Power vs. Energy

- Which is actually important?
  - Both!

- Energy is just the product of power and time
  - The site’s bill depends on it
  - May be limited based on a fixed budget
  - However, may also be abundant at year’s end
  - Used for billing centers

- Power is a limited resource
  - Practically, there will be only so much power available
  - Limits in the incoming Power Grid, Center Infrastructure, System Architecture
  - Often also closely connected to thermal limits
  - Runtime constraint (maximum consumption over time interval)
LRZ Total Cost of Ownership looking towards Exascale

Billing HPC Users

- CPU-hours as traditional billing metric
  - Does not reflect actual cost of usage

- Center is billed by Energy costs

- This could be passed through to the HPC user in the future
  - Projects allocated by KWh not CPU-hours
  - This metric is more fair since not each CPU hour has the same cost.

- “Virtual Change” no change to how the system is actually used.

Why is This an Application Developer’s Problem?

- Should be managed in hardware and be transparent
  - That’s how it has been so far but will be insufficient at exascale

- Significant improvements from the hardware technology coming to an end
  - Moore’s Law?

- Use of specialized hardware will increase
  - Implicit power management (e.g., Turbo Mode)
  - Power efficient accelerators

- Application developers should be aware:
  - Application need to contribute for maximum efficiency
  - Application runtime depending on power/energy saving optimizations and mechanisms
  - Performance heterogeneity (e.g., Turbo Mode)

The 4 Pillar Framework – General Overview
The 4 Pillar Framework – Work

External Influences/Constraints

Data Center

Pillar 1
Building Infrastructure

Pillar 2
HPC System Hardware

Pillar 3
HPC System Software

Pillar 4
HPC Applications

Results

Heat

Work

External Influences/Constraints

Data Center

Neighboring Buildings

Utility Providers

Others

Pillar 4
Applications

Pillar 3
HPC System Software

Pillar 2
HPC System Hardware

Pillar 1
Building Infrastructure

Pillar 2
HPC System Hardware

Pillar 4
Applications

Where does Our Energy go?

Goal of this Tutorial

- Raise awareness of the drastic shift in system usage we are expecting
  - Power will be one of the constrained resource
  - Won’t go unnoticed to the user
- Past, current and future trends in power aware HPC
  - From hardware to applications
  - With an eye on what this means for the application developer/user
- What can users expect going forward
  - What impacts them directly
  - What can they do to help the system (or themselves)
- Share our experience in working in this area for at least a decade

- Need to understand each pillar
- Optimize and measure (KPIs) for each
- Need global approach for optimal results
  - Includes utility provider
  - Define operating points
  - Keep infrastructure efficiency constant over the whole operating range
  - Measure and assess

Goal: Reduce Total Cost of Ownership

- Initial support (1st, 2nd)
- Ongoing support (3rd, 4th, 5th)
- Power [W1 – W5]
- Smart Grid
- Water [18°C – 27°C]
- Air

Presenters

- Dieter Kranzmüller, LMU/LRZ, Germany
- Torsten Wilde, LRZ/TUM, Germany
- Matthias Maiterth, Intel/LMU, Germany

Content Contributors

- Barry Rountree, LLNL, USA
- David Lowenthal, University of Arizona, USA
- Martin Schulz, LLNL, USA

Outline

- Section 1: Power-aware architecture and their impact on Applications
  - Hardware trends and developments
  - How this impacts applications developers
- Section 2: Power and Energy Aware HPC Centers
  - New technology in HPC Centers
  - Concerns that drive policy decisions, which are ultimately exposed to users
- Section 3: Power-aware resource management
  - Transparent power shifting or what the system will do to your application
  - Power awareness as an integral part of the job scheduling system
- Section 4: Power-aware runtimes
  - Techniques to reduce energy and maximize power usage
  - What is hidden, what can be hidden, and what should be exposed

“Rules of Engagement”

- This is the third time we are giving this tutorial
  - We want feedback
  - We can deviate from the slides if useful
- Let’s keep this interactive
  - Ask questions
  - Share your experiences
Blue Gene/Q active power well under power allocation

Linpack uses up to 2.3MW

3-minute sample intervals

97.8% of samples > 50% peak power

Behavior observed across all recent platforms

All other codes use much less power

Similar or worse utilization expected at 20MW

Extreme scale architectures capped at 20MW

Current practice

Result:
- Wasted money
- Underutilized infrastructure

Ideal power-limited architectures use all provisioned power

Possible Solution:
Overprovision hardware

Not all nodes can run at max power simultaneously (bad for Linpack)
All nodes can be run at (say) 50% power (great for everything else)

