Optimizing the RJMS of an Academic HPC & Research Computing Facility

Dr. S. Varrette, Dr. E. Kieffer, Dr. F. Pinel

University of Luxembourg (UL), Luxembourg

https://hpc.uni.lu

21st IEEE Intl. Symp. on Parallel and Distributed Computing (ISPDC'22)

July 13th, 2022, Basel, Switzerland

Summary

1 Introduction: Context and Motivations

2 Toward Improved User Job Management through a Novel RJMS configuration

3 Impact Analysis of the Updated Models and Policies through Workload Performance Evaluation

4 Conclusion & Perspectives

Summary

1. **Introduction: Context and Motivations**

2. Toward Improved User Job Management through a Novel RJMS configuration

3. Impact Analysis of the Updated Models and Policies through Workload Performance Evaluation

4. Conclusion & Perspectives

UL HPC Facility

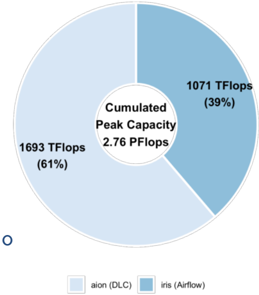
- **Managed and operated since 2007** (Dr. S. Varrette & Co.)
  ↪ 2nd Largest HPC facility in Luxembourg after EuroHPC MeluXina

hpc.uni.lu

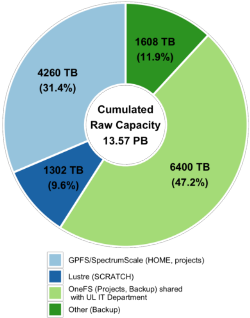
Technical Docs:
hpc-docs.uni.lu

ULHPC Tutorials:
ulhpc-tutorials.rtfd.io

UL HPC Supercomputers (2022)

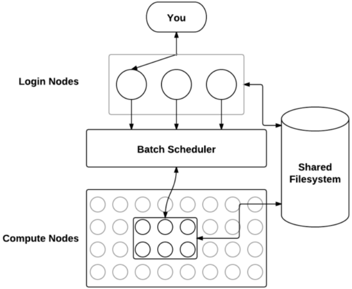


UL HPC Storage FileSystems (2022)



Resource and Job Management Systems

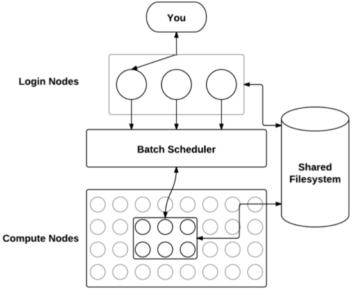
- **Resource and Job Management System (RJMS)**
  - ↪ *Glue* for a parallel computer to execute parallel jobs
  - ↪ **Goal**: satisfy users demands for computation
    - ✓ assign resources to user jobs with an efficient manner



Resource and Job Management Systems

- **Resource and Job Management System (RJMS)**
  - ↪ *Glue* for a parallel computer to execute parallel jobs
  - ↪ **Goal**: satisfy users demands for computation
    - ✓ assign resources to user jobs with an efficient manner

- **HPC Resources**:
  - ↪ Nodes (typically a unique IP address)
    - ✓ Sockets / Cores / Hyperthreads
    - ✓ Memory
    - ✓ Interconnect/switch resources
  - ↪ Generic resources (e.g. GPUs)
  - ↪ Licenses
- **Strategic Position**
  - ↪ Direct/constant knowledge of resources
  - ↪ Launch and otherwise manage jobs



Slurm on ULHPC clusters

- ULHPC uses Slurm for cluster/resource management and job scheduling
  - ↪ **Simple Linux Utility for Resource Management** `https://slurm.schedmd.com/`
  - ↪ Reference RJMS serving most of Top500 systems
    - ✓ official documentation, tutorial, FAQ

Slurm on ULHPC clusters

- ULHPC uses Slurm for cluster/resource management and job scheduling
  - ↪ **Simple Linux Utility for Resource Management** `https://slurm.schedmd.com/`
  - ↪ Reference RJMS serving most of Top500 systems
    - ✓ official documentation, tutorial, FAQ

- **Seminal configuration part of flagship `iris` production release (in 2017)**
  - ↪ migration from OAR RJMS
  - ↪ Slurm provides superior scalability and performance [JSSPP12], inherent compatibility with multiple distributed libraries (Dask, IPyparallel. . .) and MPI suits (though PMI[x])
    - ✓ brings a more convenient and flexible interface for non-specialists
  - ↪ inspired from other HPC centers (LLNL, Niflheim, CSCS. . .) & Simulators [PEARC18]

[JSSPP12] Y. Georgiou, M. Hautreux "*Evaluating scalability and efficiency of the resource and job management system on large HPC clusters*", in W. on Job Scheduling Strategies for Parallel Processing (LNCS JSSPP'12), Springer, pp. 134–156 (2012).
[PEARC18] N. A. Simakov & al. "*Slurm Simulator: Improving Slurm Scheduler Performance on Large HPC Systems by Utilization of Multiple Controllers and Node Sharing*" in Proc. of the ACM Practice and Experience on Advanced Research Computing (PEARC'18) (2018).

