Energy Efficiency in HPC
Resource Management and Scheduling aspects

Yiannis Georgiou
Architect R&D

Ecole E3-RSD

26-05-2016

Introduction

- ▶ High Performance Computing Systems run on large amounts of power
- ▶ Faster supercomputer in the world: Tianhe-2 in National University of Defense Technology, China

  – Performance:

  33.86 PF/s Linpack (55 PF/s Peak)

  – Power consumption:

  17.8 MW (plus 24 MW cooling)

  – Electricity cost

  (~ 45k€ per day)

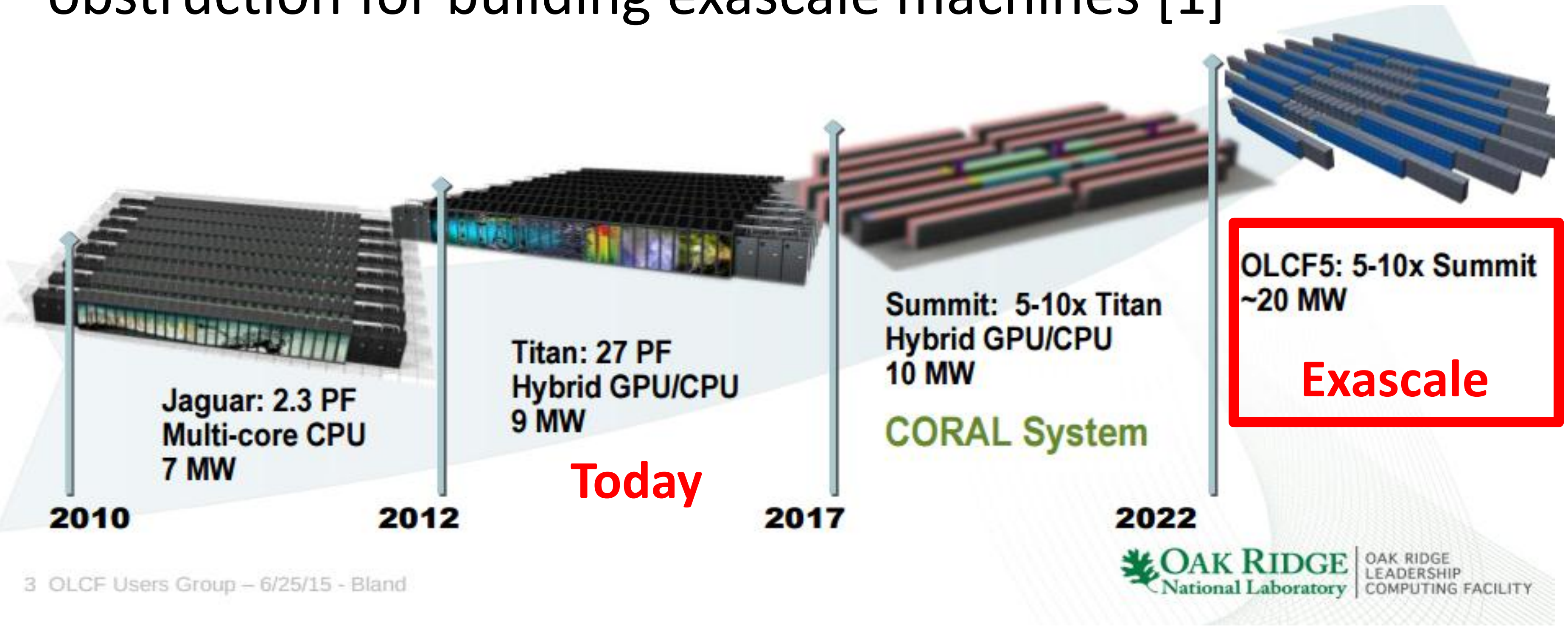
Sources: http://top500.org, http://www.scmp.com/news/china/article/1543226/chinas-world-beating-supercomputer-fails-impress-some-potential-clients

Introduction

- Half of the cost of a Petascale system comes from energy consumption, and today, it costs about **1 million dollars a year** to run a **1 MW** system.
- This means that the electricity bill is roughly equal to the hardware cost of such platforms.
- Cost not the only problem. Heat generation because of density and energy consumption is difficult to disseminate

Introduction

- The energy consumption is the most important obstruction for building exascale machines [1]



Buddy Bland, Present and Future Leadership Computers at OLCF, June 2015

[1] J. Dongarra et al., “The international exascale software project roadmap,” in *International Journal of High Performance Computing Applications*, 2011.

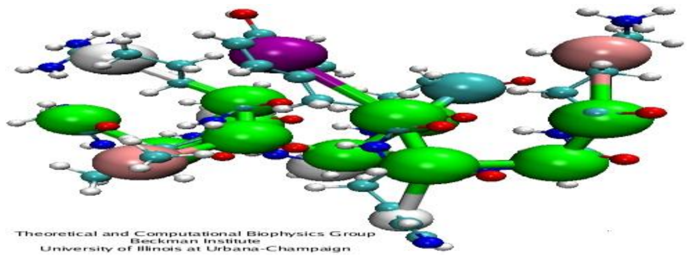
Introduction

- **Energy efficiency** major requirement in all levels of **hardware and software design**
- Ultimate goal: a maximum throughput within a given energy budget.
- Additional constraints :
  - maximum power capping or constant power consumption with only small perturbation.
    - If those constraints are met, a power supplier can often provide a significantly lower price, thus increasing the efficiency in terms of TCO.

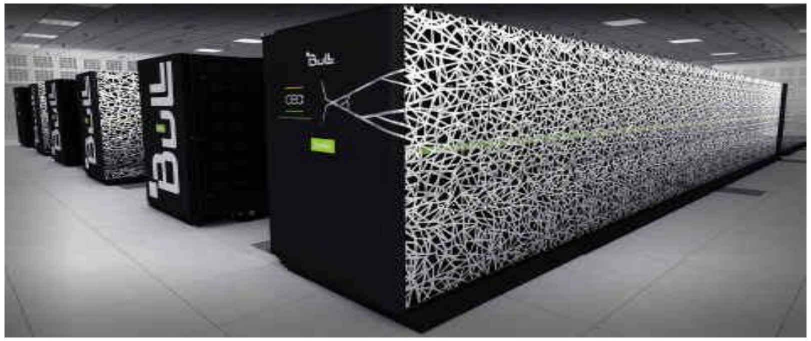
State of the art

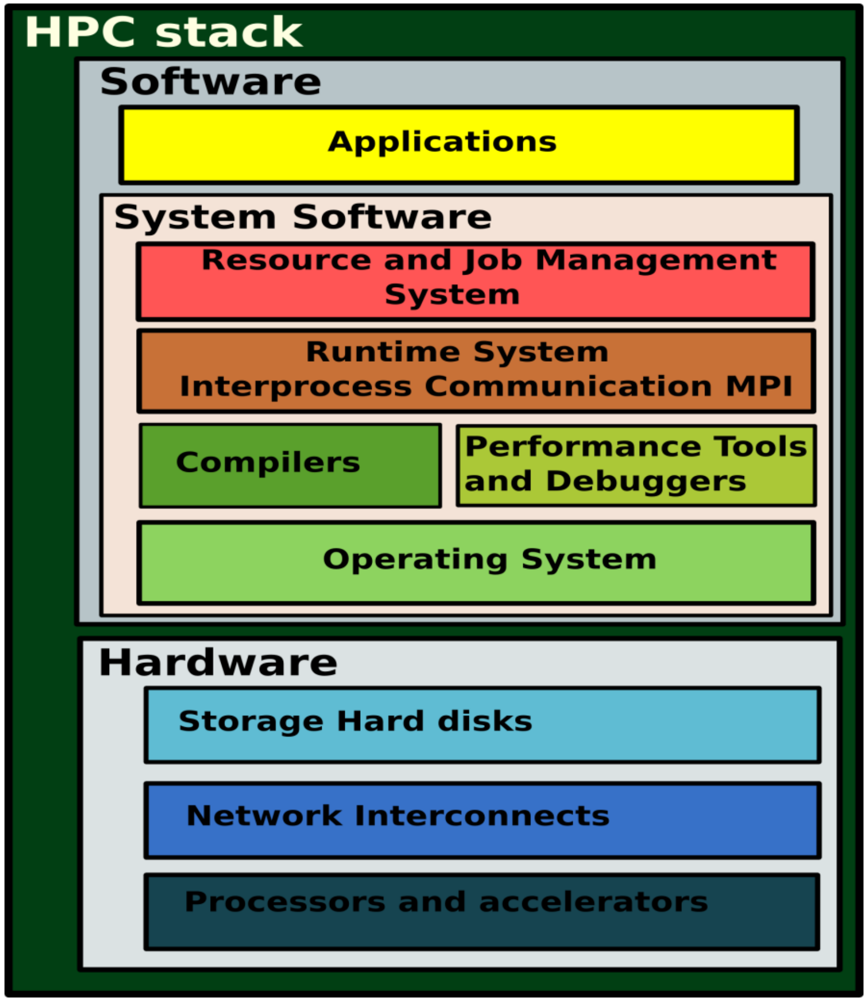
- There are 2 main approaches for energy efficiency in HPC :
  - **Static power management** which deals with designing hardware operating on efficient energy levels
  - **Dynamic power management** in which the software dynamically adapts its consumption based on the usage of the resources.

High Performance Computing Systems


Theoretical and Computational Biophysics Group
Beckman Institute
University of Illinois at Urbana-Champaign

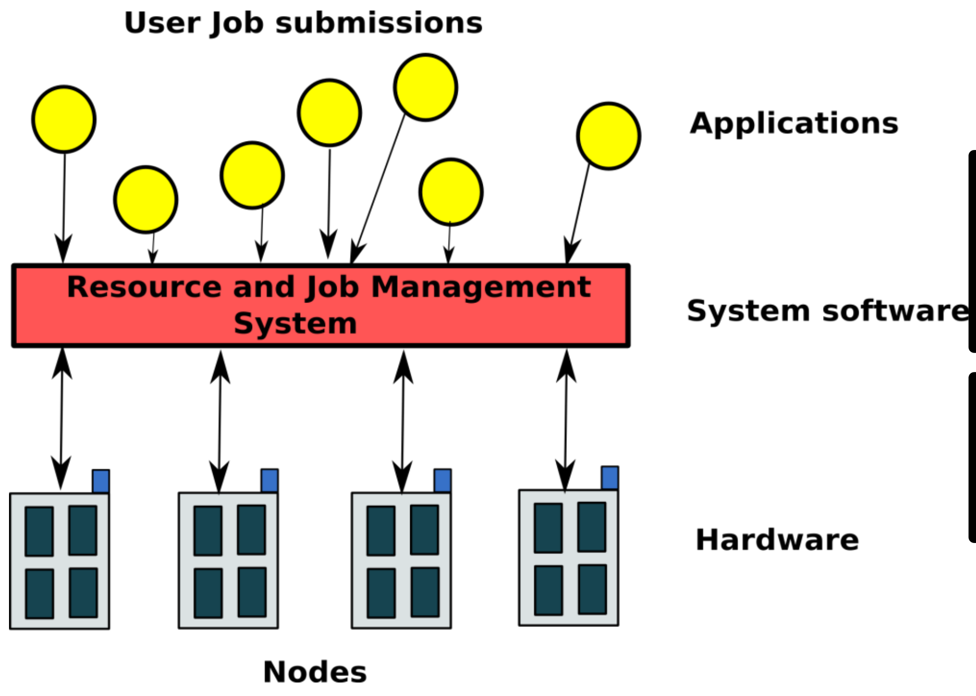
- **System Software:**
- Operating System, Runtime
- System, Resource Management,
- I/O System, Interfacing to External
- Environments





Resource and Job Management System

•The goal of a Resource and Job Management System (RJMS) is to satisfy users' demands for computation and assign resources to user jobs with an efficient manner.

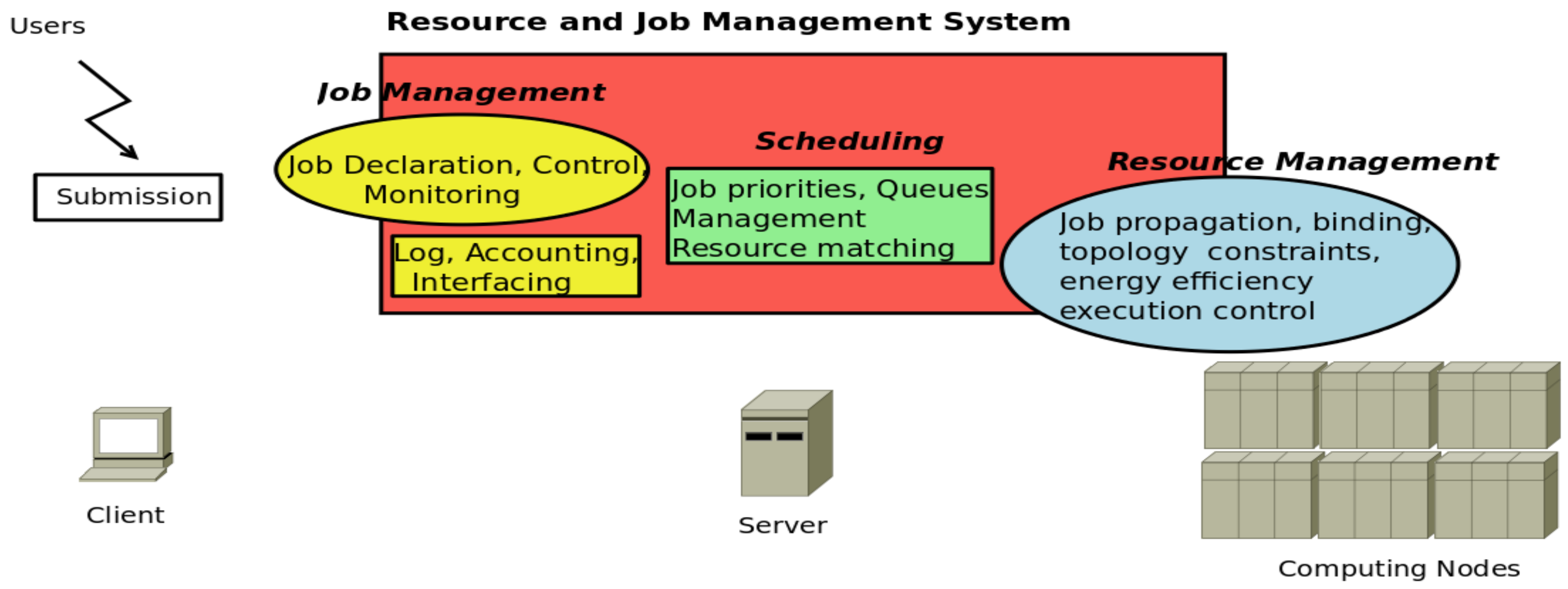


RJMS Importance
Strategic Position, responsibility for the overall system performance

Direct and constant knowledge of resources and application needs

Resource and Job Management System Layers

This assignement involves three principal abstraction layers:
•**Job Management:** declaration of a job and demand of resources and job characteristics,
•**Scheduling**: matching of the jobs upon the resources,
•**Resource Management :** launching and placement of job instances upon the computation resources along with the job's control of execution


Users
Resource and Job Management System
Job Management
Submission
Job Declaration, Control, Monitoring
Log, Accounting, Interfacing
Scheduling
Job priorities, Queues Management Resource matching
Resource Management
Job propagation, binding, topology constraints, energy efficiency execution control
Client
Server
Computing Nodes

Objectives

- Deal with energy efficiency in HPC from the RJMS side
  - Power and Energy **measurement system**
  - Allow the user to **control the energy efficiency** of his/her executions
  - Deal with power and energy as **dynamic resources** and provide internal mechanisms to adapt the scheduling
- Studies with simulations and emulations along with **implementations** upon a widely known open-source Resource and Job Management System: **SLURM**

SLURM scalable and flexible RJMS

- **Scalability**: Designed to operate in a heterogeneous cluster with up to tens of millions of processors.
- **Performance**: Can accept 1,000 job submissions per second and fully execute 500 simple jobs per second (depending upon hardware and system configuration).
- **Free and Open Source**: Its source code is freely available under the GNU General Public License.
- **Portability**: Written in C with a GNU autoconf configuration engine. While initially written for Linux, Slurm has been ported to a diverse assortment of systems.
- **Power Management**: Job can specify their desired CPU frequency and power use by job is recorded. Idle resources can be powered down until needed.
- **Fault Tolerant**: It is highly tolerant of system failures, including failure of the node executing its control functions.
- **Flexibility**: A plugin mechanism exists to support various interconnects, authentication mechanisms, schedulers, etc.

https://**github**.com/**SchedMD/slurm**

SLURM History and Facts

- Initially developed in LLNL since 2003, passed to SchedMD in 2011
- Multiple enterprises and research centers have been contributing to the project (LANL, CEA, HP, BULL, BSC, CRAY etc)
- Large international community, active mailing lists (support by main developers)
  - Contributions (various external software and standards are integrated upon SLURM)
- As of the June 2015 **Top500** supercomputer list, SLURM is being used on **six of the ten** most powerful computers in the world including the **no1 system, Tianhe-2** with 3,120,000 computing cores.