Try to use all available power in a power-limited system

Processor and job configuration matter

CFD solver kernel

Not-power aware configuration (red)

Max cores, power per processor

Only 24 nodes, cannot use all allocated power
**Processor and job configuration matter**

**Optimal configuration (blue)**

95W power bound, 14 cores

Result: 2x faster

Hardware overprovisioning leads to far better performance in a power-limited system.
Impact on Applications
(shown at 80W and 65W processor power bound)

Census across 2386 processors
NAS Parallel Benchmarks: mg - multigrid, ep - embarrassingly parallel
Normalized to fastest unbounded run
x-axis = slowdown, y-axis=CPU Freq.

Power-Aware Architectures can Mess Up Your Day!

- All systems are Power Aware in one way or the other
- As soon as you want to leverage Power-Aware Architectures there are lots of factors to consider
- HPC Community has to deal with this, but:
  - Where? (What part of the system?)
  - When? (Configuration / Runtime / Analysis?)
  - How?
- If done right: Cost saving & performance improvement

Some more Facts

- 3160.5 m² (34 019 ft²) IT Equipment Floor Space (6 rooms on 3 floors)
- 6393.5 m² (68 819 ft²) Infrastructure Floor Space
- 2 x 10 MW 20kV Power Supply
- Powered Entirely by Renewable Energy
- > 5M € (> 6M US$) Annual Power Bill

The Leibniz Supercomputing Centre
Data Center Sustainability

Sustainability as a goal:
“Get work done without environmental impact and conserve natural resources”

And/or:
“Be beneficial to the power grid”

4 Pillar Framework for Energy Efficient HPC Data Centers

Energy Consumption in Data Centers

- Data Centers are "Heaters with integrated logic"
SuperMUC Phase1 HPL Energy Consumption

Outside Conditions and LRZ Cooling Efficiency

Cooling Technology Efficiency

Why Is the Infrastructure Important?

- Determines data center overheads
  - Matching average operating power consumption with cooling infrastructure will reduce overheads
- Infrastructure limits the possible cooling technologies for the HPC systems
  - Being set up only for air cooling will not allow an easy switch to water cooled systems
- Trade-offs to reduced overhead can be a source for additional costs later on
  - Switching of power conditioning to reduce overheads can allow brown-outs to shutdown or damage system parts
- Mistakes made here can be costly in the long run
  - HPC system replaced every 3-5 years
  - Infrastructure replaced every 10-20 years
- Infrastructure is a major long term investment, OR is it?
Sustainable Exa-Scale Computing

- Pre-exa-scale:
  - Build Data Center (DC) to support multiple HPC system generations
  - One digit MW systems
  - Difficult to justify increased CAPEX for OPEX savings since budgets disjoint
  - High safety margins (over specification) in DC design to insure stable operation
  - No direct interaction between building automation system and HPC system

- Co-design of Exa-Scale system and Data Center
  - Build and optimize data center to reduce OPEX as much as possible
  - 20-50MW (system or system+data center?)
    - 35MW (LRZ 0.15€) ~ 46M€ per year (nearly as much as LRZ data center)
    - Additional CAPEX investment can save a substantial amount of OPEX
  - None or very small safety margin in DC design
  - Need turn-key solution
  - Software required to integrate all 4 Pillars
  - Re-use of IT waste heat

Early Work Air vs Direct Liquid Cooling

- CoolMUC (178 nodes):
  - Measurement for single scale
  - Two AMD Opteron 6228 HE CPUs (MagnyCours) with 8 cores each
  - 2400 MHz clock frequency
  - 24GB RAM (8x 2GB DDR3 modules)
  - Characteristics will change with CPU and System architecture

- LRZ operation:
  - Winter 36°C intake
  - Summer 40°C intake

Cooling SuperMUC
Other efforts:
- TU Dresden: HDEEM
- CINECA: Examon
- LLNL

- Lenovo NeXtScale Water Cool (WCT) system technology
- High Temperature Direct Liquid Cooled (HT-DLC) with Water inlet temperatures 30°C – 50°C, all season chiller-less cooling
- 384 compute nodes, 466 TFlop/s peak performance, #356 on the Top500 list (June 2016)
- Adsorption chiller, at least twice as efficient as traditional mechanical chillers
Average COP Analysis CoolMUC-2 Cooling (Jan-Nov 2016)

