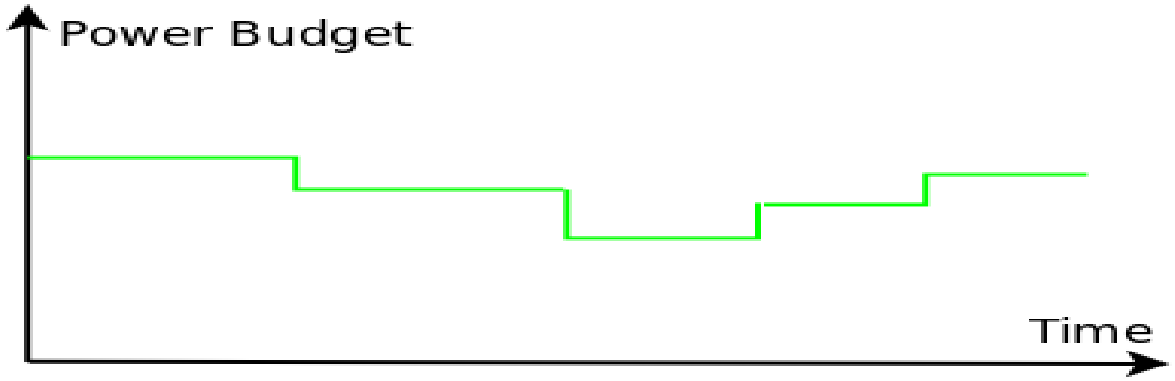
Power Adaptive Scheduling

Yiannis Georgiou (BULL)
David Glesser (BULL)
Matthieu Hautreux (CEA)
Denis Trystram (LIG)

15-09-2015

Motivations

- Need for **centralized mechanism** to dynamically **adapt the instantaneous power** consumption of the whole platform
  - Reducing the number of usable resources or running them with lower power
- Technique to plan in advance for **future power adaptations**
  - In order to align upon dynamic power provisioning and **electricity prices**



Introduction

- Power adaptive scheduling within SLURM is a new feature appearing in 15.08
  - Initial algorithms and prototype made by CEA in 2013
  - A second prototype (extended version of the first) has been studied, experimented and published in *[Georgiou et al. HPPAC-2015] by BULL + LIG*
- *Final implementation (BULL) based upon the layouts framework and its API functions (CEA)*

Yiannis Georgiou, David Glesser, Denis Trystram
Adaptive Resource and Job Management for limited power consumption
In proceedings of IPDPS-HPPAC 2015

Power adaptive scheduling in 15.08

The implementation appeared in 15.08 has the following characteristics:

- Based upon layouts framework
  - for internal represantation of resources power consumption
  - Only logical/static represantation of power
  - Fine granularity down to cores
- Reductions take place through following techniques coordinated by the scheduler:
  - Letting Idle nodes
  - Powering-off unused nodes (using default SLURM mecanism)
  - Running nodes in lower CPU Frequencies (respecting ––cpu-freq allowed frequencies)

Set/Modify/View Powercap Value

- Initially with parameter in slurm.conf

```
[root@nd25 slurm]#cat /etc/slurm.conf |grep Power
PowerParameters=cap_watts=INFINITE
```

- Dynamically with scontrol update

```
[root@nd25 slurm]#scontrol update powercap=1400000
```

- In advance with watts reservation (scontrol create res)

```
[root@nd25 slurm]#scontrol create res FLAG=ANY_NODES starttime=now+11minutes
duration=16 Watts=532224 Users=root
```

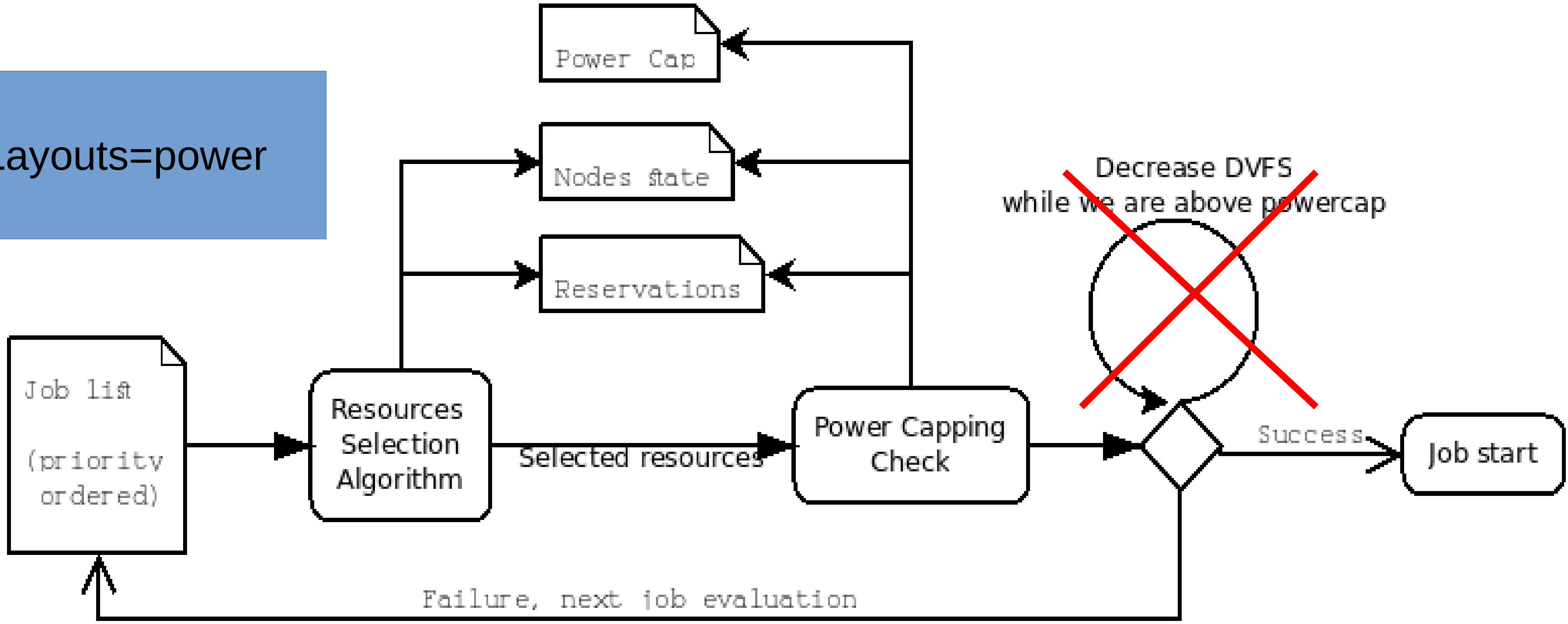
- View with scontrol show

```
[root@nd25 slurm]#scontrol show powercap
MinWatts=564480 CurrentWatts=809934 PowerCap=INFINITE PowerFloor=0
PowerChangeRate=0AdjustedMaxWatts=1774080 MaxWatts=1774080
```