Witnessed Limitations of the Initial Configuration

After 3 years of production on `iris`... (1/2)

- **Over-complexified setup** for partition/QOS
  ↪ integration of hetegeneous hardware (GPU nodes, Large-memory) starting 2018

Witnessed Limitations of the Initial Configuration

After 3 years of production on `iris`... (1/2)

- **Over-complexified setup** for partition/QOS
  - ↪ integration of hetegeneous hardware (GPU nodes, Large-memory) starting 2018
  - ↪ **under vs. over used partitions**
    - ✓ `batch` partition (Dual-CPU regular nodes) **saturated** and **over**-used
    - ✓ dedicated resources for `interactive` or `long` (**regular nodes only**) **under**-used

Witnessed Limitations of the Initial Configuration

After 3 years of production on `iris`... (1/2)

- **Over-complexified setup** for partition/QOS
  - ↪ integration of hetegeneous hardware (GPU nodes, Large-memory) starting 2018
  - ↪ **under vs. over used partitions**
    - ✓ `batch` partition (Dual-CPU regular nodes) **saturated** and **over**-used
    - ✓ dedicated resources for `interactive` or `long` (**regular nodes only**) **under**-used
  - ↪ for each partition, associated **QOS queue named** `qos-<partitionname>[-XXX]`
    - ✓ allows for specific research groups/industrial partners privileged/exclusive access
    - ✓ **Ex**: `qos-batch` (default), `qos-batch-[001-...]` (Group Prof. / Partner $XX_{\{1-..\}}$)
    - ✓ **Ex**: `qos-gpu` (default), `qos-gpu-[001-...]` (Group Prof. / Partner $YY_{\{1-..\}}$)
    - ✓ **Ex**: `qos-covid` ultra high priority jobs to support fight against COVID-19
  - ↪ In practice, this setup brought **frustration, jealousy & confusion within users**

Witnessed Limitations of the Initial Configuration

After 3 years of production on `iris`... (2/2)

- **No cross-partition QOS** except `best-effort` (preemptible jobs):
  - ↪ interactive jobs on **non-regular** nodes (GPU, large-memory) artificially complexified
  - ↪ each specific usage treated by dedicated QOS (Ex: `qos-batch-0*`, `qos-covid`)
    - ✓ no global priority level (low→urgent), unreadability of QOS objectives vs. partition

Witnessed Limitations of the Initial Configuration

After 3 years of production on `iris`... (2/2)

- **No cross-partition QOS** except `best-effort` (preemptible jobs):
  - ↪ interactive jobs on **non-regular** nodes (GPU, large-memory) artificially complexified
  - ↪ each specific usage treated by dedicated QOS (Ex: `qos-batch-0*`, `qos-covid`)
    - ✓ no global priority level (low→urgent), unreadability of QOS objectives vs. partition
- Fairsharing relying on the **Depth-Oblivious Fair-share Factor** **algorithm**
  - ↪ variant of classical fair-share factor, increases usable priority ranges
    - ✓ very complex algorithm, hard to explain impacts/issues with job priority
  - ↪ **no policy for raw-share attribution** lead to unfair situation
  - ↪ hard to evaluate/size associated Slurm parameters Ex: `PriorityWeight*` (among others)

Witnessed Limitations of the Initial Configuration

After 3 years of production on `iris`... (2/2)

- **No cross-partition QOS** except `best-effort` (preemptible jobs):
  - ↪ interactive jobs on **non-regular** nodes (GPU, large-memory) artificially complexified
  - ↪ each specific usage treated by dedicated QOS (Ex: `qos-batch-0*`, `qos-covid`)
    - ✓ no global priority level (low→urgent), unreadability of QOS objectives vs. partition
- Fairsharing relying on the **Depth-Oblivious Fair-share Factor algorithm**
  - ↪ variant of classical fair-share factor, increases usable priority ranges
    - ✓ very complex algorithm, hard to explain impacts/issues with job priority
  - ↪ **no policy for raw-share attribution** lead to unfair situation
  - ↪ hard to evaluate/size associated Slurm parameters Ex: `PriorityWeight*` (among others)
- **Incomplete Account hierarchy**
  - ↪ not able to cover new workload requests associated to funded projects/training events
  - ↪ novel auditing capabilities on the platform usage / cost model requested
    - ✓ ... and need to be integrated in the Slurm configuration

Toward a Novel RJMS Setup

- **Acquisition & integration of new liquid-cooled supercomputer** `aion`
  - ↪ occasion to *deeply* review and **optimize** the seminal configuration
  - ↪ **mitigate the identified pitfalls** & take advantage of experience gained

Toward a Novel RJMS Setup

- **Acquisition & integration of new liquid-cooled supercomputer** `aion`
  - ↪ occasion to *deeply* review and **optimize** the seminal configuration
  - ↪ **mitigate the identified pitfalls** & take advantage of experience gained

In this talk: ULHPC Slurm configuration 2.0

1. Re-defining the partition, QOS and accounting model
   - ↪ offer a more focused and easy-to-use configuration
2. Rewiew fair-sharing model giving **incentives** to good practices
   - ↪ take into account monetary contribution to increase priority
   - ↪ tribute to past **efficient** usage
3. rethinking global resources limits set to the partitions / accounts associations / QOS
4. consolidating the RJMS setup for HA services
5. define common [federated vs. multi-cluster] scheduling / accounting policy

Summary

1. Introduction: Context and Motivations
2. **Toward Improved User Job Management through a Novel RJMS configuration**
3. Impact Analysis of the Updated Models and Policies through Workload Performance Evaluation
4. Conclusion & Perspectives

ULHPC Slurm Partitions 2.0 -p, -partition=<partition>

- **Tied to global types/classes of available computing nodes** batch[,gpu][,bigmem]
- New **floating** partition interactive across all nodes for (short and quick) tests
  - ↪ selection of the expected resource type through **feature** -C broadwell,skylake,gpu,volta[32]...
  - ↪ **backfill scheduling** enabled and optimized to favor interactive and/or small jobs if queued
- **Max wall clock time** for user job on non-floating partition **reduced** from 5 **to 2 days**

AION Partition	**Type**	**#Node**	**PriorityTier**	**DefaultTime**	**MaxTime**	**MaxNodes**
all (sysadmins)	**hidden**	318	100	10h	5d	UNLIMITED
interactive	floating	318	100	30min	2h	2
batch		318	1	2h	48h	64

IRIS Partition	**Type**	**#Node**	**PriorityTier**	**DefaultTime**	**MaxTime**	**MaxNodes**
all (sysadmins)	**hidden**	196	100	10h	5d	UNLIMITED
interactive	floating	196	100	30min	2h	2
batch		168	1	2h	48h	64
gpu		24	1	2h	48h	4
bigmem		4	1	2h	48h	1

