Integrating Layouts Framework in SLURM

Slurm 2014 User Group

Thomas Cadeau, Bull
Yiannis Georgiou, Bull
Matthieu Hautreux, CEA

Motivations

- **The RJMS needs a way to integrate additional resources related information easily**
  - Ease the addition and usage of new information when necessary
  - Ease the integration and management of new type of resources
  - Ease the maintenance of the code

- **Layout Framework ?**
  - An answer to this problematic within SLURM

Goals

- **Describe the components of a supercomputer**
  - Generic notion of **« entity »** for each component
  - An entity has a key-value set associated to carry useful information
  - A single pool of « entities » represents the system

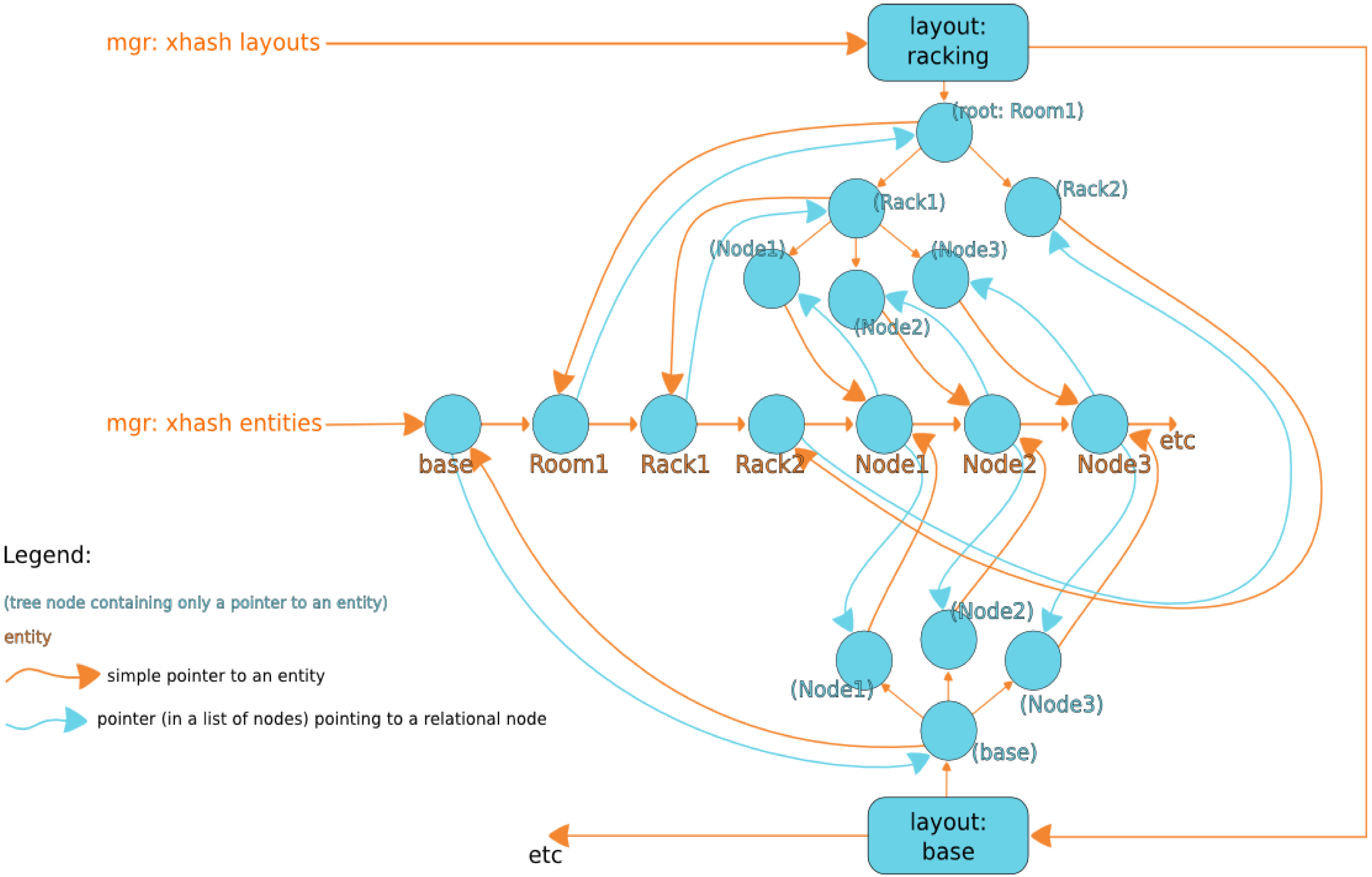
- **Describe relations between components**
  - Generic notion of << layout >>
    - every aspect of a cluster can have a dedicated « layout »
  - Federating a set of entities using a relational structure (Tree,Multi-Tree?)
  - Enhancing its federated « entities » from its aspect details (key-value entities)
  - Multiple layouts for multiple aspects / views
    - Federating entities from a common pool

