Towards multi-parameter resource selection for HPC platforms

Master Research Thesis

Dineshkumar RAJAGOPAL
(PDES - MoSIG)

advised by

Yiannis GEORGIOU

Big Data and Security(BDS) lab, BULL-SAS

September 1, 2015

1 Introduction and Motivation

2 Background
- Related Works
- SLURM Architecture
- LAYOUTS Framework

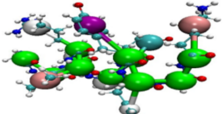
3 Resource Selection
- LAYOUTS based consumable resource selection
- Multi-parameter resource selection

4 Experimentation and Performance Evaluation
- Cons_res_layout vs Cons_res Analysis
- Multi-parameter vs Cons_res Analysis

5 Conclusion and Future Work
- Conclusion
- Future Work

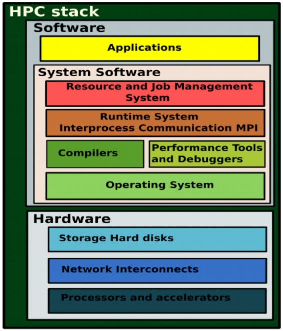
HPC System Software Stack

1. Supercomputer is a **HPC Cluster**
2. **Usage :** Computationally intensive tasks in **Scientific Experiments (Quantum mechanics, Weather Prediction, etc)**



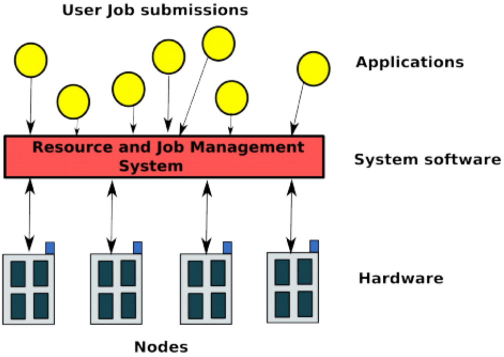
Theoretical and Computational Biophysics Group
Beckman Institute
University of Illinois at Urbana-Champaign

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


HPC stack
Software
Applications
System Software
Resource and Job Management System
Runtime System Interprocess Communication MPI
Compilers
Performance Tools and Debuggers
Operating System
Hardware
Storage Hard disks
Network Interconnects
Processors and accelerators

What is RJMS?

1. 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
2. **RJMS** knows the **complete details** about the Jobs and Resources of HPC system



Motivation of Resource Selection

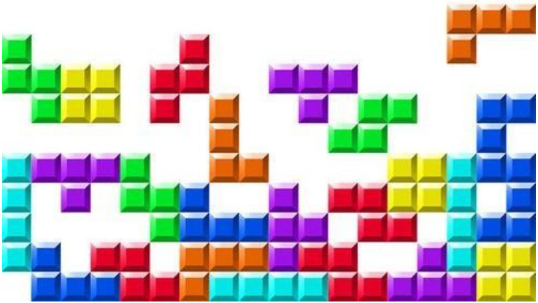
1. Resource Management is **dynamic** and **complex**
   - Evolution of HPC platform (fast complex networks, usage of accelarators,etc)
   - Evolution of internal nodes architecture (multi-core sockets, deeper memory hierarchies, etc)
2. Improper resource management **hides resources information** to lose global view of resource selection
3. LAYOUTS to manage resources properly and **reveal hidden information**

Multi-parameter Resource Selection

1. Power wall problem and the increasing number of nodes in HPC systems, **energy efficiency** is the important criteria
2. Multi-parameter resource selection to satisfy different criteria to **allocate resources perfectly**

Multi-parameter Resource Selection

1. Power wall problem and the increasing number of nodes in HPC systems, **energy efficiency** is the important criteria
2. Multi-parameter resource selection to satisfy different criteria to **allocate resources perfectly**
3. N-dimensional game of **Tetris** = Multi-parameter resource selection
4. Random sequence of **Tetromino** = Future Job workload



Related Works

1. Basic **architecture** is same, But **implementation** has different objective

Related Works

1. Basic **architecture** is same, But **implementation** has different objective
2. OAR used high level tools(Perl, Database) for flexible and scalable system implementation
3. SLURM used(C) custom implementation for scalable and performance oriented system

Related Works

1. Basic **architecture** is same, But **implementation** has different objective
2. OAR used high level tools(Perl, Database) for flexible and scalable system implementation
3. SLURM used(C) custom implementation for scalable and performance oriented system
4. OAR managing resources hierarchically in database for flexible resource allocation
5. SLURM managing resources linearly in custom data structure for performance and scalable resouce selection
6. OAR resource selection by SQL query (generic implementation)
7. SLURM resource selection by custom implementation of specific algorithm

1. Introduction and Motivation

2. **Background**
   - Related Works
   - **SLURM Architecture**
   - LAYOUTS Framework

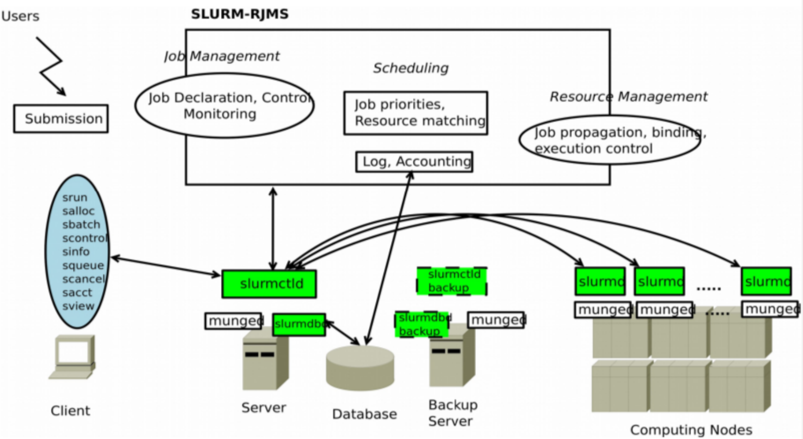
3. Resource Selection
   - LAYOUTS based consumable resource selection
   - Multi-parameter resource selection

