DE LA RECHERCHE À L'INDUSTRIE

Slurm Layouts Framework

status

www.cea.fr

Slurm Layouts Framework
Agenda

Layouts Framework

Features added in slurm-15.08

Features under review, ongoing or planed

Slurm Layouts Framework

Layouts Framework

Layouts Framework

Motivations

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

Goals

- Add a generic/extensible way to describe facets of supercomputers
- Propose facets details to the resource manager for
  - Advanced management
  - Advanced scheduling
- Ease facets 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 facets

Layouts

- Layouts correspond to the managed facets
  - Example: racking facet, power provisioning facet, ...
- Provide a relational logic to link managed components
- Provide a set of properties to enhance components information

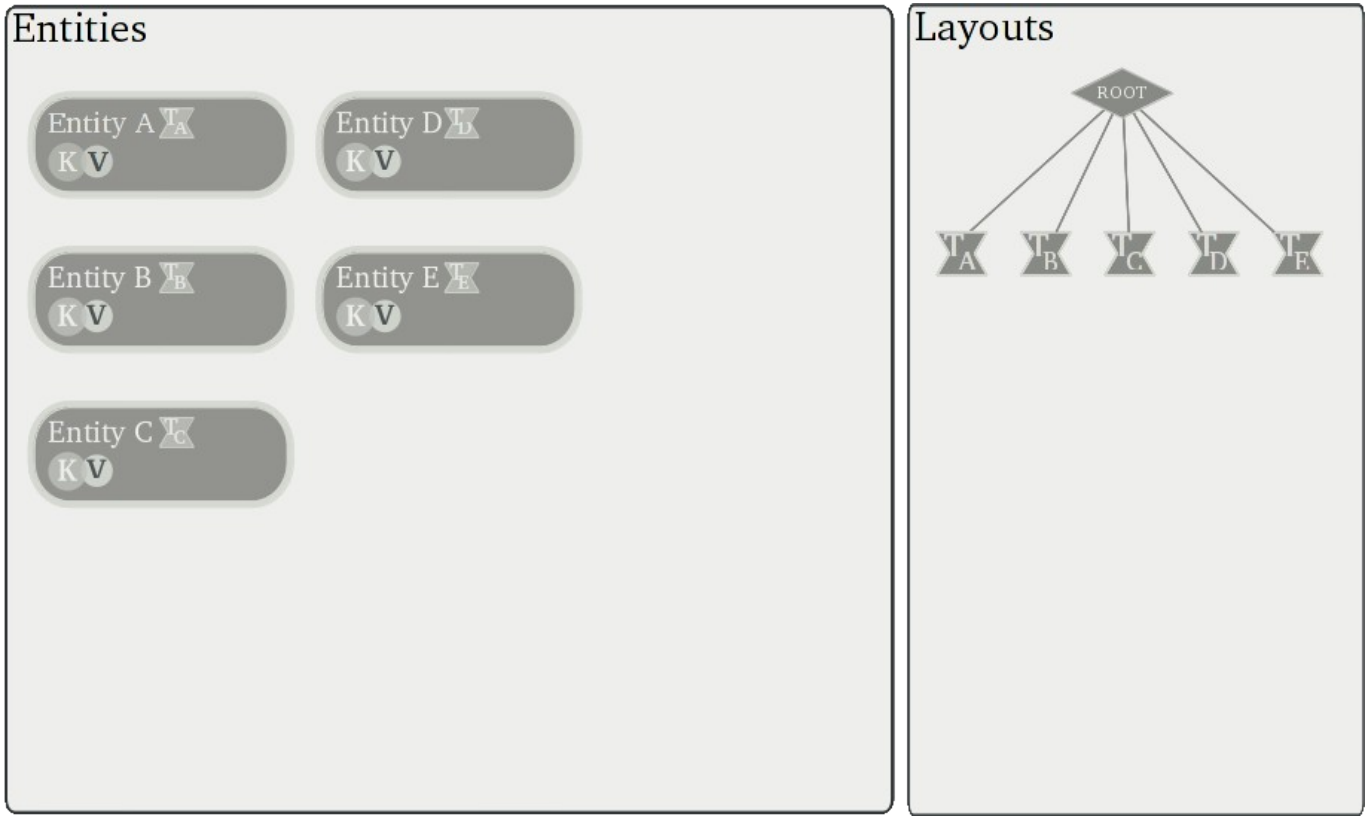
Layouts Framework

Entities

- Hold key-value entries for the associated layouts
- Hold relational structure pointers of the associated layouts
  - All entities are not necessarily part of all the layouts

