Slurm Power Management Support

Morris Jette
SchedMD LLC

Slurm User Group Meeting 2015

Power Management Overview

- Power availability for HPC has become critical issue
- Ideally we want to manage:
  - Maximum power consumption
  - Minimum power consumption and
  - Rate of change in power consumption
- Controlling CPU frequency only manages maximum power consumption
- Other mechanisms are available to manage minimum and maximum power consumption by motherboard

Cray Power Management

- Currently supported only on Cray systems
- Provides mechanism to cap a cluster's power consumption
- Dynamically re-allocates power available per node based upon <u>actual real-time usage</u>
  - 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
- No forecasting of a pending job's power requirements, which typically would vary through time
- Configuration options to control various thresholds and change rate
- NOTE: Only the compute node power consumption is managed by Slurm

Slurm Plugins

- Implemented using Slurm plugins to support various infrastructures
  - Cray – Uses Cray-specific APIs and commands
  - Common – Common power management infrastructure available the various plugins
  - Additional plugins likely in the future

Slurm Configuration: slurm.conf

- New *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”)

PowerParameter 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 <u>compute</u> nodes

PowerParameter 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%.

PowerParameter Options (3 of 3)

- job_level - All compute nodes associated with every job will be assigned the same power cap. Nodes shared by multiple jobs will have a power cap different from other nodes allocated to the individual jobs. By default, this is configurable by the user for each job.
- job_no_level - Power caps are established independently for each compute node. This disabled the "--power=level" option available in the job submission commands. By default, this is configurable by the user for each job.
- recent_job=# - If a job has started or resumed execution (from suspend) on a compute node within this number of seconds from the current time, the node's power cap will be increased to the maximum. The default value is 300 seconds.

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 watts – 100 watts) x 30%) while the maximum rate of increase would be increase 20 watts ((300 watts – 100 watts) x 10%).

User Tools

- salloc, sbatch, and srun
  - --power=level – All nodes allocated to job have same power cap. May be disabled by global configuration parameter, PowerParameters
  - --cpu-freq=[minimum[-maximum]:]governor]
    - Frequency can be low, medium, highm1 (second highest available frequency), high, or KHz value
    - Governor can be conservative, ondemand, performance, or powersave
    - These are user requests, subject to system constraints

```
$ sbatch –cpu-freq=2400000-3000000 …
$ salloc –cpu-freq=powersave ...
$ srun –cpu-freq=highm1 ...
```

User Tools

- sview and “scontrol show node”
  - Displays current power consumption and power cap information for each compute node