Performance of Different Cooling Technologies (Jan-Nov 2016)
(Combined 13th/16 + 8/9th/15 heat + 9/8th Storage Cost)

- Free Cooling: 16.72
- Compression Cooling: 3.91
- Mixed (Compression+Free): 7.84
- Adsorption Cooling (Total): 18.38

Other Factors - Leakage currents

<table>
<thead>
<tr>
<th>HT-DLC Inlet Temperature (°C)</th>
<th>El. Power (W)</th>
<th>Increase (%)</th>
<th>Die Temp (°C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>4815</td>
<td></td>
<td>50.14</td>
</tr>
<tr>
<td>35</td>
<td>4844</td>
<td>0.6</td>
<td>55.35</td>
</tr>
<tr>
<td>40</td>
<td>4877</td>
<td>1.3</td>
<td>60.15</td>
</tr>
<tr>
<td>45</td>
<td>4901</td>
<td>1.8</td>
<td>65.25</td>
</tr>
<tr>
<td>50</td>
<td>4915</td>
<td>2.1</td>
<td>67.31</td>
</tr>
</tbody>
</table>

Heat Transfer Relative to Cluster Power Consumption (Acceptance test Dec 2015 with 5 adsorption chillers)

Traditional CoolMUC2 Cooling Setup (30°C Inlet) (January 2016 till November 2016 - COP 7.95)
Exascale Power Variability

Prediction using current SuperMUC behavior:

Assuming Exascale system (compute only) with average power consumption of 35MW (at 2.3GHz) and chiller-less cooling infrastructure (LRZ free cooling COP between 10 and 25 translating to 10% to 4% overhead)

System variability (including cooling but not including electrical transmission and conversion losses) would be:

- Idle Power: (11.65MW – 12.3MW) (probably much less with newer compute architectures)
- Average Power: (36.4MW – 38.5MW)
- Full Power: (48MW - 50.8MW)
- System Power Variability: (50.8MW – 11.65MW) = 39.15MW

Running a Data Center on 100% Renewable Energy

Date of maximum total and peak solar power production (in GW and GWh): Friday 6th of June

Date of maximum total and peak wind power production (in GW and GWh): Friday 12th of December

**Heat Re-Use**

- Is throwing away energy the best we can do?
- Heat re-use with air cooling and low temperature water is hard
- Data center needs to be located close to possible consumers
  - Heat requirement average single family home 150m² \* 2.5m²
  - Outside -10°C (inside 20°C) 15.66 kW
- SuperMUC Phase1 (2.4MW) waste heat would be sufficient to heat 153 single family homes under those conditions
- LRZ office building heating

**More Re-Use Possibilities**

- Ice-Cream production: 30-40°C
- Cloth washing: 60-80°C
- Decaffeinated coffee: 22-100°C
- Surface treatment:
  - anodizing: 5-42°C
  - plating cooper: 30-100°C
  - Pre-heating of boiler feed water: 30-100°C
- Adsorption cooling (currently prototype, production ready installation)
- Beer production: 7-76°C

**Summary**

**Computing Center Perspective:**

- Need holistic approach for power awareness and energy efficiency
- Optimize and measure (KPIs) for each component
- Includes utility provider
- Define operating points
- Keep infrastructure efficiency constant over the whole operating range
- Measure, assess, optimize
Summary

User Perspective:
- Compute cycles (and storage capacities) have a value and a cost (in USD/EUR)
- The costs of your cycles are (often) paid for by the computing center
- More capacities and new investments at the computing center require money
- Code with good performance is in most cases also energy efficient
- If available, use power aware scheduling and energy saving functionalities
- Use your cycles with care (scaling at exascale)

How Can We Address the Power Issue?
- Build power-efficient data center
- Build more power-efficient hardware
- Manage power in software
  - Control different power-scalable components
    - CPU, memory, disk, etc.
    - Most work has focused on the CPU using dynamic voltage and frequency scaling
  - Before Job submission
    - Job Configuration
    - Scheduling
  - During Execution
    - Runtime
    - Feedback from Application and Hardware