BULL and SLURM

- BULL initially started to work with SLURM in 2005
- About 6 SLURM-dedicated engineers since 2013
  - –**Research** upon the field of Resource Management and Job Scheduling (National/European financed projects, PhDs) and definition of RoadMap
  - –**Development** of new SLURM features: all code dropped in the open-source
  - –**Support** upon clusters : Training, Configuration, Bug correction, Feature Requests, etc
- Integrated as the default RJMS into the **BULL- HPC software stack** since 2006
- Close development **collaboration** with SchedMD and CEA
- Organaziation of Slurm User Group (SUG) Conference (User, Admin Tutorials + Technical presentation for developpers) http://www.schedmd.com/slurmdocs/publications.html

Overview

- Power/Energy Monitoring and Control
  - Measurement System
  - Energy Accounting
  - Power Profiling
  - User level control of power and energy
- Power adaptive and Energy aware scheduling
  - User Incentives for energy aware scheduling
  - System-level control of power and energy
  - Power adaptive scheduling
- Ongoing Works and Road to Exascale
  - Dynamic Runtime Energy Optimizations
  - Towards energy budget control
  - Energy Efficiency and road to exascale

Power/Energy Monitoring and Control

Power and Energy Management

- Issues that we wanted to deal with:
- Attribute **power and energy data** to HPC components
- Calculate the **energy consumption of jobs** in the system
- Extract **power** consumption **time series of jobs**
- **Control** the Power and Energy usage of jobs and workloads

Power and Energy Measurement System

- Power and Energy monitoring per node
- Energy accounting per step/job
- Power profiling per step/job
- CPU Frequency Selection per step/job

- **How this takes place :**
- In-band collection of energy/power data (IPMI / RAPL plugins)
- Out-of-band collection of energy/power data (RRD plugin)
- Power data job profiling (HDF5 time-series files)
- Parameter for CPU frequency selection on submission commands

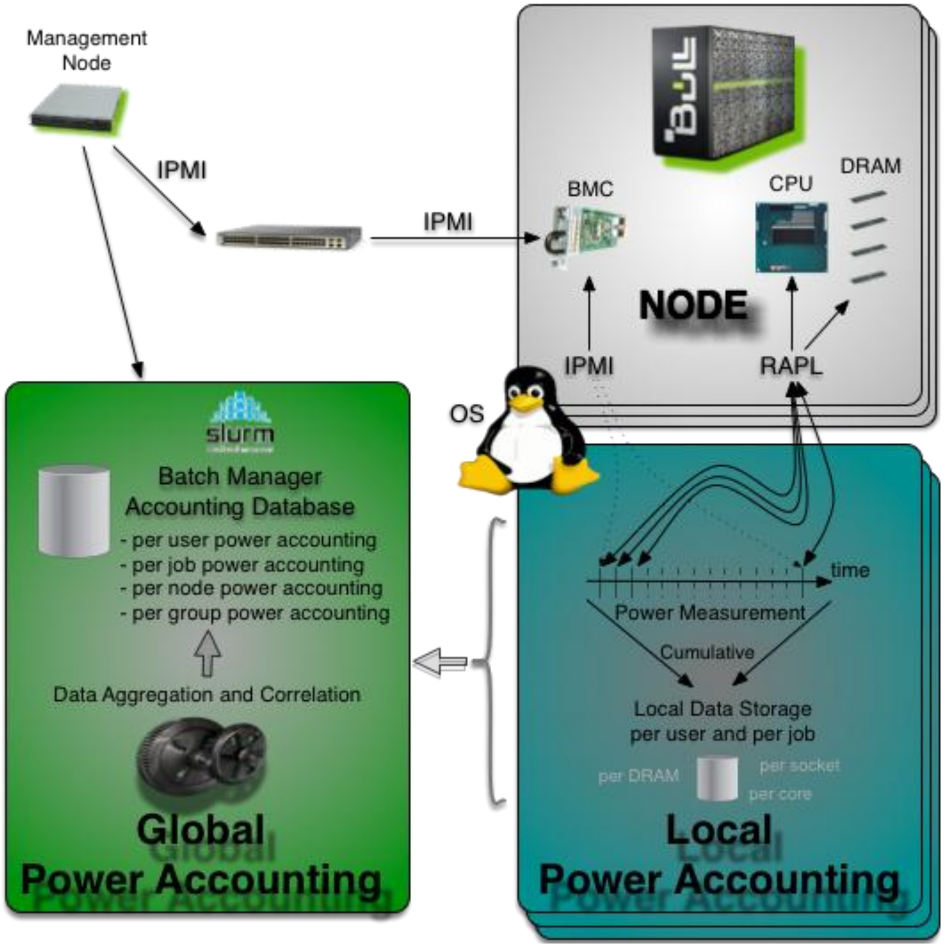
Power and Energy Measurement System

- Power and Energy monitoring per node
- Energy accounting per step/job
- Power profiling per step/job

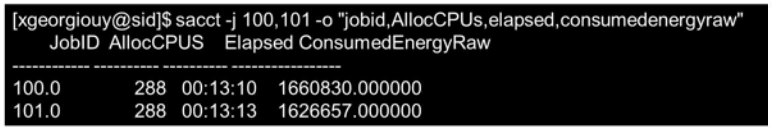
- **Overhead:** In-band Collection
- **Precision:** measurements and internal calculations
- **Scalability:** Out-of band Collection

- **How th**
- In-band collection of energy/power data (IPMI / RAPL plugins)
- Out-of-band collection of energy/power data (RRD plugin)
- Power data job profiling (HDF5 time-series files)
- SLURM Internal power-to-energy and energy-to-power calculations

Energy accounting per job with Slurm

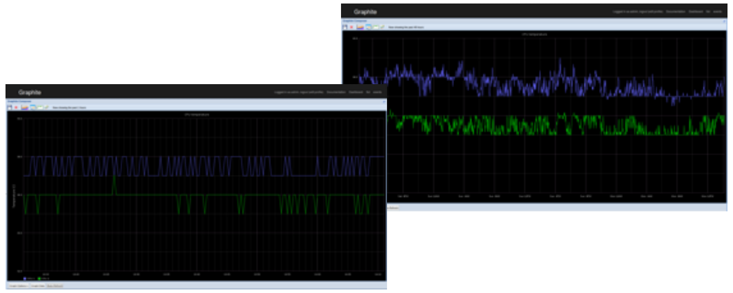

Management Node
IPMI
IPMI
BMC
CPU
DRAM
NODE
IPMI
RAPL
OS
slurm
Batch Manager Accounting Database
- per user power accounting
- per job power accounting
- per node power accounting
- per group power accounting
Data Aggregation and Correlation
Global Power Accounting
time
Power Measurement
Cumulative
Local Data Storage per user and per job
per DRAM
per socket
per core
Local Power Accounting

- Total amount of energy consumption per job is stored in Slurm accounting database



```
[xgeorgiouy@sid]$ sacct -j 100,101 -o "jobid,AllocCPUs,elapsed,consumedenergyraw"
       JobID  AllocCPUS    Elapsed ConsumedEnergyRaw
------------ ---------- ---------- -----------------
100.0               288   00:13:10     1660830.000000
101.0               288   00:13:13     1626657.000000
```

- Data can be displayed in Graphite



Power and Energy Measurement System

```
[root@cuzco108 bin]# $ scontrol show n=mo38 | grep ConsumedJoules
   CurrentWatts=105 LowestJoules=105 ConsumedJoules=17877

[root@cuzco108 bin]# sacct -o
"JobID%5,JobName,AllocCPUS,NNodes%3,NodeList%22,State,Start,End,Elapse
d,ConsumedEnergy%9"
JobID    JobName  AllocCPUS NNodes               NodeList      State
Start                End     Elapsed ConsumedEnergy
----- ---------- ---------- --- ---------------------- ---------- ----
--------------- -------------------- ---------- ---------
  127    cg.D.32         32   4     cuzco[109,111-113]  COMPLETED
2013-09-12T23:12:51 2013-09-12T23:22:03   00:09:12   490.60KJ

[root@cuzco108 bin]# cat extract_127.csv
Job,Step,Node,Series,Date_Time,Elapsed_Time,Power
13,0,orion-1,Energy,2013-07-25 03:39:03,0,126
13,0,orion-1,Energy,2013-07-25 03:39:04,1,126
13,0,orion-1,Energy,2013-07-25 03:39:05,2,126
13,0,orion-1,Energy,2013-07-25 03:39:06,3,140
```

In-band collection of power/energy data with IPMI

- **IPMI** is a message-based, hardware-level interface specification (may operate in-band or out-of-band)
- Communication with the Baseboard Management Controller **BMC**
- SLURM support is based on the FreeIPMI (opensource)
- Data collected in Watts
- SLURM individual polling frequency (>=1sec)
  - direct usage for power profiling
  - internal SLURM calculations for energy reporting per job

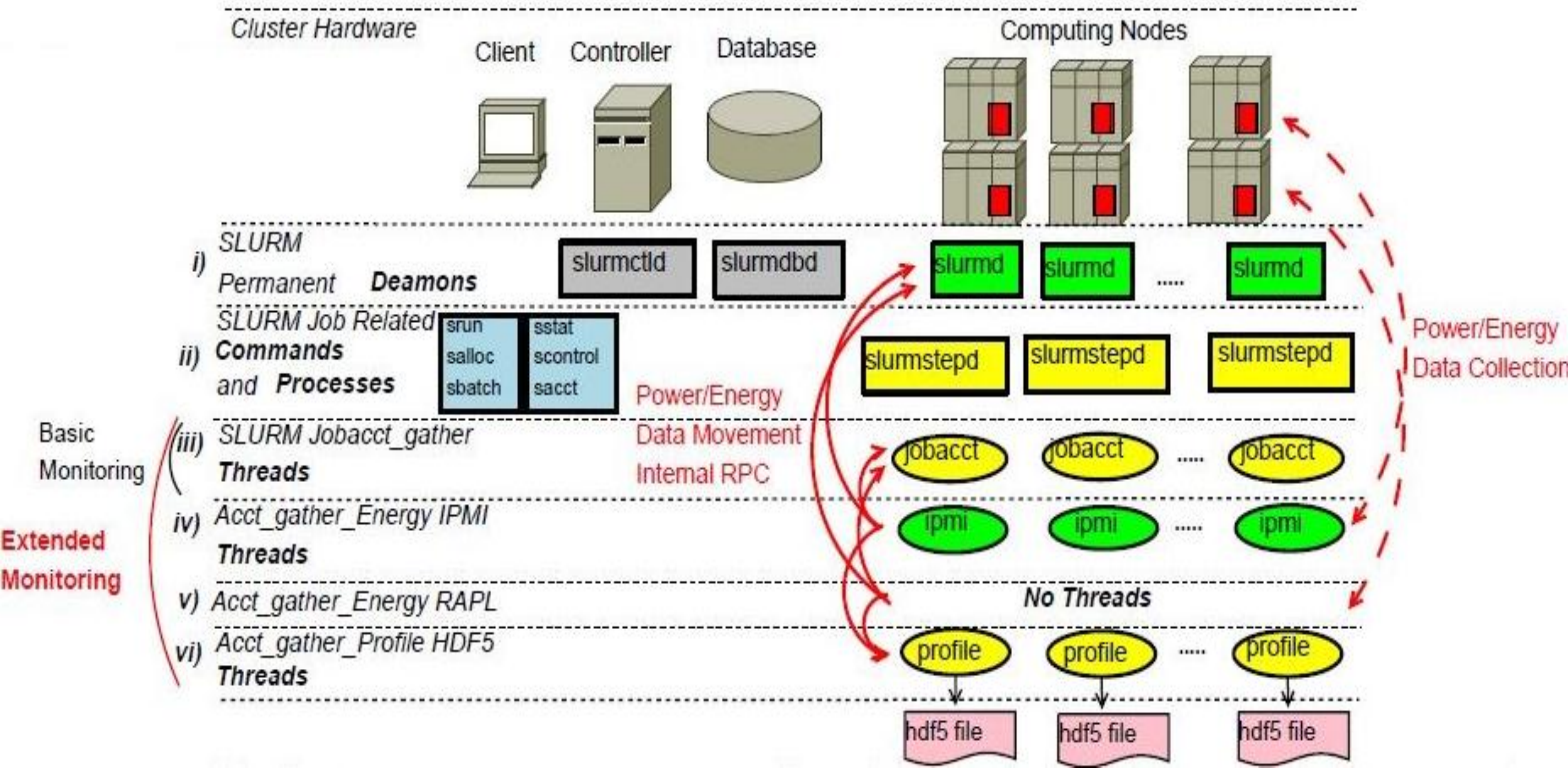
In-band collection of power/energy data with RAPL

- **RAPL** (Running Average Power Limit) interface implemented mainly for power-cap on socket level
- Interfaces can estimate current energy usage based on a software model
- The data collected from RAPL is energy consumption in Joules
- SLURM individual polling frequency (>=1sec)
  - direct usage for energy reporting per job
  - but internal SLURM calculations for power reporting

Power Profiling with HDF5

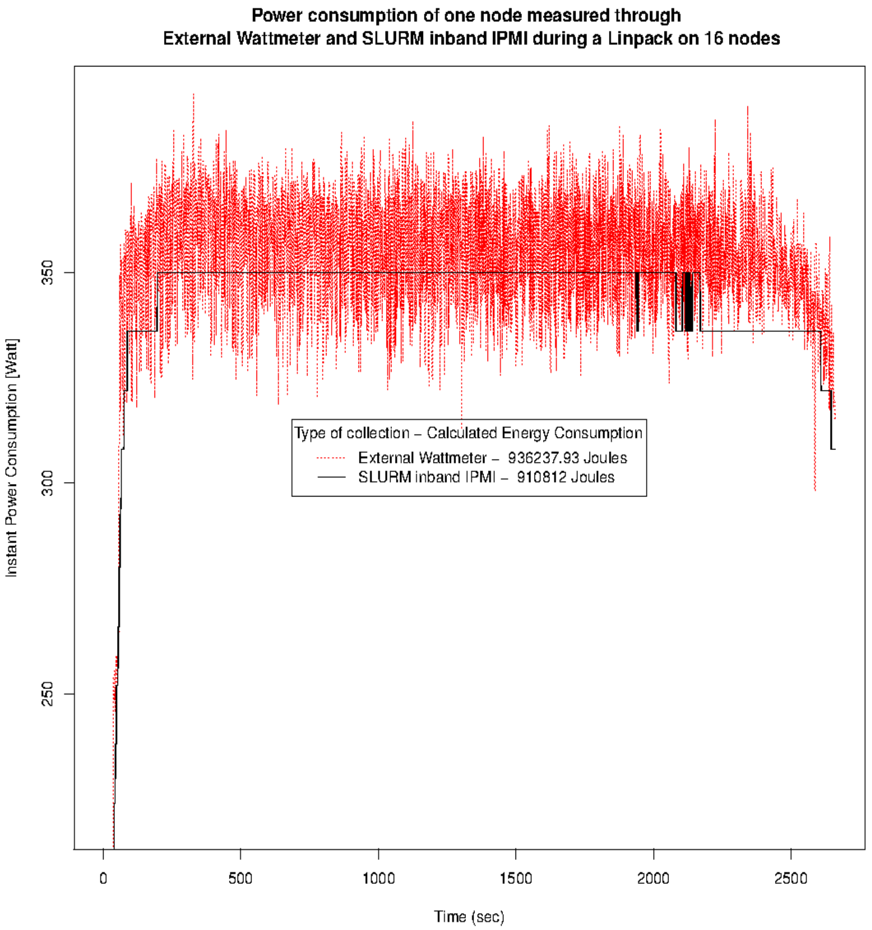
- Job profiling to periodically capture the task’s usage of various resources like CPU, Memory, Lustre, Infiniband and Power per node
- Resource Independent polling frequency configuration
- Based on **hdf5** file format (opensource)
- Profiling per node (one hdf5 file per job on each node)
- Aggregation on one hdf5 file per job (after job termination)
- Slurm built-in tools for extraction of hdf5 profiling data

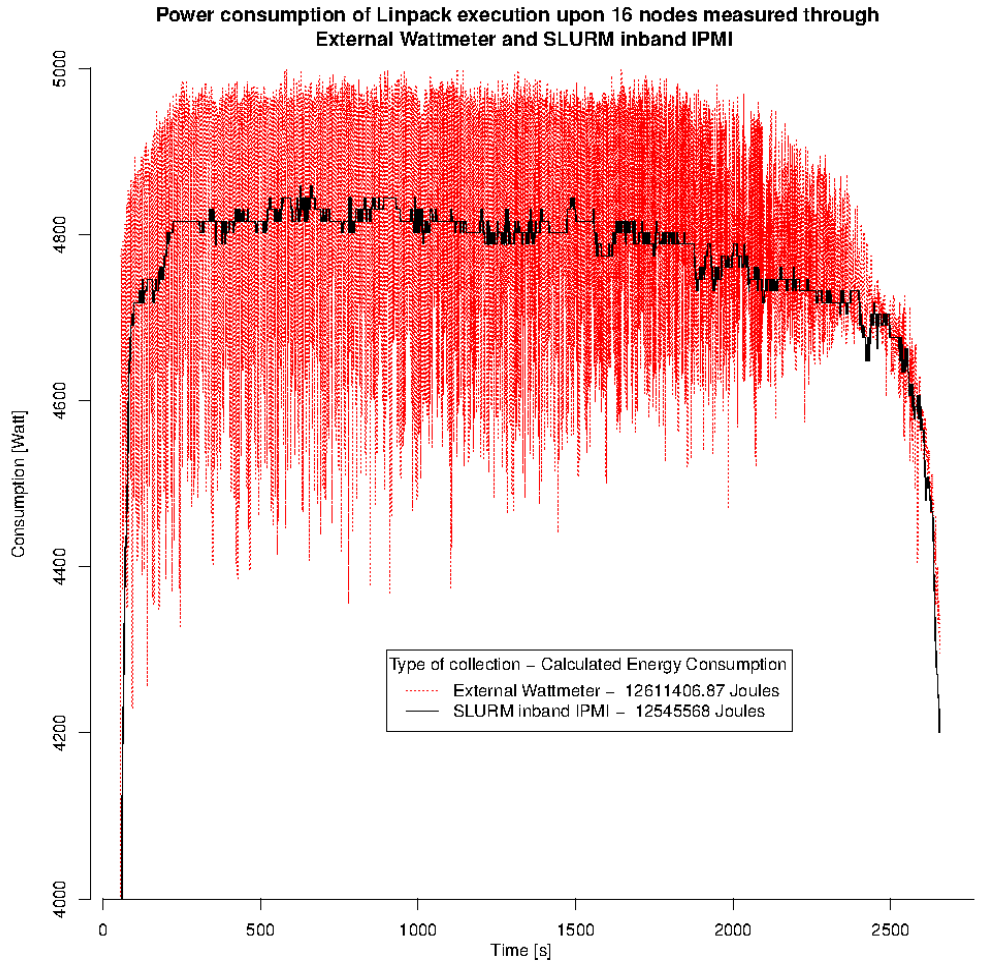
Energy Accounting and Power Profiling Architecture



