ITEA 2
INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

Bull
atos technologies

Improving resource allocation to reduce data movement overhead in applications with Slurm

Thomas Cadeau – Bull/Atos

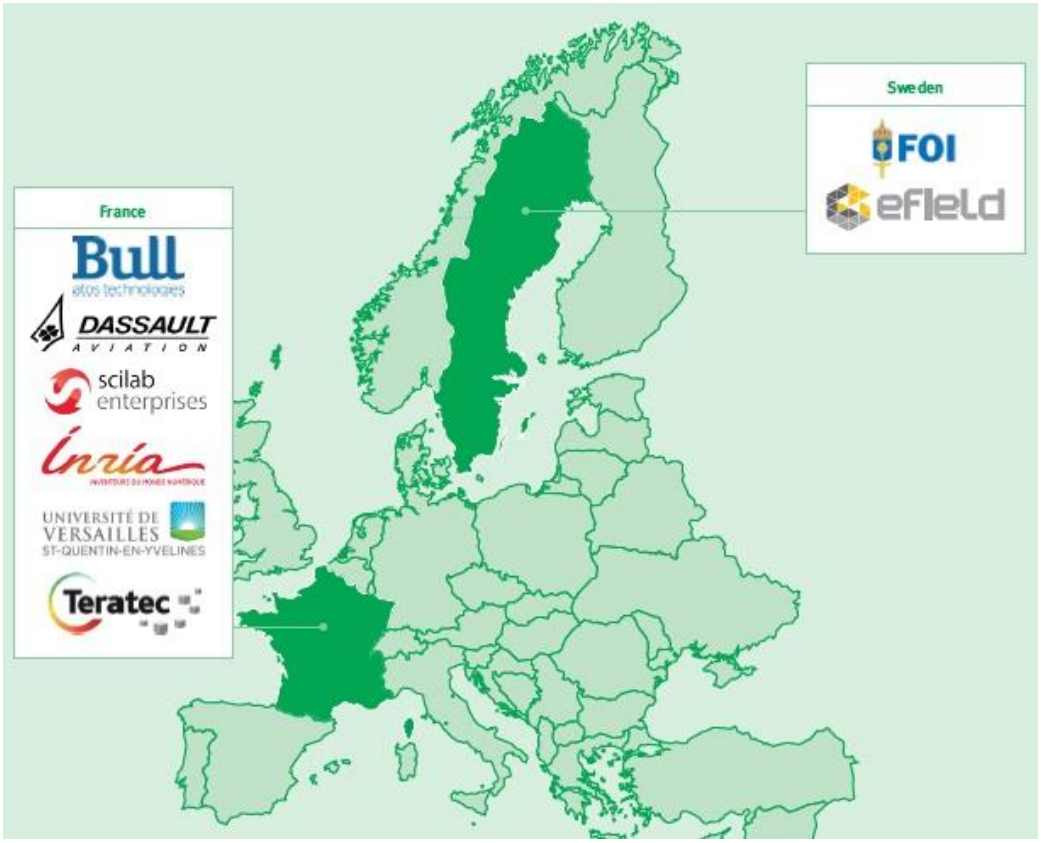
The COncurrency and LOcality Challenge

Forum TERATEC 2017, 27-28 juin 2017

- The COLOC consortium
- Slurm: scalable and flexible RJMS
- Extend Slurm to deal with data locality in resource selection:
  - Take into account application communication patterns
  - Take into account data reads/writes
- Conclusions

- The COLOC consortium
- Slurm: scalable and flexible RJMS
- Extend Slurm to deal with data locality in resource selection
  - Take into account application communication patterns
  - Take into account data reads/writes
- Conclusions

The COncurrency and LOcality Challenge
The consortium



Start Date	**1st July 2014**
End Date	**31st October 2017**

	France	Sweden
Academic partners	UVSQ	
Research Labs & Software Institutes	INRIA	FOI: Defense Research Agency
HPC promotion association	Teratec	
Software Tools Editors	ESI/SCILAB	
Simulation Software Editors		**ESI/Efield**
Industrial HPC users	Dassault Aviation	
HPC Platform Provider	**Bull (leader)**	