4. Experimentation and Performance Evaluation
   - Cons_res_layout vs Cons_res Analysis
   - Multi-parameter vs Cons_res Analysis

5. Conclusion and Future Work
   - Conclusion
   - Future Work

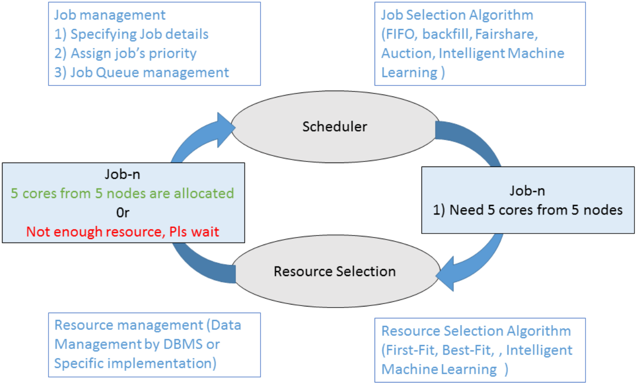
SLURM Architecture

1. SLURM is an **open source RJMS** for Supercomputer



Batch Scheduling

1. **Scheduling behaviour** depends on the Job scheduler and Resource selector behaviour
2. **Resource Selection** is an internal operation of scheduling



Job management
1) Specifying Job details
2) Assign job's priority
3) Job Queue management

Job Selection Algorithm (FIFO, backfill, Fairshare, Auction, Intelligent Machine Learning)

Scheduler

Job-n
1) Need 5 cores from 5 nodes

Resource Selection

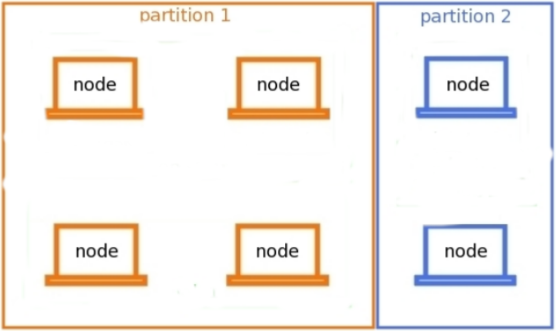
Job-n
5 cores from 5 nodes are allocated
Or
Not enough resource, Pls wait

Resource management (Data Management by DBMS or Specific implementation)

Resource Selection Algorithm (First-Fit, Best-Fit, , Intelligent Machine Learning)

SLURM Entities

1. SLURM resource and job management entities

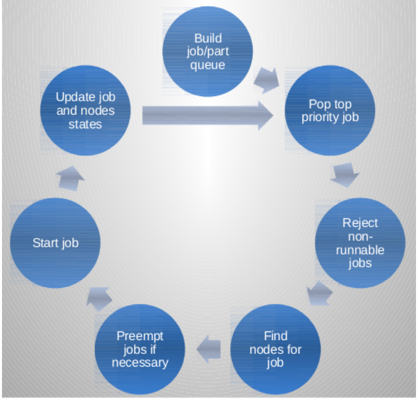


Resource Selection Cycle

1. **Select and Topology plugin** work together to allocate **topology aware resources**
2. Topology plugin has information of **Switch and Node relationship**
3. **Selection plugin mechanism** is followed in the below image

Resource Selection Cycle

1. **Select and Topology plugin** work together to allocate **topology aware resources**
2. Topology plugin has information of **Switch and Node relationship**
3. **Selection plugin mechanism** is followed in the below image



1. Introduction and Motivation
2. Background
   - Related Works
   - SLURM Architecture
   - LAYOUTS Framework
3. Resource Selection
   - LAYOUTS based consumable resource selection
   - Multi-parameter resource selection
4. Experimentation and Performance Evaluation
   - Cons_res_layout vs Cons_res Analysis
   - Multi-parameter vs Cons_res Analysis
5. Conclusion and Future Work
   - Conclusion
   - Future Work

LAYOUTS Framework

1. LAYOUTS is the resource management framework in SLURM
2. **Terminology:**
    - LAYOUTS - Framework
    - Layout(s) or layout(s) - Resource management plugin

LAYOUTS Framework

1. LAYOUTS is the resource management framework in SLURM
2. **Terminology:**
   - LAYOUTS - Framework
   - Layout(s) or layout(s) - Resource management plugin
3. **Physical and logical** entities can be manageable

LAYOUTS Framework

1. LAYOUTS is the resource management framework in SLURM
2. **Terminology:**
   - LAYOUTS - Framework
   - Layout(s) or layout(s) - Resource management plugin
3. **Physical and logical** entities can be manageable
4. Entities relation is **tree** (Inspiration of OAR resource management)

LAYOUTS Framework

1. LAYOUTS is the resource management framework in SLURM
2. **Terminology:**
    - LAYOUTS - Framework
    - Layout(s) or layout(s) - Resource management plugin
3. **Physical and logical** entities can be manageable
4. Entities relation is **tree** (Inspiration of OAR resource management)
5. Entity attribute is called in LAYOUTS **key**

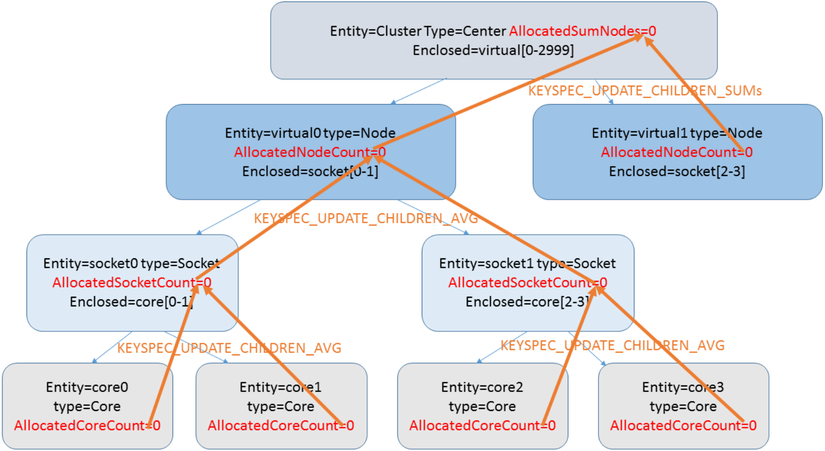
LAYOUTS Framework

1. LAYOUTS is the resource management framework in SLURM
2. **Terminology:**
   - LAYOUTS - Framework
   - Layout(s) or layout(s) - Resource management plugin
3. **Physical and logical** entities can be manageable
4. Entities relation is **tree** (Inspiration of OAR resource management)
5. Entity attribute is called in LAYOUTS **key**
6. Entity, Entity Keys and Layouts plugin list are in **hash data structure**

