Slurm Version 15.08

Jacob Jenson
jacob@schedmd.com

SchedMD LLC
http://www.schedmd.com

Slurm

Rank	Workload Manager	System
1	***Slurm***	Tianhe-2
2	-	Titan
3	**Slurm**	Sequoia
4	-	K Computer
5	-	Mira
6	***Slurm***	Piz Daint
7	***Slurm***	Stampede
8	-	JUQUEEN
9	***Slurm***	Vulcan
10	***Slurm***	Cray (Gov)

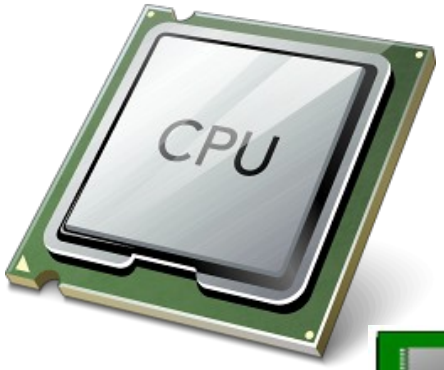
- QOS
- Power Management
- Fault Tolerant
- Scalable
- Resizable Jobs
- Topology Optimized
- Accounting
- Phi Support
- Suspend/Resume
- Checkpoint/Restart
- GPU Support
- Cgroups

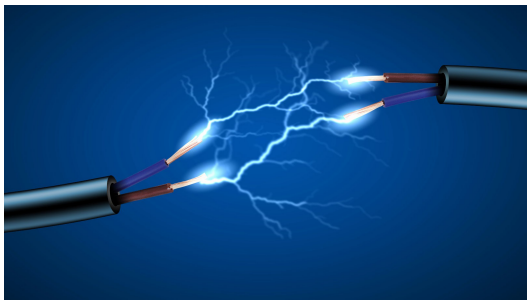
V15.08 – Road Map

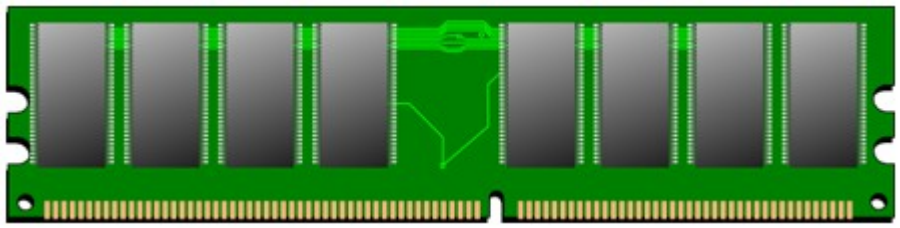
- Expanded charging options
- Improved recovery from communication failures
- Support for PMI Exascale (PMIx)
- Data Warp
- Power Management
- Bonus Round

V15.08 – Charging options

- Historically charging has been based on CPU time.
- Changing to a general system billing unit
  - Computed as a function of many different resources
  - CPUs, memory, power and GPUs