Bull
atos technologies

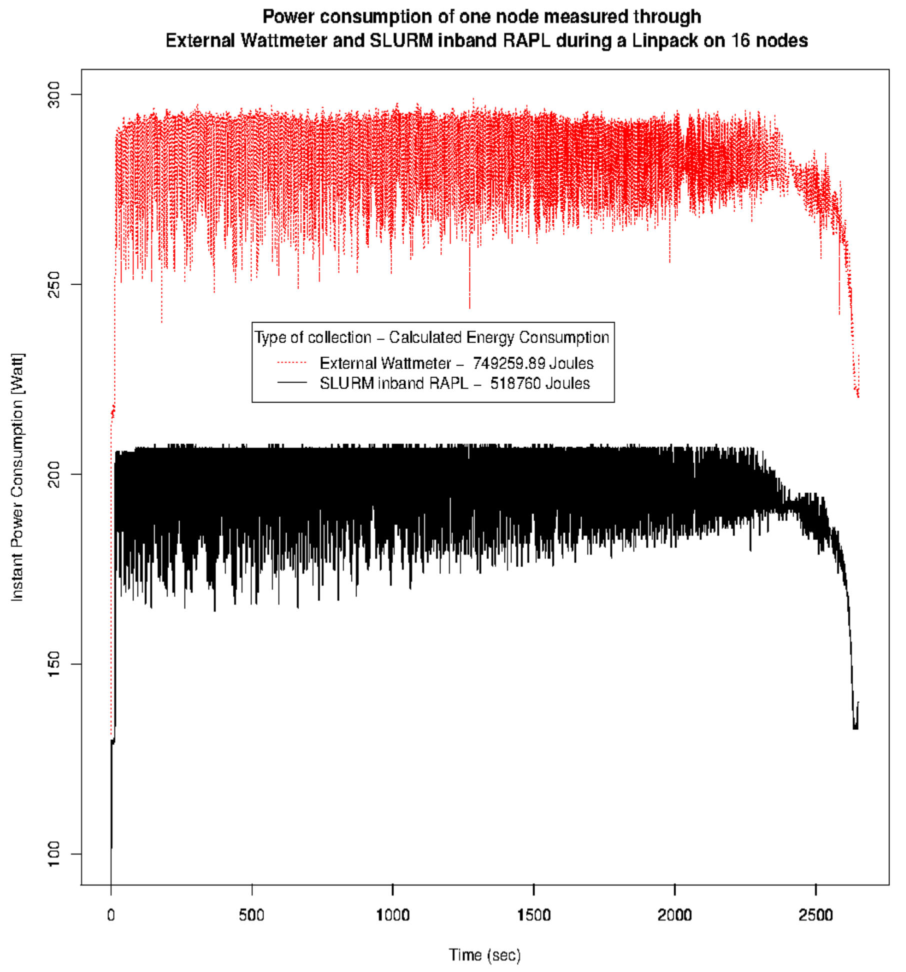
Power and Energy Measurement System

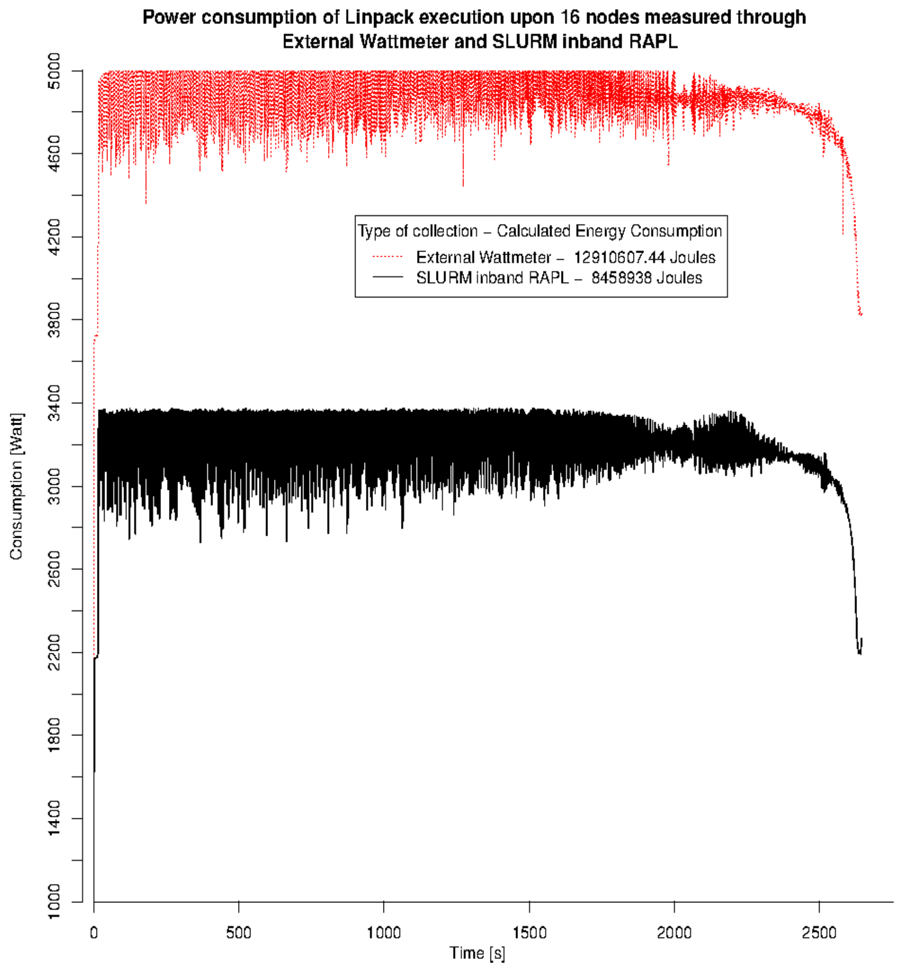




Yiannis Georgiou, Thomas Cadeau, David Glesser, Danny Auble, Morris Jette and Matthieu Hautreux
Energy Accounting and Control with SLURM Resource and Job Management System
(In proceedings of ICDCN 2014)

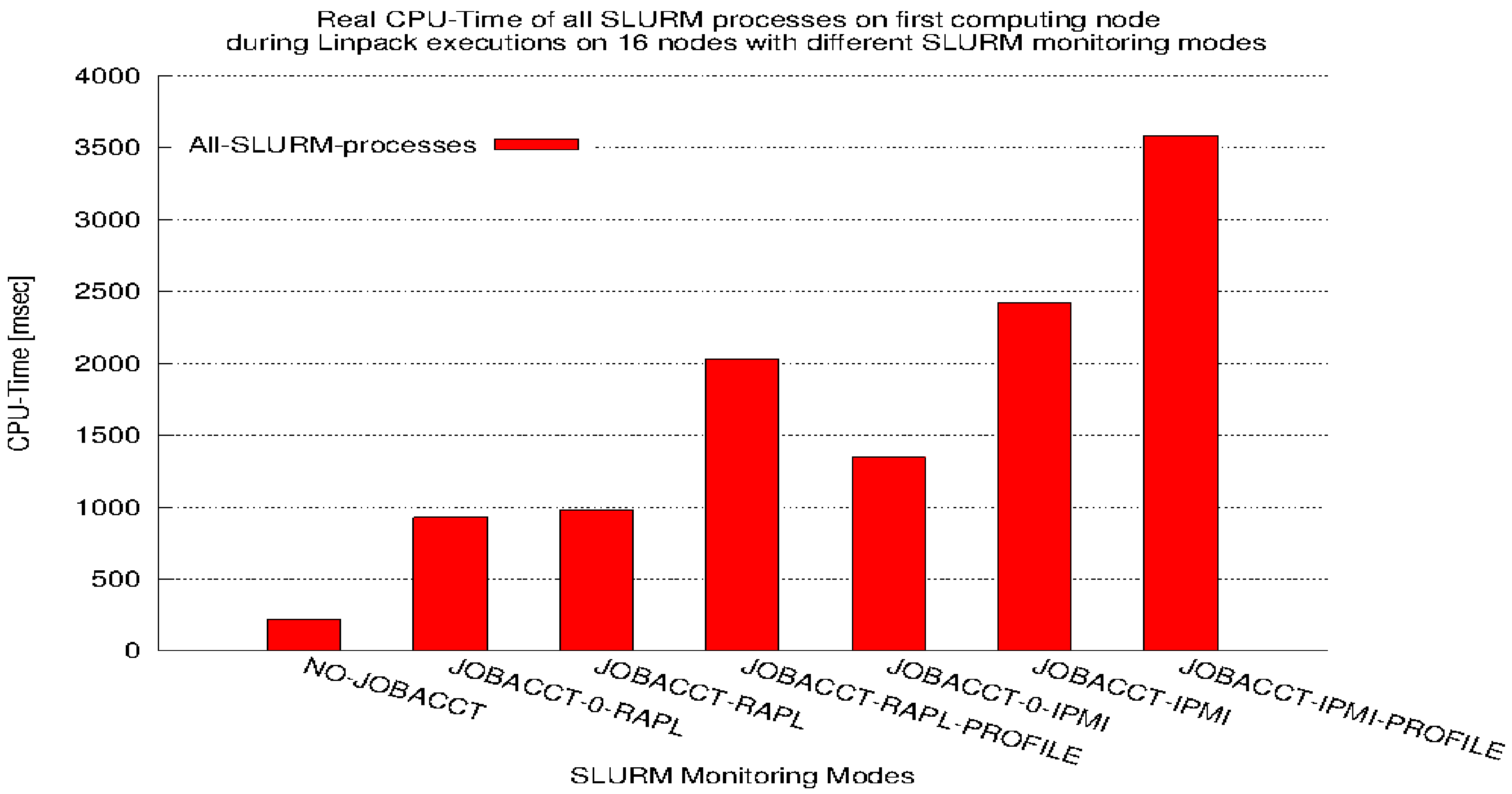
Power and Energy Measurement System





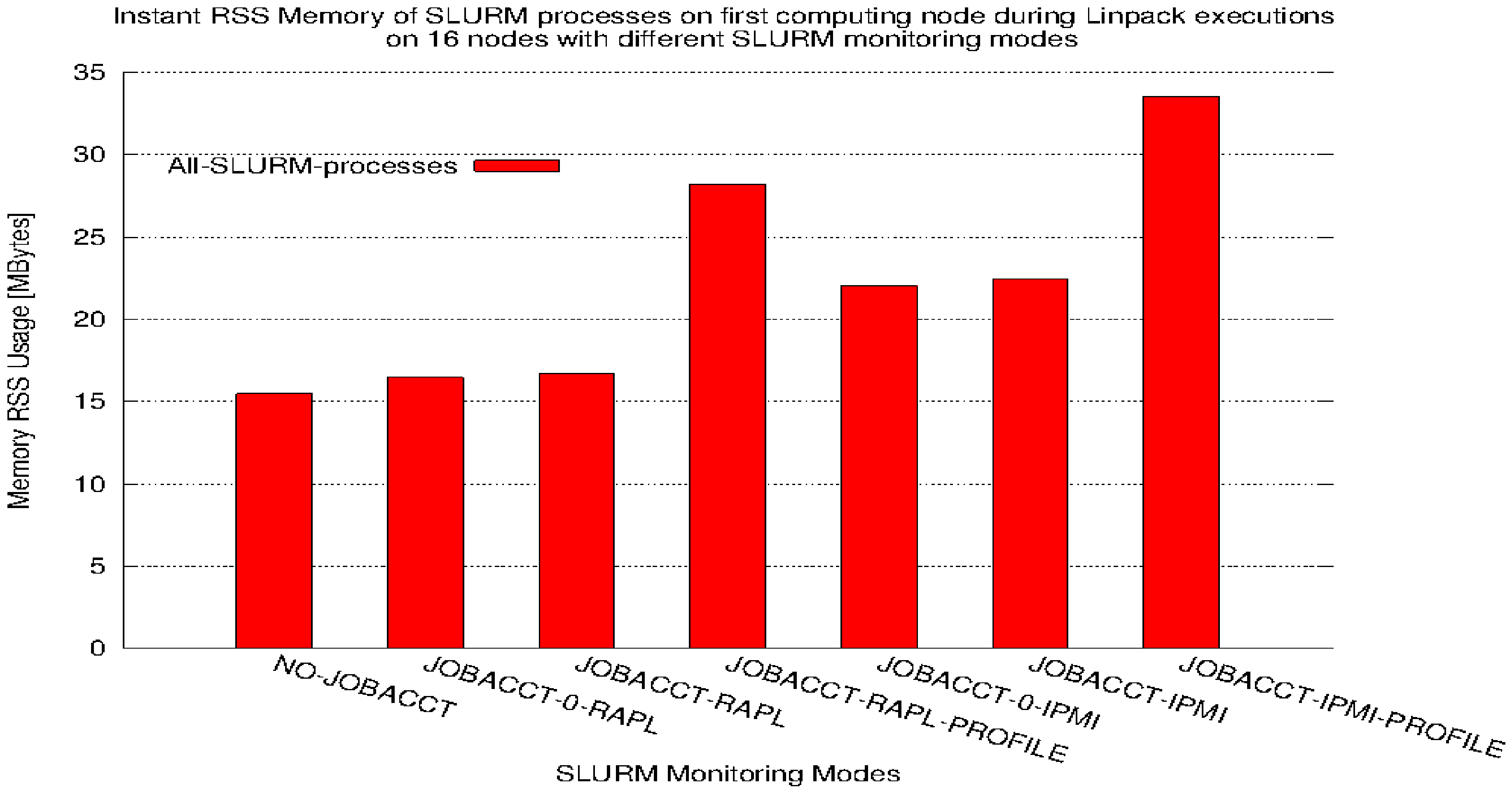
Yiannis Georgiou, Thomas Cadeau, David Glesser, Danny Auble, Morris Jette and Matthieu Hautreux
Energy Accounting and Control with SLURM Resource and Job Management System
(In proceedings of ICDCN 2014)

Power and Energy Measurement System
CPU Overhead for IN-Band techniques



Yiannis Georgiou, Thomas Cadeau, David Glesser, Danny Auble, Morris Jette and Matthieu Hautreux
Energy Accounting and Control with SLURM Resource and Job Management System
(In proceedings of ICDCN 2014)

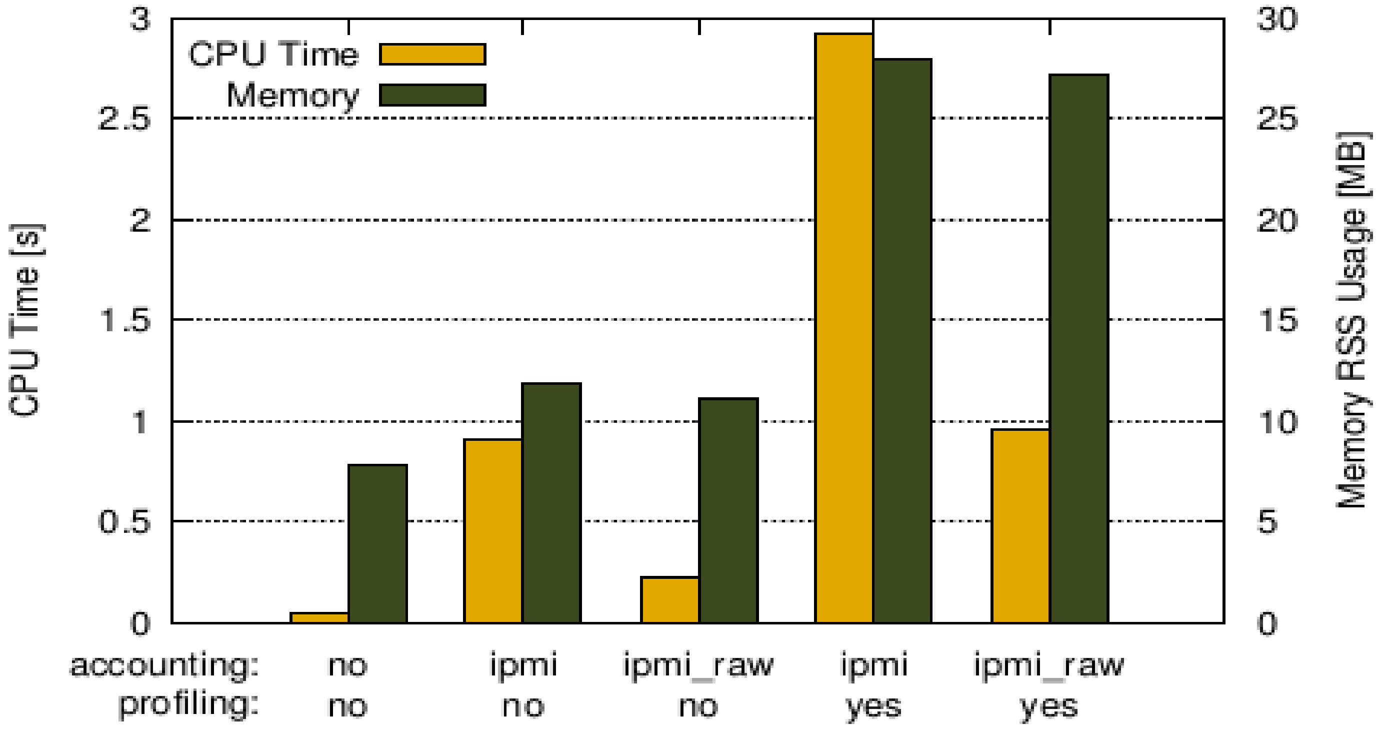
Power and Energy Measurement System
Memory Overhead for IN-Band techniques



Yiannis Georgiou, Thomas Cadeau, David Glesser, Danny Auble, Morris Jette and Matthieu Hautreux
Energy Accounting and Control with SLURM Resource and Job Management System
(In proceedings of ICDCN 2014)

Optimizations of Power and Energy Measurement System

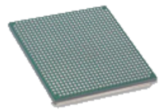
- Based on TUD/BULL - BMC firmware optimizations
  - sampling to 4Hz
  - No overhead for accounting