ULHPC Slurm QOS 2.0

`--qos=<qos>`

- New **Cross-partition QOS**, mainly tied to **priority level** (`low` → `urgent`)
  - ↪ Simpler names than before (i.e. no more `qos-` prefix)
  - ↪ special **preemptible QOS** kept for best-effort jobs: `besteffort`
  - ↪ new `long` QOS allows to run jobs for up to 14 days (instead of the default 2 days)
- Further limits on Slurm **Trackable RESources** (**TRES**)

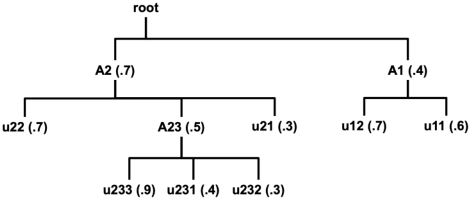
QOS	Partition	Allowed [L1] Account	Prio	GrpTRES	MaxTresPJ	MaxJobPU	Flags
besteffort	*	**ALL**	1			100	NoReserve
low	*	**ALL** (default for CRP/externals)	10			2	DenyOnLimit
normal	*	**Default** (UL,Projects,. . .)	100			50	DenyOnLimit
long	*	UL,Projects,etc.	100	node=12	node=2	4	DenyOnLimit,PartitionTimeLimit
debug	interactive	**ALL**	150	node=8		2	DenyOnLimit
high	*	(**restricted**) UL,Projects,Industry	200			10	DenyOnLimit
urgent	*	(**restricted**) UL,Projects,Industry	1000			100	DenyOnLimit
admin	all	(**restricted**) sysadmins	1000				DenyOnLimit

ULHPC Fairsharing 2.0

- Three fairsharing algorithms implemented in Slurm:
  - ↪ Classic Fairshare, Depth-Oblivious Fair-share (***initial* setup**) and Fair Tree
  - ↪ Thourough evaluation of all 3 fairshare algorithms (Python-based simulator)

ULHPC Fairsharing 2.0

- Three fairsharing algorithms implemented in Slurm:
  - ↪ Classic Fairshare, Depth-Oblivious Fair-share (***initial* setup**) and Fair Tree
  - ↪ Thourough evaluation of all 3 fairshare algorithms (Python-based simulator)
- In **Fair Tree**, all users from higher priority account receive higher fair share factor
  - ↪ . . . when compared to all users from a lower priority account
  - ↪ Done with rooted plane tree (rooted ordered tree)
    - ✓ logically created then sorted by fairshare level with highest fairshare values on the left
    - ✓ tree is then visited in a depth-first traversal way



ULHPC Fairsharing 2.0

- Three fairsharing algorithms implemented in Slurm:
  - ↪ Classic Fairshare, Depth-Oblivious Fair-share (***initial* setup**) and Fair Tree
  - ↪ Thourough evaluation of all 3 fairshare algorithms (Python-based simulator)

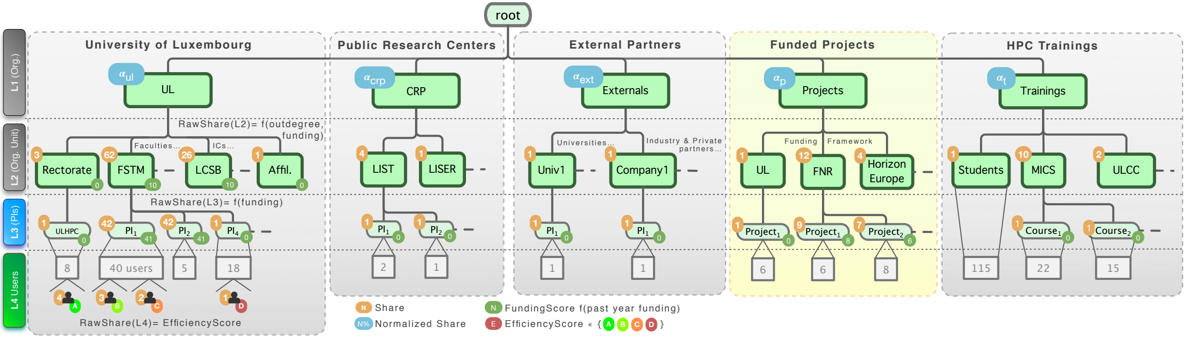
- **New configuration with Multifactor Priority Plugin and Fair tree algorithm**
  - ↪ **efficiency**: new jobs are immediately assigned a priority
  - ↪ fairshare levels are more easily understandable
  - ↪ **YET quite sensitive to the raw shares associated to each user account**

- Necessity to **deeply restructure** associations & shares in the accounting DB
  - ↪ formalize consistent rules to attribute raw shares
  - ↪ give novel **incentives** to good practices
    - ✓ take into account monetary contribution to increase priority
    - ✓ tribute to past **efficient** usage

Account Hierarchy 2.0

- **Accounting records re-organized as a hierarchical tree** (3 layers $L_{1,2,3}$ + leafs)
  - ↪ **L1**: **Organization Level**: UL, CRPs, Externals, Projects, Trainings
    - ✓ guarantee 85% of the shares for core UL activities
  - ↪ **L2**: **Organizational Unit** (Faculty, ICs, External partner, Funding program. . .)
    - ✓ raw share depends on **outdegree** and **funding score**
  - ↪ **L3**: Principal Investigator (**PIs**), **Projects**, **Course**
    - ✓ raw share depends on **funding score** (different weight)
    - ✓ eventually restricted **only** to projects and courses
  - ↪ **L4**: **End User** (ULHPC login)
    - ✓ Raw share based on **efficiency score**

Account Hierarchy 2.0



Funding Score (L2/L3)

- Associated with account A belonging to level L in the hierarchy
  - ↪ yearly updated at the beginning of the year
  - ↪ depreciation based on contribution type, weighted by level threshold β_L

$$\texttt{FundingScore}_L(A) = \left\lfloor \beta_L \frac{Investment_A(Year - 1)}{\#months} \right\rfloor$$

Funding Score (L2/L3)