Bull
atos technologies

Power adaptive scheduling – algorithm simple version -

Layouts=power


Power Cap
Nodes state
Reservations
Job list (priority ordered)
Resources Selection Algorithm
Selected resources
Power Capping Check
Decrease DVFS while we are above powercap
Success
Job start
Failure, next job evaluation

- ▶Reductions only by keeping nodes idle and shut-down (if powersave mode activated)
- ▶Considering power consumption per node level

Power adaptive scheduling
– algorithm simple version -

Logic within the Powercapping Check

- Calculate what power consumption the cluster would have if the job was executed
- If lower than the allowed power budget, proceed with job
- Else keep job pending and check next one

Architecture of the Powercapping Check

- Based upon the different nodes bitmaps states
- Using Layouts only for keeping the values for the nodes power consumption (only get, no set)

Power adaptive scheduling – algorithm simple version – Configuration

▶ Set parameter within slurm.conf

```
[root@nd25 slurm]#cat /etc/slurm.conf |grep power
Layouts=power
```

▶ Set new /etc/layouts.d/power.conf file

- Examples exist in slurm code base in etc/layouts.d.*example

```
[root@nd25 slurm]#cat /etc/layouts.d/power.conf

Entity=Cluster Type=Center IdleSumWatts=0 MaxSumWatts=0 Enclosed=virtual[0-5039]

Entity=virtual[0-5039] Type=Node  IdleWatts=103 MaxWatts=308
```

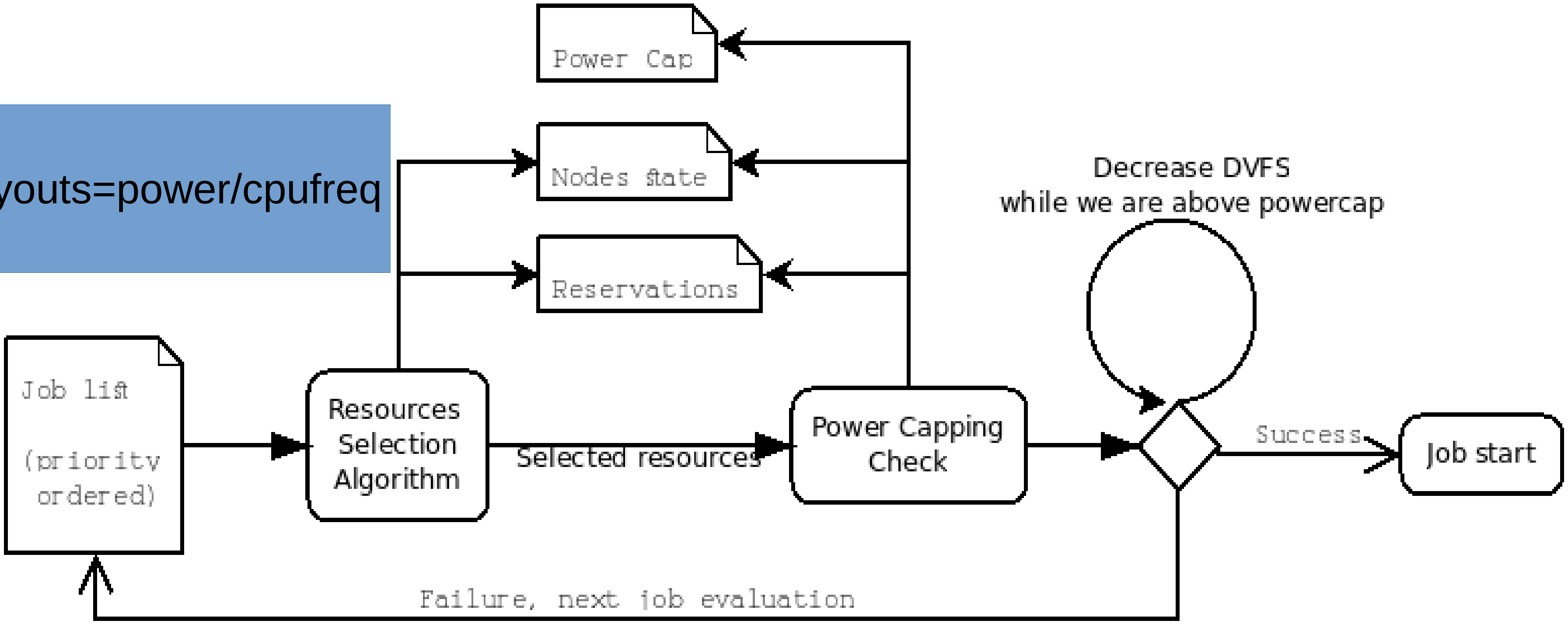
Layouts Power code structure (truncated)