Current Status

- **Core logic of the framework:**
  - CEA and Bull work
  - Already in slurm-14-11

- **Integration in Slurm:**
  - Set of API functions
  - scontrol commands
  - Implement a first set of example layouts
  - First integration of a layout
    - Power capping

Layouts implementation



mgr: xhash layouts

layout: racking

(root: Room1)

(Rack1)

(Rack2)

(Node1)

(Node2)

(Node3)

mgr: xhash entities

base

Room1

Rack1

Rack2

Node1

Node2

Node3

etc

(Node2)

(Node1)

(Node3)

(base)

layout: base

etc

Legend:

(tree node containing only a pointer to an entity)

entity

simple pointer to an entity

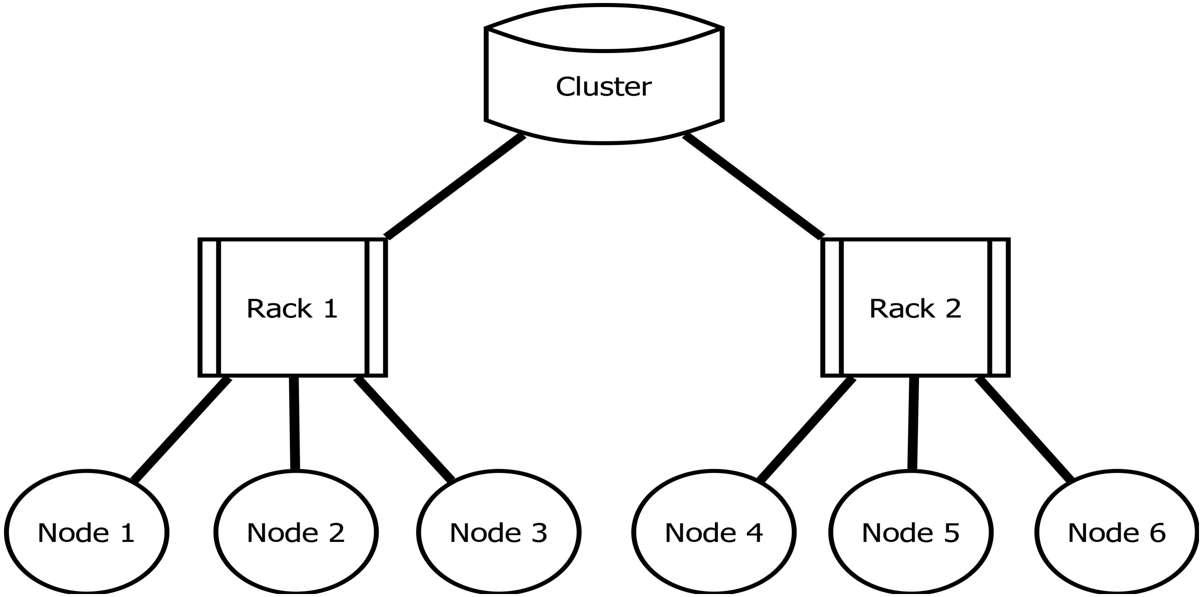
pointer (in a list of nodes) pointing to a relational node

A (very) simple layout

```
Priority=10
Root=Cluster

Entity=Cluster Type=Cluster Enclosed=Rack[1-2] CurrentPower=200
Entity=Rack1 Type=Rack Enclosed=Node[1-3] CurrentPower=200
Entity=Rack2 Type=Rack Enclosed=Node[4-6] CurrentPower=200
Entity=Node[1-6] Type=Node CurrentPower=0 Frequency=0
```