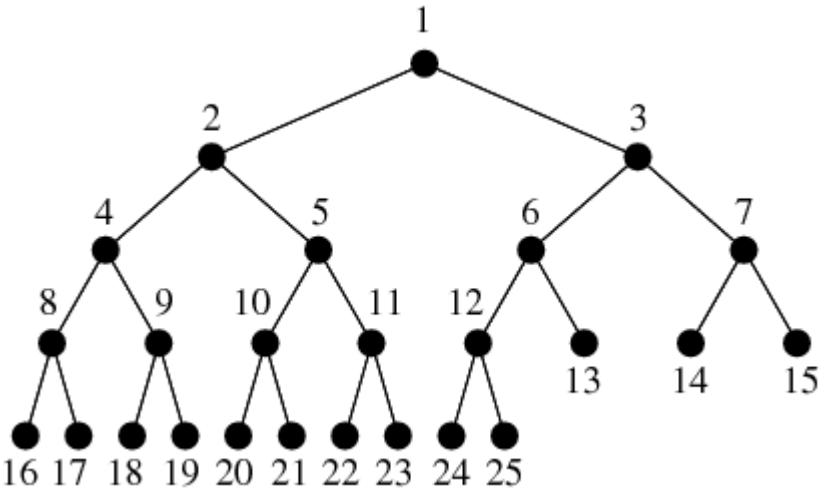


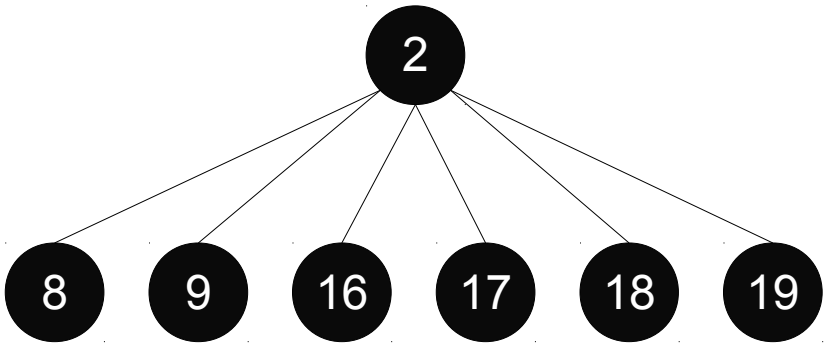


V15.08 – Improved communication failure recovery

- Improve recovery time for communication failures when large numbers of nodes fail simultaneously.

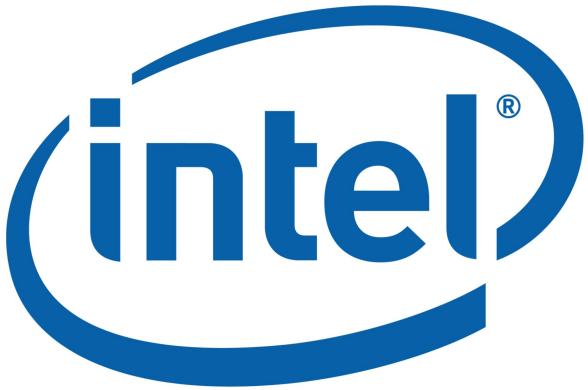
14.11 vs 15.08





V15.08 – PMI Exascale (PMIx)

- Support for PMIx to improve MPI scalability.
- The vision for Slurm is to extend workload management functionality to address Exascale requirements.



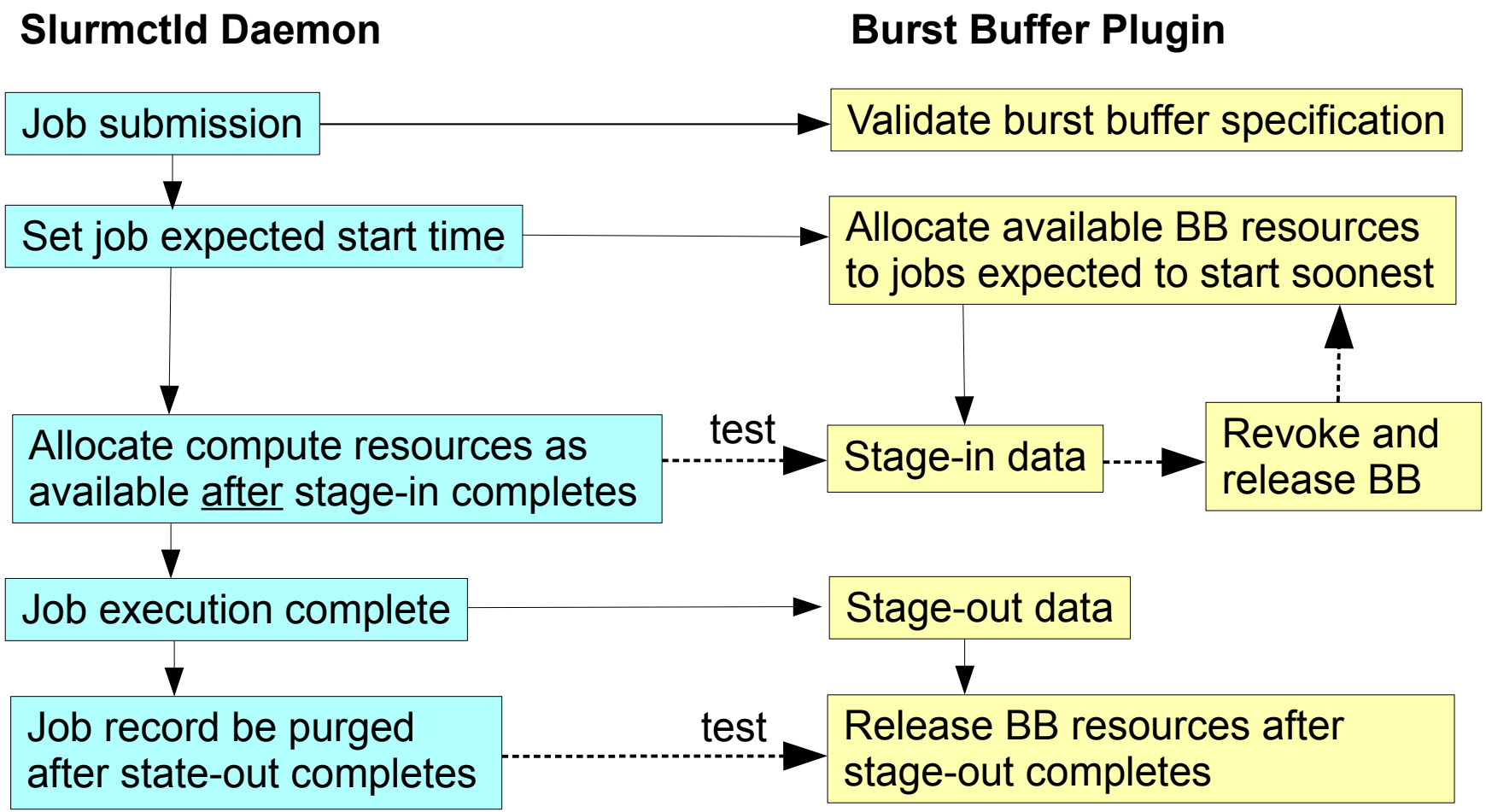
Data Warp (Burst Buffers)