Layouts

- Start from a root and link entities that are related to the layout
  - Only tree relational supported right now
- Each entity can be used as an entry point to discover its neighborhood in associated layouts



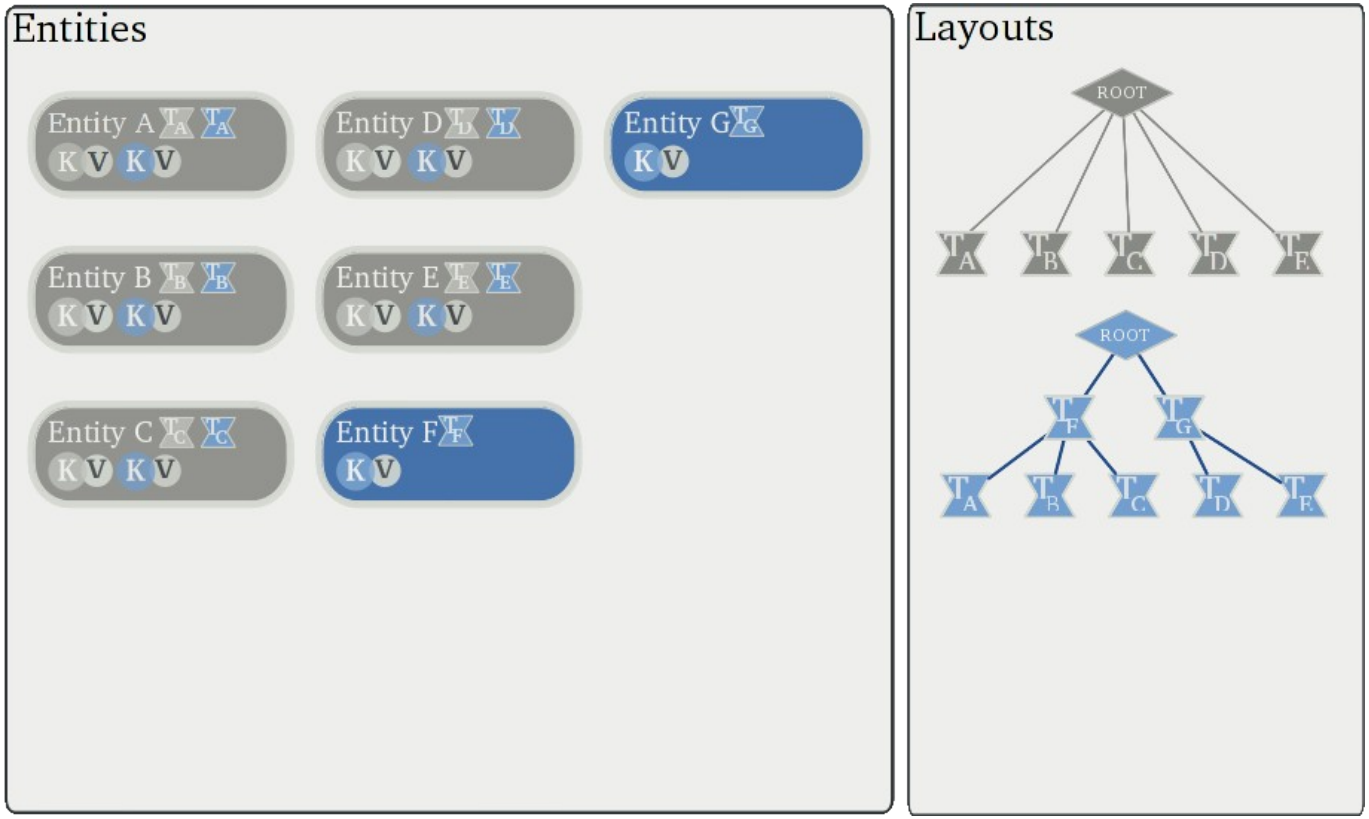
Layouts Framework

Entities

- Hold key-value entries for the associated layouts
- Hold relational structure pointers of the associated layouts
  - All entities are not necessarily part of all the layouts

Layouts

- Start from a root and link entities that are related to the layout
  - Only tree relational supported right now
- Each entity can be used as an entry point to discover its neighborhood in associated layouts