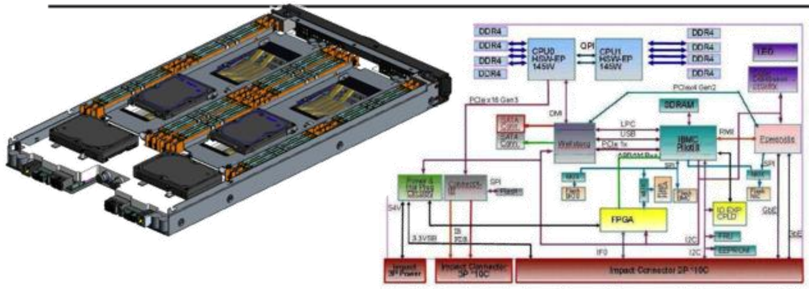


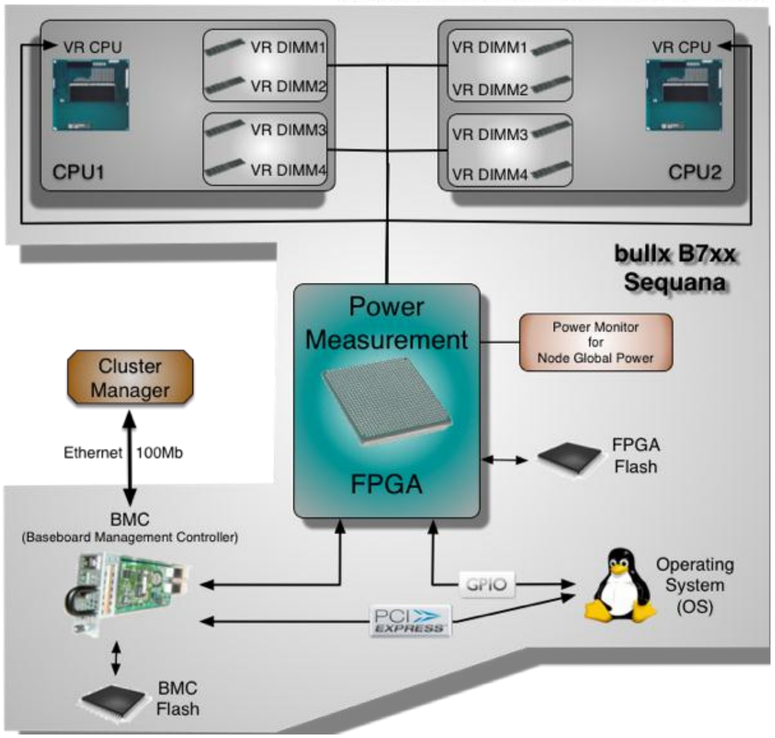
TECHNISCHE UNIVERSITÄT DRESDEN

Bull
atos technologies

FPGA for power measurement



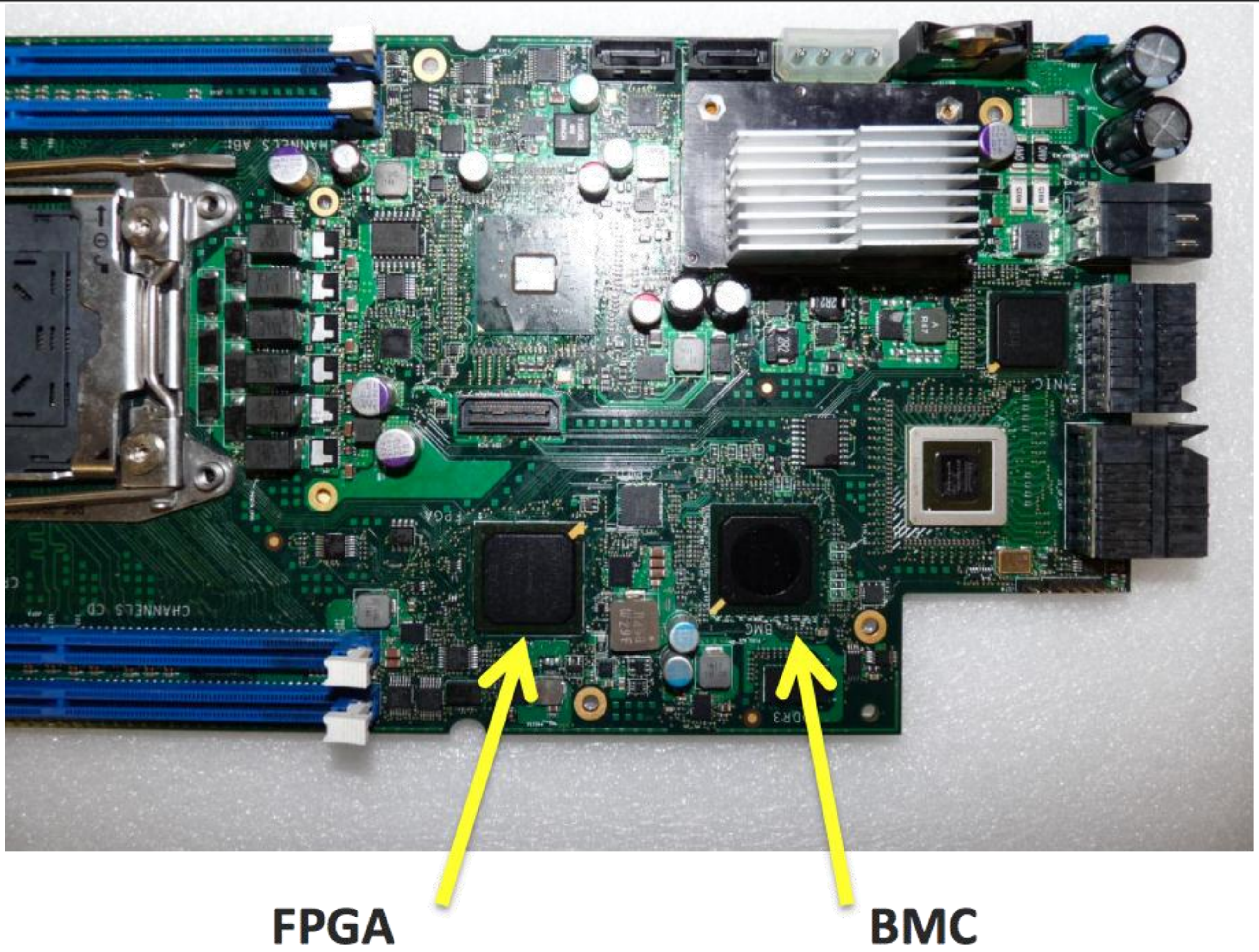




- On bullx B7xx and Bull Sequana platform a power measurement FPGA is integrated in each compute node
- Provides a sampling up to:
  - 1000 sample per second for global power including sockets, DRAM, SSD and on-board
  - 100 sample per second for voltage regulators (VR) – 6 VR: one per socket + 4 for DRAM (one / 2 lanes)
- High accuracy with 2-3% of uncertainty after calibration
  - 2% for blades
  - 5% for VR
- Time stamped measurements

Bull
atos technologies

Hardware implementation
bullx B7xx



HDEEM
High Definition Energy Efficiency Monitoring

with FPGA power measurement

- Collaborative project with Technical University of Dresden (ZIH)
- C API ease to use to gather power data
  - Start / Stop / Print / Check / Clear
- Goal is to be able to integrate power measurement in application performance traces tool(s) and also in resource manager accounting and profiling without performance overhead
  - Measurement and buffering is done on BMC side for several hours

Starting time →

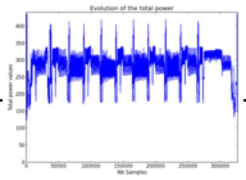
Number of values →

```
HDEEM_VERSION, 2.1.5
BMC address, localhost
==== HDEMM status ====                ,
Last polling start time            , 2015-10-02 15:22:30.268162727
Started by                       , GPIO
…
Total blade values                 , 403974
Pending blade in BMC                 , 403974
Total VR values                    , 40397
Pending VR in BMC                    , 40397
Freq blade values      (#Measure/s) , 1000
Freq vr values        (#Measure/s) , 100

Blade
----------
      1,     92.000
      2,     92.375
      3,     92.125
      4,     92.000
      5,     92.000
      6,     96.250
      7,    101.625
      8,    101.000
      9,    100.875
     10,    101.125
```

Measurement from node

- Per function analysis
  - Real test-bed on BigDFT (700 functions) +/- 6.5W on 400W



Bull
atos technologies

Power and Energy through SLURM IPMI-RAW plugin

- High Definition energy efficiency monitoring based on new FPGA architecture supported through ipmi-raw
  - Improved accuracy for both power profiling per components (100Hz) and nodes (1000Hz)
  - Improved precision for energy consumption per job based on nodes (1000Hz) measurements
  - Decrease overhead on the application (CPU and Memory) since the collection is done internally within the FPGA
- To be released in upcoming slurm version

HDEEM
High Definition Energy Efficiency Monitoring

Out-of-band collection of power/energy data

- **External Sensors** Plugin to allow out-of-band monitoring of cluster sensors
- Possibility to Capture energy usage and temperature of various components (nodes, switches, rack-doors, etc)
- Framework generic but initial support for RRD
- Plugin to be used with real wattmeters or out-of-band IPMI capturing
- Power data captured used for per node power monitoring (scontrol show node) and per job energy accounting (Slurm DB)
  - direct usage for energy reporting per job
  - but internal SLURM calculations for power
- Currently used in MontBlanc project with ARM

Accounting – Profiling
Support of multiple energy sensors

- Support for one sensor per node (until 14.11)

```
$ ipmi-sensors
62 | Power          | Current    | 175.80    | W          | 'OK'
```

- Support for multiple sensors per node (from 15.08)

```
$ipmi-sensors
85 | CPU0 Pwr           | Power Supply       | 10.00    | W     | 'OK'
86 | CPU1 Pwr           | Power Supply       | 6.00     | W     | 'OK'
87 | CPU0 DIM01 Pwr     | Power Supply       | 2.00     | W     | 'OK'
88 | CPU0 DIM23 Pwr     | Power Supply       | 0.00     | W     | 'OK'
89 | CPU1 DIM01 Pwr     | Power Supply       | 1.00     | W     | 'OK'
90 | CPU1 DIM23 Pwr     | Power Supply       | 0.00     | W     | 'OK'
91 | Blade Pwr          | Power Supply       | 112.00   | W     | 'OK'
```

User-level control of power and energy through CPU Frequency setting parameter

- Job "--cpu-freq" option now supports minimum frequency (in addition to maximum frequency and governor) and supported for salloc and sbatch (for power adaptive scheduling)
- --cpu-freq =<p1[-p2[:p3]]>
  - p1 is current options or minimum frequency
  - optional p2 is maximum
  - optional p3 is scaling governor
- New configuration parameter "CpuFreqGovernors" identifies allowed governors

Set the CPU frequency

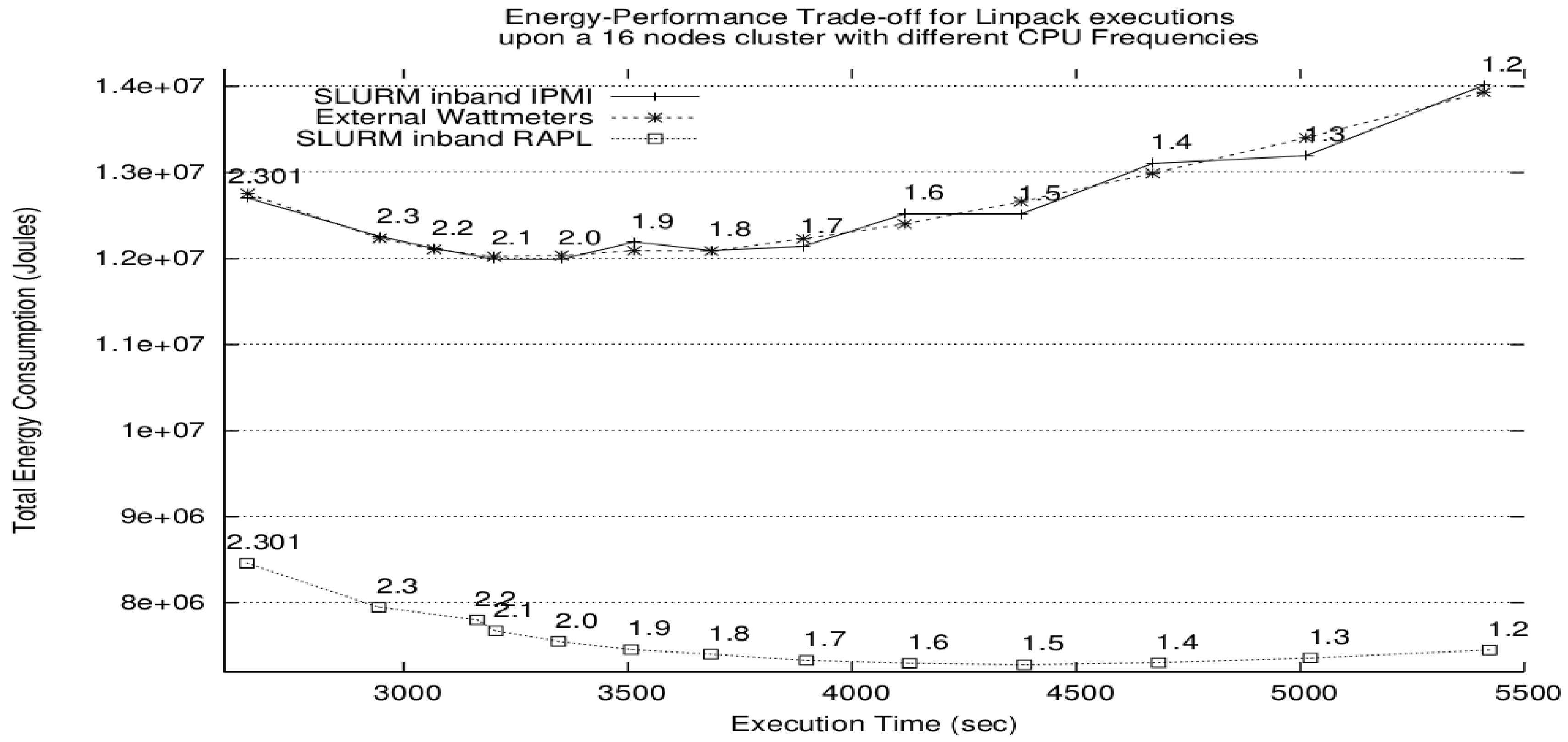
```
$# srun --cpu-freq=2700000 --resv-ports -N2 -n64 ./cg.C.64&
$# sacct -j 58 -format=jobid,elapsed,aveCPUFreq,consumedenergy
       JobID    Elapsed AveCPUFreq ConsumedEnergy
------------ ---------- ---------- --------------
          66   00:00:49    2640340          19668
```

Real CPU freq.

Job energy consumption

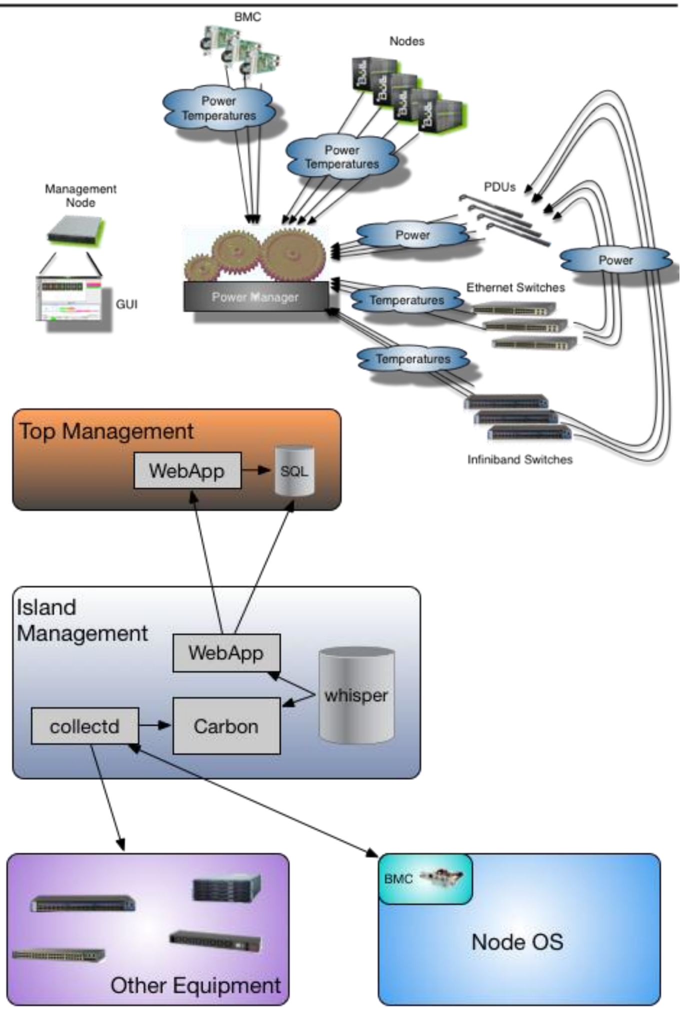
User-level: Find the good configuration…

- Using Slurm Energy Accounting to find the right trade off
- Specifying the optimal CPU using srun parameter



Ongoing Work: Global power measurement

- Data collection is, by default, done out-of-band to avoid any system noise
  - IPMI, SNMP
- Collectd is the major component for data acquisition
  - Scalable by aggregating data per island
  - Consolidation per island
  - Modified collectd to synchronize data acquisition (using NTP and epoch reference)
  - Customizable rate (default 20 sec, up to 1sec)
- Covers nodes (compute and service) but also others equipment in the system
- On-site tuning to adapt configuration



Power adaptive and Energy aware scheduling

Overview

- Power/Energy Monitoring and Control
  - Measurement System
  - Energy Accounting
  - Power Profiling
  - User level control of power and energy
- Power adaptive and Energy aware scheduling
  - User Incentives for energy aware scheduling
  - System-level control of power and energy
  - Power adaptive scheduling
- Ongoing Works and Road to Exascale
  - Dynamic Runtime Energy Optimizations
  - Towards energy budget control
  - Energy Efficiency and road to exascale

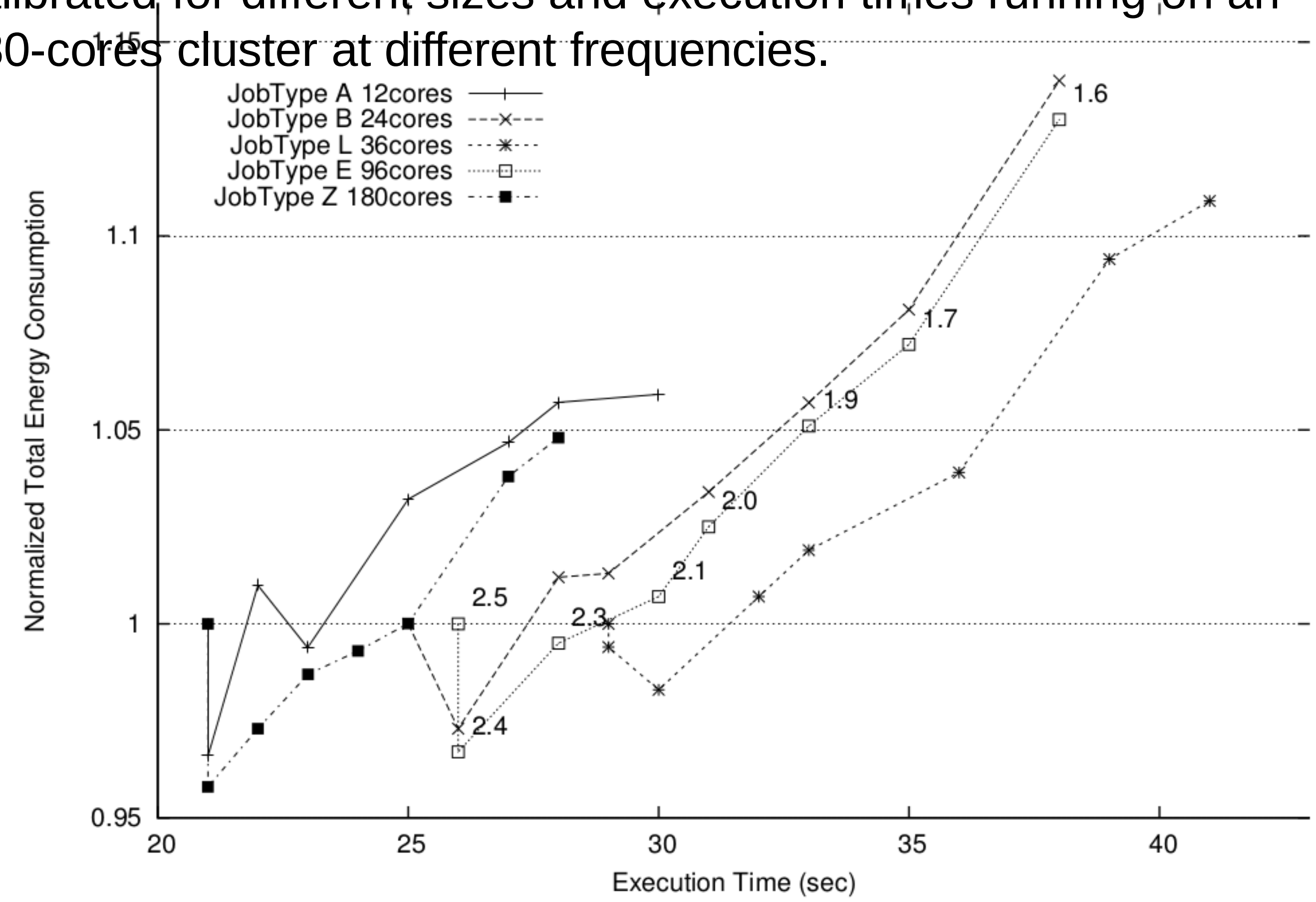
Energy Fairsharing

- ►Fairsharing is a common scheduling prioritization technique
- ►Exists in most schedulers, based on past CPU-time usage
- ►Our goal is to do it for **past energy usage**
- ►Provide **incentives to users** to be more energy efficient
    - Based upon the energy accounting mechanisms
    - Accumulate past jobs energy consumption and align that with the shares of each account
    - Implemented as a new multi-factor plugin parameter in SLURM

- ►Energy efficient users will be favored with lower stretch and waiting times in the scheduling queue