LAYOUTS Framework

1. LAYOUTS is the resource management framework in SLURM
2. **Terminology:**
   - LAYOUTS - Framework
   - Layout(s) or layout(s) - Resource management plugin
3. **Physical and logical** entities can be manageable
4. Entities relation is **tree** (Inspiration of OAR resource management)
5. Entity attribute is called in LAYOUTS **key**
6. Entity, Entity Keys and Layouts plugin list are in **hash data structure**
7. Entity and immediate entities **keys relation** are defined

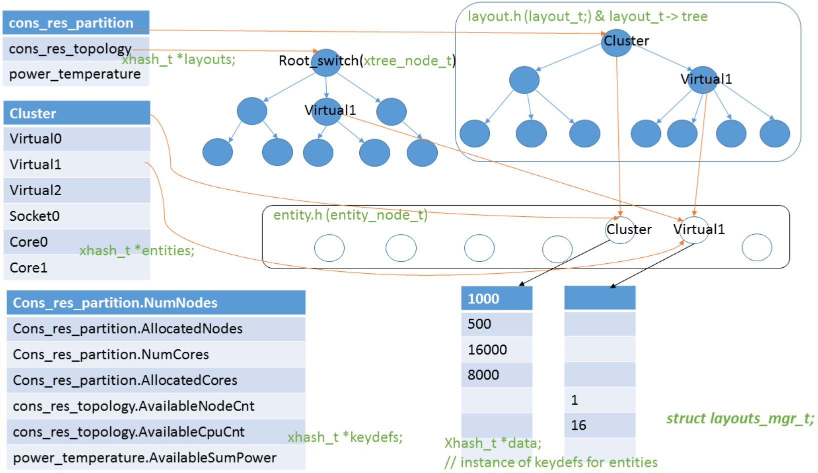
LAYOUTS Entity Keys and Key Relation



LAYOUTS Aggregate Keys

- **Child aggregate functions** - Specific operations performed on the children key value and update the calculated value in the parent′s key
  1. KEYSPEC_UPDATE_CHILDREN_SUM
  2. KEYSPEC_UPDATE_CHILDREN_AVG
  3. KEYSPEC_UPDATE_CHILDREN_MIN
  4. KEYSPEC_UPDATE_CHILDREN_MAX
  5. KEYSPEC_UPDATE_CHILDREN_COUNT
  6. KEYSPEC_UPDATE_CHILDREN_MASK
- **Parent aggregate functions**
  1. KEYSPEC_UPDATE_PARENTS_SUM
  2. KEYSPEC_UPDATE_PARENTS_AVG
  3. KEYSPEC_UPDATE_PARENTS_MIN
  4. KEYSPEC_UPDATE_PARENTS_MAX
  5. KEYSPEC_UPDATE_PARENTS_FSHARE
  6. KEYSPEC_UPDATE_PARENTS_MASK

LAYOUTS Internal Architecture



cons_res_partition
cons_res_topology
power_temperature

Cluster
Virtual0
Virtual1
Virtual2
Socket0
Core0
Core1

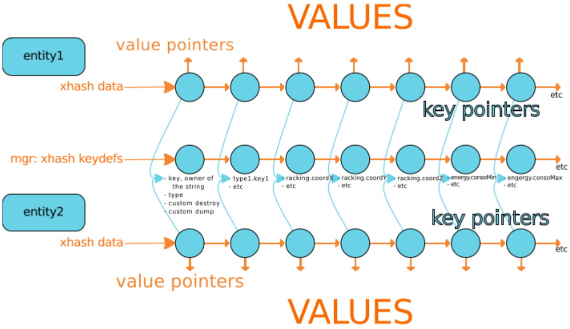
Cons_res_partition.NumNodes
Cons_res_partition.AllocatedNodes
Cons_res_partition.NumCores
Cons_res_partition.AllocatedCores
cons_res_topology.AvailableNodeCnt
cons_res_topology.AvailableCpuCnt
power_temperature.AvailableSumPower

1000
500
16000
8000

1
16

LAYOUTS Entity Data Management

1. Entity key names are unique
2. Entity and entity keys are in hash data structure
3. Entity has **different layouts plugin** information in the central place



LAYOUTS APIs

LAYOUTS APIs for plugin developers.

1. **layouts_entity_get_kv() - get** key value
2. **layouts_entity_set_kv() - set** key value
3. **layouts_entity_get_mkv() - get** multiple key value
4. **layouts_entity_pull_get_kv() - get** key value by key relation update
5. **layouts_entity_set_push_kv() - set** key value and key relation update
6. **layouts_entity_pull() - update** key relations value

1. Introduction and Motivation
2. Background
   - Related Works
   - SLURM Architecture
   - LAYOUTS Framework
3. Resource Selection
   - LAYOUTS based consumable resource selection
   - Multi-parameter resource selection
4. Experimentation and Performance Evaluation
   - Cons_res_layout vs Cons_res Analysis
   - Multi-parameter vs Cons_res Analysis
5. Conclusion and Future Work
   - Conclusion
   - Future Work

Cons_res_layout Implementation

1. **Cons_res_layout** is the **new** consumable resource selection plugin based on LAYOUTS
   - **Cons_res** consumes internal resources of nodes(cores, memory, etc)
   - **Cons_res** used list and bitmap to keep resource information