- Associated with account A belonging to level L in the hierarchy
  - ↪ yearly updated at the beginning of the year
  - ↪ depreciation based on contribution type, weighted by level threshold β_L

$$\texttt{FundingScore}_L(A) = \left\lfloor \beta_L \frac{Investment_A(Year-1)}{\#months} \right\rfloor$$

- **Ex1**: Exceptional contribution of 120K€ performed in 2021 by a faculty (L2 account A)
  - ↪ depreciation: 12 months (*default*)
  - ↪ **funding score in 2022**: $\left\lfloor \beta_{L_2} \frac{120000}{12} \right\rfloor = \lfloor \beta_{L_2} \times 10000 \rfloor$.
- **Ex2**: let P be a project granted in 2021 to start in 2022 for a duration of 36 months
  - ↪ **budget**: 27K€ allocated for HPC costs
  - ↪ **funding score for the years 2022, 2023 and 2024**: $\left\lfloor \beta_{L_3} \frac{27000}{36} \right\rfloor = \lfloor \beta_{L_3} \times 750 \rfloor$

Efficiency Score (L4)

- **Updated every year based on past jobs efficiency.**
  - ↪ Similar notion of "nutri-score': A(very good - 3), B (good: 2), C (bad, 1), D(very bad - 0)
- Proposed Metric for **user** U: **Average Wall-time Accuracy (WRA)** (higher the better)
  - ↪ Defined for a given time period (past year)

```
sacct -u <U> -X -S <start> -E <end> [...] # --format User,JobID,state,time,elapsed
```

$$\begin{aligned} S_{\text{efficiency}}(U, Year) &= \text{WRA}(U, Year) \\ &= \frac{1}{N} \sum_{JobID} \frac{T_{\text{elapsed}}(JobID)}{T_{\text{asked}}(JobID)} \end{aligned}$$

Efficiency Score (L4)

- **Updated every year based on past jobs efficiency.**
  - ↪ Similar notion of "nutri-score': A(very good - 3), B (good: 2), C (bad, 1), D(very bad - 0)
- Proposed Metric for **user** U: **Average Wall-time Accuracy (WRA)** (higher the better)
  - ↪ Defined for a given time period (past year)

```
sacct -u <U> -X -S <start> -E <end> [...] # --format User,JobID,state,time,elapsed
```

$$S_{\text{efficiency}}(U, Year) = \text{WRA}(U, Year) = \frac{1}{N} \sum_{JobID} \frac{T_{\text{elapsed}}(JobID)}{T_{\text{asked}}(JobID)}$$

- U raw share: $1 + S_{\text{efficiency}}(U, Year)$

Score	Avg. WRA
A (3) very good	$S_{\text{efficiency}} \geq 75\%$
B (2) good	$50\% \leq S_{\text{efficiency}} < 75\%$
C (1) bad	$25\% \leq S_{\text{efficiency}} < 50\%$
D (0) very bad	$S_{\text{efficiency}} < 25\%$

Efficiency Score (L4)

- **Updated every year based on past jobs efficiency.**
  - ↪ Similar notion of "nutri-score": A(very good - 3), B (good: 2), C (bad, 1), D(very bad - 0)
- Proposed Metric for **user** U: **Average Wall-time Accuracy (WRA)** (higher the better)
  - ↪ Defined for a given time period (past year)

```
sacct -u <U> -X -S <start> -E <end> [...] # --format User,JobID,state,time,elapsed
```

$$S_{\text{efficiency}}(U, Year) = \text{WRA}(U, Year)$$
$$= \frac{1}{N} \sum_{JobID} \frac{T_{\text{elapsed}}(JobID)}{T_{\text{asked}}(JobID)}$$

- U raw share: $1 + S_{\text{efficiency}}(U, Year)$

Score	Avg. WRA
A (3) very good	$S_{\text{efficiency}} \geq 75\%$
B (2) good	$50\% \leq S_{\text{efficiency}} < 75\%$
C (1) bad	$25\% \leq S_{\text{efficiency}} < 50\%$
D (0) very bad	$S_{\text{efficiency}} < 25\%$

- **WIP**: integrate other efficiency metrics (CPU, mem, GPU efficiency)

Job Accounting and Billing

- **Utilization** of the University computational resources is **charged in Service Unit (SU)**
  - ↪ 1 SU $\simeq$ 1 hour on 1 physical processor core on regular computing node
  - ↪ Usage charged **0,03€ per SU (VAT excluded)** (external partners, funded projects etc.)
- A **job is characterized*** (and thus billed) according to the following elements:
  - ↪ $T_{\textbf{exec}}$: Execution time (in hours)
  - ↪ $N_{\textbf{Nodes}}$: number of computing nodes, and **per node**:
    - ✓ $N_{\textbf{cores}}$: number of CPU cores allocated per node
    - ✓ *Mem*: memory size allocated per node, in GB
    - ✓ $N_{\textbf{gpus}}$: number of GPU allocated per node
  - ↪ associated weighted factors $\alpha_{cpu}, \alpha_{mem}, \alpha_{GPU}$ defined as `TRESBillingWeight` in Slurm
    - ✓ account for consumed resources other than just CPUs, taken into account in fairshare factor
    - ✓ α_{cpu}: normalized relative perf. of CPU processor core (reference: skylake 73,6 GFlops/core)
    - ✓ α_{mem}: inverse of the average available memory size per core
    - ✓ α_{GPU}: weight per GPU accelerator

Job Accounting and Billing

Number of SU associated to a job

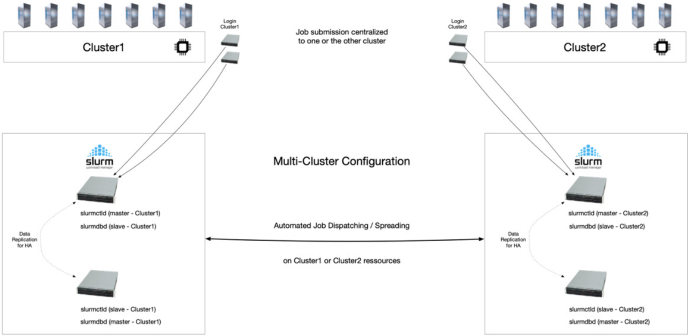
$$N_{\text{Nodes}} \times [\alpha_{cpu} \times N_{\text{cores}} + \alpha_{mem} \times Mem + \alpha_{gpu} \times N_{\text{gpus}}] \times T_{\text{exec}}$$