Energy Fairsharing

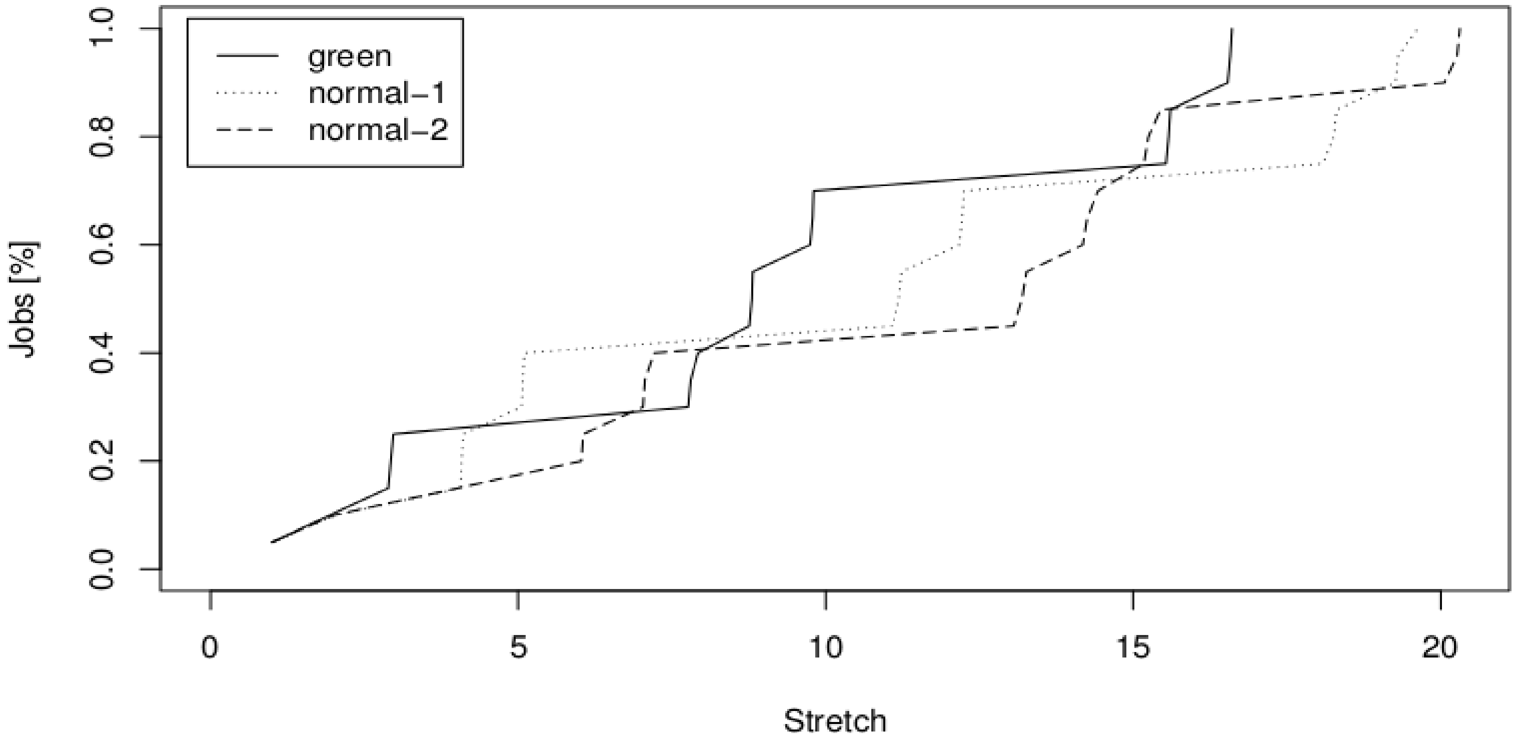
Performance vs. energy tradeoffs for Linpack applications as calibrated for different sizes and execution times running on an 180-cores cluster at different frequencies.



Yiannis Georgiou, David Glesser, Krzysztof Rzadca, Denis Trystram
A Scheduler-Level Incentive Mechanism for Energy Eciency in HPC
(In proceedings of CCGRID 2015)

Energy Fairsharing

Cumulated Distribution Function for Stretch with EnergyFairShare policy running a submission burst of 60 similar jobs with Linpack executions by 1 energy-efficient and 2 normal users (ONdemand and 2.3GHz)



Yiannis Georgiou, David Glesser, Krzysztof Rzadca, Denis Trystram
A Scheduler-Level Incentive Mechanism for Energy Eciency in HPC
(In proceedings of CCGRID 2015)

Overview

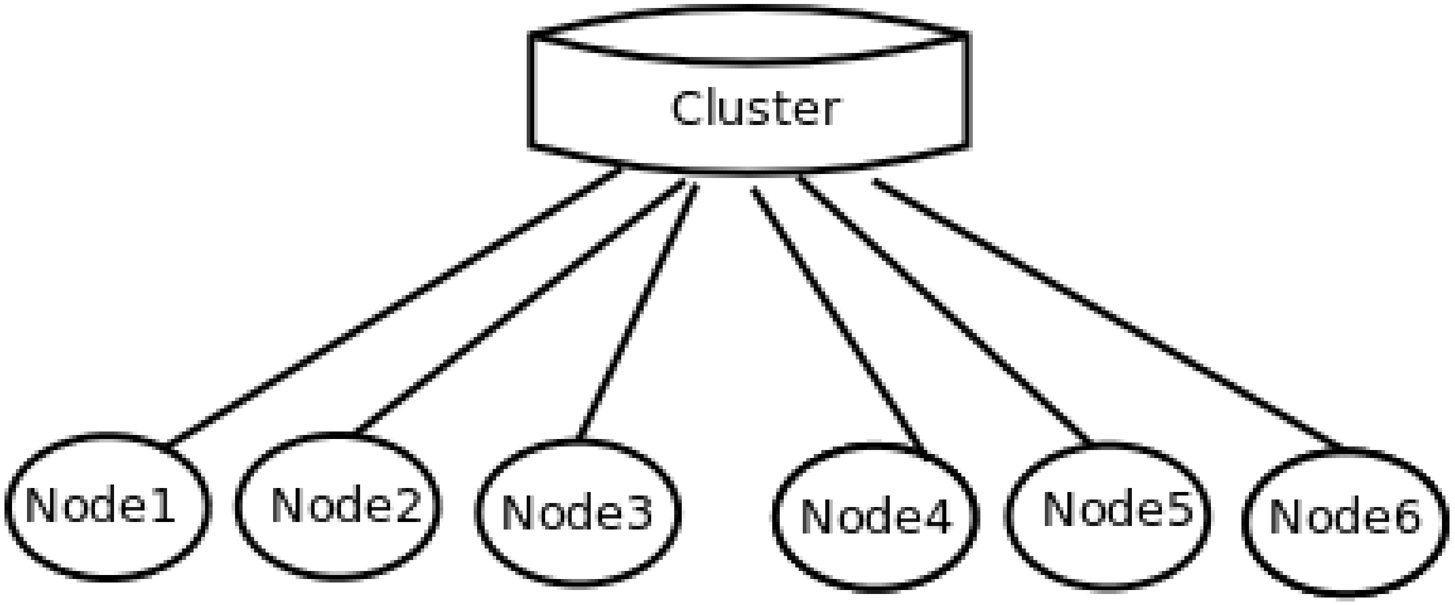
- Power/Energy Monitoring and Control
  - Measurement System
  - Energy Accounting
  - Power Profiling
  - User level control of power and energy
- Power adaptive and Energy aware scheduling
  - User Incentives for energy aware scheduling
  - System-level control framework for power and energy
  - Power adaptive scheduling
- Ongoing Works and Road to Exascale
  - Dynamic Runtime Energy Optimizations
  - Towards energy budget control
  - Energy Efficiency and road to exascale

Layouts Framework

- **Supercomputers become more complex structures**
  - Resources have a lot of characteristics that **are not** currently **taken into account** by the RJMS:
    - Power Consumption per Component, Electrical Connections, Communications roles

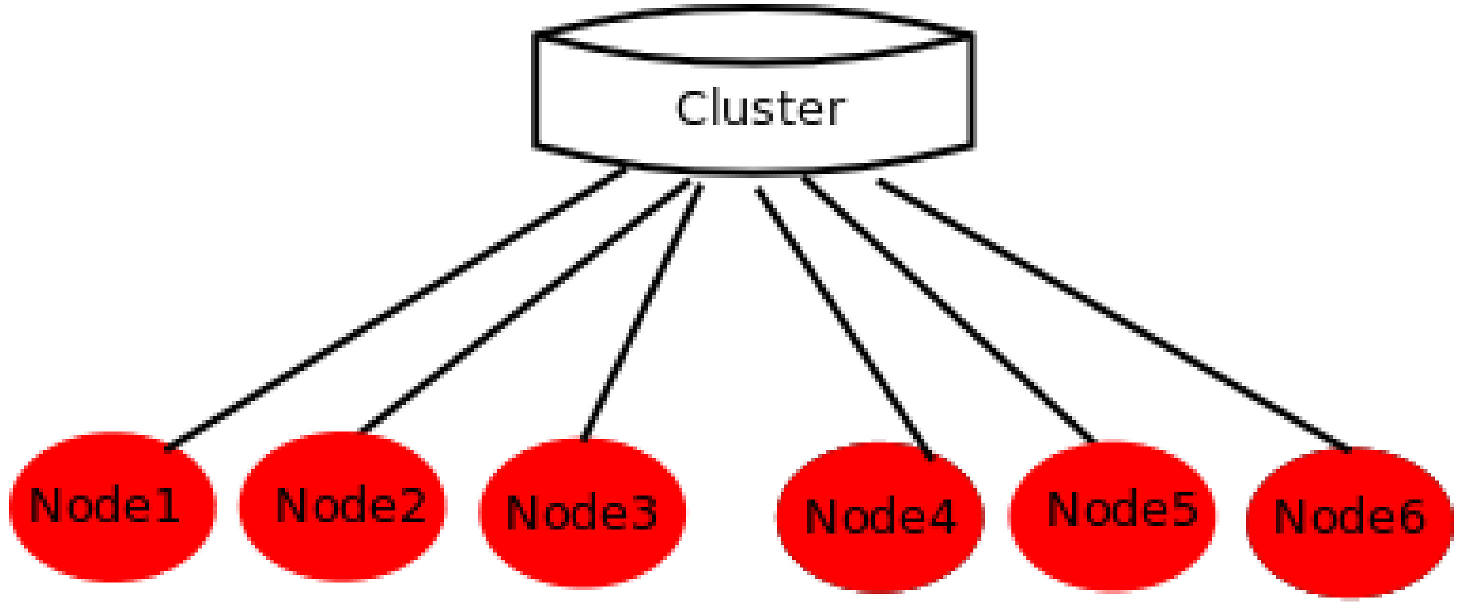
- Infrastructure characteristics may impact the way resources should be used or provided
  - Available power, cooling capacity, ...
- Those characteristics may provide **valuable information** that may be used to **optimize automatic decisions:**
  - Scheduling, Energy Efficiency, Scalability

Motivations

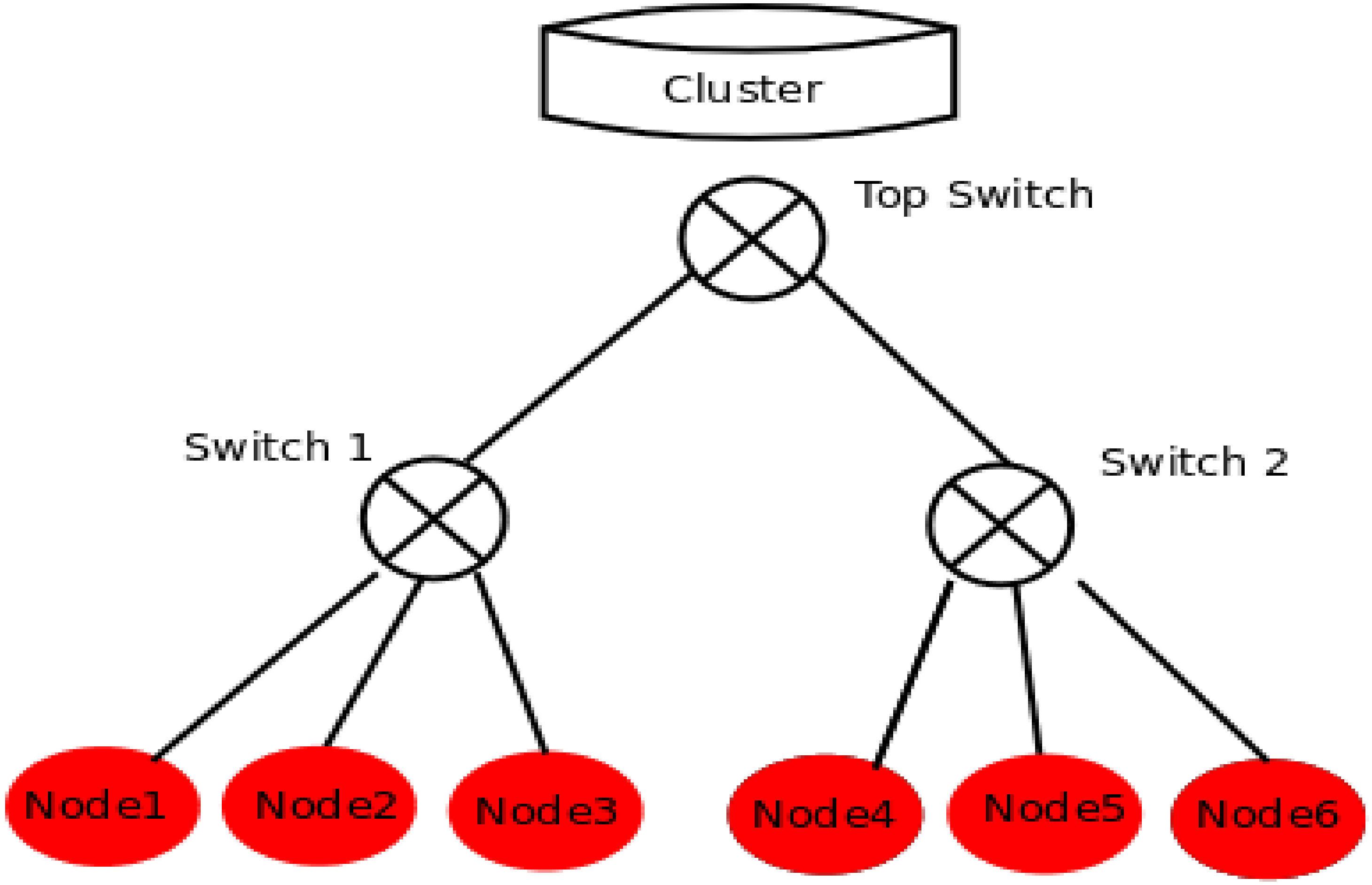


Motivations

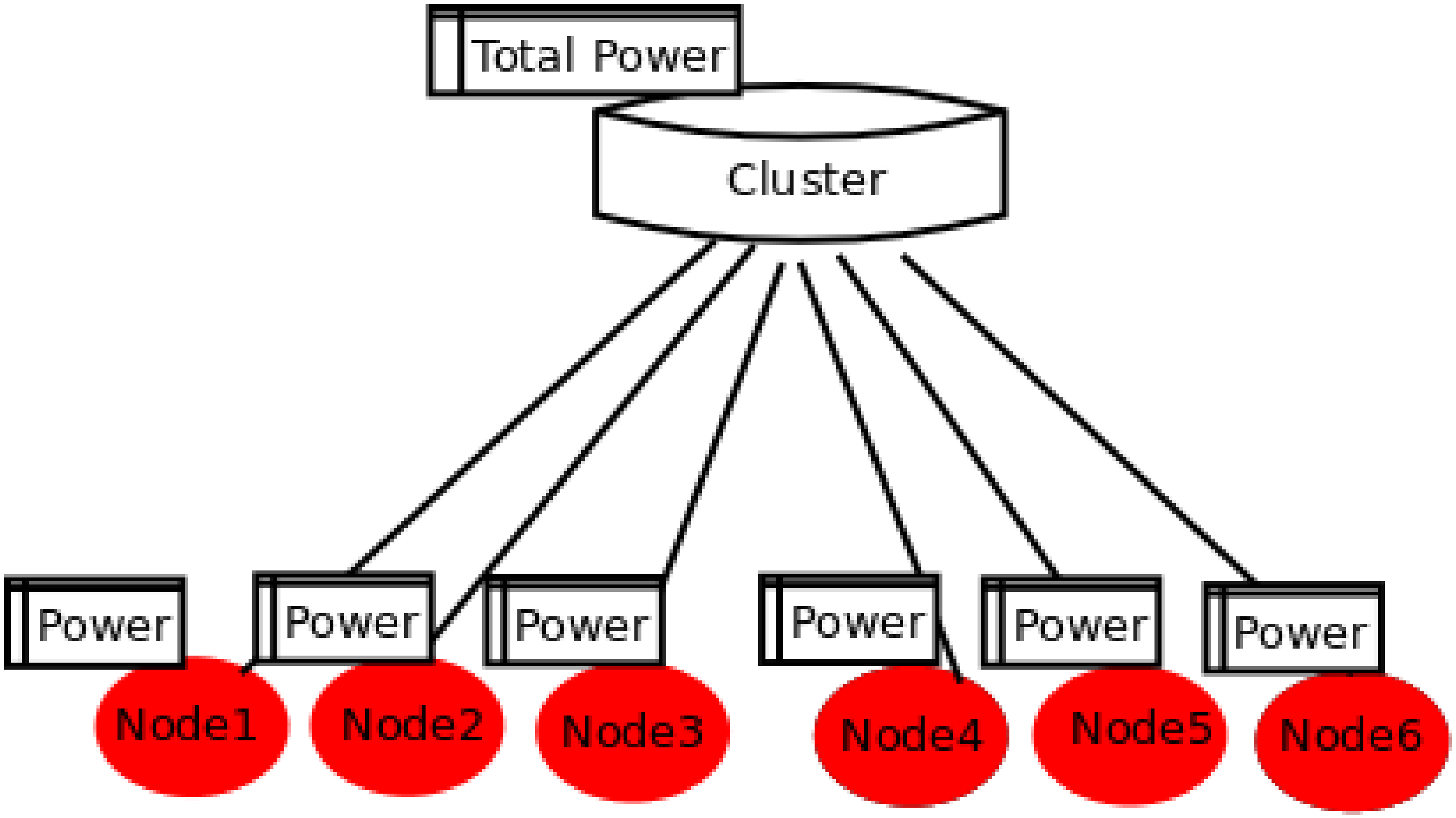
- One Cluster
    - **with multiple views**
    - of the same type of resources