Measuring and altering Hardware
- Changing Hardware Behavior / Control Techniques:
  - DVFS
  - RAPL
- Measuring Energy Consumption & Power
  - Hardware Counters / MSRs
  - Wall Plug Power Meters
  - Infrastructure Measurements (to coarse, except for post mortem analysis)
- APIs and Standard Interfaces
  - MRSs (accessible via libmsr [https://github.com/LLNL/libmsr])
  - PAPI (can now read Power related msrs)
  - PowerAPI (Specification for Platform idendepence)
The Classic Technique: Controlling CPU Power using Dynamic voltage and frequency scaling (DVFS)

- Reduce frequency & voltage
  - Reduces CPU power & performance
  - Energy-time tradeoff

- Why is this a good idea?
  - Applications may not be CPU-bound
  - CPU is large power consumer

- Goal: Find optimal operating point

\[ \text{power} \propto \text{frequency} \times \text{voltage}^2 \]

Is CPU scaling a win?

\[ \text{time} \]

\[ \begin{align*}
\text{P}_{\text{CPU}} & \quad \text{E}_{\text{CPU}} \\
\text{P}_{\text{other}} & \quad \text{E}_{\text{other}} \\
\text{P}_{\text{system}} & \quad \text{E}_{\text{system}}
\end{align*} \]

Is CPU scaling an energy win?

\[ \text{time} \]

\[ \begin{align*}
\text{P}_{\text{CPU}} & \quad \text{E}_{\text{CPU}} \\
\text{P}_{\text{other}} & \quad \text{E}_{\text{other}} \\
\text{P}_{\text{system}} & \quad \text{E}_{\text{system}}
\end{align*} \]

Optimizing Frequency, Energy vs Time

Averaged across all LRZ workloads:
Multidimensional Optimization

- Dependent on:
  - Workload
  - Job size

How do we automate selection of frequency?

- When do we get a benefit?
- Want to limit user involvement
- Want to select frequency automatically, via run-time system, based on application characteristics
  - First steps „energy tags”
  - Next: Full analysis / using malleability and moldability of jobs

Scheduling using Energy-TAGs at LRZ

- Energy-tag added to job script
  ```ruby
  @energy_policy_tag = my_energy_tag
  @minimize_time_to_solution = true
  ```
- Execution runs with standard frequency setting (2.3 GHz)
- Runtimes and Energy saving is assessed:
  - if Runtime decrease is more than:
    - 2.5% → frequency set to 2.4 GHz
    - 5% → frequency set to 2.5 GHz
    - 8.5% → frequency set to 2.6 GHz
    - 12% → frequency set to 2.7 GHz
- Relearning of energy settings possible.

2.3 GHZ as SuperMUCs sweet-spot

- Energy Policy Selection
- Energy Policy Selection
- Energy Policy Selection
- Energy Policy Selection
Optimization by Job Configuration and Scheduling

- Job Configuration requires application knowledge
- Different inputs can change behavior
- Possible input to Scheduler:
  - Scheduling information based on Time, Node requirements AND Power information.
  - Application Characteristics as Scheduling information. (Phases, I/O, Communication)
- Requirements from the Scheduler:
  - Information from User
  - Moldable and Maleable Job

Resource Management Challenges

- Hardware overprovisioning with power bounds
- Utilize the procured resource better and minimize waste
- Users care about fairness and turnaround time
  - Fair and transparent job-level power allocation
  - Minimize execution time, reduce queue wait time
- HPC Data Centers care about utilization and throughput
  - Maximize utilization of available nodes and power or energy
  - Minimize average turnaround time for job queue

Taking a Look at Two Approaches: RMAP and PARM

- Aimed at future power-constrained systems
  - Following prototypes implemented within SLURM
- Adding power awareness to the scheduler:
  - Two approaches:
    - (1) ILP at each stage (PARM)
    - (2) power-aware backfilling (RMAP)
  - Both improve system power utilization and optimize execution time under a job-level power bound
  - Both assume (at least) job moldability
- Both lead to significantly faster turnaround times than baseline SLURM

Power-Aware Speedup in PARM

- Basic idea: develop expression for speedup that takes resources into account
- Speedup relative to instance with minimum number of nodes and minimum amount of power
  - As long as application speeds up with nodes (it’s strong scaling), more resources means job runs faster
  - Sum of all jobs’ power-aware speedups is maximized, but power limit constraints prevent scheduler from running all jobs as fast as possible
- Estimate power-aware speedup for a job using regression
Performance Results (courtesy of Osman Sarood)