A (very) simple layout

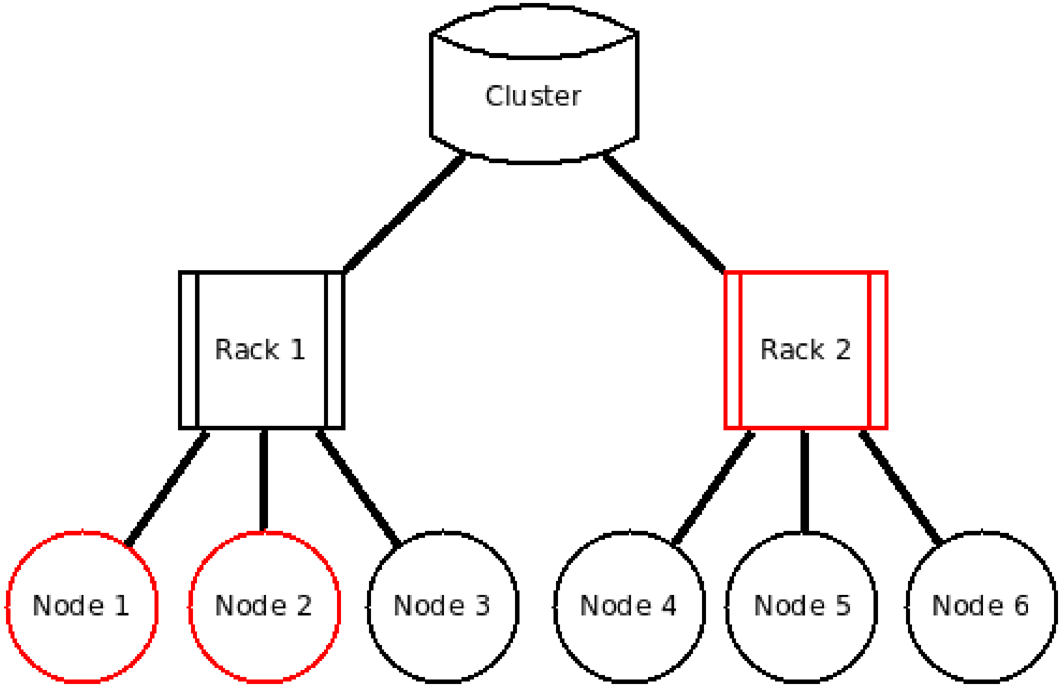
```
Priority=10
Root=Cluster

Entity=Cluster Type=Cluster Enclosed=Rack[1-2] CurrentPower=200
Entity=Rack1 Type=Rack Enclosed=Node[1-3] CurrentPower=200
Entity=Rack2 Type=Rack Enclosed=Node[4-6] CurrentPower=200
Entity=Node[1-6] Type=Node CurrentPower=0 Frequency=0
```



API : Get

Get the values for:

- a layout
- a uniq key
- one or several entities

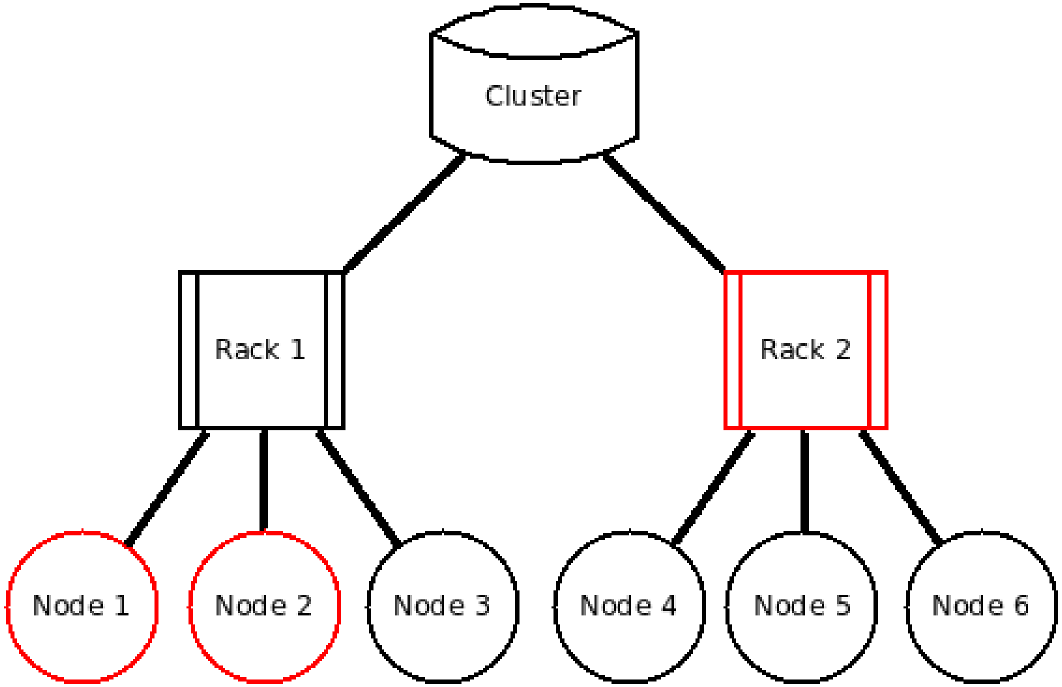


Layout :
Power
Key:
CurrentPower
Entities:
Rack2
Node1
Node2

API : Set

Set the values for:

- a layout
- a uniq key
- one or several entities



Layout :
    Power
Key:
    CurrentPower
Entities:
    Rack2
    Node1
    Node2

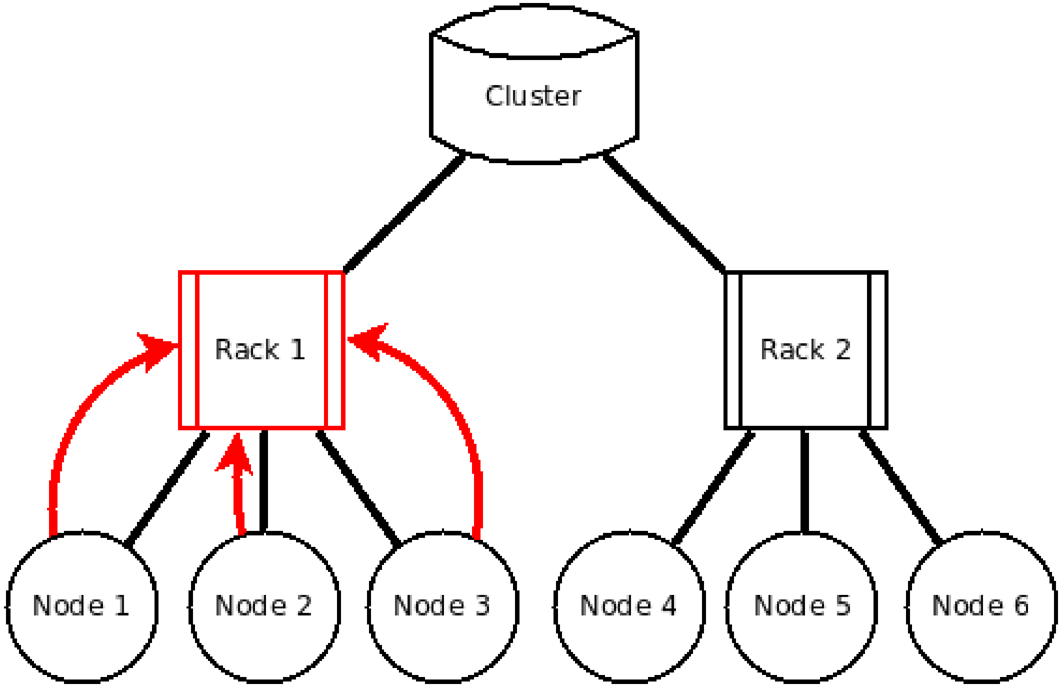
API : Update and Get

Recursive update, and get the values for:

a layout

a uniq key

one or several entities



Layout :

Power

Key:

CurrentPower

Entities:

Rack1

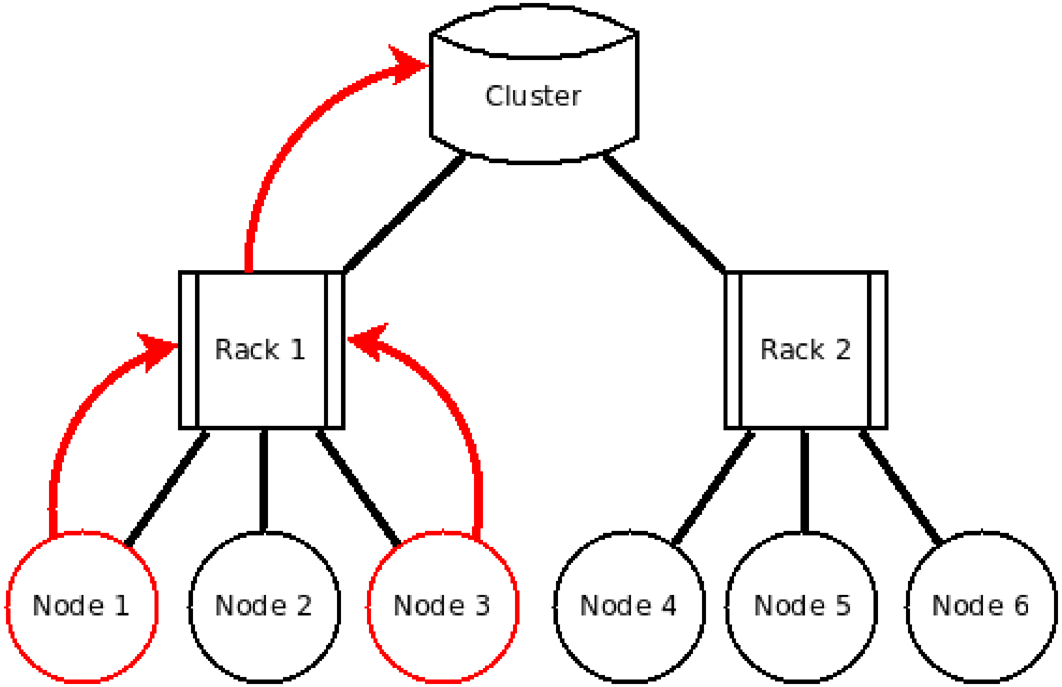
API : Set and Update

Set the values and propagate information for:

a layout

a uniq key

one or several entities



Layout :

Power

Key:

CurrentPower

Entities:

Node1

Node3

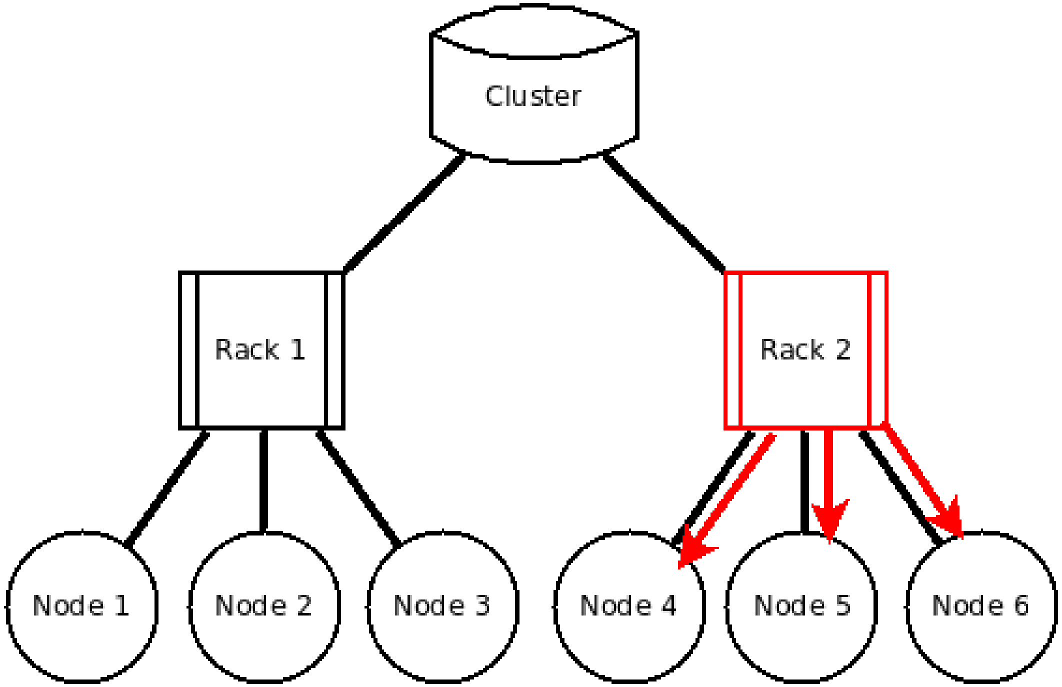
API : Update

Propage the values for:

a layout

a uniq key

from one or several entities



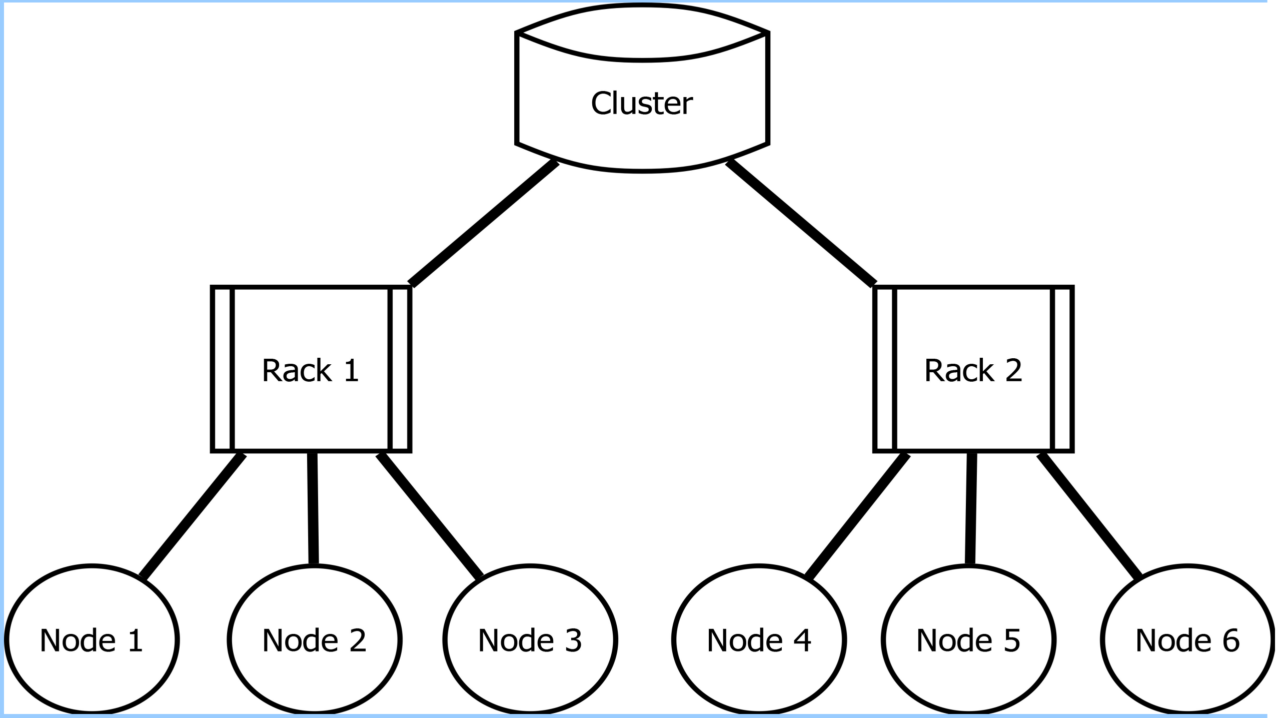
Layout :
Power
Key:
CurrentPower
Entities:
Node1
Node3