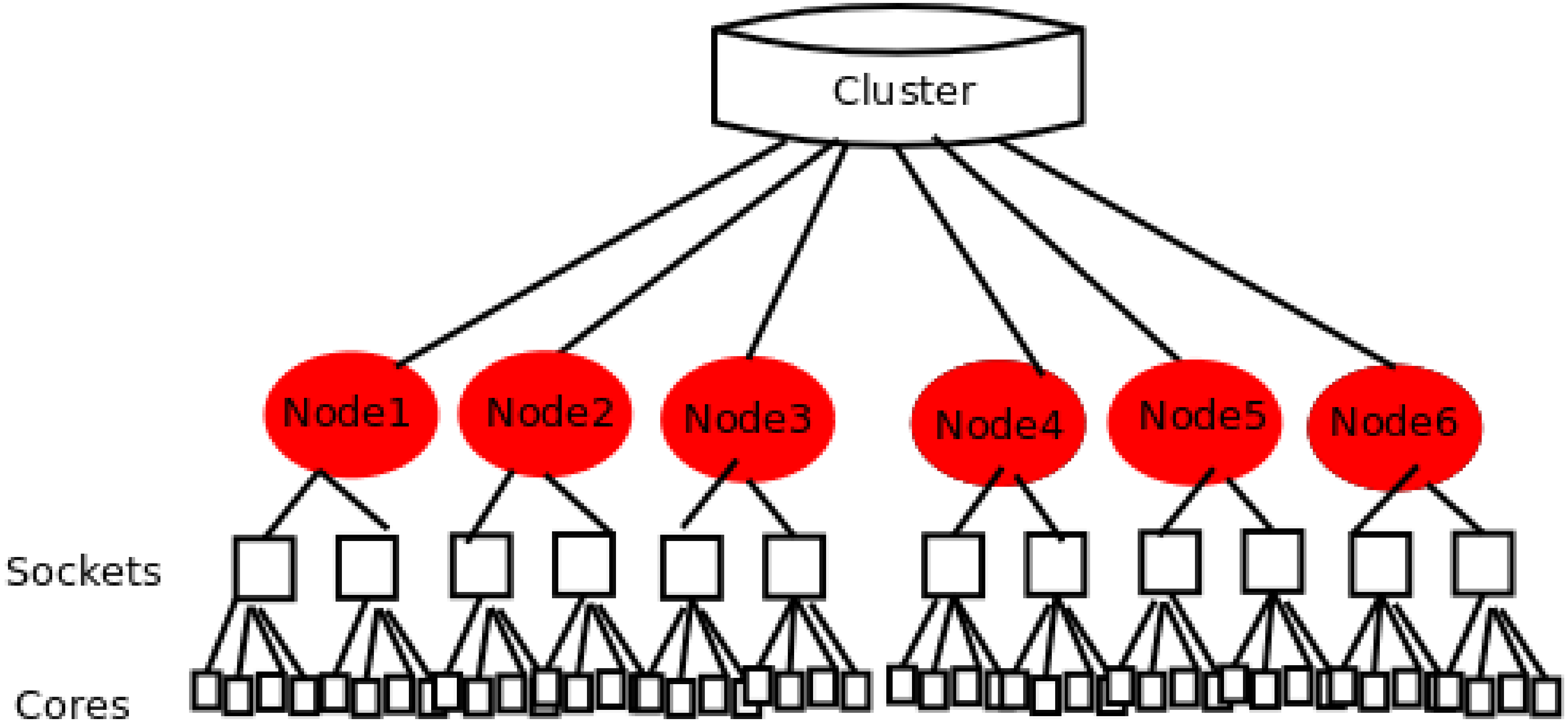
Network topology view



Power Consumption view



Consumable Resources view



Layouts Framework

- Motivations
  - Supercomputers **size and complexity** are increasing
  - Acquisition and **running costs** can/must be optimized
  - **Multiple views** of supercomputers can be leveraged

- Goals
  - Add a **generic/extensible** way to describe views of supercomputers
  - Propose views details to the resource manager for
    - Advanced **management**
    - Advanced **scheduling**
  - Ease views information update to take into account **system dynamics**

Layouts Framework

- Entities
  - Each **component of a supercomputer** can be an entity
    - A single pool of entities to manage all the components
  - Each entities can have a **set of properties** (Key-Value entries)
    - Associated to the different views
- Layouts
  - Layouts correspond to the **managed views**
    - Example: racking view, power provisioning view, ..
  - Provide a **relational logic** to link managed components
  - Provide a set of properties to enhance components information

Slurm Layouts Framework
Features added in slurm-15.08

- Read-Only Key/Value entries (Key-spec)
  - Provide a way to ask for immutable properties
    - Forbidding any update using “scontrol update layouts ...”

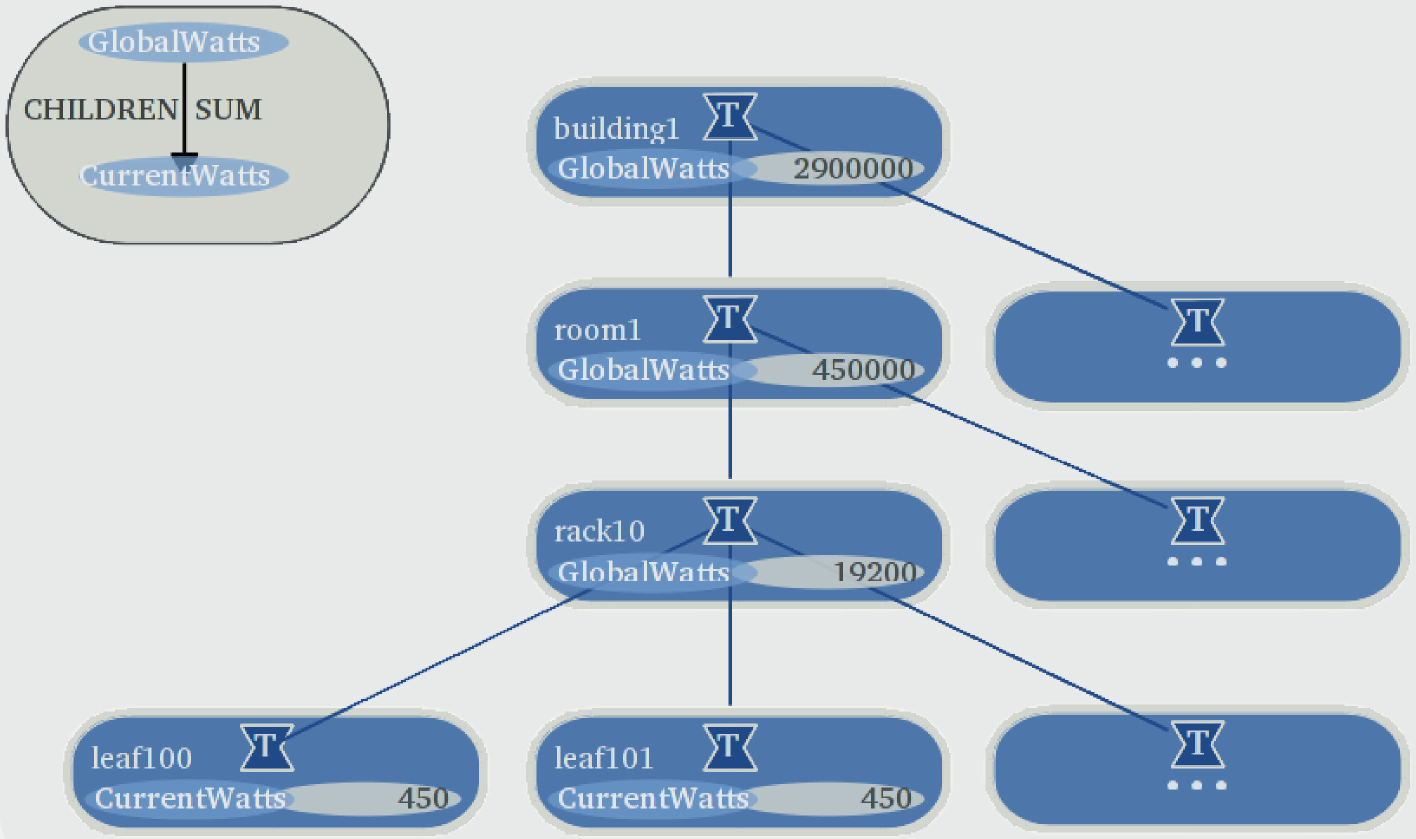
- Key/Value inheritance model
  - Define Key/Value inheritance property over a layout relation model
    - Tree based only right now
    - Examples of inheritance properties (mutually exclusive)
      - CHILDREN_SUM / CHILDREN_{MIN,MAX,AVG} / CHILDREN_COUNT
      - PARENTS_SUM / PARENTS_{MIN,MAX,AVG} / PARENTS_FSHARE

Slurm Layouts Framework
Features added in slurm-15.08

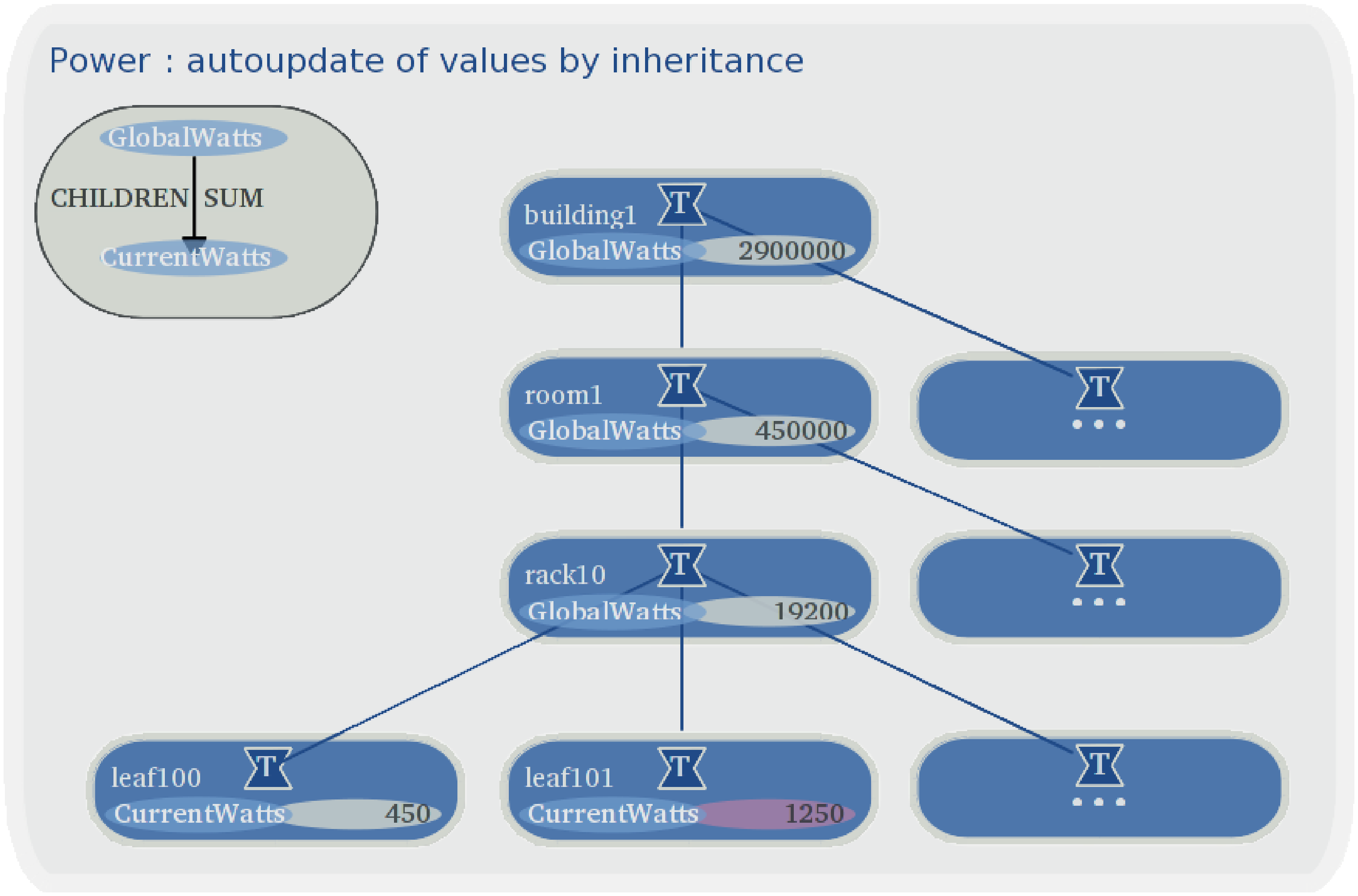
- **Key/Value Automatic Updates**
  - Leverage Key/Value **inheritance model** to provide global layout consistency
    - Automatically **updates** of the entity's neighborhood
    - Ensure **consistency** of the internal representation after any modification
  - Selected logic: 2 consecutive stages
    - Start with Top-Down Parents Inheritances
    - Followed by Bottom-Up Children Inheritances
  - Optional
    - Layout plugins have to ask for “autoupdate” support for that
    - Can have **performance penalties** for very large layouts
      - Need more evaluations / improvements

Slurm Layouts Framework
Example of Automatic Update

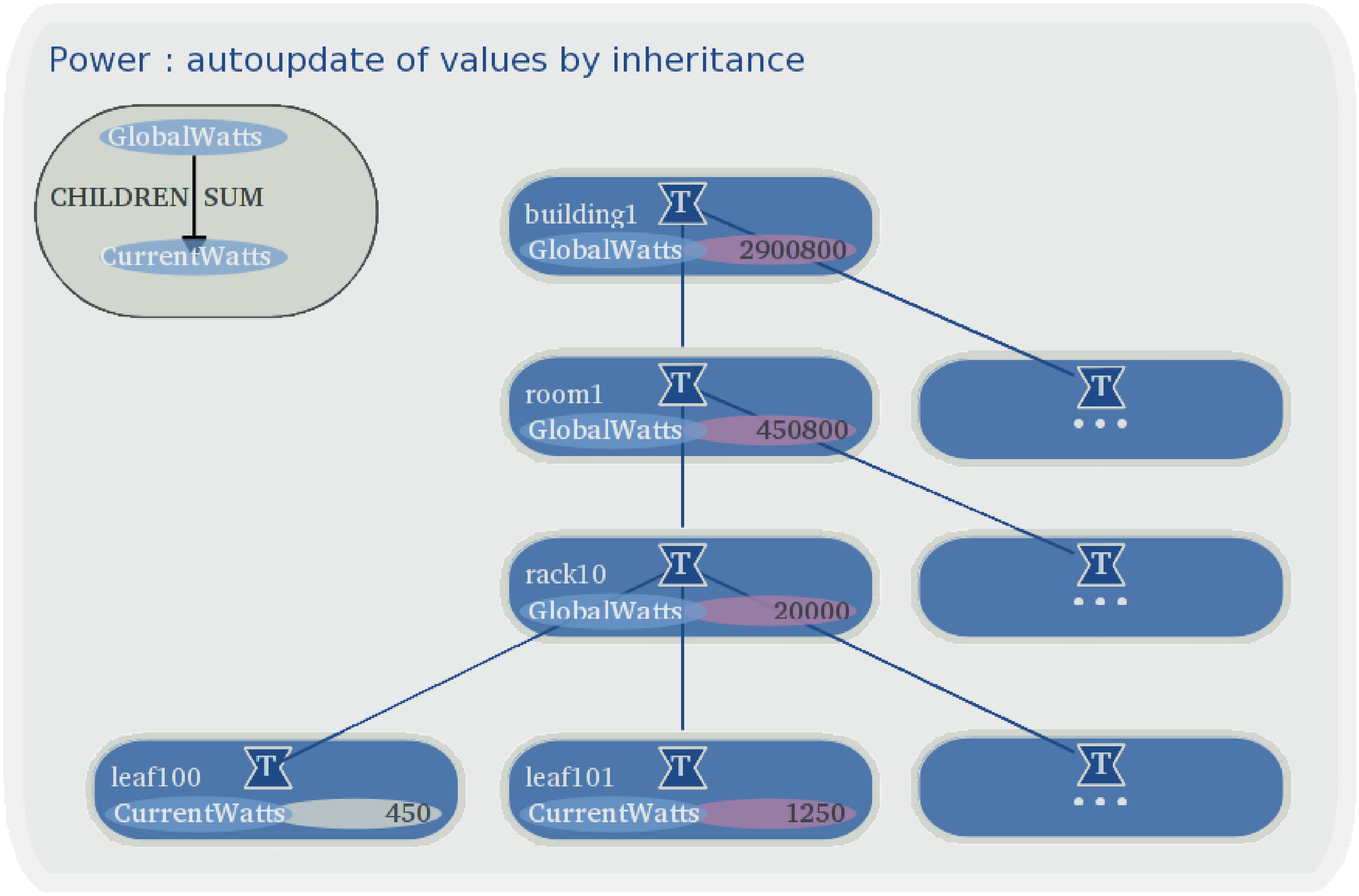
Power : autoupdate of values by inheritance



Slurm Layouts Framework
Example of Automatic Update



Slurm Layouts Framework
Example of Automatic Update



Overview

- Power/Energy Monitoring and Control
  - Measurement System
  - Energy Accounting
  - Power Profiling
  - User level control of power and energy
- Power adaptive and Energy aware scheduling
  - User Incentives for energy aware scheduling
  - System-level control framework for power and energy
  - Power adaptive scheduling
- Ongoing Works and Road to Exascale
  - Dynamic Runtime Energy Optimizations
  - Towards energy budget control
  - Energy Efficiency and road to exascale

Power adaptive scheduling

- ►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

►The implementation appeared in Slurm v15.08 has the following characteristics:

►Based upon layouts framework

-for internal represantation of resources power
-Only logical/static represantation of power
-Fine granularity down to cores

►Power 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
```

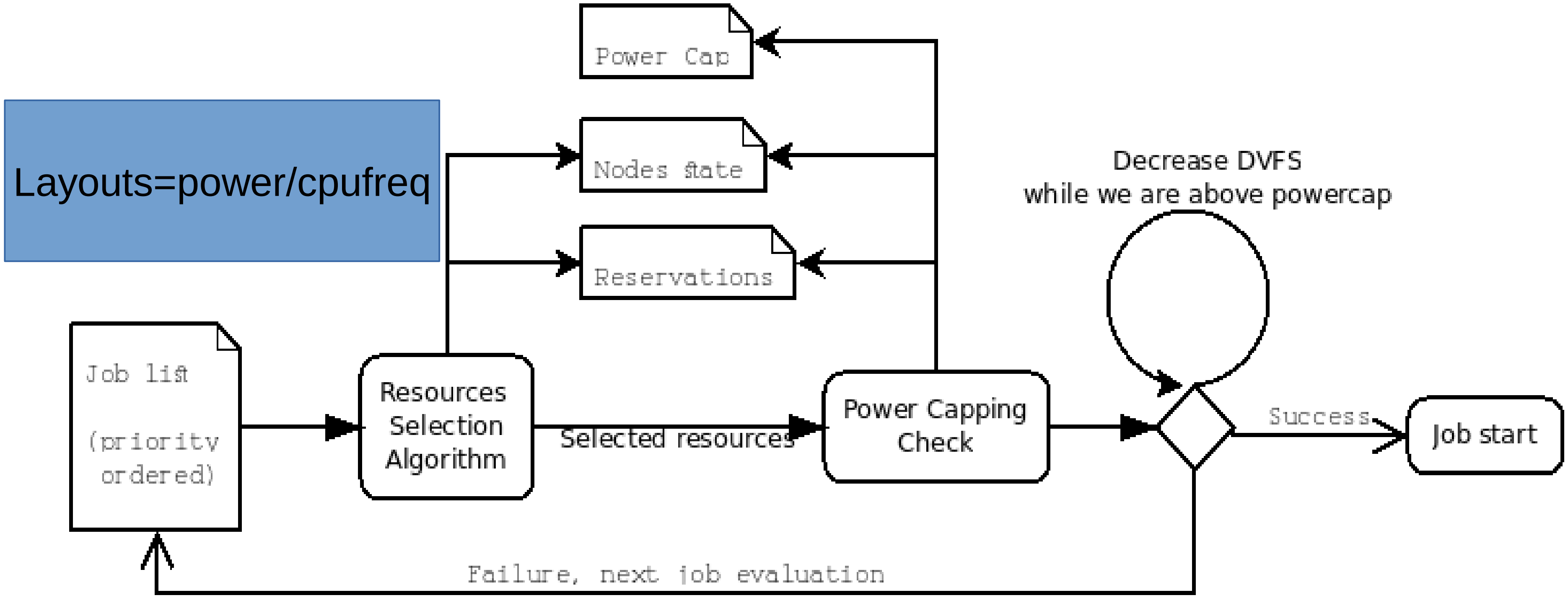
▸In advance with watts reservation (scontrol create