Lulesh, AMR, LeanMD, Jacobi and Wave2D
38-node Intel Sandy Bridge Cluster, 3000W budget

1.7X improvement in throughput

RMAP Insight: Power-Aware Backfilling
(using power instead of nodes as limiting resource)

RMAP Inputs and Adaptive Policy

- Inputs: Number of nodes, time; job level power bound derived based on node count
- If enough power is available, allocate the best overprovisioned configuration under the derived job-level power bound
- Otherwise, allocate a suboptimal overprovisioned configuration with available power
- Users can specify an optional performance slowdown threshold for potentially faster turnaround times. Default is no slowdown (0%)
- Result: 7% better turnaround than Traditional scheduling (0% slowdown) and 27% improved turnaround time with 20% allowed slowdown.

Inter-Job Power Optimization (Co-Scheduling)

- Can this be done effectively?
- Suppose there are repeatable phases
  - Can one application donate power to another app (in exchange for what?)
  - A global runtime system manages this
- Opportunities:
  - I/O phases
  - Low speedup phases
  - Sequential phases

**Variable Power Usage in I/O Phases**

(Paradis; Graph courtesy of Lee Savoie)

---

**Staggering Jobs**

- Can control when jobs enter I/O phase

---

**Controlling Jobs**

- Can prevent too many I/O phases from occurring simultaneously.
- Assumes that timing of I/O phases can be controlled externally.

---

**Results**

**Fifty Applications – Max Improvement**

![Graph showing percent improvement over no sharing for various factors like spread, stagger, control, and priority.](image)
Going Beyond I/O

- Similar techniques can be useful for any kind of periodic behavior
  - Multiphase algorithms (e.g., stencil and FFT)
  - Different application packages
  - Multi-physics codes

- Annotations by the developer can help
  - No need for power specific annotation
  - Phase markers are often enough

- Many things can also be done transparently

Scheduling Summary

- Focus on turnaround time, not execution time
  - Choose lower-power configurations; queue wait time decreases more than execution time increases
  - Scheduler needs a way to determine performance of different configurations

- Help from the Application
  - Optimization hints for Job Configuration
  - Moldability and Maleability

Optimization during Runtime

- What about after the jobs are scheduled—can we further optimize
  - Yes! During Runtime

- Two approaches
  - App independent approach
  - Using App Information for optimization
    - Imbalance
    - Semantic information / Annotations
Power Scheduling independent of Jobs

- POWSched
  - Job Scheduler not PowerAware
  - Power Scheduler runs independent
  - Power Rescheduled among all nodes in the cluster

- Considerations how to combine this with Power Aware Schedulers.
  - No Direct interaction with User/App
  - Implications for User?

- Work by Daniel Ellsworth

POWSched Results: Static vs Dynamic

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Runtime</th>
<th>Stddev</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>115W static</td>
<td>278.26</td>
<td>9.57</td>
<td>0.7%</td>
</tr>
<tr>
<td>115W dynamic</td>
<td>276.24</td>
<td>4.84</td>
<td></td>
</tr>
<tr>
<td>90W static</td>
<td>284.63</td>
<td>3.20</td>
<td>2.6%</td>
</tr>
<tr>
<td>90W dynamic</td>
<td>277.13</td>
<td>5.04</td>
<td></td>
</tr>
<tr>
<td>70W static</td>
<td>323.83</td>
<td>4.90</td>
<td></td>
</tr>
<tr>
<td>70W dynamic</td>
<td>278.02</td>
<td>4.97</td>
<td>14.1%</td>
</tr>
<tr>
<td>50W static</td>
<td>407.21</td>
<td>18.00</td>
<td></td>
</tr>
<tr>
<td>50W dynamic</td>
<td>371.92</td>
<td>13.23</td>
<td>8.7%</td>
</tr>
</tbody>
</table>

Basic Idea

- Monitor power usage of all applications in the system
  - OS level power monitors
  - Scalable aggregation
    - In a hierarchy representing applications
    - In a hierarchy representing the system

- Detect “wasted” power
  - Applications that don’t use their allocation

- Detect “needed” power
  - Applications that run against their power limit

- Shift “wasted” power to applications that need it

POWSched Summary

- Power Optimization != Power Bound Enforcement
- Static power allocation may not be optimal
- Dynamic power reallocation can reduce time to solution
**Scheduling Power Along the Critical Path**

