Layouts Framework

Francois Chevallier
Matthieu Hautreux
Yiannis Georgiou

19/09/13
- Motivations and Goals
- Architecture and Current Status
- Performance Evaluations
- Ongoing and Future Works
Motivations and Goals

Architecture and Current Status

Performance Evaluations

Ongoing and Future Works
Introduction

• 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

• 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
• 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
Layout Framework: Examples of targeted usages

• **Communications optimization**
  – advanced hierarchical communications
    • Components roles » layout: gateway nodes aggregating/spreading the messages
  – optimized tree communications
    • Admin network » layout: generic Tree -> Adapted Tree

• **Scheduling**
  – Racking / Power awareness: « racking » layout
    • Free full racks when possible to power off infrastructure equipments and reduce useless consumptions (reduce PUE)
  – Power awareness: « power supply » layout
    • Adapt job placement to available power
- Motivations and Goals
- Architecture and Current Status
- Performance Evaluations
- Ongoing and Future Works
Current Status

• Study started in 2012
  – Student in an internship at CEA
  – continued in 2013 at BULL

• First milestones
  – Implement the core logic of the framework
  – Implement a first set of layouts
  – Roles, Racking, Power Supply (, Resources)
  – Reuse the layouts in the internals of SLURM
  – Adv hierarchical comms, power aware sched, ...
Current Status

• Completed milestones
  – Implement the core logic of the framework
    • Basic required structures (hash table, tree) in slurm2.5
    • Entities / Layouts parsing, generation and management
  – Implement a first set of layouts
    • Racking, Power Supply
• Not a plugin, a new framework
  – Containing layouts as plugins
  – generic and simple insertion of new information types;
• Features:
  – Easy browsing: simple browsing inside entities relations;
  – fast browsing: indexed and constant time browsing, optimized access;
  – quick creation of layouts: code factorization of main workflow;
  – configuration extension: extended |slurm| parser.
Current Status

VALUES

entity1

value pointers

xhash data

mgr: xhash keydefs

- key, owner of
  - type
  - custom destroy
  - custom dump

type1.key1
  - etc

racking.coordX
  - etc

racking.coordY
  - etc

racking.coordZ
  - etc

energy.consoMin
  - etc

energy.consoMax
  - etc

etc

VALUES

entity2

value pointers

xhash data

eetc

Current Status

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
Motivations and Goals

Architecture and Current Status

Performance Evaluations

Ongoing and Future Works
Performance Evaluation Tests

• Simulation of usage with real / synthetic configurations
  – Curie, racking layout of >5k nodes
  – Fictive, racking and energy layouts for different cluster sizes

• Evaluation of 5 steps of the workflow:
  – **init**: loads a layout plugin, instantiate structures and variables;
  – **phase 1**: parse configuration, read entities, merge them, root vertex to layout structure;
  – **phase 2**: build relations (tree);
  – **walk entities**: entities walk in global hash table, access attributes;
  – **walk layouts**: layouts walk, entities names;
Performance Evaluation Results

• Racking layout for Curie cluster

Loading times for racking.conf.curie

Loading time in microseconds

init  phase1  phase2  dump_entities  dump_layouts

© Bull, CEA 2013
Performance Evaluation Results

• Racking layout for simple fictive cluster
  ....
  Entity=chassis1 Type=Chassis CoordsY=1 Enclosed=asterix[0-49]
  Entity=asterix[0-49] Type=Node CoordsZ=[1-50]

• Energy layout for simple fictive cluster
  ...
  Entity=chassis1 Type=Chassis ConsoMIN=10 ConsoMED=40 ConsoMAX=50
  Enclosed=asterix[0-49]
  Entity=asterix[0-49] Type=Node ConsoMIN=10 ConsoMED=80 ConsoMAX=400
Performance Evaluation Results

• Racking layout for simple cluster with 10K nodes and different complexities:

Entity=Node[0-9999] Type=Node

Entity=Node0 Type=Node
Entity=Node1 Type=Node
Entity=Node2 Type=Node
...

Loading times for racking.conf.size-complex0

Loading times for racking.conf.size-complex4
Performance Evaluation Results

• Racking layout for Curie >5K nodes and fictive with 10K nodes
Performances Feedback

• Phase 1 might be time consuming
  – But « only » 350ms for 10k nodes
  – Only at startup / reload
• Entities and layouts walks are very fast
  – Interesting as the target is to use these calls very often
    • For scheduling
    • For communications
    • ...

© Bull, CEA 2013
Motivations and Goals

Architecture and Current Status

Performance Evaluations

Ongoing and Future Works
Ongoing and Future Works

• Validate / Enhance the API
  – Still a prototype
• Roles, admin network
  – Continue the Implementation of a first set of layouts
• Integrate the layouts logic in the internals of SLURM
  – Advanced hierarchical communications, power aware scheduler
People Involved

-Francois Chevallier (BULL)
-Matthieu Hautreux (CEA)
-Thomas Cadeau (BULL)
-Yiannis Georgiou (BULL)
slurmctl

type/implems
- init
  slurm_layouts_init
- global variables initialization
- loop on type/implem names
  (base, racking, etc)
  plugin_context_create
  plugin_context_create
- merge entities' data keys

configuration
  slurm_layouts_load_config

entity

phase 1: loop on loaded plugins
- parse config
  slurm config parser hash table
- entity data merging
  entity_init
  data_get / add_data
- store "Enclosed" strings
- add root entity to layout's tree

phase 2: loop on loaded plugins
- build relational structures
  - from the root node, extend tree until completion

temporary phase: dump