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

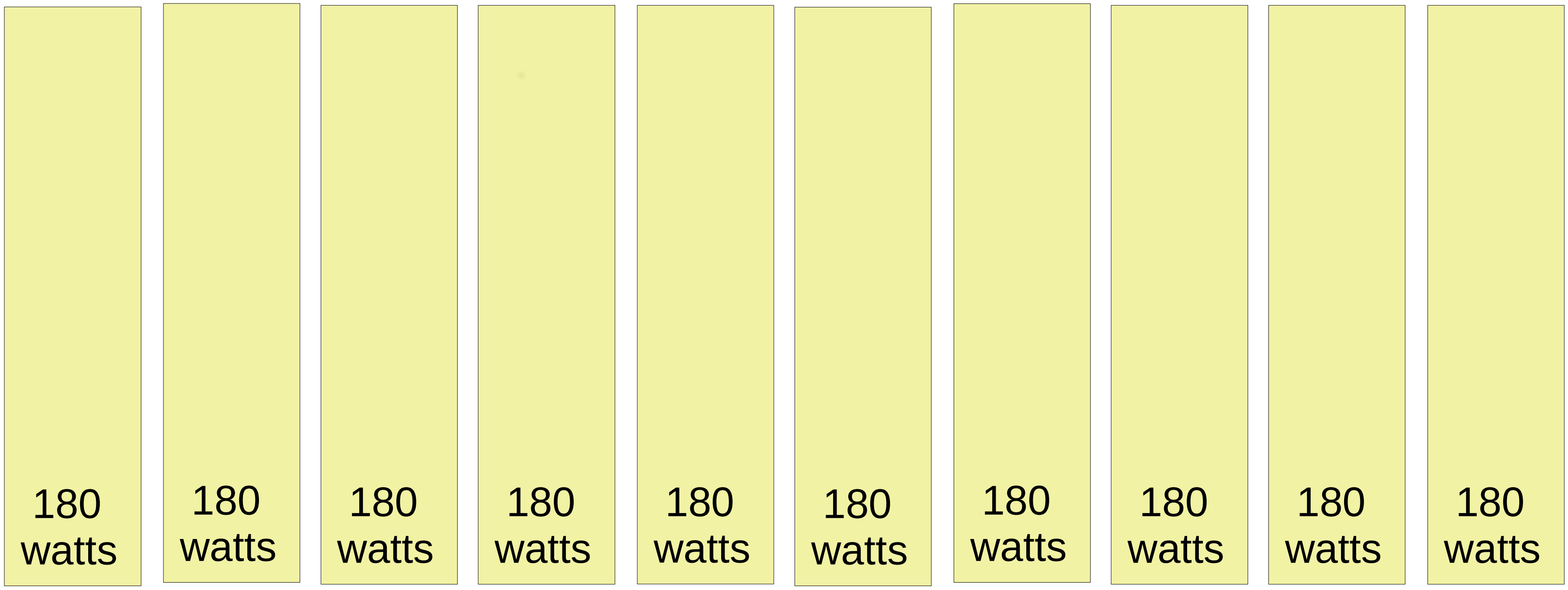
Example
Time 0, Initial state

- 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)