- **Current billing weights:**

Cluster	Node Type	Partition	#Cores/node	CPU	α_{cpu}	α_{mem}	α_{GPU}
Iris, Aion	Regular	`interactive`	28/128	`n/a`	0	0	0
Iris	Regular	`batch`	28	`broadwell`	0.522	$\frac{1}{4} = 0, 25$	0
Iris	Regular	`batch`	28	`skylake`	1.0	$\frac{1}{4} = 0, 25$	0
Iris	GPU	`gpu`	28	`skylake`	1.0	$\frac{1}{27}$	50
Iris	Large-Mem	`bigmem`	112	`skylake`	1.0	$\frac{1}{27}$	0
Aion	Regular	`batch`	128	`epyc`	0,57	$\frac{1}{1.75}$	0

Federated vs. Multi-Cluster Scheduling Strategy

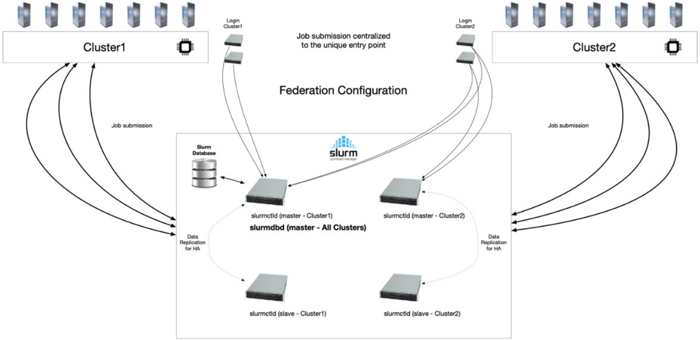
- **Two possible approaches** to integrating `aion` into the existing RJMS configuration
  - ↪ Multi-Cluster: allow to submit jobs across each clusters `sbatch -M {iris,aion} [...]`


Login Cluster1
Job submission centralized to one or the other cluster
Login Cluster2
Cluster1
Cluster2
slurm
Multi-Cluster Configuration
slurm
slurmctld (master - Cluster1)
slurmdbd (slave - Cluster1)
Automated Job Dispatching / Spreading
slurmctld (master - Cluster2)
slurmdbd (slave - Cluster2)
Data Replication for HA
on Cluster1 or Cluster2 ressources
Data Replication for HA
slurmctld (slave - Cluster1)
slurmdbd (master - Cluster1)
slurmctld (slave - Cluster2)
slurmdbd (master - Cluster2)

[Source : JOG, Oct 2020]

Federated vs. Multi-Cluster Scheduling Strategy

- **Two possible approaches** to integrating `aion` into the existing RJMS configuration
  - ↪ Multi-Cluster: allow to submit jobs across each clusters `sbatch -M {iris,aion} [...]`
  - ↪ Federation: P2P replication of job submitted locally, distributed schedule attempt
    - ✓ unified job ID


Cluster1
Login Cluster1
Job submission centralized to the unique entry point
Login Cluster2
Cluster2
Federation Configuration
Job submission
Job submission
Slurm Database
slurm
slurmctld (master - Cluster1)
slurmdbd (master - All Clusters)
slurmctld (master - Cluster2)
Data Replication for HA
Data Replication for HA
slurmctld (slave - Cluster1)
slurmctld (slave - Cluster2)

[Source : JOG, Oct 2020]

Federated vs. Multi-Cluster Scheduling Strategy

- **Two possible approaches** to integrating `aion` into the existing RJMS configuration
  - ↪ Multi-Cluster: allow to submit jobs across each clusters `sbatch -M {iris,aion} [...]`
  - ↪ Federation: P2P replication of job submitted locally, distributed schedule attempt
    - ✓ unified job ID

Selected Setup: Hybrid Multi-Cluster Scheduling Strategy

- **Redundant master/slave RJMS controllers** associated to each cluster
- **Shared** (like for federations), the **same Slurm database service** (`slurmdb`)
  - ↪ facilitate and centralize accounting information and management

Summary

1. Introduction: Context and Motivations

2. Toward Improved User Job Management through a Novel RJMS configuration

3. **Impact Analysis of the Updated Models and Policies through Workload Performance Evaluation**

4. Conclusion & Perspectives

Performance Evaluation and Experimental Setup

- **Impact of the updated RJMS policy and setup particularly hard to qualify**
  - ↪ Approach 1: **Slurm simulators** of workloads execution. Ex: BSC `slurm_simulator` [PMBS18]
    - ✓ **YET** old/obsolete version of Slurm featured (17.x)
    - ✓ novel proposed contributions absent and thus to implement...

[PMBS18] A. Jokanovic & al. "*Evaluating SLURM Simulator with Real-Machine SLURM and Vice Versa*", in W. on Perf. Modeling, Benchmarking and Simulation of HPC Systems (PMBS'18). pp. 72–82 (2018)
[PEARC18] N. A. Simakov & al. "*Slurm Simulator: Improving Slurm Scheduler Performance on Large HPC Systems by Utilization of Multiple Controllers and Node Sharing*" in Proc. of the ACM Practice and Experience on Advanced Research Computing (PEARC'18) (2018).

