

Workload Scheduling and Power Management



Alejandro Sanchez, Morris Jette
[alex,jette]@schedmd.com
SchedMD, LLC

Slurm User Group Meeting 2018

Outline



- Power monitoring
- Reporting stats
- Power saving support
- User cpu frequency requests
- Power management and capping

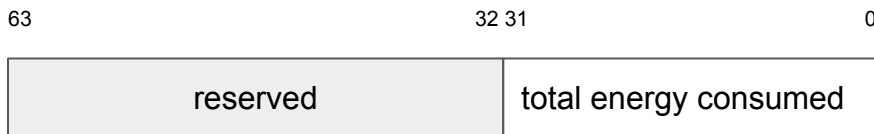
Power Monitoring



- Slurm provides energy accounting plugins for different infrastructure options
 - **Cray** - Uses existing Cray infrastructure that provides per-node power and energy data from the head node
 - `/sys/cray/pm_counters/power` → Point-in-time power, in watts.
 - `/sys/cray/pm_counters/energy` → Accumulated energy, in joules.
 - **IBM AEM** - Uses IBM's Systems Director Active Energy Manager (AEM). Power and energy measurements available on each node
 - `/sys/devices/platform/aem.0/{energy1_input,power1_average}`
 - Newer IBM POWER systems use OCC (On Chip Controller) to collect system data.

Power Monitoring

- **IPMI** - Gets data from BMC (Baseboard Management Controller) using the IPMI (Intelligent Platform Management Interface) API.
- **RAPL** - Uses Running Average Power Limit (RAPL) sensors on two hardware domains:
 - Package RAPL domain (sockets)
 - DRAM RAPL domain (memory)
 - May require MSR driver
 - Example MSR_PKG_ENERGY_STATUS MSR (Intel IA manual):



Comparison of IPMI and RAPL



- Energy measurement accuracy
 - IPMI - Good
 - RAPL - Good, but only for CPUs and memory
- Power measurement accuracy
 - IPMI - Poor
 - RAPL - Excellent, but only for CPUs and memory
- Overhead
 - IPMI - Relatively low
 - RAPL - Better than IPMI, no extra Slurm pthread required

Power Monitoring



- **Lenovo XCC** - Gets data from the XClarity Controller.
 - Embedded in every ThinkSystem server on a separate microprocessor.
 - IPMI variant where RAW hexadecimal values are used as commands to access Lenovo's specific OEM functionalities.
 - Ongoing work performed by Felip Moll. Working proof of concept already in place.

Plugin Configuration



- These plugins can be enabled in slurm.conf
- Sample configuration

```
AcctGatherEnergyType=acct_gather_energy/ipmi  
AcctGatherNodeFreq=30
```

Power Consumption Reporting

- Available in node state information through scontrol or sview

```
$ scontrol show node
  nodeName=nid00001 ...
  CurrentWatts=180 CapWatts=185
  LowestJoules=56 ConsumedJoules=123456
```

- Live job consumption with sstat
- Included in accounting reports with sreport
- Available for accounting and fair-share resource allocations

Power Consumption Reporting



- Available for job profiling
 - Data collected on admin/user-configurable interval
 - JobAcctGatherFrequency=energy=60
 - srun --acctg-freq=energy=30 (overrides previous one)
 - Written to HDF5 format file (or InfluxDB timeseries database since 18.08)
 - Different options configurable in acct_gather.conf
 - Tools available to plot and analyze per-node power consumption through time
 - HDFView, Grafana, ...

Power Saving Mechanism



- Supports powering down nodes that have been idle for some configurable period of time
- Nodes powered up as required
- Configurable rate at which nodes can be powered up or down

https://slurm.schedmd.com/power_save.html

Power Saving Mechanism



- slurm.conf excerpt

```
SuspendTime=120  
SuspendRate=60  
ResumeRate=300  
SuspendProgram=/path/to/suspend_prog  
ResumeProgram=/path/to/resume_prog  
SuspendTimeout=30  
ResumeTimeout=60  
SuspendExcNodes=nid[0001-0050]  
SuspendExcParts=debug
```

Dynamic Power Management



- Provides a mechanism to cap a cluster's power consumption
- Dynamically re-allocates power available per node based upon actual real-time usage
 - Starts by evenly distributing power cap across all nodes, periodically lowers the cap on nodes using less power and redistributes that power to other nodes
- Nodes using most of their power cap have the cap increased
- Nodes with newly initiated jobs have power cap reset

Dynamic Power Management



- Configurable iteration time and change rates
- Optimizes throughput within power cap with little to no user input and responds quickly to changes in application power consumption
- Currently only available on Cray systems
 - capmc utility used underneath alongside the json-c library
- Used at KAUST on Shaheen II

Power Management Configuration



- `slurm.conf` options:
 - `DebugFlags=power` - Enable plugin-specific logging
 - `PowerParameters` - Defines power cap, various thresholds, rate of changes, etc. (more on next slides)
 - `PowerPlugin` - Define the plugin to use (e.g. “power/cray”)

Power Parameter Options (1 of 3)