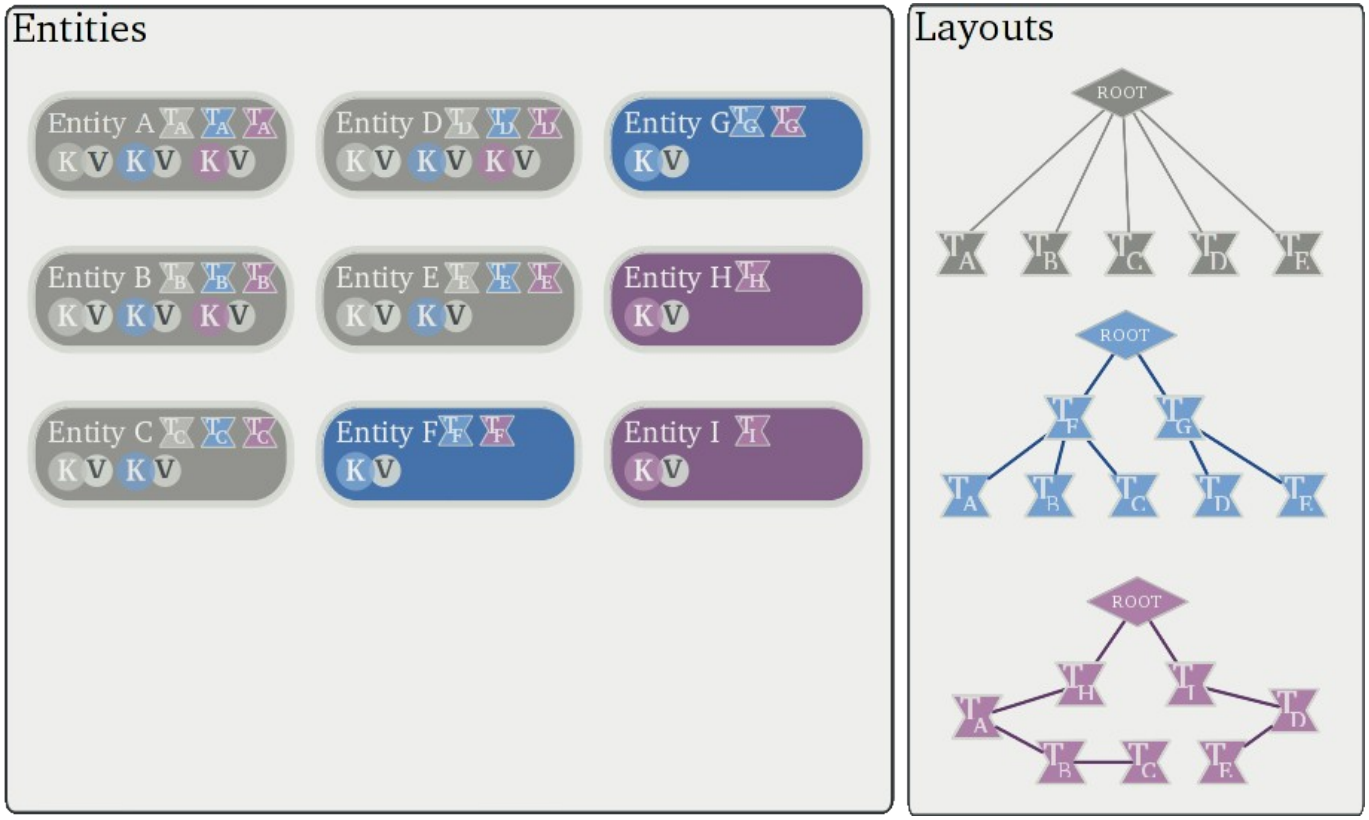
Layouts Framework

Entities

- Hold key-value entries for the associated layouts
- Hold relational structure pointers of the associated layouts
  - All entities are not necessarily part of all the layouts