Performance Evaluation and Experimental Setup

- **Impact of the updated RJMS policy and setup particularly hard to qualify**
  - ↪ Approach 1: **Slurm simulators** of workloads execution. Ex: BSC `slurm_simulator` [PMBS18]
    - ✓ **YET** old/obsolete version of Slurm featured (17.x)
    - ✓ novel proposed contributions absent and thus to implement...
  - ↪ Approach 2: **Slurm Replay Engines**. Ex: CSCS `slurm-replay` [SC18]
    - ✓ suffers from the same pittfalls: old/obsolete version of Slurm featured (18.x)

[PMBS18] A. Jokanovic & al. "*Evaluating SLURM Simulator with Real-Machine SLURM and Vice Versa*", in W. on Perf. Modeling, Benchmarking and Simulation of HPC Systems (PMBS'18). pp. 72–82 (2018)
[PEARC18] N. A. Simakov & al. "*Slurm Simulator: Improving Slurm Scheduler Performance on Large HPC Systems by Utilization of Multiple Controllers and Node Sharing*" in Proc. of the ACM Practice and Experience on Advanced Research Computing (PEARC'18) (2018).
[SC18] M. Martinasso & al. "*RM-Replay: A High-Fidelity Tuning, Optimization and Exploration Tool for Resource Management*", in Proc. of the Intl. Conf. for High Performance Computing, Networking, Storage, and Analysis (SC'18) (2018)

Performance Evaluation and Experimental Setup

- **Impact of the updated RJMS policy and setup particularly hard to qualify**
  - ↪ Approach 1: **Slurm simulators** of workloads execution. Ex: BSC `slurm_simulator` [PMBS18]
    - ✓ **YET** old/obsolete version of Slurm featured (17.x)
    - ✓ novel proposed contributions absent and thus to implement...
  - ↪ Approach 2: **Slurm Replay Engines**. Ex: CSCS `slurm-replay` [SC18]
    - ✓ suffers from the same pittfalls: old/obsolete version of Slurm featured (18.x)

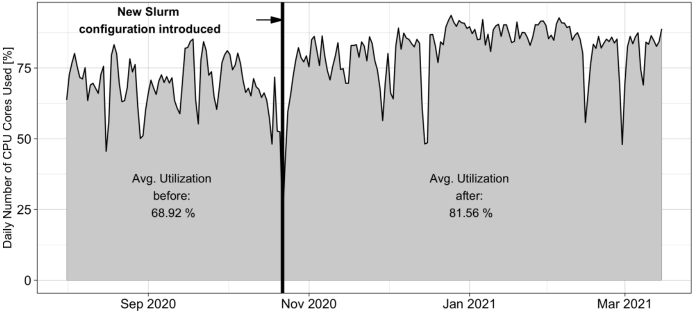
Our chance (Approach 3)

- **All the modifications**/proposals **applied at once within our production systems!**
  - ↪ on **Oct 22, 2020** during a maintenance session.
  - ↪ 1 supercomputer (`iris`) has thus seen **both** configs for a sufficient amount of time
    - ✓ workload analysis able to reasonably capture impact of the changes on RJMS perf.

Impact on the Total Utilization

- Impact of the updated Slurm configuration on the ULHPC relative utilization:
  - ↪ aggregates traces from several months of **uninterrupted** HPC services
  - ↪ **daily number of CPU cores used increased by 12.64%**: ≃ **81.56%** over 6 months

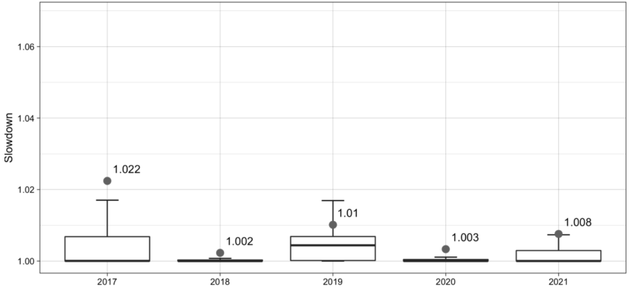
UL HPC Facility Relative Utilization (CPUHours)



Impact on the Job Slowdown

- **Slowdown**: traditional performance metrics for job scheduling strategies [Feitelson15]
  - ↪ RJMS response time normalized by the runtime: slowdown $= \frac{T_w + T_r}{T_r}$
  - ↪ **average yearly slowdown very close to the optimal value (1)**

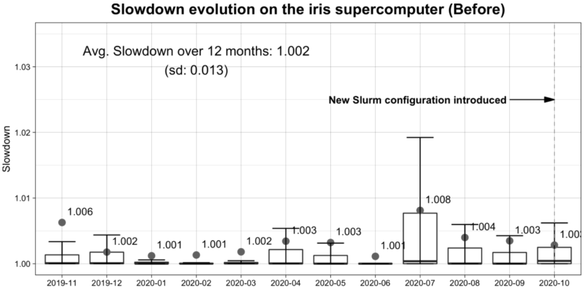
Slowdown evolution on the iris supercomputer



[Feitelson15] D. G. Feitelson, "*Workload Modeling for Computer Systems Performance Evaluation*, 1st ed. USA: Cambridge University Press (2015).

Impact on the Job Slowdown

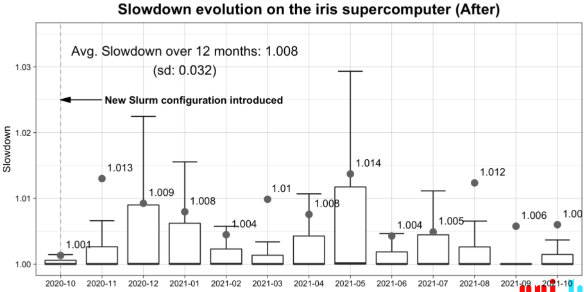
- More detailed analysis to better capture the impact of the updated RJMS config.
  - ↪ **Monthly slowdown** for all `iris` jobs run for 1 year period **before** and **after** the changes
  - ↪ **Fairly negligible impact**: observed avg. job slowdown **only increased by 0.59%**



Slowdown evolution on the iris supercomputer (Before)

Avg. Slowdown over 12 months: 1.002
(sd: 0.013)