Cons_res_layout Implementation

1. **Cons_res_layout** is the **new** consumable resource selection plugin based on LAYOUTS
    - **Cons_res** consumes internal resources of nodes(cores, memory, etc)
    - **Cons_res** used list and bitmap to keep resource information

2. **Cons_res_partition** layout plugin - Internal nodes details
    - **Nodes** - Physical nodes in HPC
    - **Partitions** - Logical group of nodes

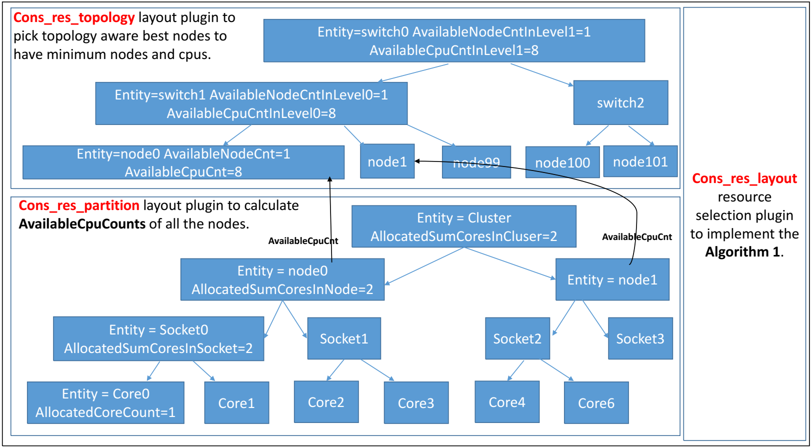
Cons_res_layout Implementation

1. **Cons_res_layout** is the **new** consumable resource selection plugin based on LAYOUTS
    - **Cons_res** consumes internal resources of nodes(cores, memory, etc)
    - **Cons_res** used list and bitmap to keep resource information

2. **Cons_res_partition** layout plugin - Internal nodes details
    - **Nodes** - Physical nodes in HPC
    - **Partitions** - Logical group of nodes

3. **Cons_res_topology** layout plugin - Cluster topology details
    1. Topology details needed for **topology aware resource selection**
    2. Tree(Normal) topology only supported

Cons_res_layout Architecture



Cons_res_topology layout plugin to pick topology aware best nodes to have minimum nodes and cpus.

Entity=switch0 AvailableNodeCntInLevel1=1 AvailableCpuCntInLevel1=8

Entity=switch1 AvailableNodeCntInLevel0=1 AvailableCpuCntInLevel0=8

switch2

Entity=node0 AvailableNodeCnt=1 AvailableCpuCnt=8

node1

node99

node100

node101

Cons_res_partition layout plugin to calculate **AvailableCpuCounts** of all the nodes.

Entity = Cluster AllocatedSumCoresInCluser=2

AvailableCpuCnt

AvailableCpuCnt

Entity = node0 AllocatedSumCoresInNode=2

Entity = node1

Entity = Socket0 AllocatedSumCoresInSocket=2

Socket1

Socket2

Socket3

Entity = Core0 AllocatedCoreCount=1

Core1

Core2

Core3

Core4

Core6

Cons_res_layout resource selection plugin to implement the **Algorithm 1**.

Cons_res_layout Resource Selection Algorithm

1. **Cons_res** same algorithm is followed, **without any change**
2. **Bestfit** to reduce fragmentation
3. **Topology aware** to increase the user application performance
4. **Algorithm 1** in the **section 4.1** of the **report** discussed the algorithm step by step

5. Naively all the operations are performed by using LAYOUTS framework
6. Naive implementation of cons_res_layout plugin performance was **25 times slower**

LAYOUTS New APIs

New APIs developed for new **functionality** and **performance** purpose.

1. **layouts_entity_get_parent_name()**
2. **layouts_multi_entity_set_kv()** - set **multiple entity** same key
3. **layouts_multi_entity_get_kv()**
4. **layouts_entity_pull_get_skv()** - update **specific key** and its relations
5. **layouts_entity_set_push_skv()**

Cons_res_layout Performance Improvements

1. Node and Core list is maintained in the **bitmap** data structure to compare and match another node list faster
2. LAYOUTS to access **entities keys** in the different levels

Cons_res_layout Performance Improvements

1. Node and Core list is maintained in the **bitmap** data structure to compare and match another node list faster
2. LAYOUTS to access **entities keys** in the different levels
3. **layouts_multi_entity_set_kv** - transfer information from one layout to another

Cons_res_layout Performance Improvements

1. Node and Core list is maintained in the **bitmap** data structure to compare and match another node list faster
2. LAYOUTS to access **entities keys** in the different levels
3. **layouts_multi_entity_set_kv** - transfer information from one layout to another
4. **layouts_entity_pull_get_skv** - update only specific key values

Cons_res_layout Performance Improvements

1. Node and Core list is maintained in the **bitmap** data structure to compare and match another node list faster
2. LAYOUTS to access **entities keys** in the different levels
3. **layouts_multi_entity_set_kv** - transfer information from one layout to another
4. **layouts_entity_pull_get_skv** - update only specific key values
5. Topology aware resource selection algorithm was adapted for **normal tree** topology

1. Introduction and Motivation
2. Background
   - Related Works
   - SLURM Architecture
   - LAYOUTS Framework
3. Resource Selection
   - LAYOUTS based consumable resource selection
   - Multi-parameter resource selection
4. Experimentation and Performance Evaluation
   - Cons_res_layout vs Cons_res Analysis
   - Multi-parameter vs Cons_res Analysis
5. Conclusion and Future Work
   - Conclusion
   - Future Work

Cons_res_power Implementation

1. **Cons_res_power** is the enhancement of the **cons_res_layout** plugin

Cons_res_power Implementation

1. **Cons_res_power** is the enhancement of the **cons_res_layout** plugin
2. New layout plugin **topology_power** for power details