COLOC Objectives & Software Technologies

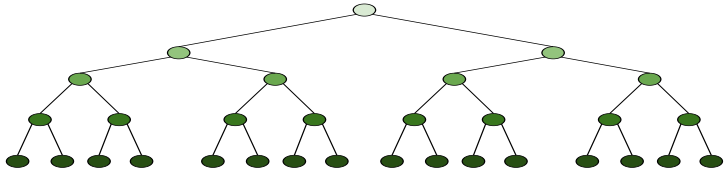
- Develop new approaches to manage concurrency and data locality
  - Data management improvement (BULL, UVSQ)
  - Process and data placement enhancement (BULL, INRIA)

- Develop new models, mechanisms and tools to better exploit the various types of resources
  - Hierarchical topology modelling (INRIA)
  - Performance profiling tools (UVSQ)

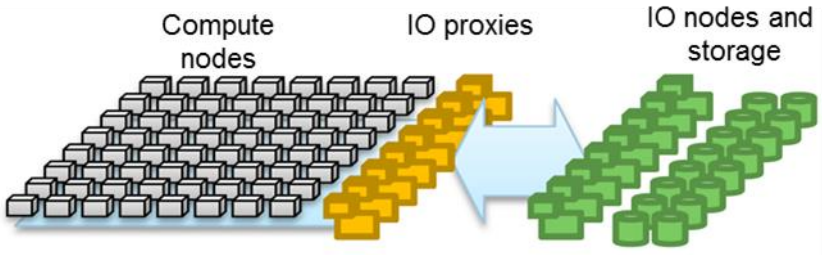
- Enhance the software infrastructure and applications
  - Adapt, enhance key components (Scilab, FOISOL, …)
  - Adapt, enhance applications & simulation software using the new libraries & facilities (Dassault Aviation, ESI)

Data Locality as a resource management challenge

- Improve exchanges of data
  - based on communication pattern
    - the expression of bytes/messages exchanged by the application processes
  - match this pattern with the available resources
    - placing processes that communicate more to cores that are closer to each other



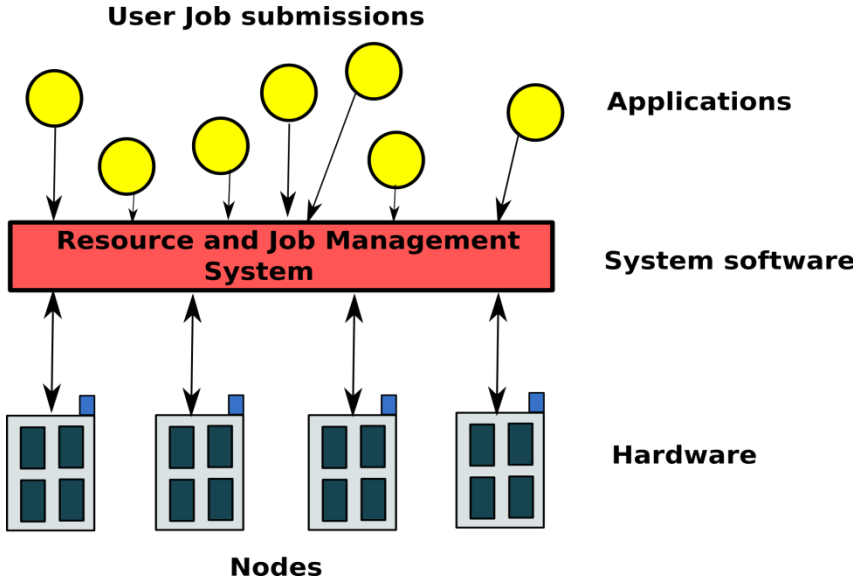
- Improve I/O on persistent storage
  - based on the needs of the application
  - through special nodes such as IO proxies
  - relevant proxies as close as possible to the compute nodes of a given job



- The COLOC consortium
- Slurm: scalable and flexible RJMS
- Extend Slurm to deal with data locality in resource selection
  - Take into account application communication patterns
  - Take into account data reads/writes
- Conclusions