- src/layouts/power/default.c -

```
[root@nd25 slurm]#vi  src/layouts/power/default.c

const layouts_keyspec_t keyspec[] = {
        /* base keys */
        {"IdleWatts", L_T_UINT32},
        {"MaxWatts", L_T_UINT32},
        /* parents aggregated keys */
        {"IdleSumWatts", L_T_UINT32,
        KEYSPEC_UPDATE_CHILDREN_SUM, "IdleWatts"},
        {"MaxSumWatts", L_T_UINT32,
        KEYSPEC_UPDATE_CHILDREN_SUM, "MaxWatts"},
        {NULL}
};
const char* etypes[] = {
        "Center",
        "Node",
        NULL
};
```

Power adaptive scheduling – algorithm extended version -

Layouts=power/cpufreq


Power Cap
Nodes state
Reservations
Decrease DVFS while we are above powercap
Job list (priority ordered)
Resources Selection Algorithm
Selected resources
Power Capping Check
Success
Job start
Failure, next job evaluation

▶ Reductions through DVFS, idle and shut-down nodes (if power-save mode activated)

▶ Considering core level power consumption

Power adaptive scheduling
– algorithm extended version -

Logic within the Powercapping Check

- Calculate what power consumption the cluster would have if the job was executed
- If higher than the allowed power budget, check if DVFS is allowed for the job (usage of –-cpu-freq parameter with MIN and MAX)
  - If yes then calculate what power consumption the cluster would have if the job was executed with its different allowed CPU-Frequencies
  - Try with the optimal CPU-Frequency which is the one that would allow all the idle resources to become allocated
- If neither the optimal nor the MIN allowed CPU-Frequency for the job results in lower power consumption than the powercap then job pending else running

Bull
atos technologies

Power adaptive scheduling
– algorithm extended version -

Architecture of the Powercapping Check

- Based upon the different nodes bitmaps states
- Using Layouts for collecting and setting nodes and cores power consumption (both get and set functions)
- Each CPU Frequency is represented/considered to have its own power consumption (based on measures or hardware provider specifications)

Power adaptive scheduling – algorithm extended version – Configuration

- Set parameter within slurm.conf

```
[root@nd25 slurm]#cat /etc/slurm.conf |grep power
Layouts=power/cpufreq
```

- Set new /etc/layouts.d/power.conf file

```
[root@nd25 slurm]#cat /etc/layouts.d/power.conf

Entity=Cluster Type=Center CurrentSumPower=0 IdleSumWatts=0 MaxSumWatts=0
Enclosed=virtual[0-5039]

Entity=virtualcore[0-80639] Type=Core CurrentCorePower=0 IdleCoreWatts=7
MaxCoreWatts=22 CurrentCoreFreq=0 Cpufreq1Watts=12 Cpufreq2Watts=13
Cpufreq3Watts=15 Cpufreq4Watts=16 Cpufreq5Watts=17 Cpufreq6Watts=18
Cpufreq7Watts=20

Entity=virtual0 Type=Node CurrentPower=0 IdleWatts=0 MaxWatts=0 DownWatts=14
PowerSaveWatts=14 CoresCount=0 LastCore=15 Enclosed=virtualcore[0-15]
Cpufreq1=1200000 Cpufreq2=1400000 Cpufreq3=1600000 Cpufreq4=1800000
Cpufreq5=2000000 Cpufreq6=2200000 Cpufreq7=2400000 NumFreqChoices=7

Entity=virtual1 Type=...
```

Layouts Power code structure (truncated)