Layouts

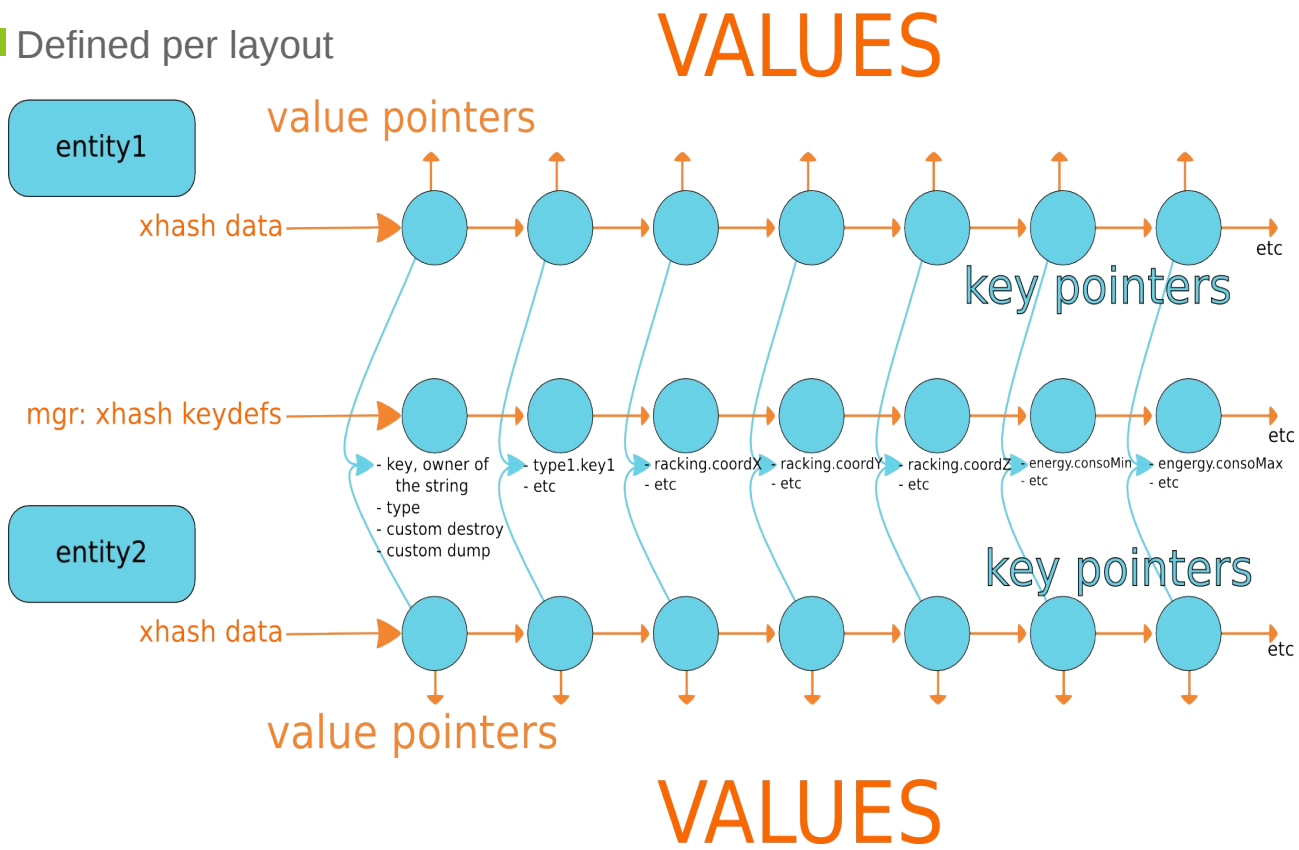
- Start from a root and link entities that are related to the layout
  - Only tree relational supported right now
- Each entity can be used as an entry point to discover its neighborhood in associated layouts



Layouts Framework

Key/Value definitions: Key-spec

- Associate a particular key to a particular type of data
  - boolean, uint16, uint32, long
  - float, double, long double
  - string, custom type
- Defined per layout
- Centralize meta-data of the K/V entries
  - Minimize memory consumption by avoiding duplication of information in entities



Availability in Slurm

- Started in 2012
  - Initial low level data structures for tree logic and hash tables (xtree, xhash)
- Followed in 2013/2014
  - Generic parsing logic for layouts
  - Entities, Layouts and Layouts Manager structures
  - Integrated in Slurm-14.11
  - (xhash [uthash wrapper] reused to decrease slurm node conf parsing time)
- Status in 2015
  - New features added but still in “Technical Preview” state

Slurm Layouts Framework

Features added in slurm-15.08

State save/restore logic

- Provide a way to serialize/deserialize layouts states
- Required to track/keep dynamic changes of entities properties (KVs)
- Selected logic
  - Generate expanded (no [1-...,10-...]) conf files representing states
  - Interests
    - Reuse conf parsing logic to deserialize states
    - Modifying/Inspecting states for debug is very easy
  - Drawbacks
    - File size can be large for high number of components
    - Float values precision limited by string conversion

Scontrol view/update layouts

- Provide a way to get layouts details externally
- Provide a way to modify key/value entries
- Required to ease dynamic changes of entities properties
- Selected logic
  - Get serialized view of layouts states (~state files) over the network
  - Send conf extracts processed by the controller conf parsing logic
    - Only KV modifications are handled in such updates
  - Add advanced operators to the parsing logic for atomicity of some ops
    - Key[+-*/]=value
    - Enable concurrent writers

Read-Only Key/Value entries (Key-spec)

- Provide a way to ask for immutable properties
  - Forbidding any update using “scontrol update layouts ...”