- A cluster-wide high-performance storage resource
- Burst buffer (BB) allocations are managed by Slurm
- Two types of BB allocations:
  - Persistent allocations used by multiple jobs or
  - Associated with a specific job
- BB allocations can exist before, during and/or after a job allocation of compute resources
  - Used to stage-in data, scratch storage, and/or stage-out data

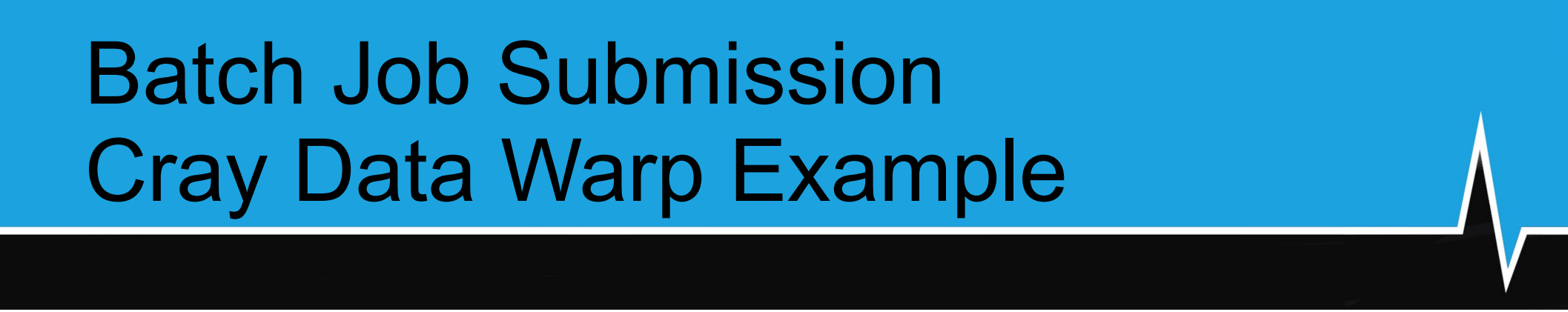
Workflow

- Job submission specifies burst buffer requirements, validated immediately at submit time
- Slurm allocates available burst buffer resources to the jobs expected to start soonest
  - May later be revoked for higher priority job
- Stage-in of files begins
- Compute nodes may be allocated after stage-in of files completes
- Stage-out of files begins upon completion of computation
- Job record may be purged after stage-out of files completes