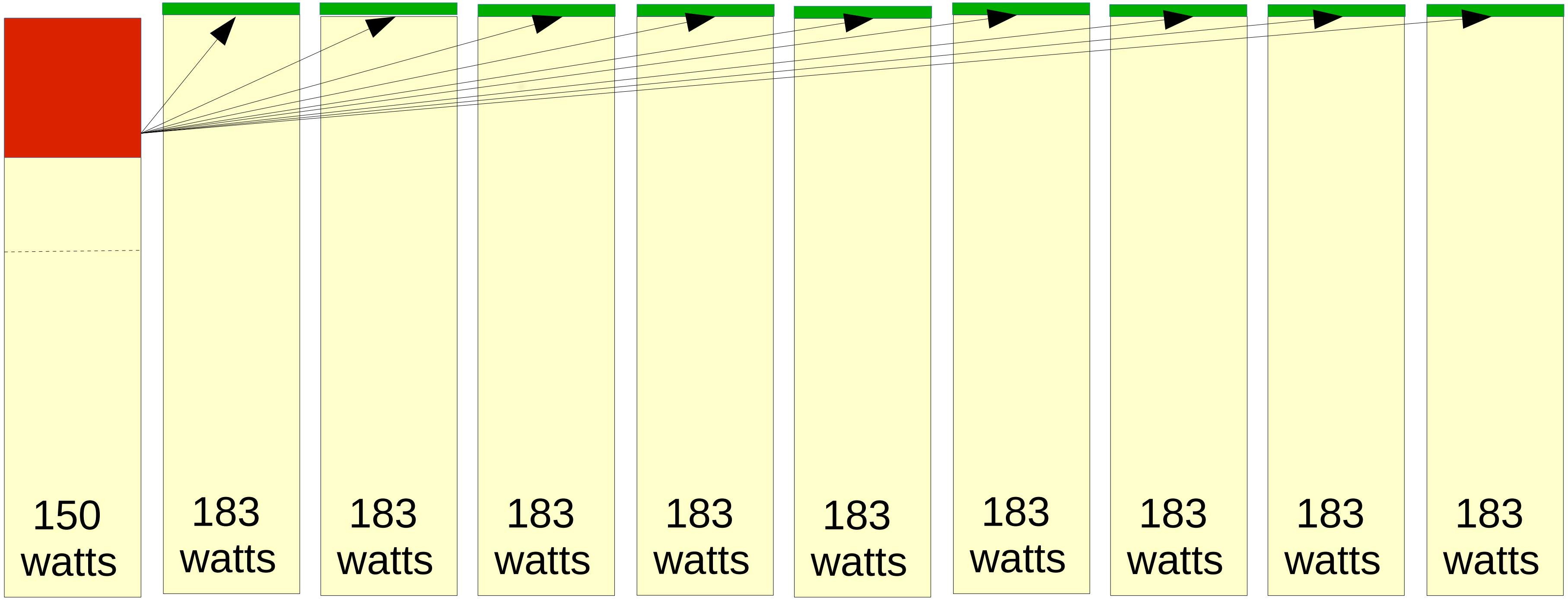
Example
Time 0, Initial state



Example
Time 60 seconds

- One node is using 110 watts, others at 180 watts
- That 110 watt node is below lower_threshold (180 watts x 90% = 162 watts), so its cap gets reduced by the lesser of half the difference ((180 watts – 110 watts) / 2 = 35 watts) or decrease_rate (200 watts -100 watts x 30% = 30 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 watts / 9 nodes)

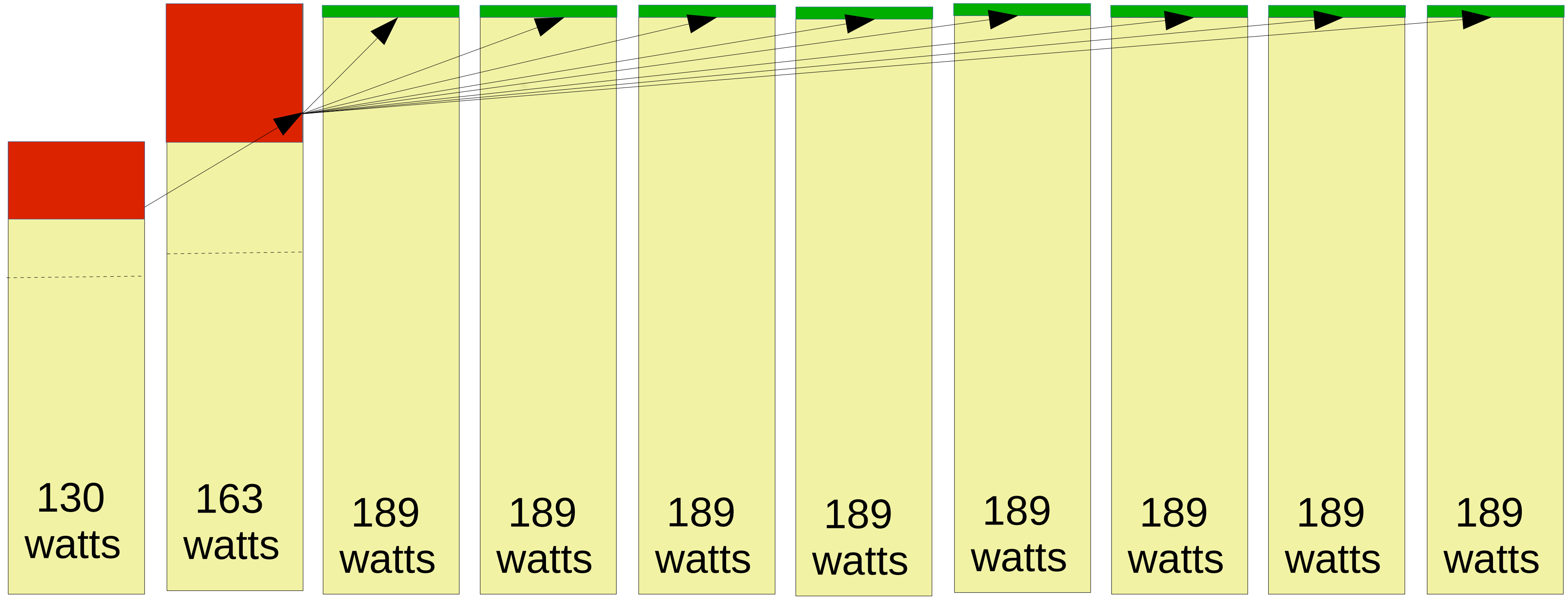
Example
Time 60 seconds



Example
Time 120 seconds

- One node using 110 watts, others at 115 watts, others at 183 watts
- Node at 110 watts is reduced by half difference from the cap ((150 watts – 110 watts) / 2 = 130 watts)
- Node at 115 watts is reduced by 30 watts based upon decrease_rate (which is less than half the difference)
- Remaining 1517 watts evenly distributed to remaining 8 compute nodes or 189 watts per node