New Slurm configuration introduced



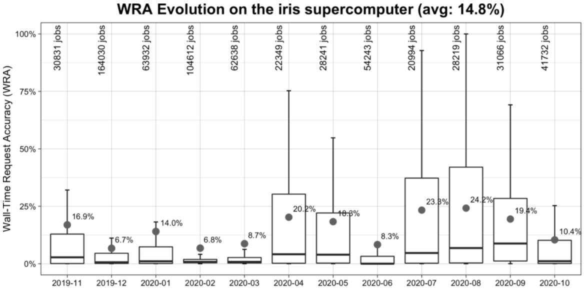
Slowdown evolution on the iris supercomputer (After)

Avg. Slowdown over 12 months: 1.008
(sd: 0.032)

New Slurm configuration introduced

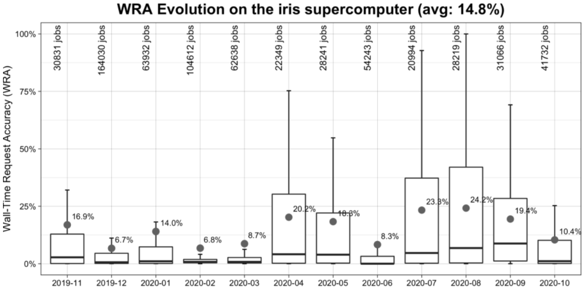
Impact on the Average Wall-time Request Accuracy

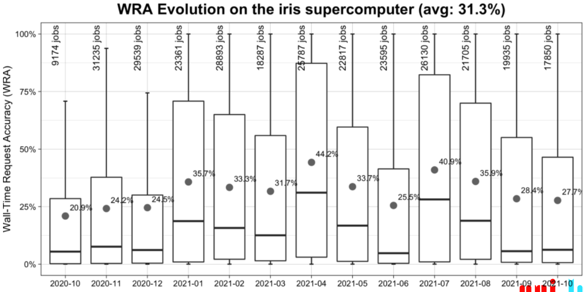
- Evaluation covering 1 year period **before** proposed configuration change



Impact on the Average Wall-time Request Accuracy

- Evaluation covering 1 year period **before** and **after** proposed configuration change
- **Avg. WRA for the processed jobs was increased by 110.81%**
  - ↪ moving from 14.8% on average to 31.3%

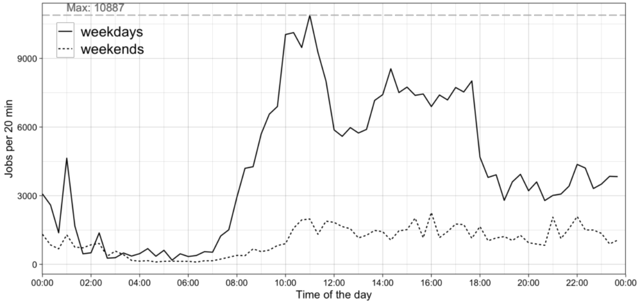

WRA Evolution on the iris supercomputer (avg: 14.8%)


WRA Evolution on the iris supercomputer (avg: 31.3%)

Current Daily Arrival Pattern

- **Proposed changes in production for 18 months and extensively used**
  - ↪ **Ex**: **Daily Arrival Pattern** extracted from `iris` **workload trace over 2021** Jan 1st → Dec 31th
    - ✓ **up to 12869 jobs submitted** (incl. 10887 outside weekends) and **processed per 20 min**

Daily job arrival patterns on the iris supercomputer



Summary

1 Introduction: Context and Motivations

2 Toward Improved User Job Management through a Novel RJMS configuration

3 Impact Analysis of the Updated Models and Policies through Workload Performance Evaluation

4 **Conclusion & Perspectives**

Conclusion

- **In this talk**:
  - ↪ **Optimization of Slurm RJMS config** when **acquiring** & **integrating** a new supercomputer
    - ✓ smooth integration within the existing HPC ecosystem
    - ✓ mitigation of the identified pittfalls of the initial configuration after 3y of production

Conclusion

- **In this talk**:
  - ↪ **Optimization of Slurm RJMS config** when **acquiring** & **integrating** a new supercomputer
    - ✓ smooth integration within the existing HPC ecosystem
    - ✓ mitigation of the identified pittfalls of the initial configuration after 3y of production
  - ↪ **Novel and flexible setup, adaptable to other HPC centers**
    - ✓ new scheme defining partitions, QOS queues, priorities & their resource limits
    - ✓ fairsharing mechanism updated, with **incentives** to good practices & monetary contribution
    - ✓ rededigned account hierarchy, **hybrid multi-cluster scheduling strategy**

Conclusion

- **In this talk**:
  - ↪ **Optimization of Slurm RJMS config** when **acquiring** & **integrating** a new supercomputer
    - ✓ smooth integration within the existing HPC ecosystem
    - ✓ mitigation of the identified pittfalls of the initial configuration after 3y of production
  - ↪ **Novel and flexible setup, adaptable to other HPC centers**
    - ✓ new scheme defining partitions, QOS queues, priorities & their resource limits
    - ✓ fairsharing mechanism updated, with **incentives** to good practices & monetary contribution
    - ✓ rededigned account hierarchy, **hybrid multi-cluster scheduling strategy**
  - ↪ **New cost model** and updated policies in production for 18 months
  - ↪ **Performance evaluation from real workload traces**

Conclusion

- **In this talk**:
  - ↪ **Optimization of Slurm RJMS config** when **acquiring** & **integrating** a new supercomputer
    - ✓ smooth integration within the existing HPC ecosystem
    - ✓ mitigation of the identified pittfalls of the initial configuration after 3y of production
  - ↪ **Novel and flexible setup, adaptable to other HPC centers**
    - ✓ new scheme defining partitions, QOS queues, priorities & their resource limits
    - ✓ fairsharing mechanism updated, with **incentives** to good practices & monetary contribution
    - ✓ rededigned account hierarchy, **hybrid multi-cluster scheduling strategy**
  - ↪ **New cost model** and updated policies in production for 18 months
  - ↪ **Performance evaluation from real workload traces**

- **Perspectives and Future directions**
  - ↪ **smooth integration with Euro-HPC infrastructures**
    - ✓ *transparently* outsource Research Computing/data analytic workflows to Tier-0 systems
  - ↪ **model & automatically offload from RJMS some of the less-demanding jobs**
    - ✓ target **dynamically allocated virtual cloud resources** (burst instances)

Thank you for your attention...

