
Power monitor
Job data
slurmctld
slurmd
Node Power Manager
Vendor-level power knobs
Power consumption normalized to TDP of CPUs x 400 nodes
Good power utilization
Bad power utilization
Wasted (idle) power
Job power
low-only
many-low
middle
many-high
high-only
only-low
only-high
400nodes (No Ovp)
540nodes (140 Ovp)
680nodes (280 Ovp)
820nodes (420 Ovp)
960nodes (540 Ovp)

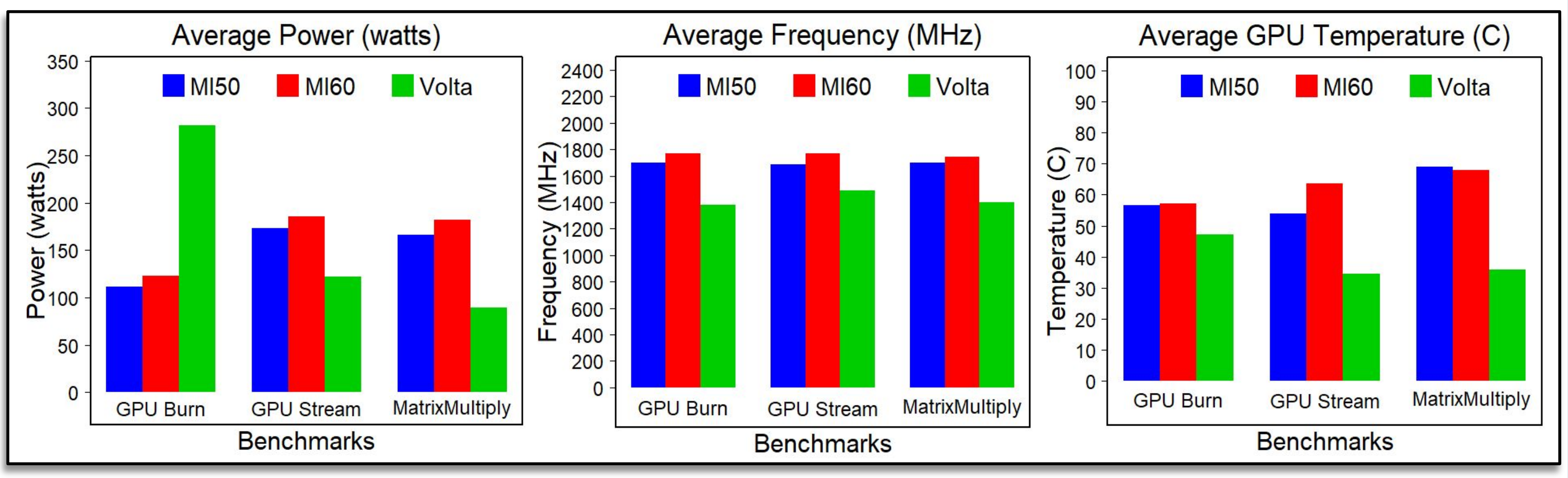
As part of PowerStack, a power-aware SLURM (GRM) has been developed and tested on 960-nodes on the HA8K supercomputer in Japan (collaborators). Results show the benefits of hardware overprovisioning at scale as well as sensitivity to the degree of overprovisioning.

Variorum Latest Features

- https://variorum.readthedocs.io
- Extensible, open-source library for exposing low-level hardware knobs and vendor-neutral API for power management
- Adds support for AMD (under NDA), allowing for 4 platforms and 10 microarchitectures to be supported.
- Extends JSON-based API support for interaction with other system software components. Additionally, integration with Kokkos, Flux, and Caliper has been implemented.

Evaluation of Nvidia and AMD GPU Power Knobs

- **Objective**: Investigate available methods for power management on GPUs from different vendors
- **Test Platforms**: MI50 and MI60 AMD GPUs on Corona system, Volta GPUs on Lassen system at LLNL
- **Results**: Observed interesting power, frequency and thermal tradeoffs with different benchmarks



This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy’s Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation’s exascale computing imperative.
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-POST-803567