**Example: Adagio - Overall approach**

- **Goal:** save energy with zero time delay and no user involvement
  - Matches HPC goals as opposed to environmental goals
- **Divide the (MPI) application into discrete tasks**
  - Task: code bounded by MPI communication operation
  - Intercept MPI calls via PMPI layer
- **Create a task graph to represent execution behavior**
  - Runtime analyzes task graph to determine what tasks can be slowed

Do all of this transparently

Adagio software is available: [https://github.com/scalability-llnl/Adagio](https://github.com/scalability-llnl/Adagio)


---

**Program Execution Time**

- **Determined by critical path** (longest path)
  - Tasks not on critical path can (potentially) be slowed
    - Running slower may not impact execution time

Tasks on **critical path** must execute at the fastest frequency

---

**Program Execution Time**

- **Determined by critical path** (longest path)
  - Tasks not on critical path can (potentially) be slowed
    - Running slower may not impact execution time

Tasks not on the critical path can stretch as long as path not lengthened
Energy Saving Runtimes: Summary

1. Must respect critical path to avoid delay
2. Load imbalance creates opportunities for saving energy.
3. The critical path can be approximated at runtime.
4. With no source code annotations, schemes like Adagio perform well

Power-Constrained Performance Optimization

- **Relevance:**
  - Hard power limit of 20MW for Exascale imposes strict limits on machine power consumption
  - Predicted power-usage much higher
  - Inherited power constraints: cluster-level vs. job-level

- **Problem definition:**
  Given a job-level power constraint and a number of nodes, how do we optimize application performance?


Naïve Scheme: Uniform, Static Power Allocation

- Equally distribute and enforce power constraint over all nodes of a job
  - Can use Running Average Power Limit (RAPL)
  - Execute as usual

- Statically select a configuration under the power constraint
  - Configuration: (Number of cores, Frequency)
  - Commonly used: Packed configuration
  - Maximum cores possible under the power constraint

Issues with Uniform, Static Power Allocation

1. Trivial node-level configurations may be inefficient
   - Example: LULESH – Power vs. execution time (single node on Cab)

   - Trivial configuration up to 30% slower than the optimal configuration
   - Needs prohibitively large number of runs of the application
Issues with Uniform, Static Power Allocation

2. Inefficient power usage on individual nodes in load-imbalanced applications
   Example: ParaDiS – Average power usage on nodes

   - Significant difference in power usage across nodes
   - Wastes unused power on nodes

Difference in power usage up to 36%!

Going Beyond an Illusion of a Homogeneous Machine

- Need to identify
  - Operations potentially on the critical path
  - Mitigate effects of application load imbalance
  - Power/performance relationship for every operation on every processor

- Difficult to select performance-optimizing configuration
  - Cannot violate the power constraint
  - Constrain overhead of reallocating power
  - Tricky to allocate power; do not want to slow down the critical path
  - Low-overhead, on-line profiling of the configuration space

- Example of a solution:
  Conductor – A run-time system for efficient power allocation

Conductor Algorithm

Start

Explore configurations

Construct Pareto frontier

Distribute new power allocation

Calculate new power headroom

Select Configuration $F_{opt}$

End monitor

Is computation non-critical?

Yes

Slow down computation

No

Start monitor

Evaluation: Configuration Selection

- Up to 30% speedup over Static scheme
- Benefits configuration-sensitive applications

Evaluation: Power Reallocation

- Up to 13% speedup over Static scheme
- Benefits from process-level imbalance of power usage


Summary of Evaluation

- LULESH, BT-MZ: Configuration-sensitive applications benefit from selecting the optimal configuration at the start (up to 30%)
- ParaDis, SP-MZ: Applications with load imbalance (and therefore, power usage imbalance) benefit from power reallocation (up to 13%)
- Overheads
  - Overall: 0.05% of total execution time
  - Configuration exploration: 1.96 seconds (two iterations)
  - Power reallocation: 566 microseconds per few time steps
- Best benefit from Optimization Strategy is application dependent!