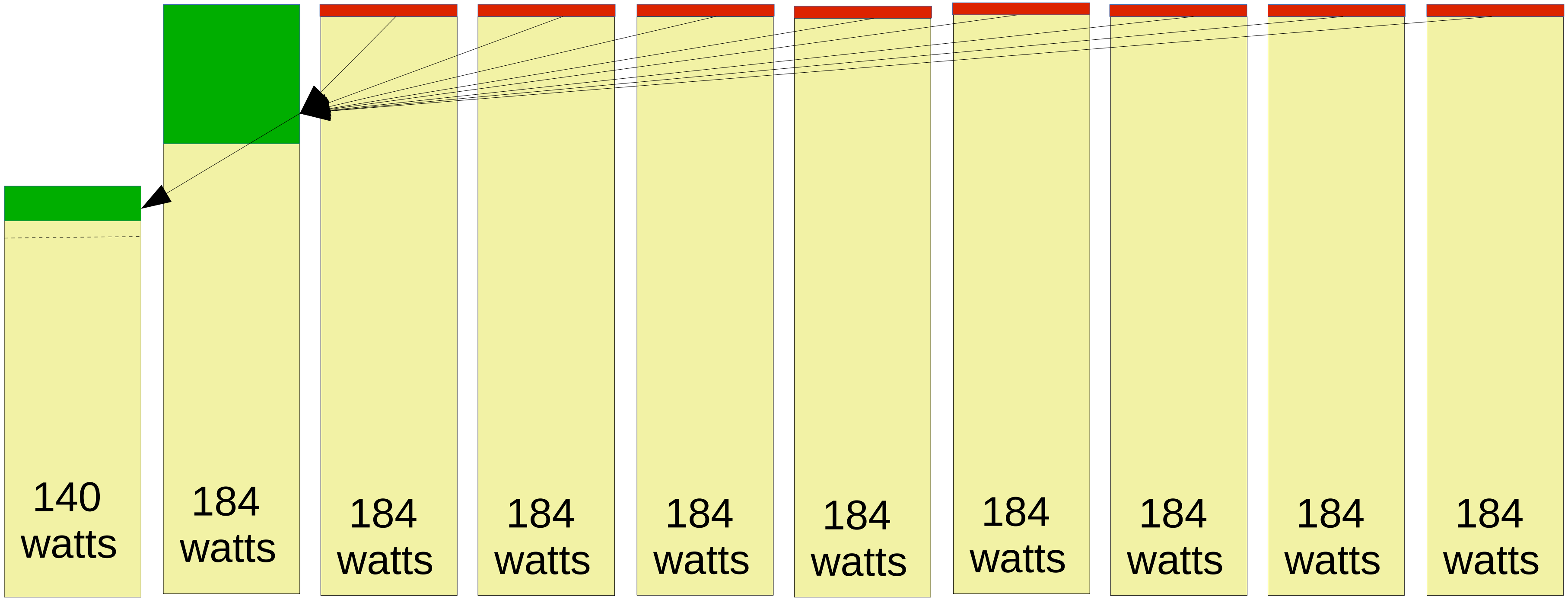
Example
Time 120 seconds



Example
Time 180 seconds

- Node previously consuming 110 watts is now consuming 128 watts, which is over upper_threshold (130 watts x 98% = 127 watts), so it's cap gets increased by increase_rate (10 watts) to 140 watts
- Node previously consuming 115 is allocated a new job, so its power cap is increased to the same as other nodes consuming all available power
- Remaining 1660 watts evenly distributed across 9 nodes or 184 watts per node

Example
Time 120 seconds



Future work

- Add support for minimum power consumption and managing rate of change
  - Dependent upon Cray infrastructure managing minimum power consumption by node
- Considering expected power consumption of pending jobs in making scheduling decisions
- Likely some merging of Bull's and SchedMD's work
  - These two approaches are complementary

Questions?