- src/layouts/power/cpufreq.c -

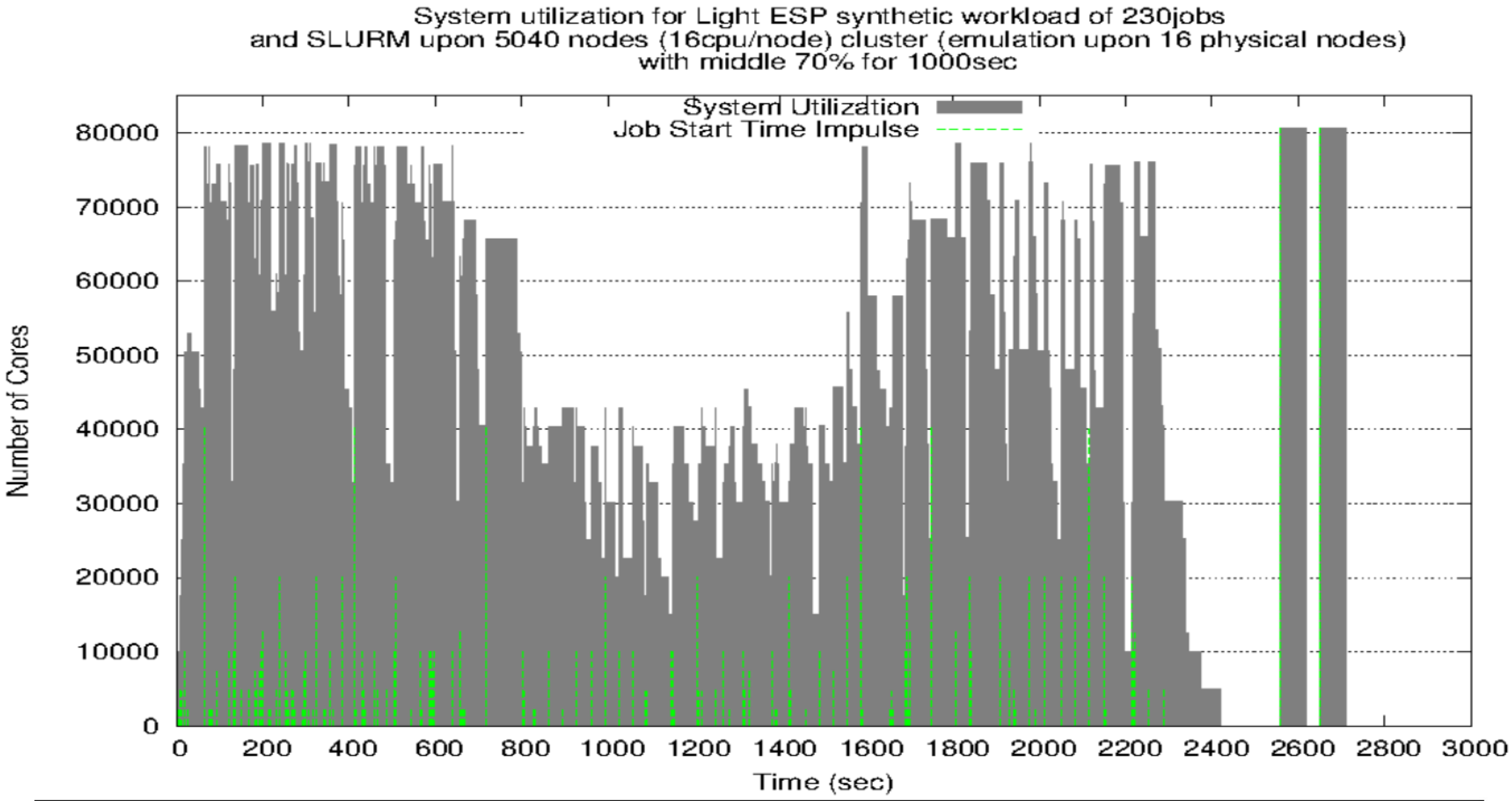
```
[root@nd25 slurm]#vi  src/layouts/power/cpufreq.c

const layouts_keyspec_t keyspec[] = {
        /* base keys */
        {"CurrentCorePower", L_T_UINT32},
        {"Cpufreq1", L_T_UINT32},
        {"Cpufreq1Watts", L_T_UINT32},
        /* parents aggregated keys */
        {"CurrentSumPower", L_T_UINT32,
        KEYSPEC_UPDATE_CHILDREN_SUM, "CurrentPower"},
       {"CurrentPower", L_T_UINT32,
        KEYSPEC_UPDATE_CHILDREN_SUM, "CurrentCorePower"},
        {NULL}
};
const char* etypes[] = {
        "Center",
        "Node",
        "Core",
        NULL
};
```

Experiments Testbed

- Consist of executing the Light-ESP synthetic workload composed of 230 jobs of 8 different job profiles (sizes, execution times)
- Deploy an emulated cluster with 5040 emulated nodes (16 cores / node) using 18 physical nodes
  - Upon an bullx B510 cluster with Intel Sandybridge (16cores/node, 64GB)
  - Using “multiple-slurmd” emulation technique
  - Layouts=power/cpufreq configured
- Experiments have as goal to:
  - Validate that powercapping works correctly
  - Compare the scaling of the powercapping logic, layouts framework and API functions