What is the best Optimization Strategy for a Runtime

- Possibly conflicting Strategies
- Center / Infrastructure / System / User
- How to implement and interface different optimization strategies?
- Is there a silver bullet? – No, if you want to be flexible.
- How flexible do we have to be?
- Different Center different goals

Global Extensible Open Power Manager

- Job level power manager: GEOPM
  - Free open source power management runtime and framework
  - Contributed to accelerate community research on power management strategies to overcome Exascale challenges
  - Plug-in architecture for extensibility in two dimensions:
    - control algorithms
    - hardware platform portability
  - Example plug-ins included which significantly improve performance and efficiency via application-awareness
GEOPM Interfaces and HPC Stack Integration

- Job power manager
  - Works with scheduler
  - Flexible objective function via plug-ins
  - Globally optimizes HW control knobs across all compute nodes of job
- Feedback-guided control system
  - Can be fully dynamic or assisted by historical data
  - Feedback from app/libs via GEOPM APIs
  - Automatic in future
  - Targets power caps, DVFS

Hierarchical Design and Communications

- Scalable tree-hierarchical design
  - Tree hierarchy of controller agents
  - All agents run in the job compute nodes
  - Each agent runs ctrl algorithm plug-in
  - Recursive control / feedback algorithms
- Flexible tree configuration
  - Tree depth, fan-out, balance, placement optimized via MPI Cartesian grid
  - Tree auto-configured for deployments ranging from Rackscale to Exascale

Runtime and Power Allocation Traces

- GEOPM power balancer plug-in speeds up the critical path in Nekbone CORAL workload, by identifying bottlenecks and re-allocating power.
- Nekbone does two CGs with different characteristics leading to re-learning of best power allocation (~iter #50).
- Compare for GEOPM publication at ISC’17 with detailed results on 7 workloads.

Extensible Runtime through Plugins

- GEOPM Framework Extensible using Plugins
  - Different Optimization Functions
  - Different Platform Plugins
- Plugins Configure-able
  - During job start up
  - & exchangeable during execution.
- Can be implemented Resource-/Center- Specific
- Runtime designed and tested from small scale to full system size jobs
Global Extensible Open Power Manager

- Open source runtime for power management and framework for HPC community collaboration. (BSD-3 license)
- Scalable, extensible through plugins!
- Contribute, use & adapt for your HPC center / users / research groups
- Everything you need to get started: http://geopm.github.io/geopm

Matthias Maiterth
(matthias.maiterth@intel.com)

Open source runtime for power management and framework for HPC community collaboration. (BSD-3 license)

Scalable, extensible through plugins!

Contribute, use & adapt for your HPC center / users / research groups

Everything you need to get started: http://geopm.github.io/geopm

Power vs. Energy

- Which is actually important?
  - Both!

- Energy is just the product of power and time
  - The site’s bill depends on it
  - May be limited based on a fixed budget
  - However, may also be abundant at year’s end

- Power is a limited resource
  - Practically, there will be only so much power available
  - Limits in the incoming Power Grid, Center Infrastructure, System Architecture
  - Often also closely connected to thermal limits
  - However, may costly infrastructure if we don’t use its capabilities

Why is This an Application Developer’s Problem?

- Should be managed in hardware and be transparent
  - That’s how it has been so far

- Significant improvements expected from the hardware
  - More efficient circuits
  - Power efficient accelerators
  - Implicit power management (e.g., Turbo Mode)

- Problem 1: Won’t be enough to reach 20 MW
  - Far away from 25x goal of improvement
  - Application need to contribute across the whole software stack

- Problem 2: Hardware “tricks” will negatively impact applications
  - Performance heterogeneity (e.g., Turbo Mode)
Global energy optimization needed

- Need to understand each pillar
- Optimize and measure (KPIs) for each
- Need global approach for optimal results
  - includes utility provider
  - define operating points
  - keep infrastructure efficiency constant over the whole operating range
  - measure and assess

Future directions

- Clear interfaces between
  - Pillars
  - Components
  - Optimization functions
- Predictive Systems
  - Schedulers
  - Control Systems
- Improved possibilities
  - Pricing and Operation (spot market etc.)
  - Application Performance

The Times When We Could Ignore Power Are Over

- Need improvements across the system stack
  - Hardware (all components, not just CPUs)
  - System software (OS, runtime)
  - Resource manager
  - Application developers

- Techniques will impact applications
  - Increase noise/jitter/variability
  - Important for any kind of performance analysis

- However, we don’t need to expose every knob to developers
  - General awareness needed / form right expectations
  - Understand the reason and the consequences behind power policies
  - Provide simple opportunities to help steer power allocations