Cons_res_power Implementation

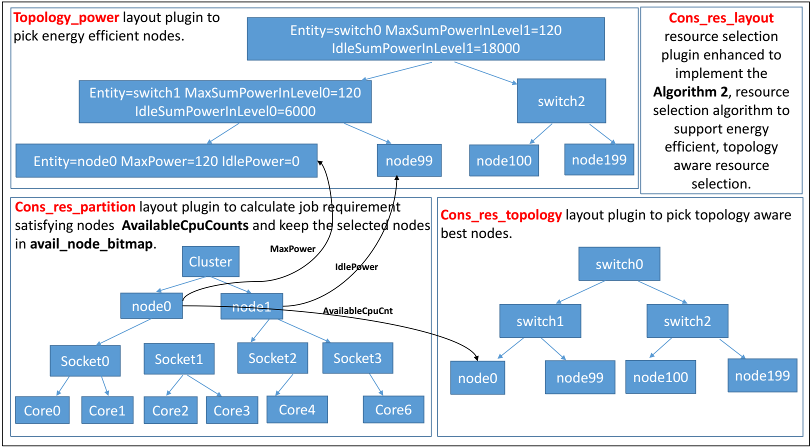
1. **Cons_res_power** is the enhancement of the **cons_res_layout** plugin
2. New layout plugin **topology_power** for power details
3. **Cons_res_partition** and **Cons_res_topology** layout plugins

Cons_res_power Resource Selection Algorithm

1. **Bestfit energy efficient** to reduce energy consumption
2. **Topology aware** to increase the user application performance
3. **Algorithm 2** in the **section 4.2** of the **report** discussed the algorithm step by step

4. **Advantage:** Multi-parameter resource selection supports **user′s performance** and **server′s energy** criterias

Cons_res_power Architecture



Topology_power layout plugin to pick energy efficient nodes.

Cons_res_layout resource selection plugin enhanced to implement the **Algorithm 2**, resource selection algorithm to support energy efficient, topology aware resource selection.

Cons_res_partition layout plugin to calculate job requirement satisfying nodes **AvailableCpuCounts** and keep the selected nodes in **avail_node_bitmap**.

Cons_res_topology layout plugin to pick topology aware best nodes.

1 Introduction and Motivation

2 Background
- Related Works
- SLURM Architecture
- LAYOUTS Framework

3 Resource Selection
- LAYOUTS based consumable resource selection
- Multi-parameter resource selection

4 Experimentation and Performance Evaluation
- Cons_res_layout vs Cons_res Analysis
- Multi-parameter vs Cons_res Analysis

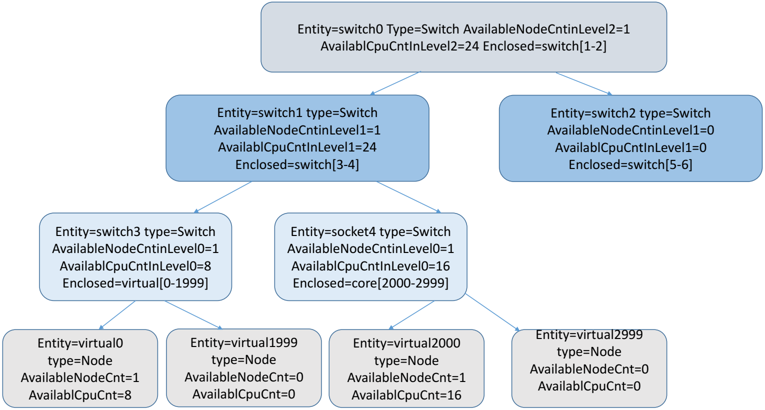
5 Conclusion and Future Work
- Conclusion
- Future Work

Experiment Environment

1. Emulate real HPC environment using **–enable-multiple-slurmd** option
2. Synthetic workload of **Enhanced System Performance**(ESP) Benchmark
3. Use **sleep, hostname** like simple application
4. **Standard Workload Format**(SWF) to store workload
5. BULL CUZCO cluster 17 nodes to emulate **5040 nodes** HPC cluster
6. Each node configured as **2 sockets, 16 cores and 32GB of memory**, more details are in **Appendix C.1**

Topology Experiment Environment

1. **Homogeneous** cluster environment
2. Simple tree topology to have **4 leaf switches** and **3 levels**



Entity=switch0 Type=Switch AvailableNodeCntinLevel2=1 AvailablCpuCntInLevel2=24 Enclosed=switch[1-2]

Entity=switch1 type=Switch AvailableNodeCntinLevel1=1 AvailablCpuCntInLevel1=24 Enclosed=switch[3-4]

Entity=switch2 type=Switch AvailableNodeCntinLevel1=0 AvailablCpuCntInLevel1=0 Enclosed=switch[5-6]

Entity=switch3 type=Switch AvailableNodeCntinLevel0=1 AvailablCpuCntInLevel0=8 Enclosed=virtual[0-1999]

Entity=socket4 type=Switch AvailableNodeCntinLevel0=1 AvailablCpuCntInLevel0=16 Enclosed=core[2000-2999]

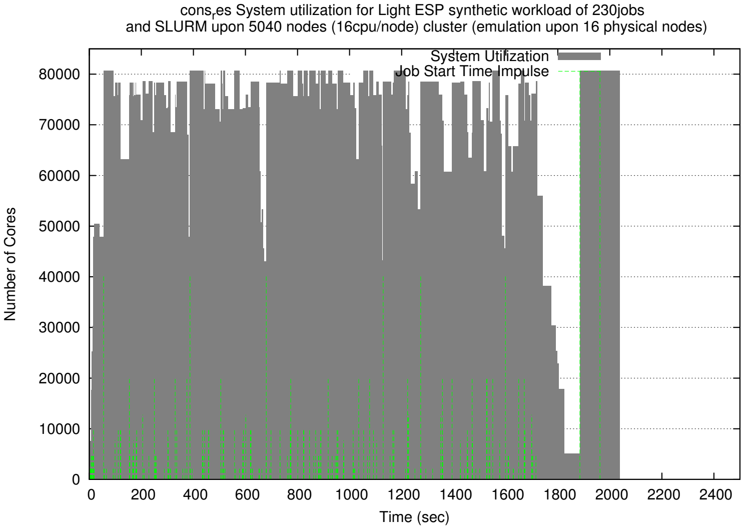
Entity=virtual0 type=Node AvailableNodeCnt=1 AvailablCpuCnt=8