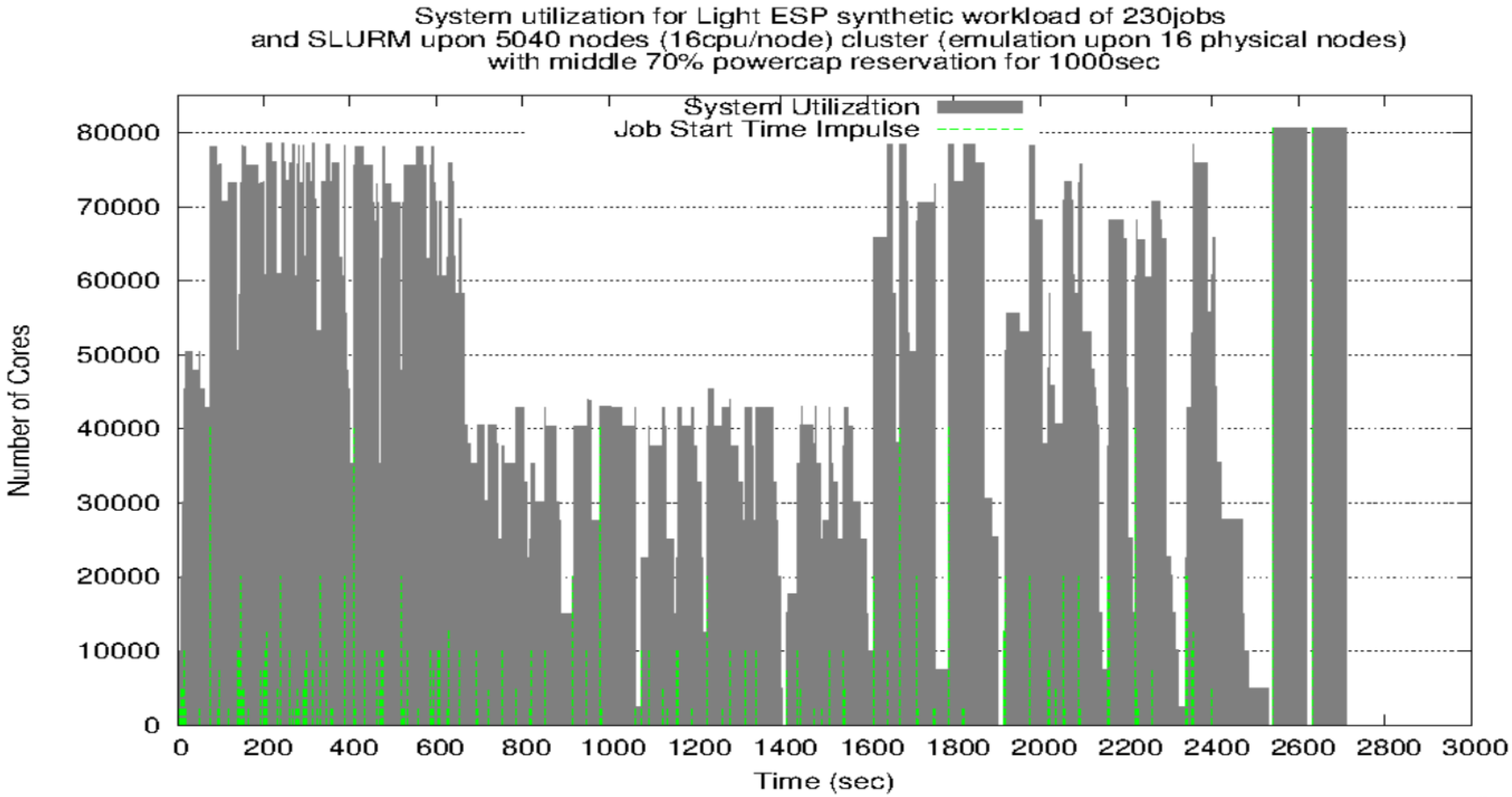
Power adaptive scheduling validation

– powercap set on-the-fly with scontrol update-



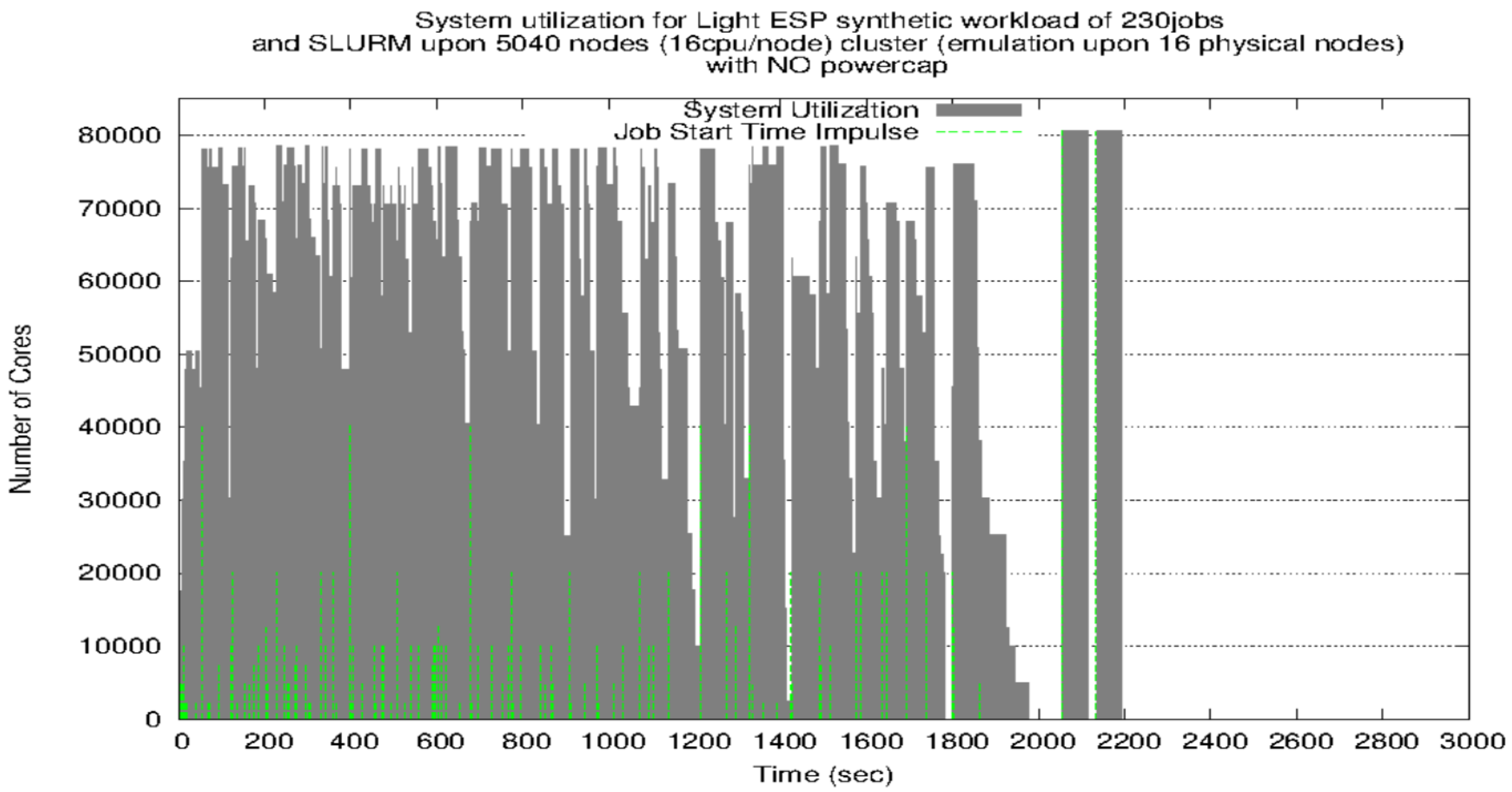