```
[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
```

Power adaptive scheduling – algorithm extended version -


Power Cap
Nodes state
Reservations
Layouts=power/cpufreq
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

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
```

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
```

Layouts Power code structure (truncated)

- src/layouts/power/default.c -

```
[root@nd25 slurm]#vi src/common/job_resources.c
…
extern int adapt_layouts(job_resources_t *job_resrcs_ptr,...
...
      layouts_entity_get_mkv("power", node_name,
                        "CoresCount,LastCore", data,
                        (sizeof(uint32_t)*2),L_T_UINT32);

 for (i = 0; i < core_cnt; i++) {
            /*core_num=LastCore+1-CoresCount*/
            core_num = data[1] + 1 - data[0] + i;
            sprintf(ename, "virtualcore%u", core_num);

layouts_entity_get_mkv("power", ename,
                              "CurrentCorePower,IdleCoreWatts",
                              vals,
                              (sizeof(uint32_t)*2) ,L_T_UINT32);
                if (new_value) {
                        if (vals[0] == 0) {
                              layouts_entity_set_kv(
                                          "power",
                                          ename,
                                          "CurrentCorePower",
                                          &vals[1],
                                          L_T_UINT32);
```

« get_mkv »
API functions
examples

« set_kv »
API functions
examples

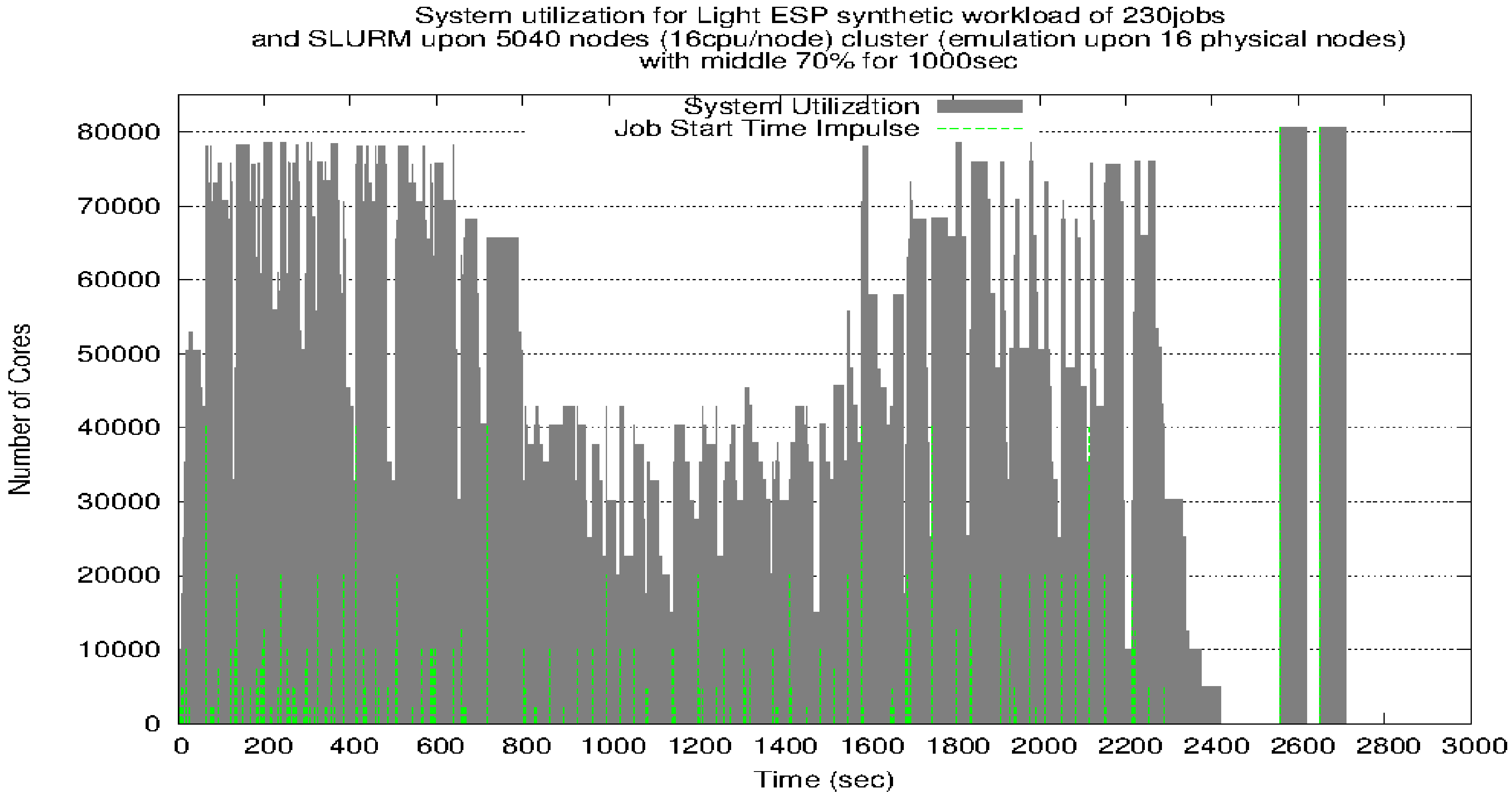
Experiments Testbed

- ►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

Bull
atos technologies

Power adaptive scheduling validation

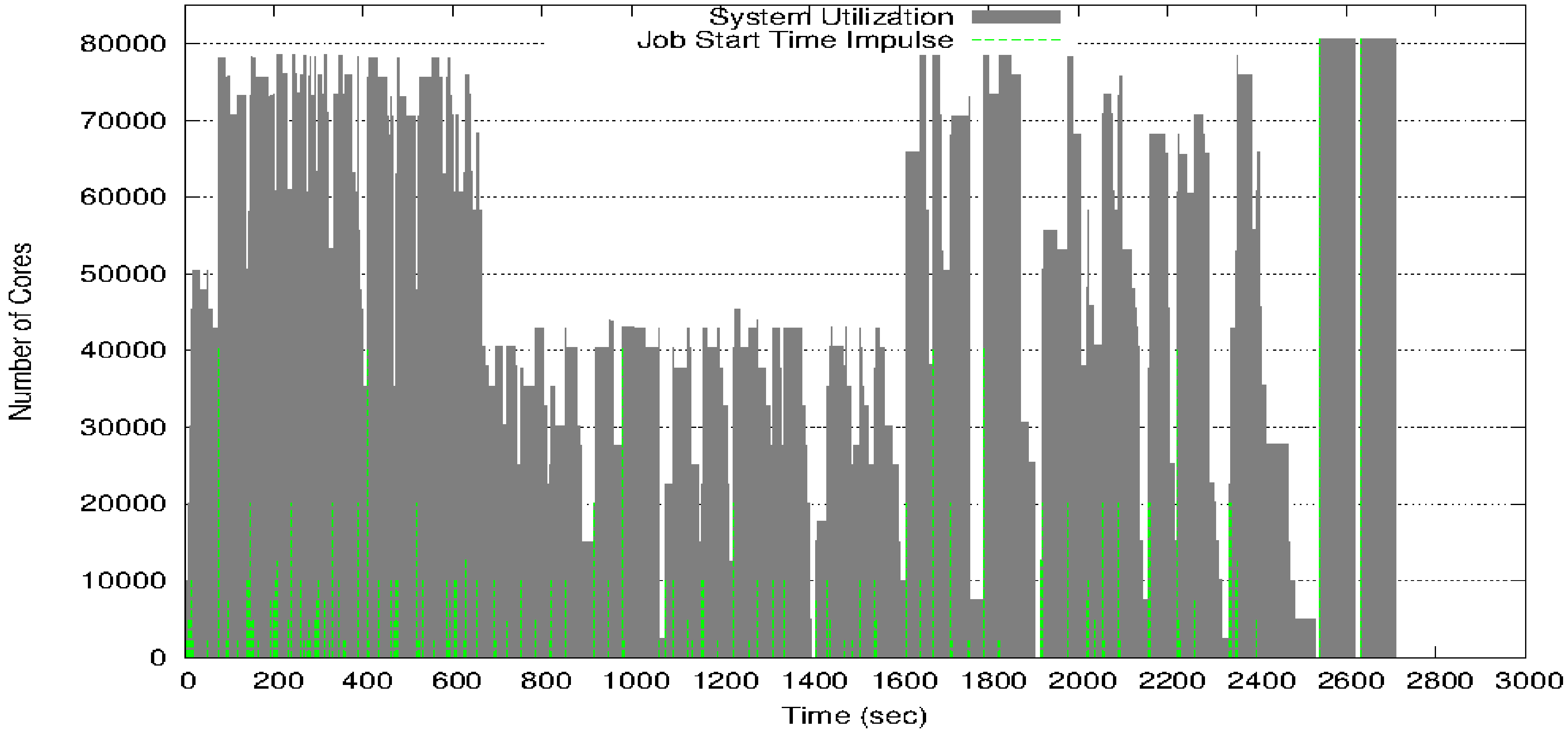
– powercap set on-the-fly with scontrol update-



Power adaptive scheduling validation

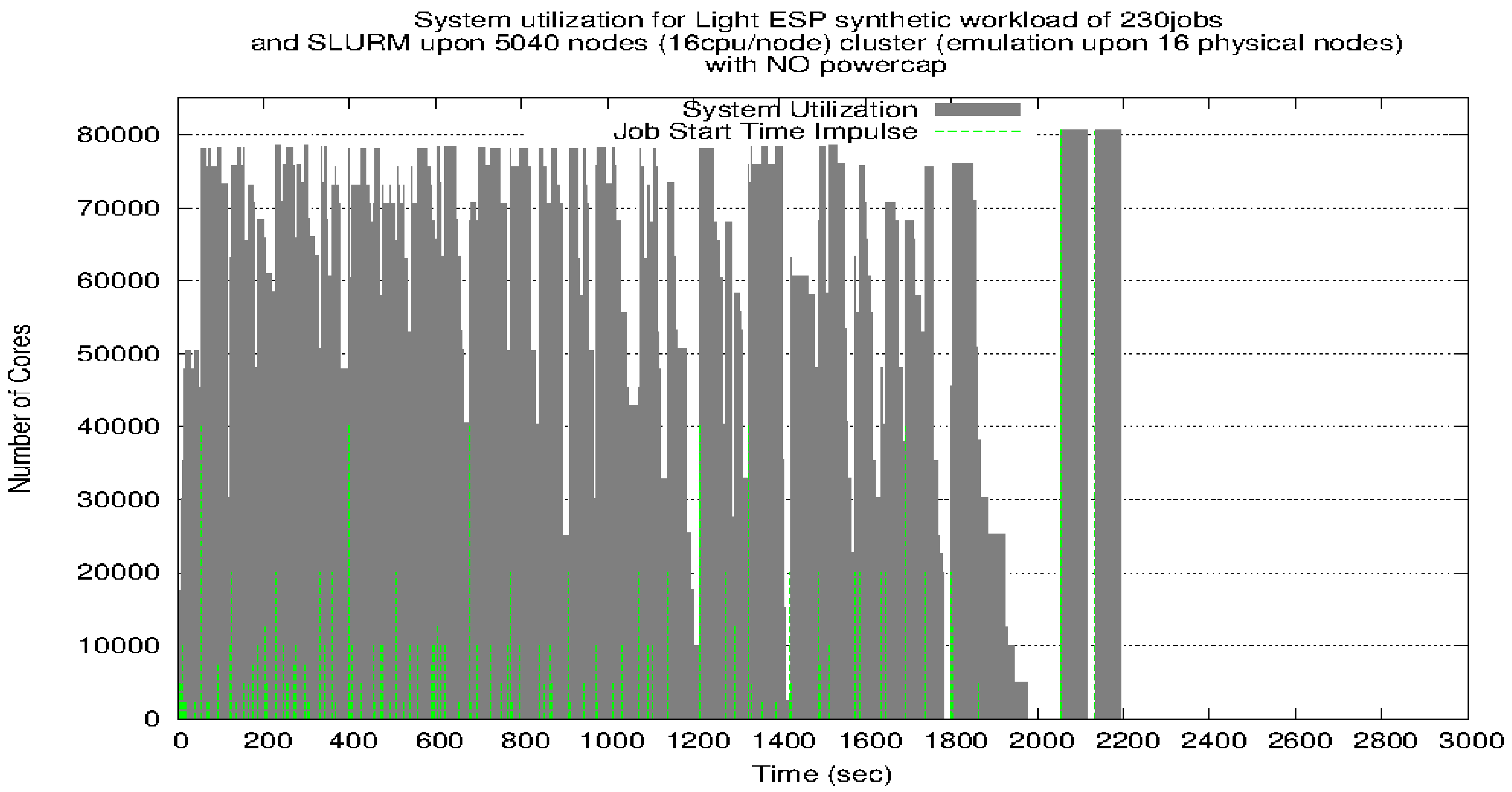
– powercap set in advance with reservation -

System utilization for Light ESP synthetic workload of 230jobs
and SLURM upon 5040 nodes (16cpu/node) cluster (emulation upon 16 physical nodes)
with middle 70% powercap reservation for 1000sec



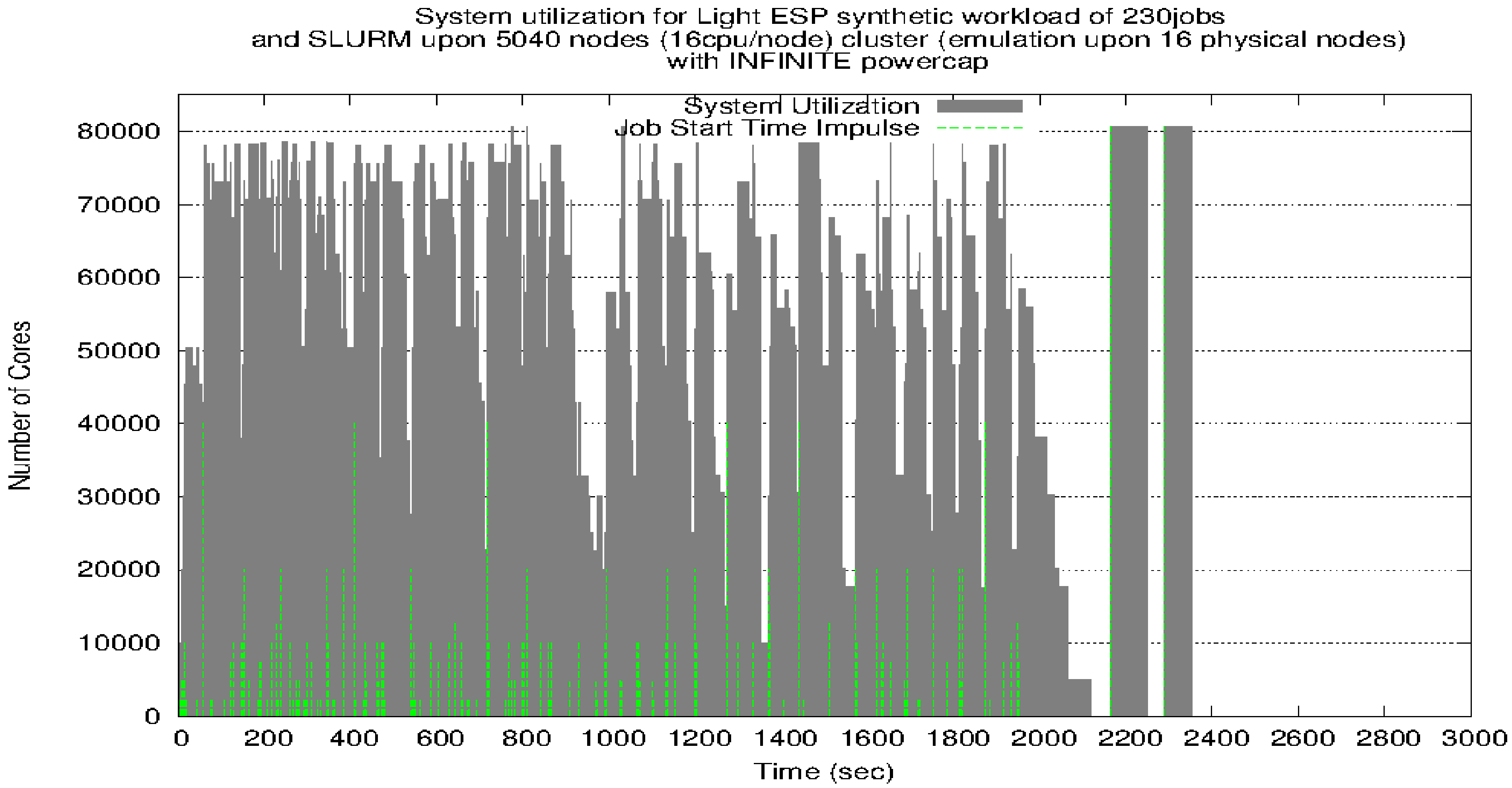
Power adaptive scheduling – scaling validation

– No powercap set -



Power adaptive scheduling scaling validation

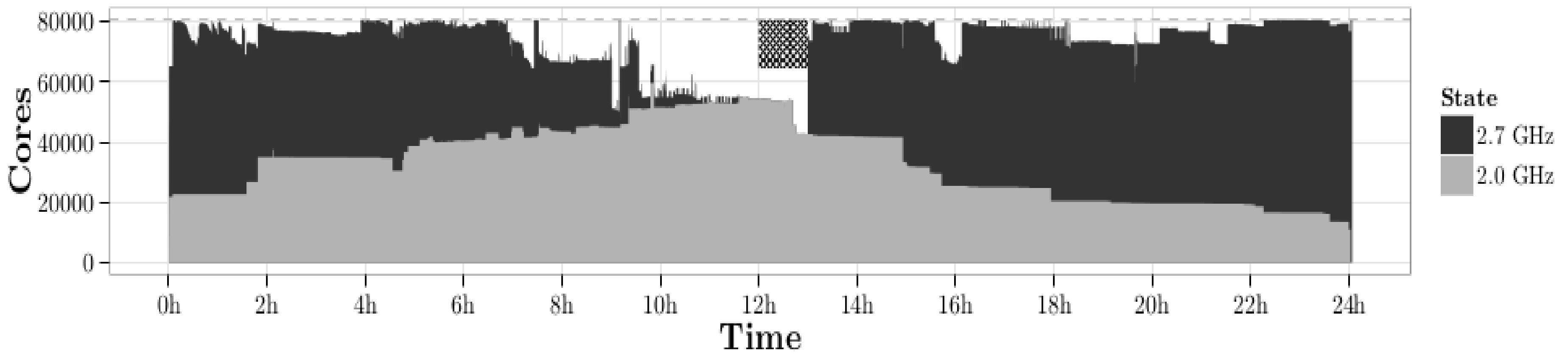
– With powercap INFINITE -

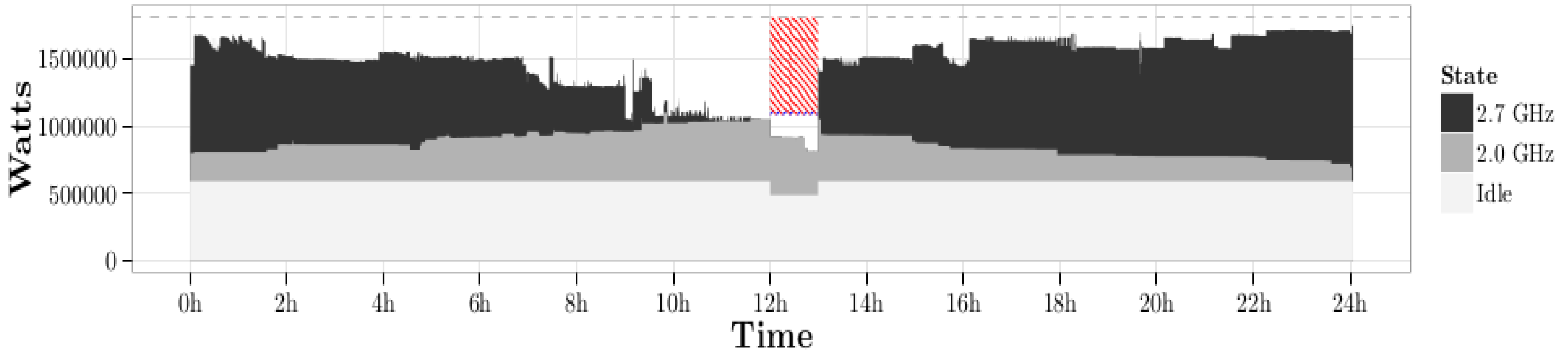


Discussion

- ►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



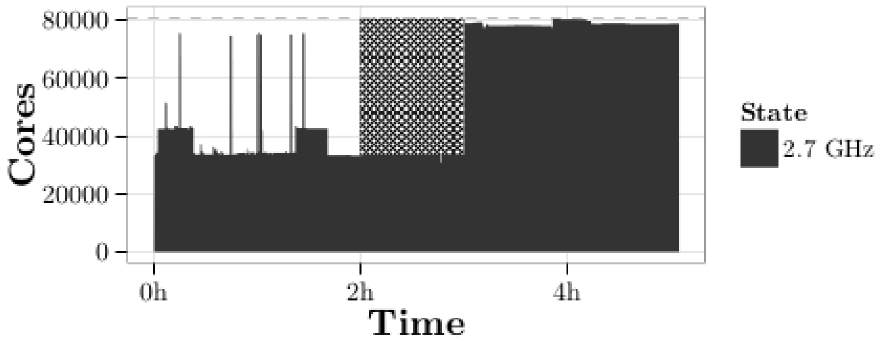


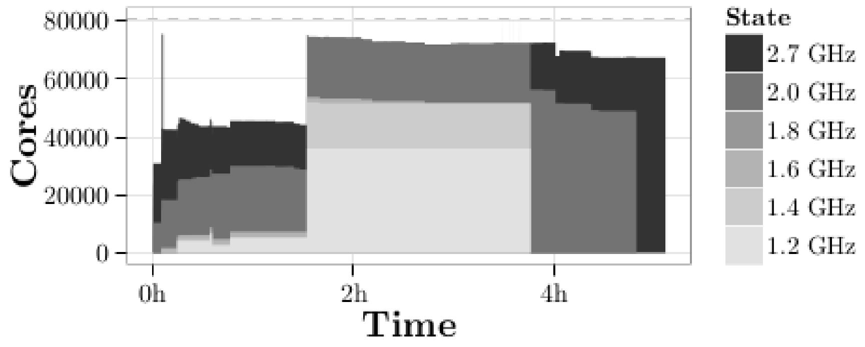
Yiannis Georgiou, David Glesser, Denis Trystram
Adaptive Resource and Job Management for limited power consumption
In proceedings of IPDPS-HPPAC 2015

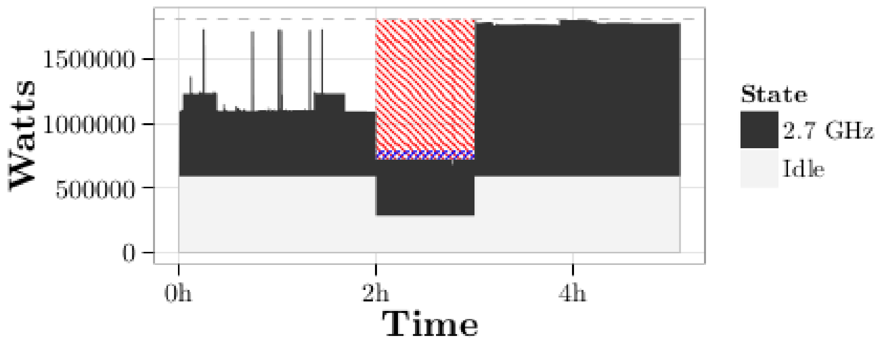
Power adaptive scheduling

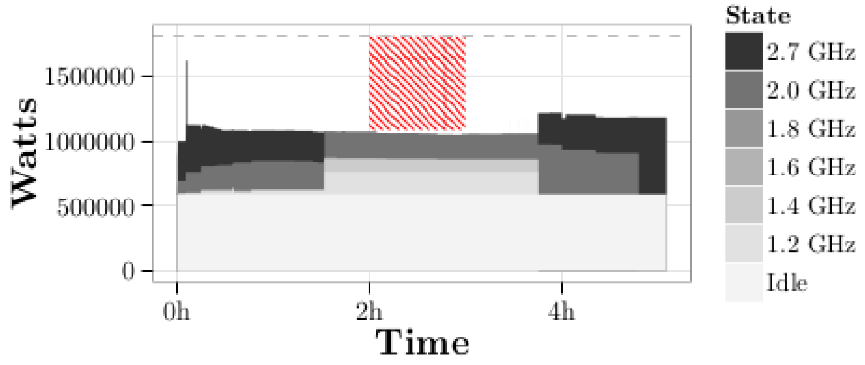
Powercap of 60% with mainly big jobs and SHUT policy

Powercap of 40% with mainly small jobs and DVFS policy









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

Ongoing 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

Ongoing Works and Road to Exascale

Overview

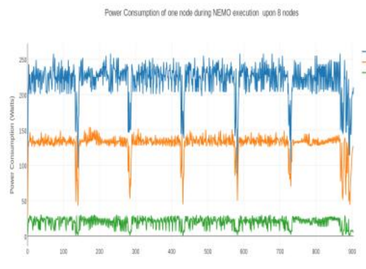
- Power/Energy Monitoring and Control
  - Measurement System
  - Energy Accounting
  - Power Profiling
  - User level control of power and energy
- Power adaptive and Energy aware scheduling
  - User Incentives for energy aware scheduling
  - System-level control of power and energy
  - Power adaptive scheduling
- Ongoing Works and Road to Exascale
  - Dynamic Runtime Energy Optimizations
  - Towards energy budget control
  - Energy Efficiency and road to exascale

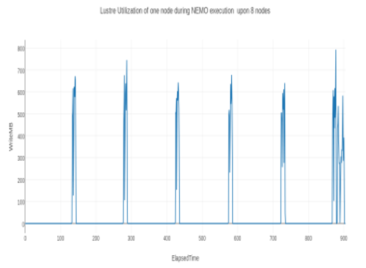
Bull
atos technologies

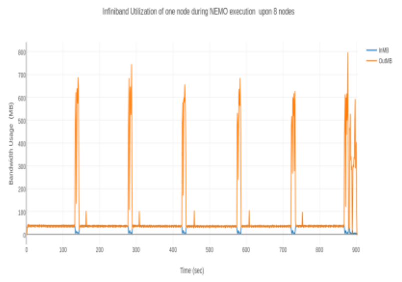
Dynamic Runtime Energy Optimizations

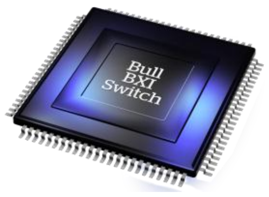
►**Goal: to reduce the energy consumption of a single application during execution.**

1. Based on initial application profiling which reveal the different phases of the application (compute, communication, IO, etc)
2. Enable dynamic runtime energy optimizations by triggering adapted actions based on the application phase (i.e. CPU, GPU, BXI reconfigurations for lower power)









Overview

- Power/Energy Monitoring and Control
  - Measurement System
  - Energy Accounting
  - Power Profiling
  - User level control of power and energy
- Power adaptive and Energy aware scheduling
  - User Incentives for energy aware scheduling
  - System-level control of power and energy
  - Power adaptive scheduling
- Ongoing Works and Road to Exascale
  - Dynamic Runtime Energy Optimizations
  - Towards energy budget control
  - Energy Efficiency and road to exascale