Workflow



Batch Job Submission
Cray Data Warp Example



- Include burst buffer directives in the batch script
  - Buffer size
  - Files to stage
  - See Cray documentation for option details
- On Cray systems, job requesting burst buffers will be allocated whole nodes

```
#!/bin/bash
#DW jobdw capacity=100gb
#DW stage_in type=file source=/what/ever/input destination=input
#DW stage_out type=file source=output destination=/what/ever/output
a.out
```

Interactive Job Submission

- Specify the burst buffer directives using the -bb option
- No file staging is supported
- On Cray systems, job requesting burst buffers will be allocated whole nodes

```
$ salloc --bb=”capacity=100gb” -N2 a.out

$ salloc –bb=”swap=2gb” -N2 a.out
```

New Job State Information

- New wait reasons
  - BurstBufferResources – Waiting for allocation of burst buffer
  - BurstBufferStageIn – Waiting for stage-in to complete
- New job error codes
  - ESLURM_BURST_BUFFER_PERMISSION - Burst Buffer permssion denied
  - ESLURM_BURST_BUFFER_LIMIT - Burst Buffer resource limit exceeded
  - ESLURM_INVALID_BURST_BUFFER_REQUEST - Burst Buffer request invalid

Example burst_buffer_generic.conf

```
# Example burst_buffer_generic.conf
#
AllowUsers=alan,brenda,jette
PrivateData=true
#
JobSizeLimit=200GB    # Applies to each job
UserSizeLimit=500GB  # Applies to each user
#
PrioBoostUse=100
PrioBoostAlloc=200
#
StageInTimeout=300     # Seconds
StageOutTimeout=300  # Seconds
#
GetSysState=/usr/local/slurm/15.08/sbin/GSS
StartStageIn=/usr/local/slurm/15.08/sbin/SSI
StartStageOut=/usr/local/slurm/15.08/sbin/SSO
StopStageIn=/usr/local/slurm/15.08/sbin/PSI
StopStageOut=/usr/local/slurm/15.08/sbin/PSO
```

Example burst_buffer_cray.conf

```
# Example burst_buffer_cray.conf
#
AllowUsers=alan,brenda,jette
PrivateData=true
#
JobSizeLimit=200GB    # Applies to each job
UserSizeLimit=500GB  # Applies to each user
#
PrioBoostUse=100
PrioBoostAlloc=200
#
StageInTimeout=300     # Seconds
StageOutTimeout=300  # Seconds
#
# No command paths required, uses Cray APIs for BB management
```

Additional BB Features

- Advanced reservation of burst buffer resources
  - Make certain that burst buffers are available for critical uses
- Burst buffer space will be added as a factor in the calculation of a job's priority
- Burst buffer space will be added as a new limit that can be configured on a per-user, account and/or QOS basis

Power Management Overview

- Currently supported only on Cray systems
- Provides mechanism to cap a cluster's power consumption
- 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
- Configuration options to control various thresholds and change rate options
- NOTE: Only the compute node power consumption is managed by Slurm

Slurm Configuration

- New *slurm.conf* options:
  - DebugFlags=power
    - Enable plugin-specific logging
  - PowerParameters
    - Defines power cap, various thresholds, rate of changes, etc.
  - 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 compute 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