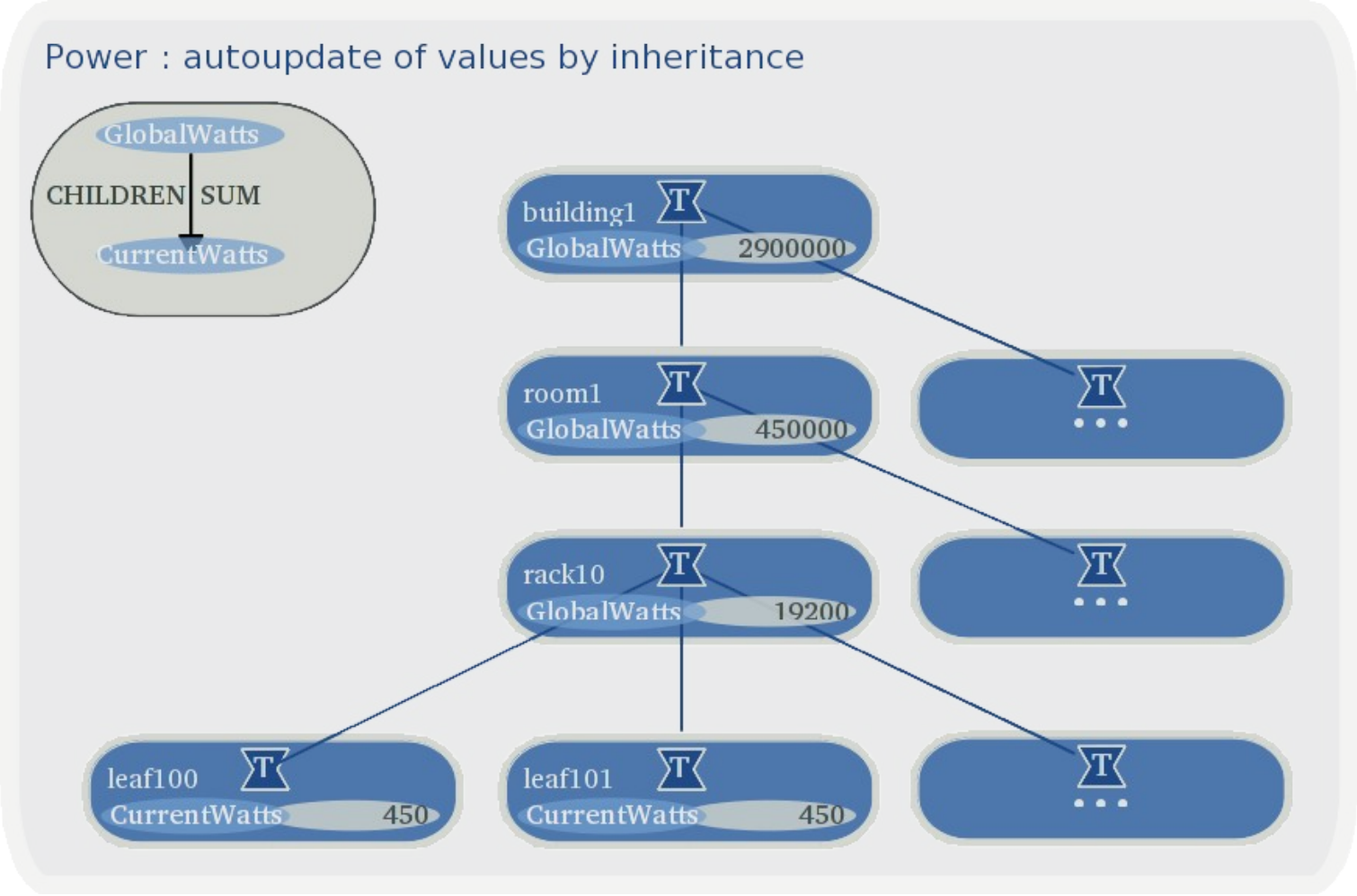
Key/Value inheritance model

- 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

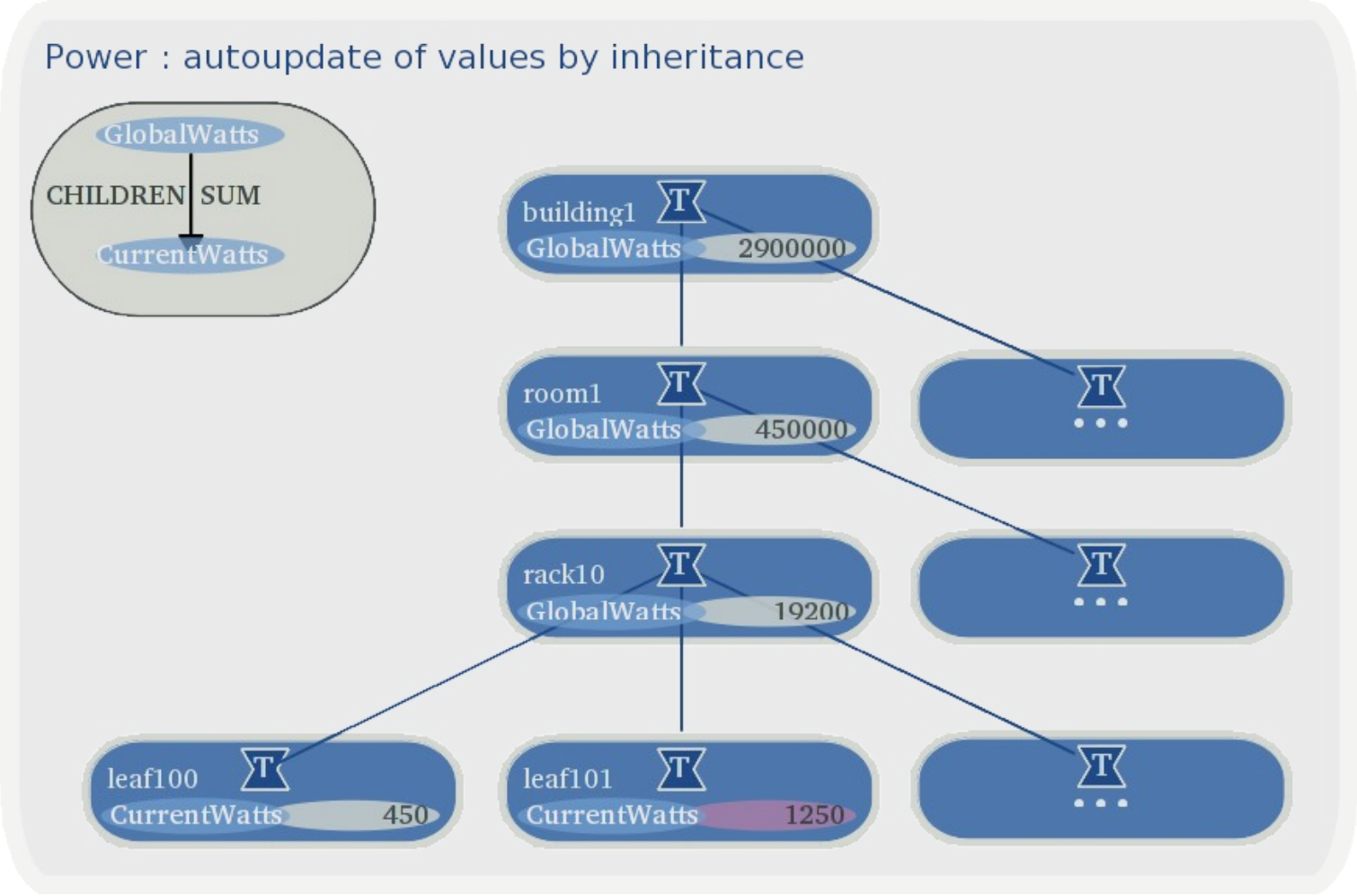
Key/Value Automatic Updates

- 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

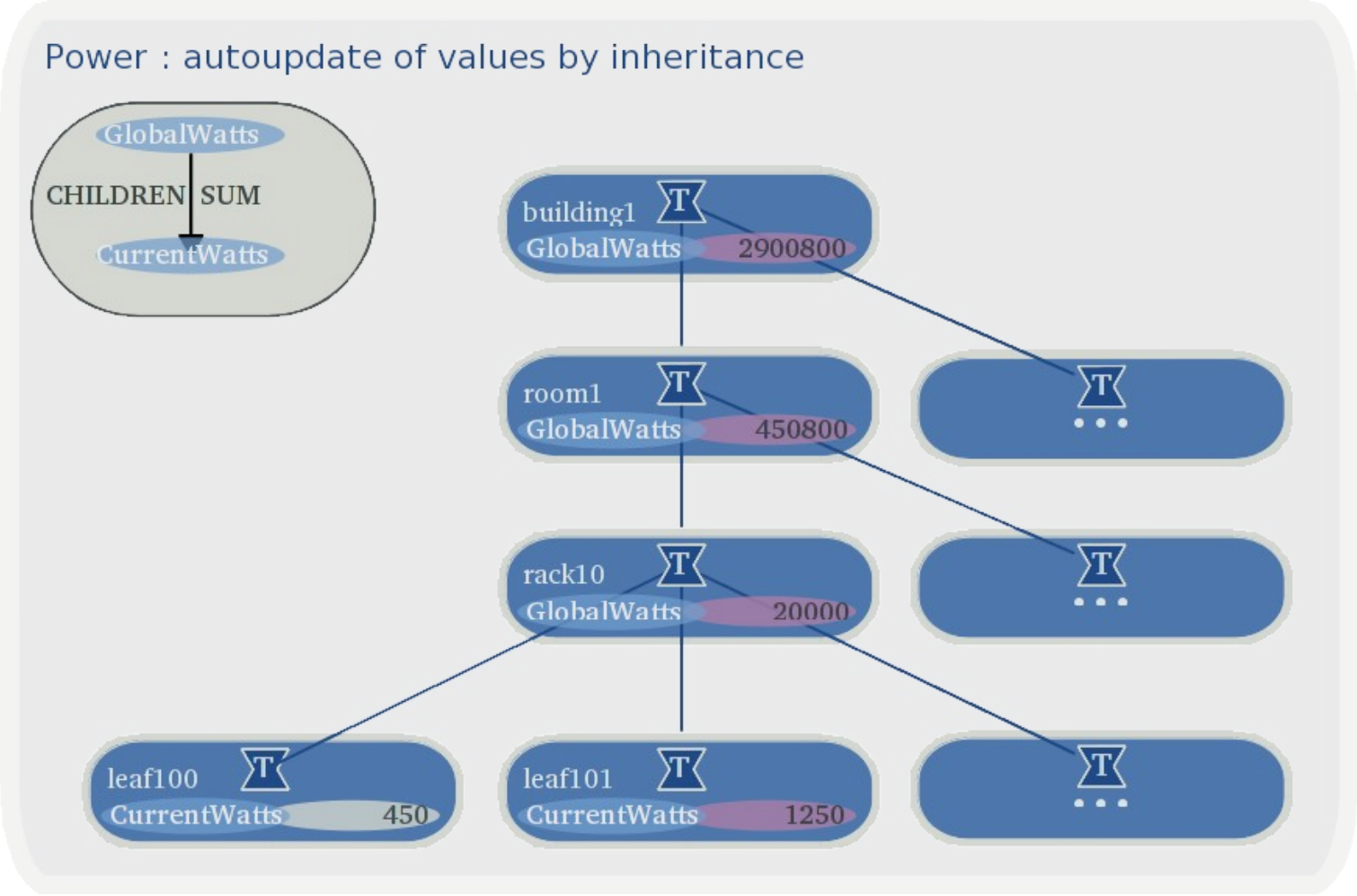
Key/Value Automatic Updates



Key/Value Automatic Updates



Key/Value Automatic Updates



Internal API modification

- Ask for states save
  - layouts_state_save(void)
- Ask for update
  - layouts_update_layout(...)
- Ask for set/get, exposing internal consistency model
  - Layouts_entity_{get,set,**pull**get,set**push**}_kv[_ref] calls
    - Enable to query without consistency support
    - Enable to request for update before getting values