Questions?

Sebastien Varrette, Emmanuel Kieffer and Frederic Pinel
Optimizing the Resource and Job Management System of an Academic HPC & Research Computing Facility – IEEE ISPDC 2022

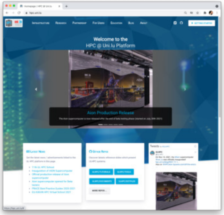
University of Luxembourg, Belval Campus
Maison du Nombre, 4th floor
2, avenue de l'Université
L-4365 Esch-sur-Alzette
mail: firsname.lastname@uni.lu

High Performance Computing @ Uni.lu
mail: hpc@uni.lu

1. Introduction: Context and Motivations
2. Toward Improved User Job Management through a Novel RJMS configuration
3. Impact Analysis of the Updated Models and Policies through Workload Performance Evaluation
4. Conclusion & Perspectives

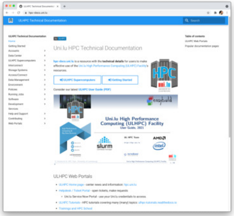
High Performance Computing @ Uni.lu

hpc.uni.lu



Welcome to the HPC @ Uni.lu Platform

Aion Production Release



Uni.lu HPC Technical Documentation

Uni.lu High Performance Computing (ULHPC) Facility User Guide, 2021

ULHPC Web Portals

ULHPC Technical Docs

hpc-docs.uni.lu

Appendix / Backup Slides

ULHPC Job Prioritization Factors

- **Age**: length of time a job has been waiting (PD state) in the queue
- **Fairshare**: difference between the portion of the computing resource that has been promised and the amount of resources that has been consumed
- **Partition**: factor associated with each node partition
  - ↪ Ex: privilege `interactive` over `batch`
- **QOS** A factor associated with each Quality Of Service (`low` ⟶ `urgent`)

```
Job_priority =
    PriorityWeightAge         * age_factor +
    PriorityWeightFairshare * fair-share_factor+
    PriorityWeightPartition * partition_factor +
    PriorityWeightQOS         * QOS_factor +
    - nice_factor
```

ULHPC Job Prioritization Factors

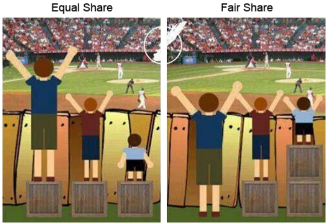
- **Age**: length of time a job has been waiting (PD state) in the queue
- **Fairshare**: difference between the portion of the computing resource that has been promised and the amount of resources that has been consumed
- **Partition**: factor associated with each node partition
  - ↪ Ex: privilege `interactive` over `batch`
- **QOS** A factor associated with each Quality Of Service (`low` ⟶ `urgent`)

```
Job_priority =
    PriorityWeightAge        * age_factor +
    PriorityWeightFairshare  * fair-share_factor+
    PriorityWeightPartition  * partition_factor +
    PriorityWeightQOS        * QOS_factor +
    - nice_factor
```

```
# Current weights on ULHPC platform
$ sprio -w # --format "%8i %5A %9F %9P %Q"
JOBID     AGE    FAIRSHARE PARTITION QOS
Weights   2000   3000      10000     1000
```

ULHPC Fairsharing 2.0

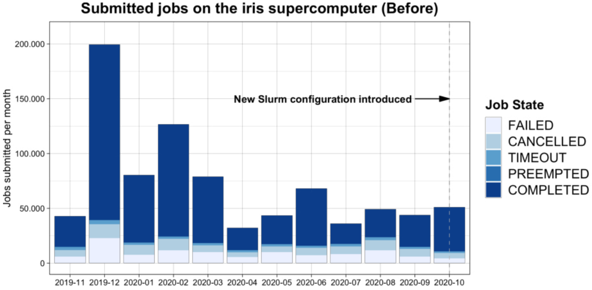
- **Fairsharing**: way of ensuring that users get their appropriate portion of a system
  - ↪ **Share**: portion of the system users have been granted.
  - ↪ **Usage**: amount of the system users have actually **used**.
  - ↪ **Fairshare score**: value the system calculates based off of user's usage.
    - ✓ difference between the portion of the computing resource that has been promised and the amount of resources that has been consumed
  - ↪ **Priority score**: priority assigned based off of the user's fairshare score.



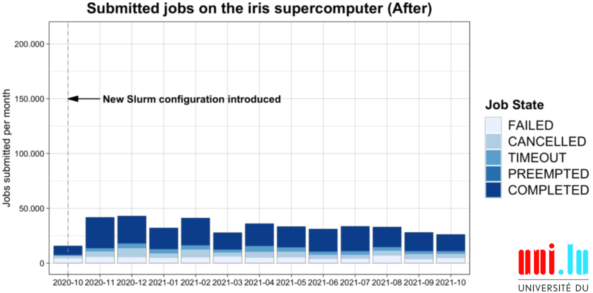
Impact on the Daily submitted Jobs

- Obviously depends on the usage pattern of the platform
  - ↪ Avg. #**monthly** submitted jobs: 71102 → 35384
  - ↪ Submission pattern changed and hard to conclude
    - ✓ `COMPLETED` job decreased. . .
    - ✓ Yet roughly eq. to `CANCELLED+TIMEOUT` increase (user-dependant)

Job State	Quantity Before	Quantity After	Difference
COMPLETED	73.3%	60.3%	**-13%**
PREEMPTED	0.12%	0.02%	**-0.1%**
FAILED	13.1%	15.2%	**+2.1%**
CANCELLED	10.4%	14.5%	**+4.1%**
TIMEOUT	3.06%	9.92%	**+6,86%**



Submitted jobs on the iris supercomputer (Before)



Submitted jobs on the iris supercomputer (After)