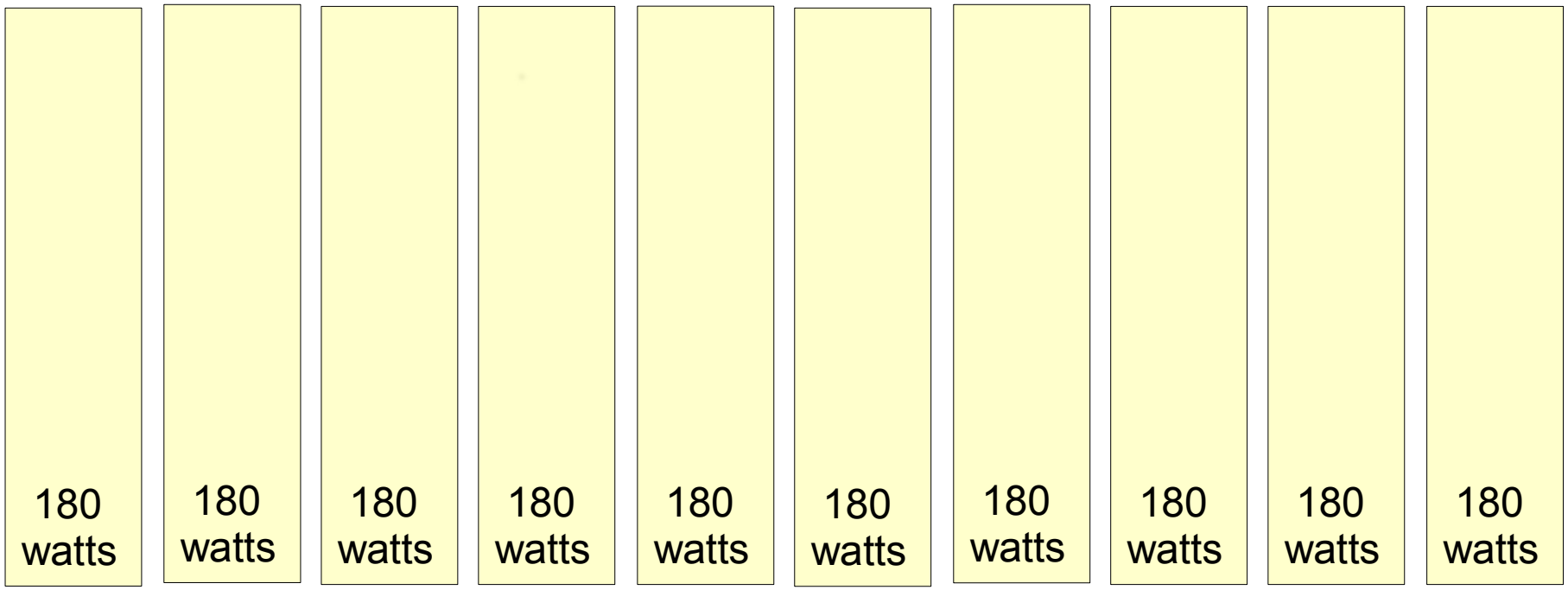
- sview and “scontrol show node”
  - Displays current power consumption and power cap information for each compute node

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

Example
Time 0, Initial state

- 10 compute nodes
  - Maximum power 200 watts
  - Minimum power 100 watts
- PowerParameters
  - balance_interval=60
  - cap_watts=1800
  - decrease_rate=30
  - increase_rate=10
  - lower_threshold=90
  - upper_threshold=98
- Configured power cap of 1800 watts available
- Set each node's power cap to 180 watts (1800 / 10)