- `balance_interval=#` - Time interval between attempts to balance power caps. Default is 30 seconds.
- `capmc_path=/...` - Fully qualified pathname of the capmc command. Default is “/opt/cray/capmc/default/bin/capmc”.
- `cap_watts=#[KW|MW]` - Power cap across all compute nodes

Power Parameter Options (2 of 3)



- `decrease_rate=#` - Maximum rate of change in power cap of a node under-utilizing its available power. Based upon difference between a node's minimum and maximum power consumption. Default value is 50%.
- `increase_rate=#` - Maximum rate of change in power cap of a node fully utilizing its available power. Default value is 20%.
- `lower_threshold=#` - Nodes using less than this percentage of their power cap are subject to the cap being reduced. Default value is 90%.
- `upper_threshold=#` - Nodes using more than this percentage of their power cap are subject to the cap being increased. Default value is 95%

Example slurm.conf

```
#  
# Select portions of a slurm.conf file  
#  
DebugFlags=power    # Use recommended only for testing PowerPlugin=power/cray  
PowerParameters=balance_interval=60,cap_watts=1800,decrease_rate=30,increase_rate=10,  
lower_threshold=90,upper_threshold=98
```

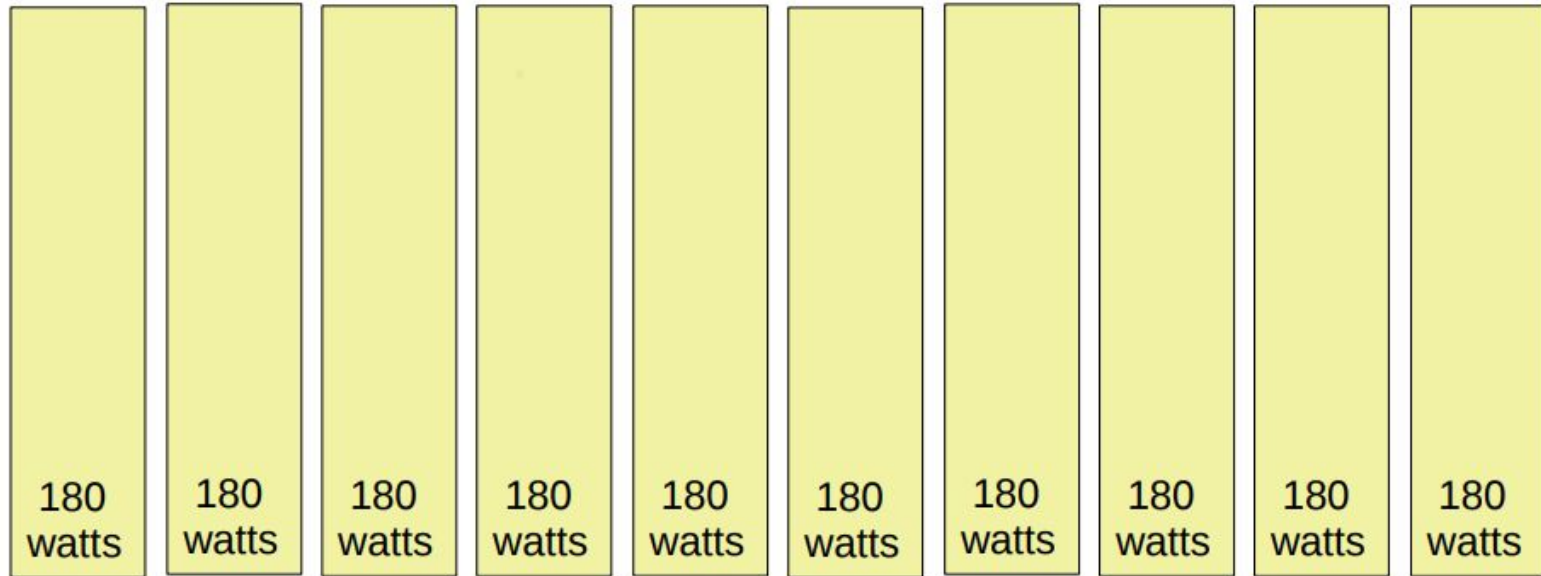
NOTE: decrease_rate and increase_rate are based upon the difference between a node's minimum and maximum power consumption. If minimum power consumption is 100 watts and maximum power consumption is 300 watts then the maximum rate at which a node's power cap would be decreased is 60 watts $((300 \text{ watts} - 100 \text{ watts}) \times 30\%)$ while the maximum rate of increase would be increase 20 watts $((300 \text{ watts} - 100 \text{ watts}) \times 10\%)$.