Entity=virtual1999 type=Node AvailableNodeCnt=0 AvailablCpuCnt=0

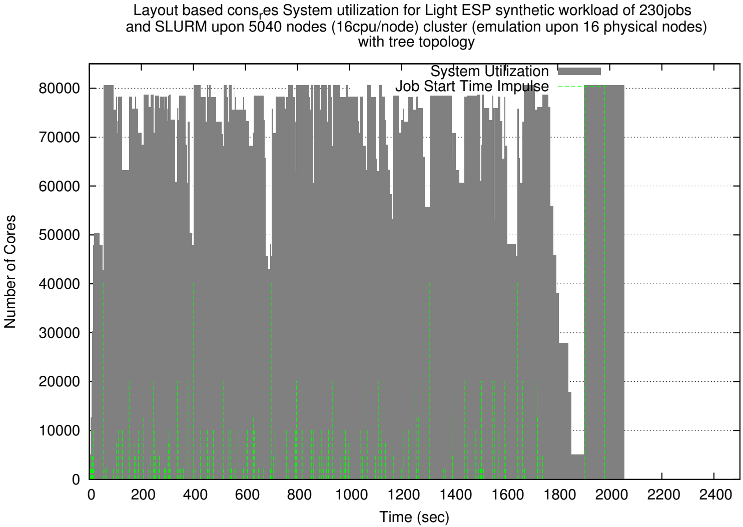
Entity=virtual2000 type=Node AvailableNodeCnt=1 AvailablCpuCnt=16

Entity=virtual2999 type=Node AvailableNodeCnt=0 AvailablCpuCnt=0

Cons_res System Utilization

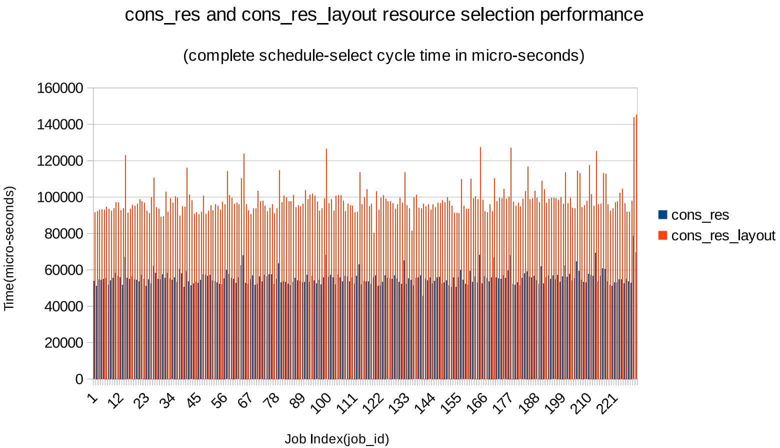


Cons_res_layout System Utilization

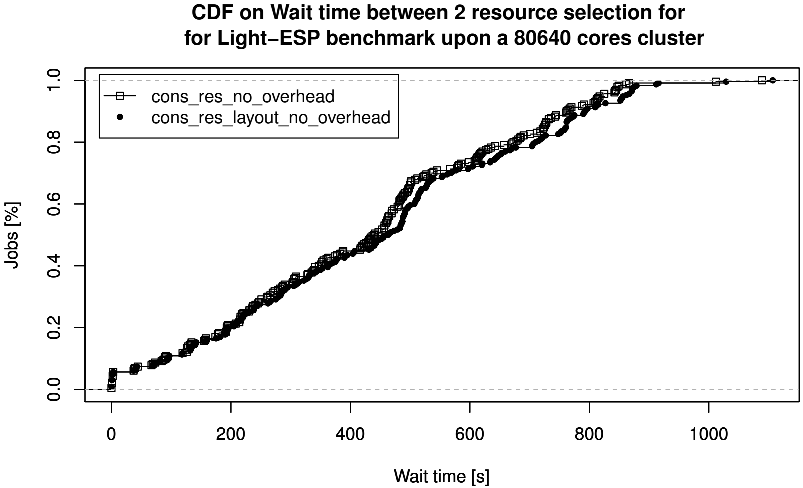


Individual Jobs Performance Comparison

1. **Cons_res_layout** individual resource selection time increased twice



Waiting Time for 2 Plugins



1. Introduction and Motivation

2. Background
   - Related Works
   - SLURM Architecture
   - LAYOUTS Framework

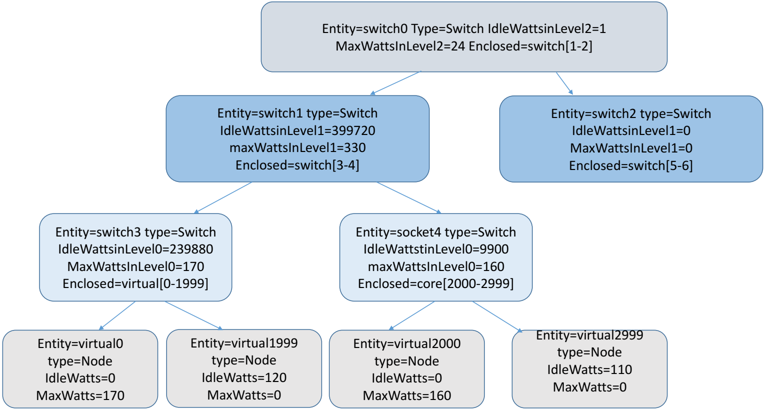
3. Resource Selection
   - LAYOUTS based consumable resource selection
   - Multi-parameter resource selection

4. Experimentation and Performance Evaluation
   - Cons_res_layout vs Cons_res Analysis
   - Multi-parameter vs Cons_res Analysis

5. Conclusion and Future Work
   - Conclusion
   - Future Work

Energy Experiment Environment

1. **Heterogeneous power consuming nodes** cluster environment
2. Simple tree topology to have **4 leaf switches** and **3 levels**
3. **Homogeneous power consuming nodes** within leaf switches