API: Options

Layouts are described by trees (for the moment)

Operation (set functions)

- Set
- Sum



Direction

- None
- Up
- Down

Consolidation

- Sum
- Mean
- Set (propagate value)

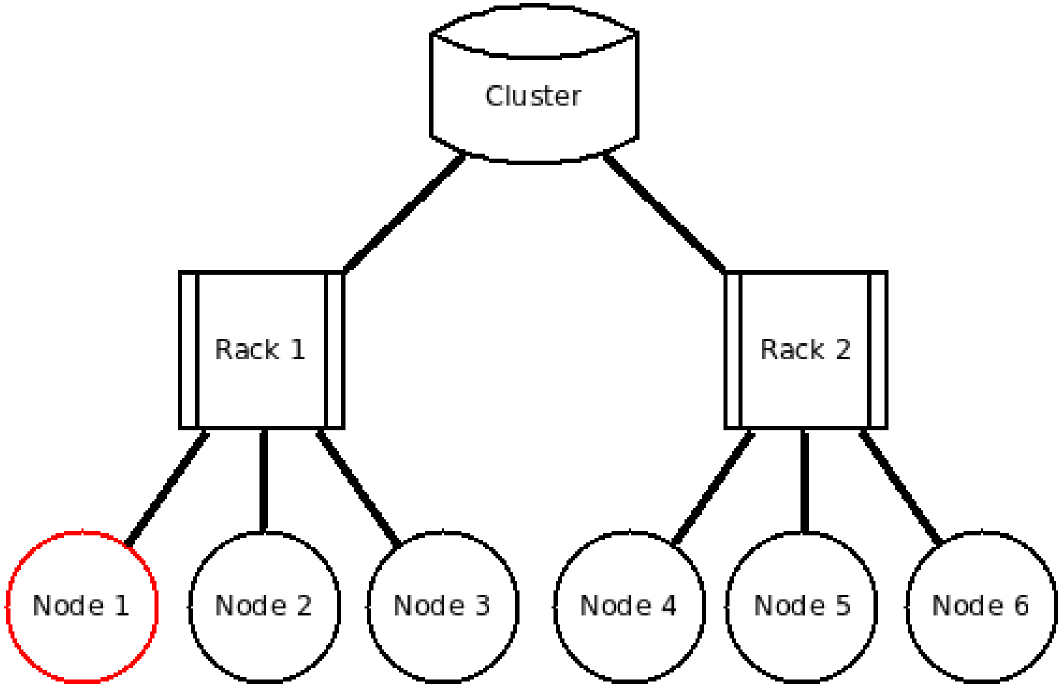
API : Multiple Get

Get the values for:

a layout

one or several keys (same type)

a uniq entities



Layout :

Power

Key:

CurrentPower

Frequency

Entities:

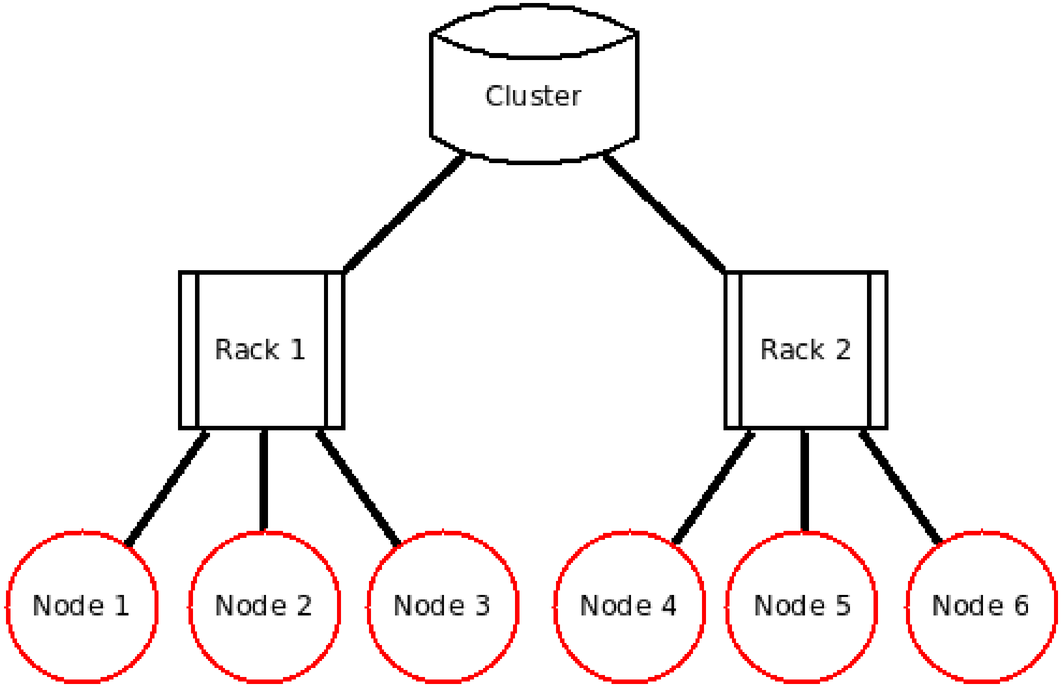
Node1

API : List entities

Get the list of entities for:

a layout

an entity type



Layout :