Example Time 0 - initial state



- PowerParameters=balance_interval=60,cap_watts=1800,decrease_rate=30,increase_rate=10,lower_threshold=90,upper_threshold=98
- 10 compute nodes each with maximum power consumption of 200 watts and minimum of 100 watts
- Configured power cap of 1800 watts available
- Set each node's power cap to 180 watts ($1800 / 10$)

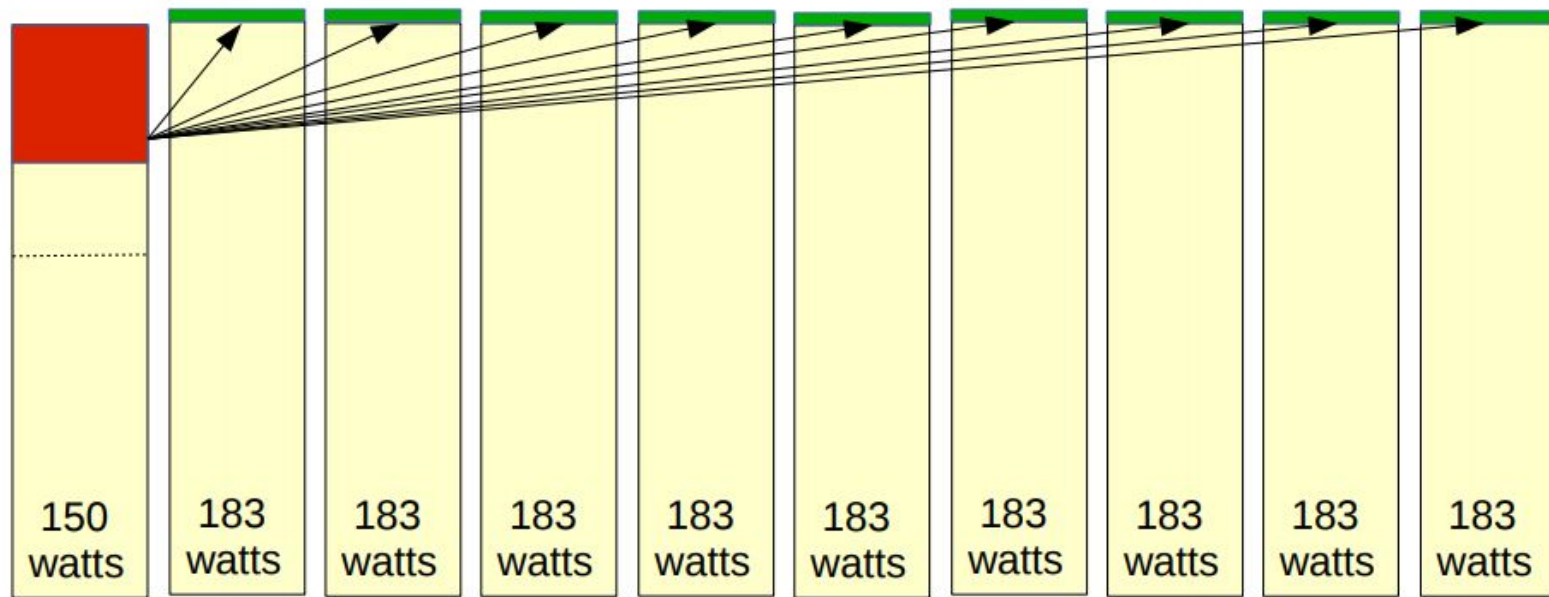
Example Time 0 - initial state



Example Time 60s - initial state

- One node is using 110 watts, others at 180 watts
- That 110 watt node is below `lower_threshold` ($180 \text{ watts} \times 90\% = 162 \text{ watts}$), so its cap gets reduced by the lesser of half the difference ($((180 \text{ watts} - 110 \text{ watts}) / 2 = 35 \text{ watts})$ or `decrease_rate` ($200 \text{ watts} - 100 \text{ watts} \times 30\% = 30 \text{ watts}$), so that node's cap is reduced from 180 watts to 150 watts.
- We now have 1650 watts available to distribute over the remaining 9 nodes, or 183 watts per node ($1650 \text{ watts} / 9 \text{ nodes}$)

Example Time 60s



Dynamic Power Management



- Guidelines, slides and more examples available here:

https://slurm.schedmd.com/power_mgmt.html

https://slurm.schedmd.com/SLUG15/Power_mgmt.pdf

User Power Management Controls



- User's can specify desired frequency range and/or CPU governor

```
$ srun --cpu-freq=low-medium:conservative ....  
$ srun --cpu-freq=performance ...  
$ srun --cpu-freq=2400 ...
```

- /sys/devices/system/cpu/cpuX/cpufreq/
 - scaling_setspeed
 - scaling_governor
 - ...

Areas of Interest



- User management of GPU frequency similar to CPU controls
- User specified power budget on jobs
- Controlled rate of change in power consumption (ramp up/down)
- Scheduled power availability changes (e.g. more power available at night)
- Schedule workload to optimize performance within power budgets through time

Areas of Concern



- Infrastructure needs to support power floor for ramp down
 - Not typically available today
- Will users specify power requirements?
- Will users specify reasonable time limits?
 - Needed for power budget scheduling
- Heuristics likely needed achieve good scheduling performance given (likely) poor guidance from users

Extra HPC PowerStack Notes



- Efficiently managing procured power on HPC is challenging due to different reasons, including
 - Processor manufacturing variability and increasing heterogeneity of node-level components
 - Vendor-specific mechanisms to measure metrics
- Some projects, most notably the Power API efforts, discuss interfaces that form a good starting point for full stack integration
- HPC community needs a *holistic* stack for power and energy management