Power adaptive scheduling validation

– powercap set in advance with reservation -



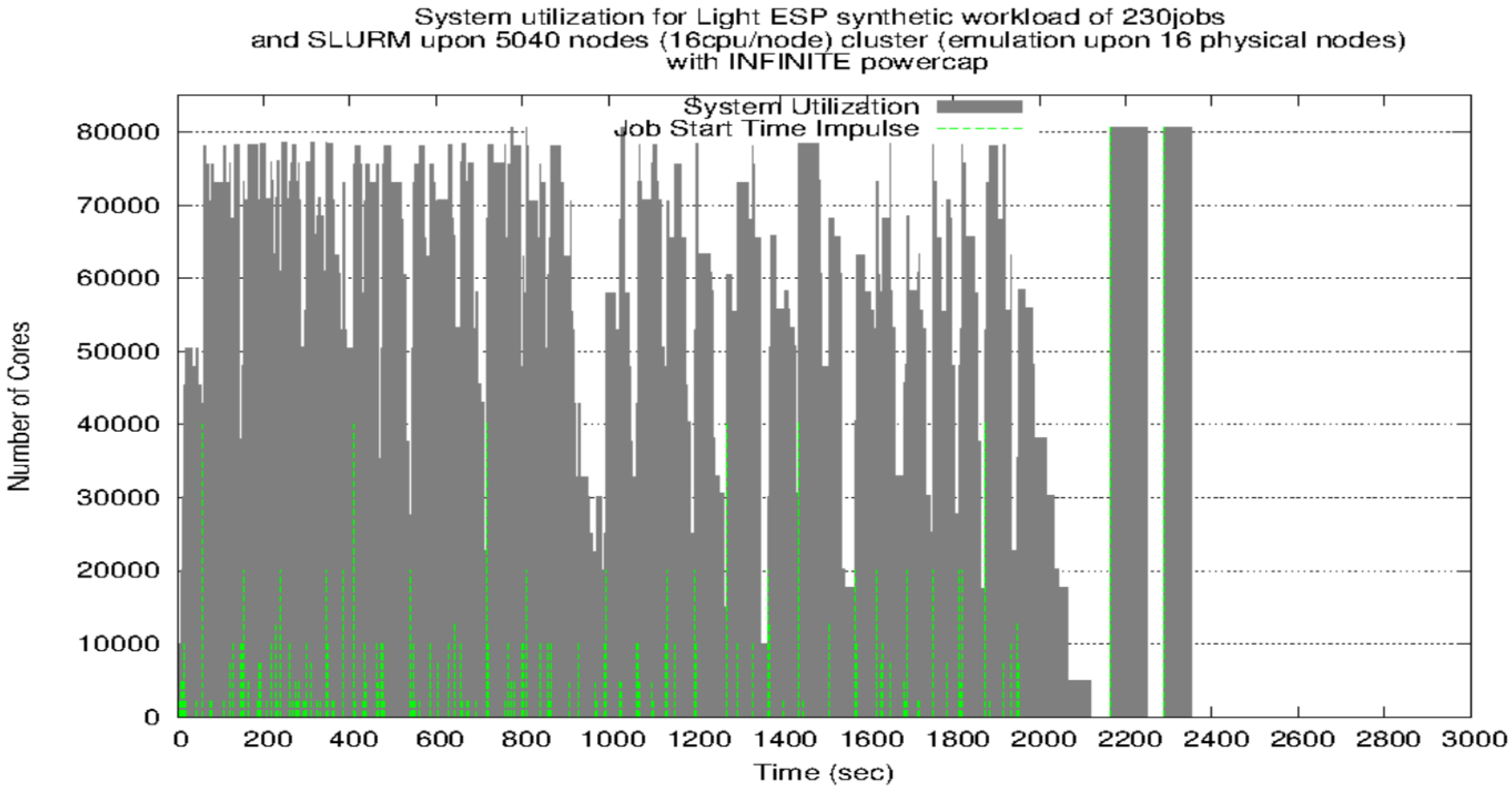
Power adaptive scheduling – scaling validation