Power

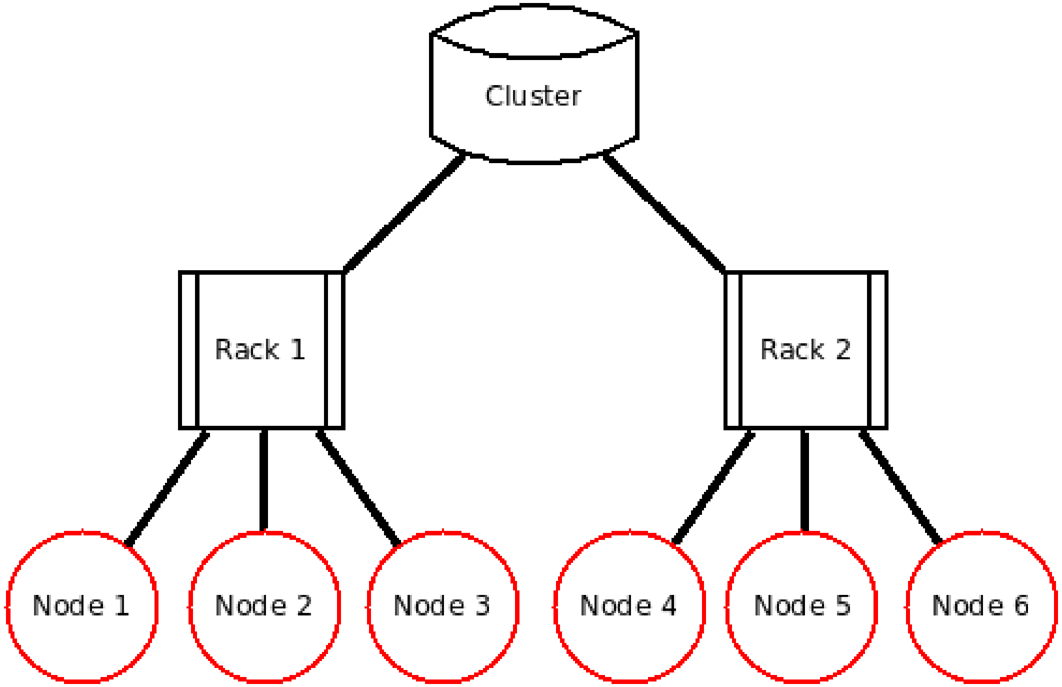
Type:

Node

scontrol update

Use set function of API

```
scontrol update layout=Power Entities=Node[1-6] Frequency=2
scontrol update layout=Power Entity_type=Node Frequency+=1
```



scontrol show

List all entities and print all pairs key/value for a layout

```
scontrol show layout=Power
```

```
Cluster
    Type=Cluster
    CurrentPower=200
Rack1
    Type=Rack
    CurrentPower=200
[...]
Node1
    Type=Node
    CurrentPower=312
    Frequency=3
[...]
```

Layout: powercap

```
Priority=10
Root=Cluster

Entity=Cluster Type=Center
        CurrentPower=0 IdleWatts=0 MaxWatts=0 Enclosed=node[0-40]

Entity=node[0-40] Type=Node
        CurrentPower=0 CurrentFreq=0
        IdleWatts=103 MaxWatts=308
        NumFreqChoices=8
        Cpufreq1=1200000 Cpufreq2=1400000
        Cpufreq3=1600000 Cpufreq4=1800000
        Cpufreq5=2000000 Cpufreq6=2200000
        Cpufreq7=2400000 Cpufreq8=2600000
        Cpufreq1Watts=172 Cpufreq2Watts=187
        Cpufreq3Watts=203 Cpufreq4Watts=226
        Cpufreq5Watts=252 Cpufreq6Watts=273
        Cpufreq7Watts=293 Cpufreq8Watts=308
```