    - Enable to request for update after setting values
- Ask for get of multiple KV entries for an entity
  - layouts_entity_get_[m]kv[_ref] call

Unit layout for validation

- Simple layout used to validate proper working of the layouts framework
  - Including all the possible typed values & available inheritances

```
$ scontrol show layouts
unit
$ scontrol show layouts unit
Root=GlobalPass
Entity=GlobalPass Type=UnitTestPass children_count=4 Enclosed=pass[1-4]
Entity=pass1 Type=UnitTestPass ... Enclosed=pass1_test[1-9]
Entity=pass2 Type=UnitTestPass ... children_sum_long=9 Enclosed=pass2_test[1-9]
Entity=pass3 Type=UnitTestPass Enclosed=pass3_test[1-9]
Entity=pass4 Type=UnitTestPass Enclosed=pass4_test[1-9]
$
```

Unit layout for validation (cont'd)

```
$ scontrol show layouts unit entity=pass2_test1
Entity=pass2_test1 Type=UnitTest string=test11 long=1 ...
$

$ scontrol show layouts unit entity=pass2_test1
Entity=pass2_test1 ... long=1 ...
$

$ scontrol update layouts=unit entity=pass2_test1 long=2
$

$ scontrol show layouts unit entity=pass2_test1
Entity=pass2_test1 ... long=2 ...
$

$ scontrol show layouts unit entity=pass2
Entity=pass2 ... children_sum_long=10 ... Enclosed=pass2_test[1-9]
```

Slurm Layouts Framework

Features under review, ongoing or planed

Enhance large configurations definition

- Previous layouts configuration description was limited to **N-to-1** or **N-to-N matching** like