– No powercap set -



Power adaptive scheduling scaling validation

– With powercap INFINITE -

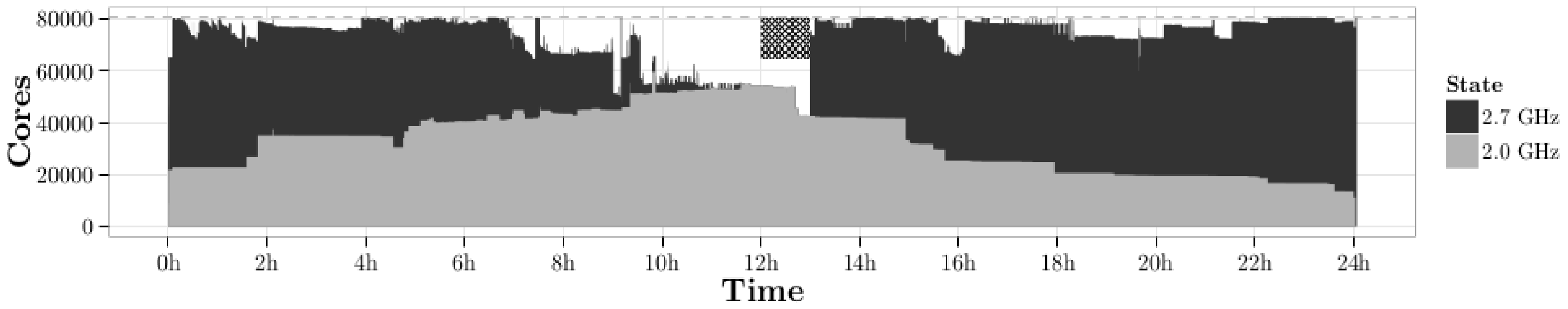


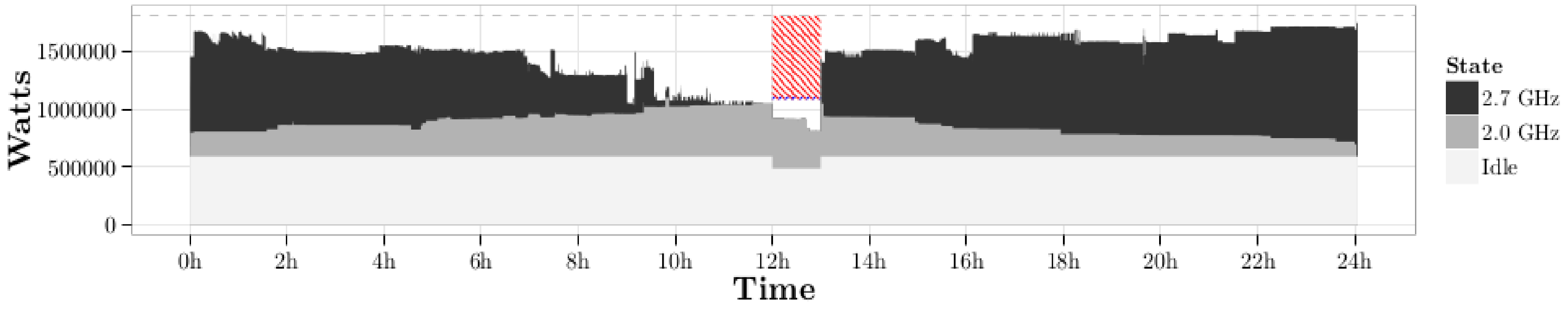
Discussion

- Power adaptive scheduling logic works fine but we can see that optimizations are needed in the layouts usage to reach the performance of bitmaps
  - This is due to the fact that we still check the power of each node individually, this should be done globally with consistent synchronization of layouts
  - The synchronization part of the layouts pull and push functions update the whole key/values store, it should update only the affected neighbours
  - Need of new layouts API functions to get/set multiple entities

Power adaptive scheduling

System utilization in terms of cores (top) and power (bottom) for MIX policy during a 24 hours workload of Curie system with a powercap reservation (hatched area) of 1 hour of 40% of total power. Cores switched-off represented by a dark-grey hatched area.