Entity=switch0 Type=Switch IdleWattsinLevel2=1
MaxWattsInLevel2=24 Enclosed=switch[1-2]

Entity=switch1 type=Switch
IdleWattsinLevel1=399720
maxWattsInLevel1=330
Enclosed=switch[3-4]

Entity=switch2 type=Switch
IdleWattsinLevel1=0
MaxWattsInLevel1=0
Enclosed=switch[5-6]

Entity=switch3 type=Switch
IdleWattsinLevel0=239880
MaxWattsInLevel0=170
Enclosed=virtual[0-1999]

Entity=socket4 type=Switch
IdleWattstinLevel0=9900
maxWattsInLevel0=160
Enclosed=core[2000-2999]

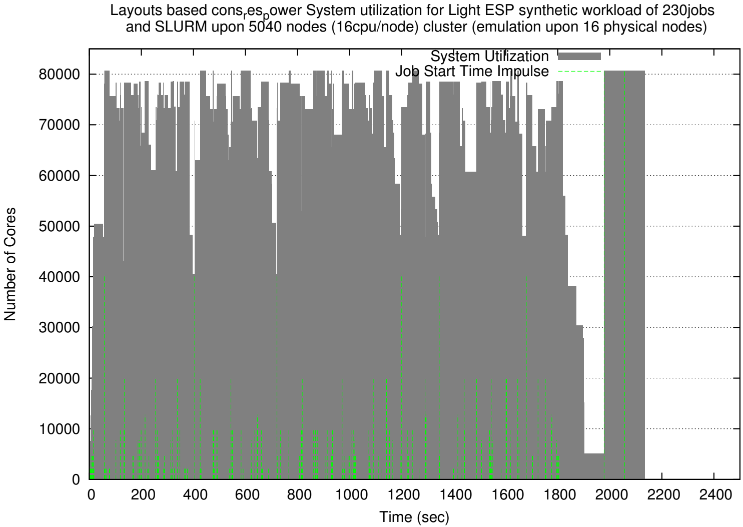
Entity=virtual0
type=Node
IdleWatts=0
MaxWatts=170

Entity=virtual1999
type=Node
IdleWatts=120
MaxWatts=0

Entity=virtual2000
type=Node
IdleWatts=0
MaxWatts=160

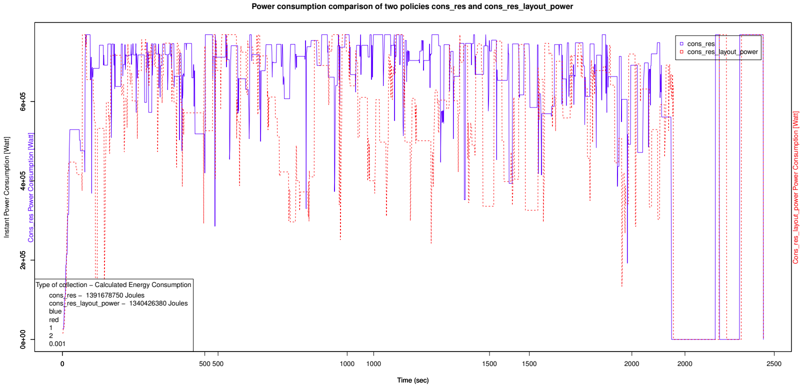
Entity=virtual2999
type=Node
IdleWatts=110
MaxWatts=0

Cons_res_power System Utilization



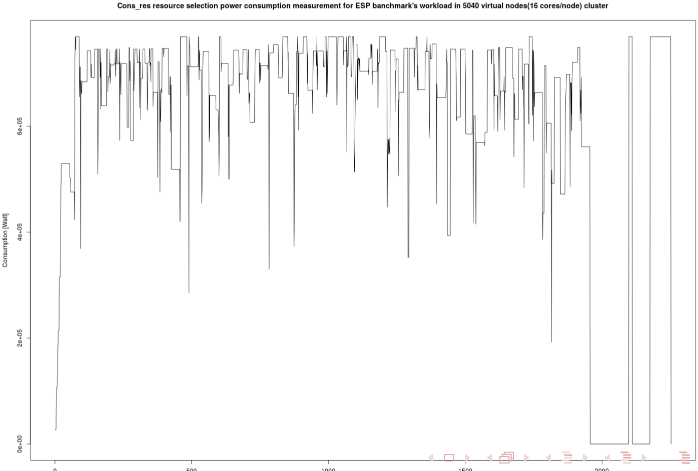
Energy Efficiency

1. **Cons_res_power** energy consumption is better than **Cons_res** by **3.8 %**