  Entity=node[1-10] key=value
  Entity=node[1-10] key=value[1-10]

- This was **not sufficient** to express **complex nested description**

- **Now we provide support for cycling or split N-to-M matching**

  Entity=node[1-10] Enclosed=node[1-10]_core[0-31]
  Entity=node[1-10]_core[0-31] core_id=[0-31] frequency=2.5

- This enable to describe complex hierarchies in **very few configuration lines**

Default racking Layout

- Propose a way to describe racking information in Slurm
  - Insights concerning valuable properties to add are welcomed
    - GPS coords ? Elevation ? ...

```
$ scontrol show layouts
unit,racking
$ scontrol show layouts racking
Root=Roaming
Entity=Roaming     Type=Center   Enclosed=Laptop
Entity=Laptop      Type=Room     Enclosed=Row[0-2]
Entity=Row0  Type=Row  x_coord=0          Enclosed=Rack0
Entity=Rack0 Type=Rack  y_coord=1  x_coord=0  Enclosed=leaf0
Entity=Row1 Type=Row   x_coord=1          Enclosed=Rack[1-2]
Entity=Rack1 Type=Rack  y_coord=1  x_coord=1  Enclosed=leaf[100000-100007]
Entity=Rack2 Type=Rack  y_coord=2  x_coord=1  Enclosed=leaf[100008-100015]
Entity=Row2 Type=Row   x_coord=2          Enclosed=Rack[3-4]
Entity=Rack3 Type=Rack  y_coord=1  x_coord=2  Enclosed=leaf[100016-100023]
Entity=Rack4 Type=Rack  y_coord=2  x_coord=2  Enclosed=leaf[100024-100031]
```

Default racking Layout (cont'd)

```
$ scontrol show layouts racking entities=*
Root=Roaming
Entity=Roaming     Type=Center   Enclosed=Laptop
Entity=Laptop      Type=Room     Enclosed=Row[0-2]
Entity=Row0  Type=Row  x_coord=0          Enclosed=Rack0
Entity=Rack0 Type=Rack  y_coord=1  x_coord=0  Enclosed=leaf0
Entity=Row1 Type=Row   x_coord=1          Enclosed=Rack[1-2]
Entity=Rack1 Type=Rack  y_coord=1  x_coord=1  Enclosed=leaf[100000-100007]
Entity=leaf100000 Type=Node   z_coord=1   x_coord=1  y_coord=1
Entity=leaf100001 Type=Node   z_coord=2   x_coord=1  y_coord=1
Entity=leaf100002 Type=Node   z_coord=3   x_coord=1  y_coord=1
Entity=leaf100003 Type=Node   z_coord=4   x_coord=1  y_coord=1
Entity=leaf100004 Type=Node   z_coord=5   x_coord=1  y_coord=1
Entity=leaf100005 Type=Node   z_coord=6   x_coord=1  y_coord=1
Entity=leaf100006 Type=Node   z_coord=7   x_coord=1  y_coord=1
Entity=leaf100007 Type=Node   z_coord=8   x_coord=1  y_coord=1
...
```

Common ancestors look-up

- Provide a way to query for the common ancestor(s) of a list of entities in a particular layout
  - Useful when searching for the aspect that could better explain an issue

```
$ scontrol show layouts racking entity=leaf[100024-100028] ancestors
Entity=Rack4 Type=Rack y_coord=2 x_coord=2 Enclosed=leaf[100024-100031]
$

$ scontrol show layouts racking entity=leaf[100023-100028] ancestors
Entity=Row2 Type=Row x_coord=2 Enclosed=Rack[3-4]
$

$ scontrol show layouts racking entity=leaf[0,100023-100028] ancestors
Entity=Laptop Type=Room Enclosed=Row[0-2]
$
```

Zipped layouts states files

- Layouts state save/restore logic may generates large files
  - Because it represents the expanded version of the layouts configurations
- Zipping state files could greatly reduce:
  - The storage space
  - The time spent to write them on disk
  - The time spent to read them from disk
- Zipping state files could still:
  - Allow easy access to states files content
  - Allow easy modification of states files content
- Planed feature, not yet started
  - Could also be used to zip layouts RPC payloads

Thank you for your attention

Questions ?

Commissariat à l’énergie atomique et aux énergies alternatives
Centre DAM-Ile de France | 91297 Bruyères-le-Châtel Cedex
T. +33 (0)1 69 26 40 00 | F. +33 (0)1 69 26 70 86

Etablissement public à caractère industriel et commercial | RCS Paris B 775 685 019