Resource and Job Management

- Supercomputers are shared amongst users and applications
- Resource and Job Management System (**RJMS**) is responsible to assign resources to applications based on their needs
- Strategic position but complex internals
  - competing optimization metrics (e.g. administrator vs user view)
  - job scheduling (space and time sharing)
  - multi-criteria selection of resources
  - process affinity and binding



Slurm history and facts

- Initially developed in LLNL since 2003, passed to SchedMD in 2011
- Multiple enterprises and research centers contribute to the project
  LANL, CEA, HP, BULL, BSC, CRAY, etc
- Large international community, active mailing lists
- Slurm used on the most powerful supercomputer in the world
  for example: Tianhe-2, n°2 on Top500
- **Bull** does research, development and support of Slurm since 2005

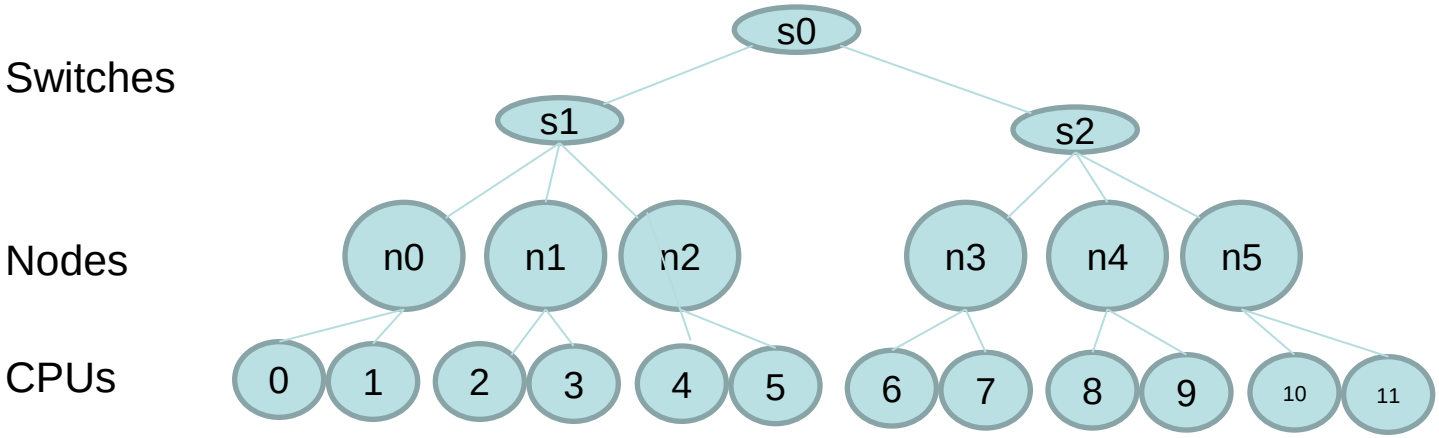
- Slurm is known to deliver
  - Scalability and Performance
  - Extensibility (plugin mechanism, Free and Open Source)

- The COLOC consortium
- Slurm: scalable and flexible RJMS
- Extend Slurm to deal with data locality in resource selection
  - Take into account application communication patterns
  - Take into account data reads/writes
- Conclusions