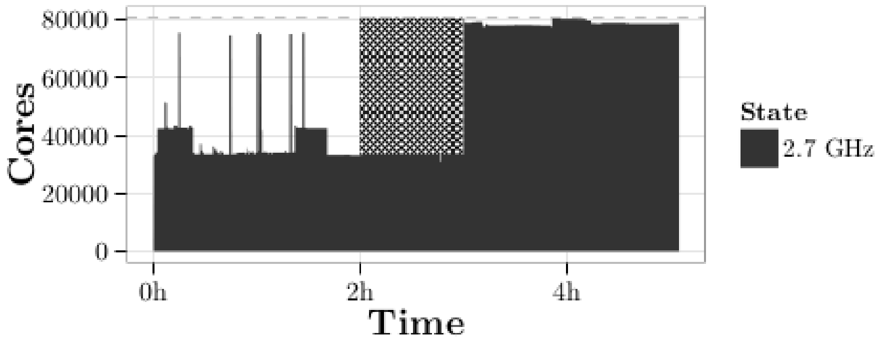


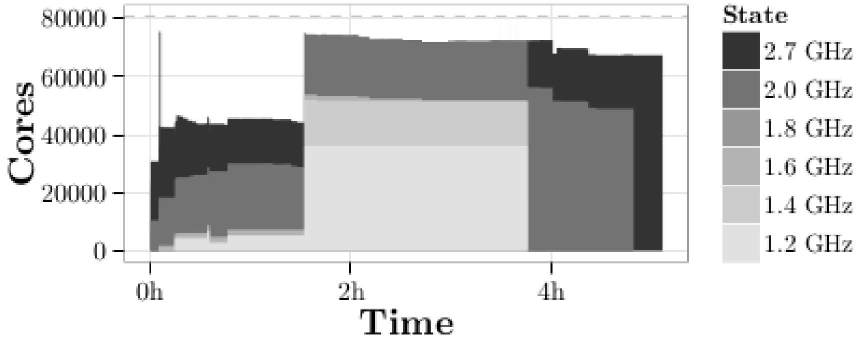


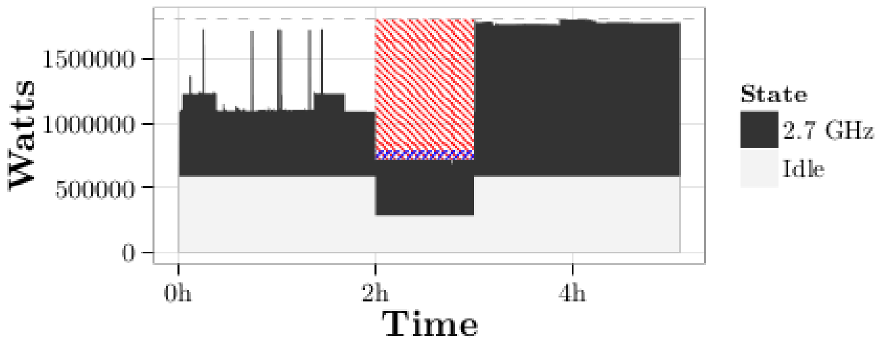
Power adaptive scheduling

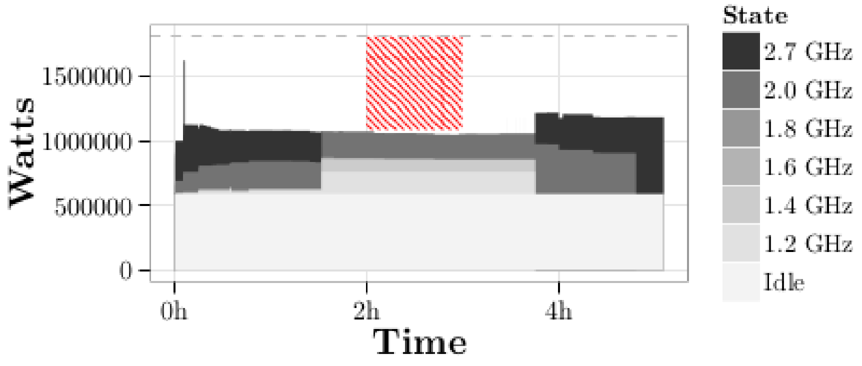
Powercap of 60% with mainly big jobs and SHUT policy

Powercap of 40% with mainly small jobs and DVFS policy









Discussion
Power Adaptive Scheduling and PowerPlugin

- The Power adaptive scheduling and the PowerPlugin logic (Cray) provide 2 different approaches for powercapping
  - The first one is based on logical, static power values and theoretical calculations for altering scheduling to achieve a global power budget
  - the former one based on the physical, real power values, and an integration to a hardware mechanism that will adapt each nodes power consumption to align to a global power budget
- Disadvantages:
  - Power adaptive scheduling is based on approximations so result may not be optimal either in system utilization or final power consumption
  - PowerPlugin will change the node configuration of jobs and depending on the executed application it will affect its turnaround time which may not be welcome
- Study ways of possible integration of both

Current and Future Works

- Further optimizations in the logic to improve scalability
- Make consistent the internal state of layouts (scontrol show layouts)
- Create new layouts API functions mainly to cover the previous points
  - Multi-entity get/set
  - Intelligent pull/push to modify only affected neighbours
- Provide ways to represent the real physical information of power consumption from the sensors to the layouts
  - Integration with real sensors data as used within AcctGatherEnergy plugin (IPMI, RAPL)
  - Add values such as -Latest20AverageWatts- or -Latest100AverageWatts- to capture time factor of an already used node
- Study and extent to dynamic DVFS support
  - Change CPU Frequency on the fly during job execution
  - This may help when both entering or coming out of powercap period

Thanks

For more information please contact:
T+ 33 1 98765432
F+ 33 1 88888888
M+ 33 6 44445678
firstname.lastname@atos.net

12-05-2015