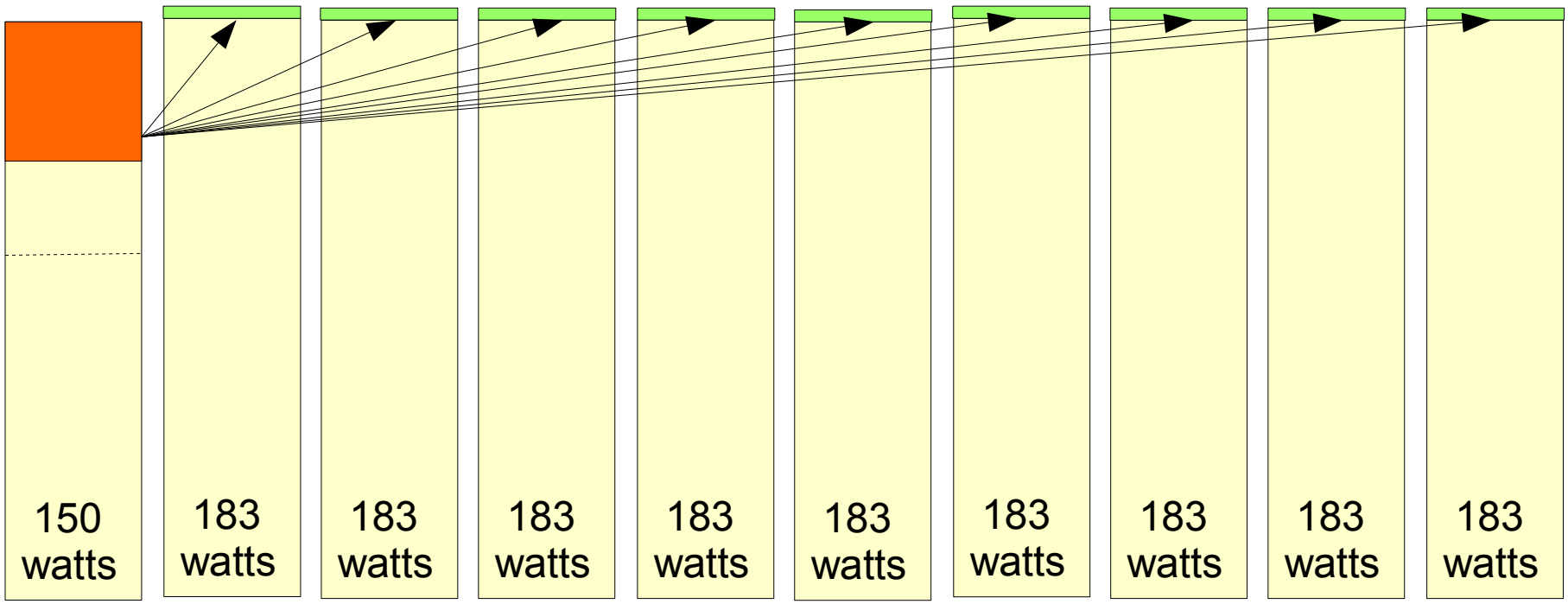
Example
Time 0, Initial state



Example
Time 60 seconds

- One node is using 110 watts, others at 180 watts
- The 110 watt node is below lower_threshold
  - 180 watts x 90% = 162 watt lower threshold
  - Reduced cap by the lesser of half the difference
    - (180 watts – 110 watts) / 2 = 35 watts
    - (200 watts -100 watts) x 30% = 30 watts
    - The node's cap is reduced from 180 watts to 150 watts.
- Ignoring upper_threshold, we now have 1650 watts available to distribute over the remaining 9 nodes, or 183 watts per node (1650 watts / 9 nodes)