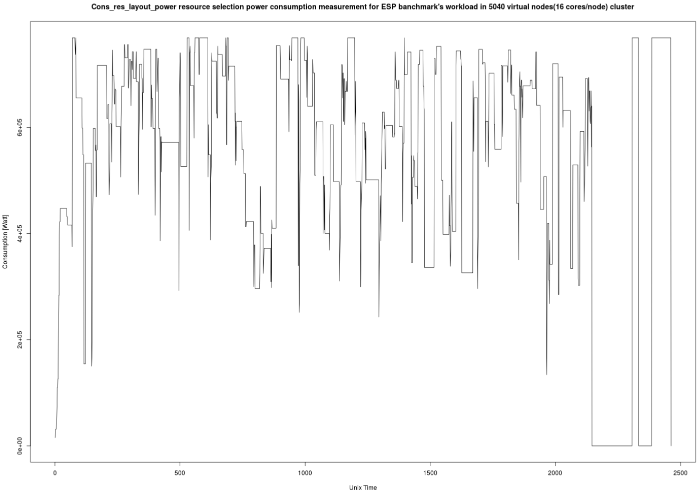


Cons_res Energy Consumption

1. For experiment, Heterogeneous cluster configured **homogeneous power consuming nodes** within the leaf switches

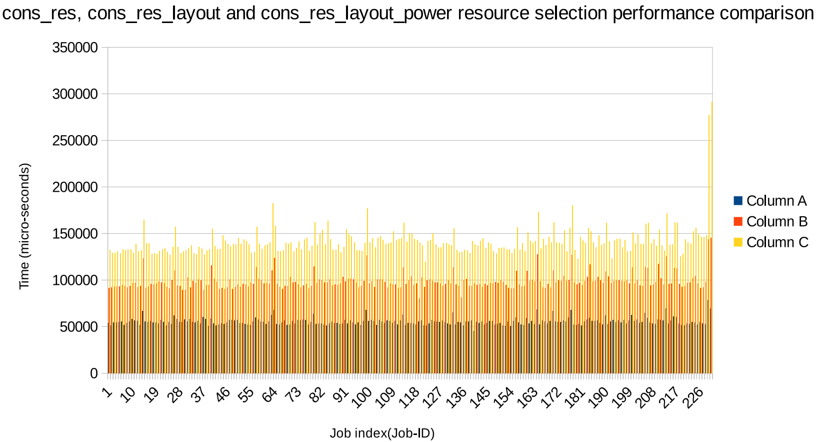


Cons_res_power Energy Consumption

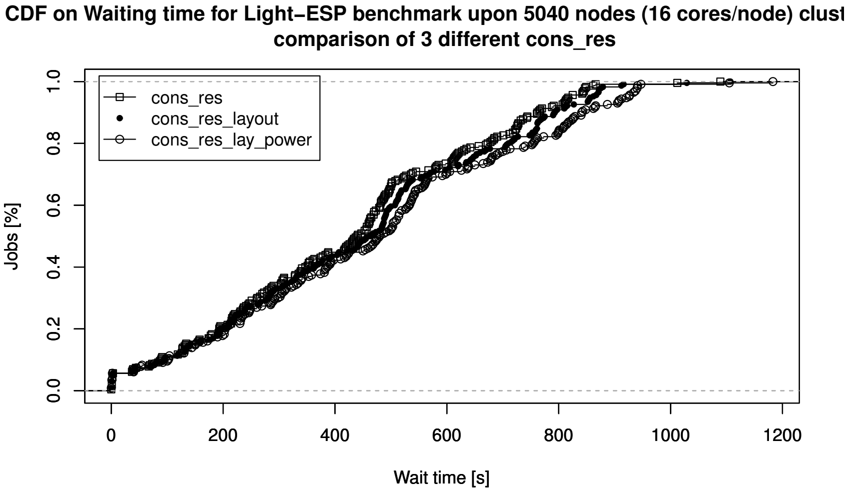


Individual Jobs Performance Comparison

1. **Cons_res_power** individual job resource selection time increased thrice than **cons_res**



Waiting Time for 3 Plugins



1. Introduction and Motivation
2. Background
   - Related Works
   - SLURM Architecture
   - LAYOUTS Framework
3. Resource Selection
   - LAYOUTS based consumable resource selection
   - Multi-parameter resource selection
4. Experimentation and Performance Evaluation
   - Cons_res_layout vs Cons_res Analysis
   - Multi-parameter vs Cons_res Analysis
5. **Conclusion and Future Work**
   - **Conclusion**
   - Future Work

Conclusion I

1. SLURM **plugin enhancement** is easier using LAYOUTS resource management framework

Conclusion I

1. SLURM **plugin enhancement** is easier using LAYOUTS resource management framework
2. **Cons_res_layout** resource selection plugin developed using LAYOUTS

Conclusion I

1. SLURM **plugin enhancement** is easier using LAYOUTS resource management framework
2. **Cons_res_layout** resource selection plugin developed using LAYOUTS
3. LAYOUTS **new APIs** developed to **increase performance** for multiple updates

Conclusion I

1. SLURM **plugin enhancement** is easier using LAYOUTS resource management framework
2. **Cons_res_layout** resource selection plugin developed using LAYOUTS
3. LAYOUTS **new APIs** developed to **increase performance** for multiple updates
4. Cons_res_layout and cons_res plugin **system utilisation and throughput** is **same**

Conclusion I

1. SLURM **plugin enhancement** is easier using LAYOUTS resource management framework
2. **Cons_res_layout** resource selection plugin developed using LAYOUTS
3. LAYOUTS **new APIs** developed to **increase performance** for multiple updates
4. Cons_res_layout and cons_res plugin **system utilisation and throughput** is **same**
5. Cons_res_layout individual resource selection performance overhead was **increased twice** than cons_res plugin
6. Cons_res_layout **job waiting time increased**, due to individual performance of resource selection

Conclusion II

1. Multi-parameter resource selection policy implemented in **Cons_res_power** plugin

Conclusion II

1. Multi-parameter resource selection policy implemented in **Cons_res_power** plugin
2. Cons_res_power plugin achieved both **application performance** and **server energy efficiency**

Conclusion II

1. Multi-parameter resource selection policy implemented in **Cons_res_power** plugin

2. Cons_res_power plugin achieved both **application performance** and **server energy efficiency**

3. Energy consumption of multi-parameter resource selection reduced by 3.8 %.

4. Cons_res_power individual resource selection performance overhead was **increased thrice** than cons_res plugin

5. Cons_res_power **job waiting time increased**, due to individual performance of resource selection

Future Work

1. Include **partition** entities in the LAYOUTS

Future Work

1. Include **partition** entities in the LAYOUTS
2. Adapt cons_res_power to support real energy values from **RAPL** or **IPMI** technique

Future Work

1. Include **partition** entities in the LAYOUTS
2. Adapt cons_res_power to support real energy values from **RAPL** or **IPMI** technique
3. Include **temperature** criteria in the resource selection

Future Work

1. Include **partition** entities in the LAYOUTS
2. Adapt cons_res_power to support real energy values from **RAPL** or **IPMI** technique
3. Include **temperature** criteria in the resource selection
4. Experiment to measure
   - Instantaneous job throughput
   - Instantaneous number of job types allocated
   - Cons_res_ power job waiting time is better than **powercapping** only approach

Thank you ..

Any questions?