Bull
atos technologies

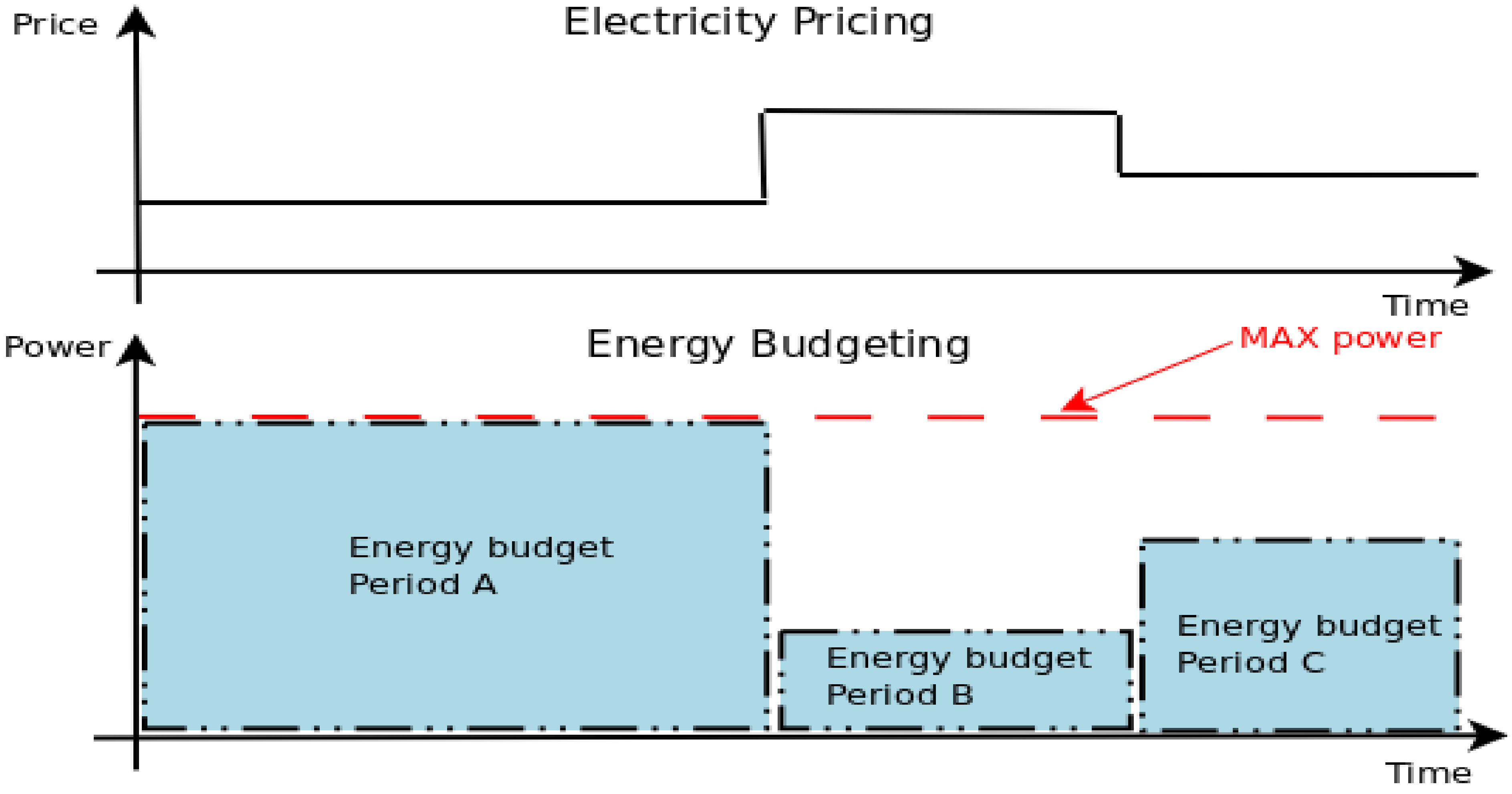
Towards Energy Budget Control

- **Background:** Scheduling with powercap (introduced in slurm 15.08) deals with maximum power
  - avoid fines from electricity provider
  - no intelligence for power usage throughout time (= energy)
- **Proposed optimization:** Scheduling with energycap enables to better align the energetic budget to the variation of electricity prices (schedule more jobs when low price)

- EnergyCap Scheduling
  - Schedule jobs under particular energetic budgets for variable time durations.
  - Extension of powercapping with the difference that we are interested to adapt the power consumption in a way that the final energy consumption of the particular time duration remains below the allowed energetic budget.

  - The actual energy consumption reductions take place through coordinated techniques such as:
    - Dynamic CPU - Frequency scaling
    - Hardware power-capping (RAPL, Sequana-blade)
    - Keeping nodes idle
    - Shut-down nodes

Towards Energy Budget Control



Overview

- Power/Energy Monitoring and Control
  - Measurement System
  - Energy Accounting
  - Power Profiling
  - User level control of power and energy
- Power adaptive and Energy aware scheduling
  - User Incentives for energy aware scheduling
  - System-level control of power and energy
  - Power adaptive scheduling
- Ongoing Works and Road to Exascale
  - Dynamic Runtime Energy Optimizations
  - Towards energy budget control
  - Energy Efficiency and road to exascale

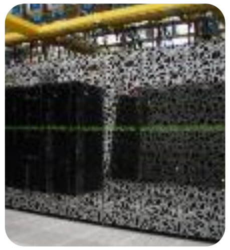
BULL current largest HPC supercomputers

CURIE - 2011
1st PRACE Petascale supercomputer
Intel E5 "Early Bird"
150 GB/s Lustre
2 PFlops peak

OCCIGEN 2015
TIER0 Supercomputer, CINES
DLC technology
2.1 PFlops peak
250 + 300GB/s – Lustre & DMF

TAURUS 2013-2014
1st BULL PetaFlops Supercomputer in Germany
1 PFlops peak
Lustre

DKRZ 2014-2016
Climate research
DLC technology
3 PFlops
45 PB @ 480 GB/s
Lustre + HPSS



CARTESIUS 2013-2014
1st Bull Petascale Supercomputer in Netherland
DLC technology
1.3 PFlops
8PB @ 220 GB/s - Lustre

HELIOS 2011-2014
ITER Community
1.7PFlops peak
X86 + PHI
+100GB/s – Lustre

BEAUFIX PROLIX 2013-2014-2015
1st Intel E5 v3 supercomputer in production ww
DLC technology
1 PFlops peak
Extension to 5 PFlops in 2016

SANTOS DUMONT 2015
Largest supercomputer in Latin America
DLC Technology
1 PFlops peak
Mobull

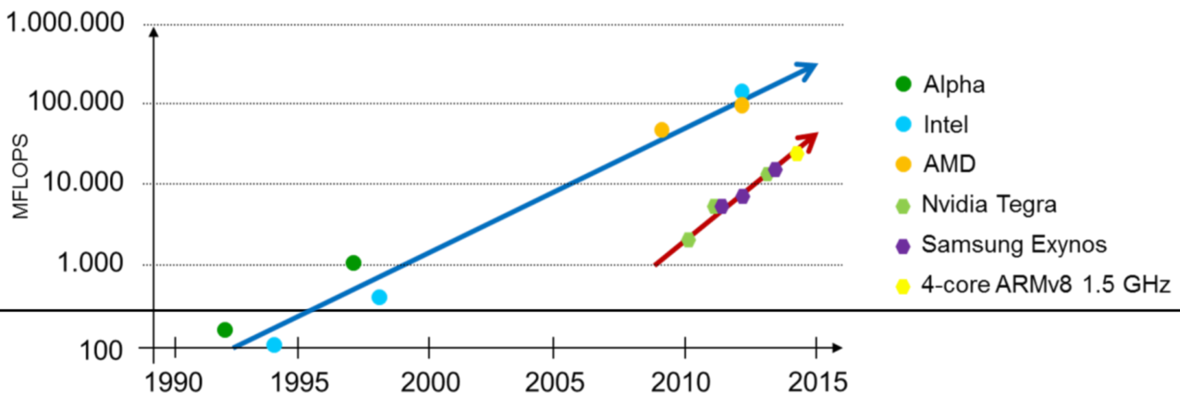
Bull sequana
the Bull exascale generation of supercomputer

- **Open and modular platform designed for the long-term**
  - ➢ To integrate current and future technologies
  - ➢ Multiple compute nodes: Xeon-EP, Xeon Phi, Nvidia GPUs, other architectures…
- **Scales up to tens of thousands of nodes**
  - ➢ Large building blocks to facilitate scaling
  - ➢ Large systems with DLC: 250-64k nodes
- **Embedding the fastest interconnects**
  - ➢ Multiple Interconnects: BXI, InfiniBand EDR-HDR
  - ➢ Optimized interconnect topology for large basic cell / DLC (288 nodes)
  - ➢ Fully non-blocking within Cell
- **Ultra-energy efficient**
  - ➢ Enhanced DLC – up to 40°C for inlet water and ~100% DLC

Bull
atos technologies

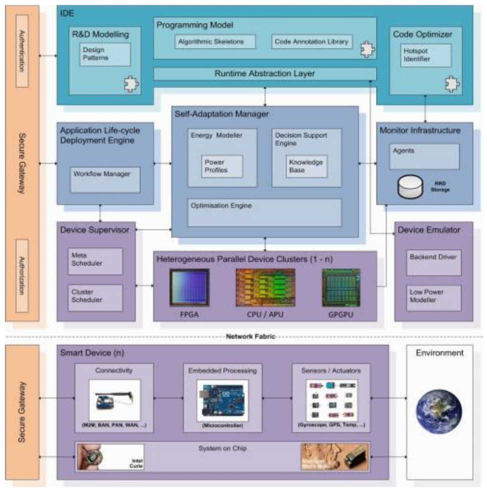
MontBlanc project targets pre-exascale systems for 2020

- BULL leading the MontBlanc project
- European approach towards energy efficient high performance
- To design a well-balanced architecture_and to deliver the design for an ARM based SoC or SoP (System on Package) capable of providing pre-exascale performance
- To introduce new high-end ARM core and accelerators implementations to efficiently support HPC applications.
- To develop the necessary software ecosystem for the future SoC.



Considering energy efficiency from code design to execution for heterogeneous architectures

- Atos/Bull leading the Tango project (Transparent Heterogeneous hardware Architecture deployment for eNergy Gain in Operation)
- Extension of currently available programming models and resource and job management systems to support complex heterogeneous architectures
- Code optimizer engine with the aim of optimizing code mapping. to reduce power consumption by the application.
- Power-awareness integrated in the whole software development optimization and execution process



Karim Djemame, Django Armstrong, Richard E. Kavanagh, Jean-Christophe Deprez, Ana Juan Ferrer, David Garcia Perez,Rosa M. Badia, Raúl Sirvent, Jorge Ejarque, Yiannis Georgiou: **TANGO: Transparent heterogeneous hardware Architecture deployment for eNergy Gain in Operation.** CoRRabs/1603.01407 (2016)

Thanks

yiannis.georgiou@atos.net

Bull
atos technologies