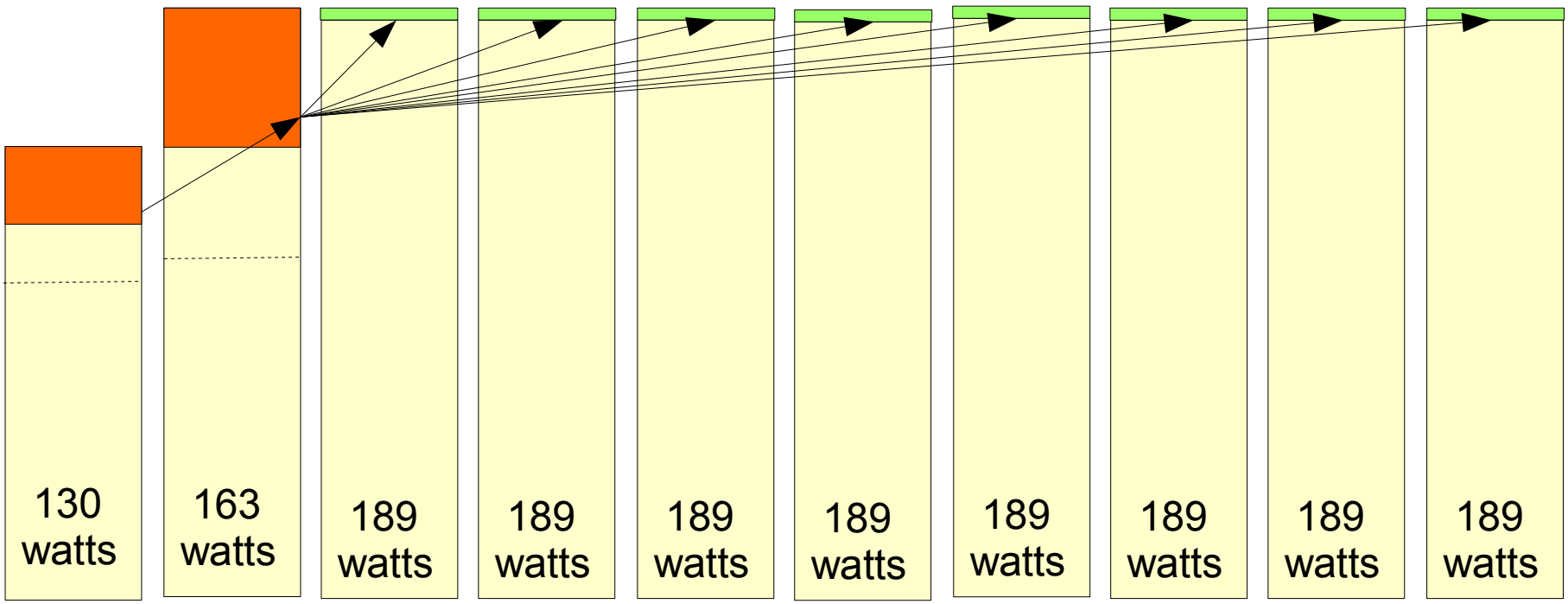
Example
Time 60 seconds



Example
Time 120 seconds

- One node using 110 watts
- One node using 115 watts
- Eight nodes using 183 watts
- Node at 110 watts reduced by half the cap difference (150 watts – 110 watts) / 2 = 20 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

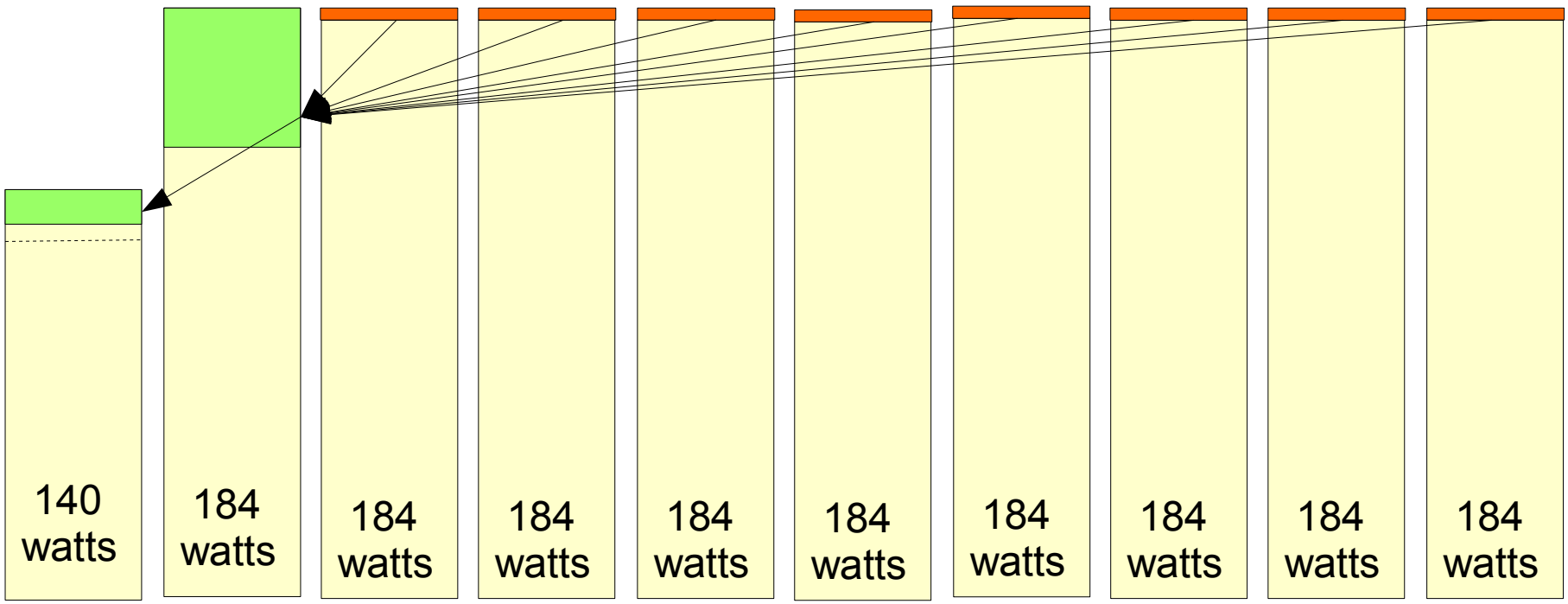
Example
Time 120 seconds



Example
Time 180 seconds

- Node previously consuming 110 watts is now consuming 128 watts
  - Over upper_threshold (130 watts x 98% = 127 watts)
  - Cap gets increased by increase_rate (10 watts) to 140 watts
- Node previously consuming 115 is allocated a new job,
  - 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



Bonus Round

- Native Slurm
  - Multiple jobs per compute node
    - Exclusive (15.08) and non-exclusive
  - Multiple simultaneous applications per job allocation
  - Active job allocations can grow and shrink on demand