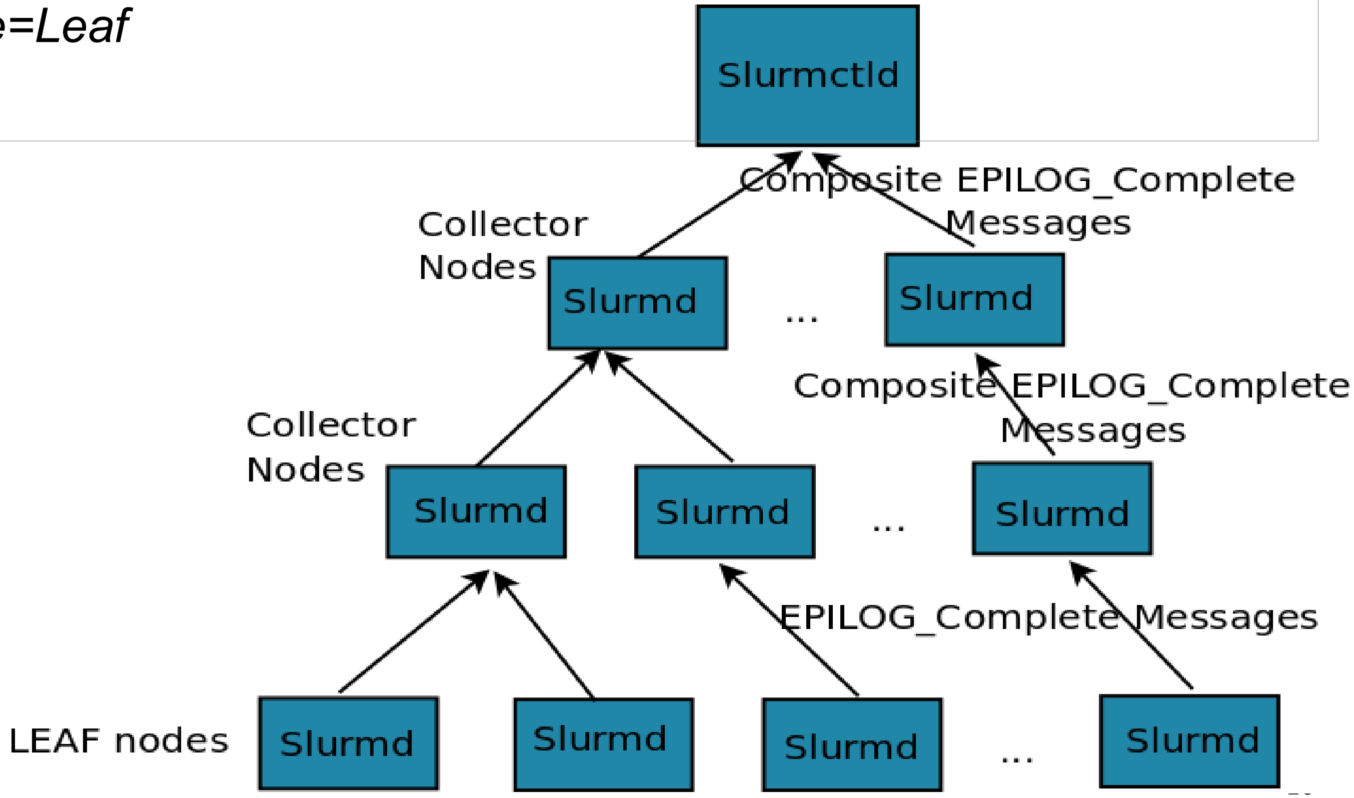
Layout: topology

```
# topology.conf

SwitchName=Top_Switch Switches=is[0-2]
SwitchName=is0 Nodes=node[0-9]
SwitchName=is1 Nodes=node[10-19]
SwitchName=is2 Nodes=node[20-29]
```

```
Priority=10
Root=Top_Switch

Entity=Top_Switch Type=Switch Enclosed=is[0-2]

Entity=is0 Type=Switch Enclosed=node[0-9]
Entity=is1 Type=Switch Enclosed=node[10-19]
Entity=is2 Type=Switch Enclosed=node[20-29]

Entity=node[0-29] Type=Node
```

Layout: forwarding

```
Priority=10
Root=node0
Entity=node0 Type=Slurmctld Enclosed=node[1,2]
Entity=node1 Type=Collector Enclosed=node[3,4]
Entity=node2 Type=Collector Enclosed=node[5,6]
Entity=node3 Type=Collector Enclosed=node[7,8]
Entity=node4 Type=Collector Enclosed=node[9,10]
Entity=node[5-10] Type=Leaf
```



Layout: associations

```
Priority=10
Root=all

Entity=all Type=account Enclosed=research,prod

Entity=research Type=account Enclosed=project1,project2,user1,user2

Entity=project1 Type=account Enclosed=user3,user4
Entity=project2 Type=account Enclosed=user5,user6
Entity=prod Type=account Enclosed=user7

Entity=user1 Type=user Role=coordinator
Entity=user2 Type=user
Entity=user3 Type=user
Entity=user4 Type=user
Entity=user5 Type=user
Entity=user6 Type=user
Entity=user7 Type=user
```

Ongoing and Future Works

Ongoing work

- Dump values for state recovery
- Validate the API with Power Capping algorithm
- Enhance the API for any needs of other layouts
- Continue the implementation of a first set of example layouts

Integrate the layouts logic in the internals of Slurm

- With new features : Advanced hierarchical communications,
- power aware scheduler...
- Updating current features : topology...

Implement other description than tree

- Graph
- Multi-tree

Bull

an atos company