Current Network Topology Aware Placement

- Slurm provides topology aware selection of resources

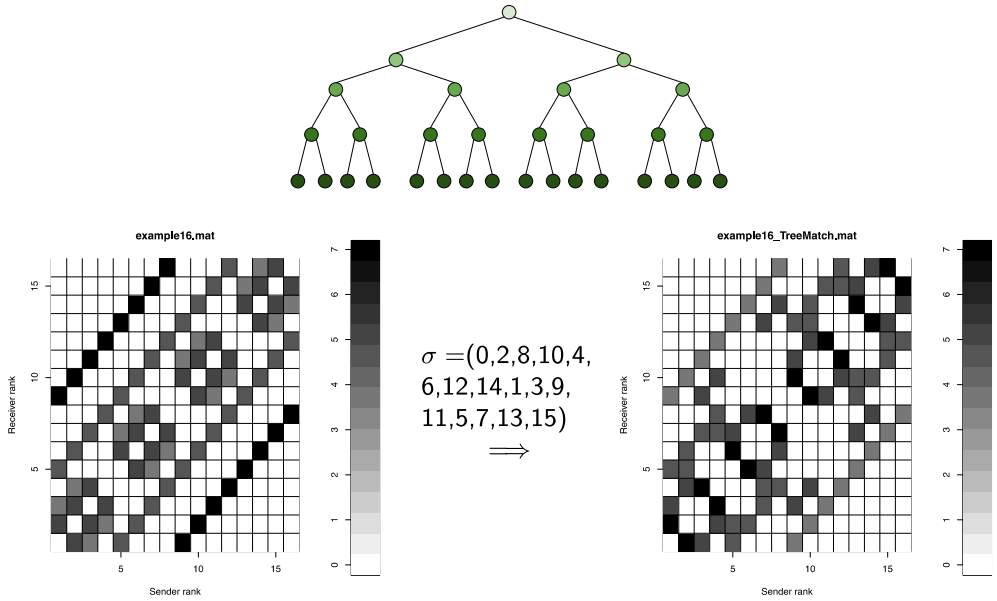
Example with tree topology



- Assumptions
  - balanced communications among processes
  - reducing number of hops can only improve performance

Improving Network Topology Aware Placement

- Not all the processes exchange the same amount of data
- The speed of the communications, and hence the performance of the application depends on the way processes are mapped to resources.



- Communication matrix + Tree Topology = Process permutation

Improving Network Topology Aware Placement

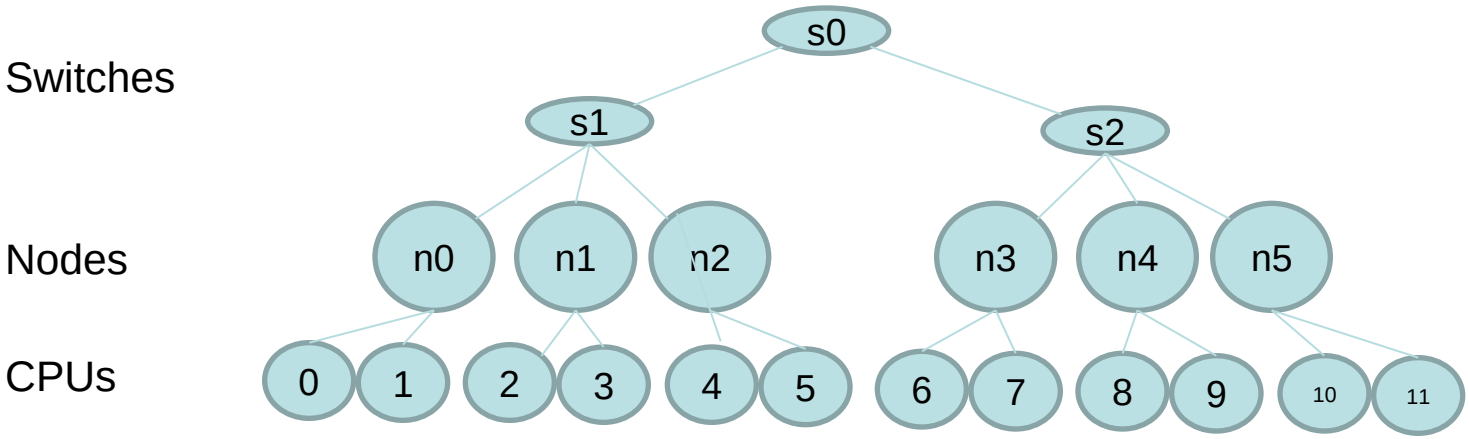
COLOC

- Input
  - Pattern of communication given by user expertise or profiling
  - Topology defined in the RJMS configuration

- Process permutation: TREEMATCH
  - Recursively from the minium level of switches
    - Find the minimum level of switches
    - Find the minimum number of switches
    - Partition the matrix of communication / group processes
  - Build permutation

Treematch Example

- Assume the following topology on a small cluster

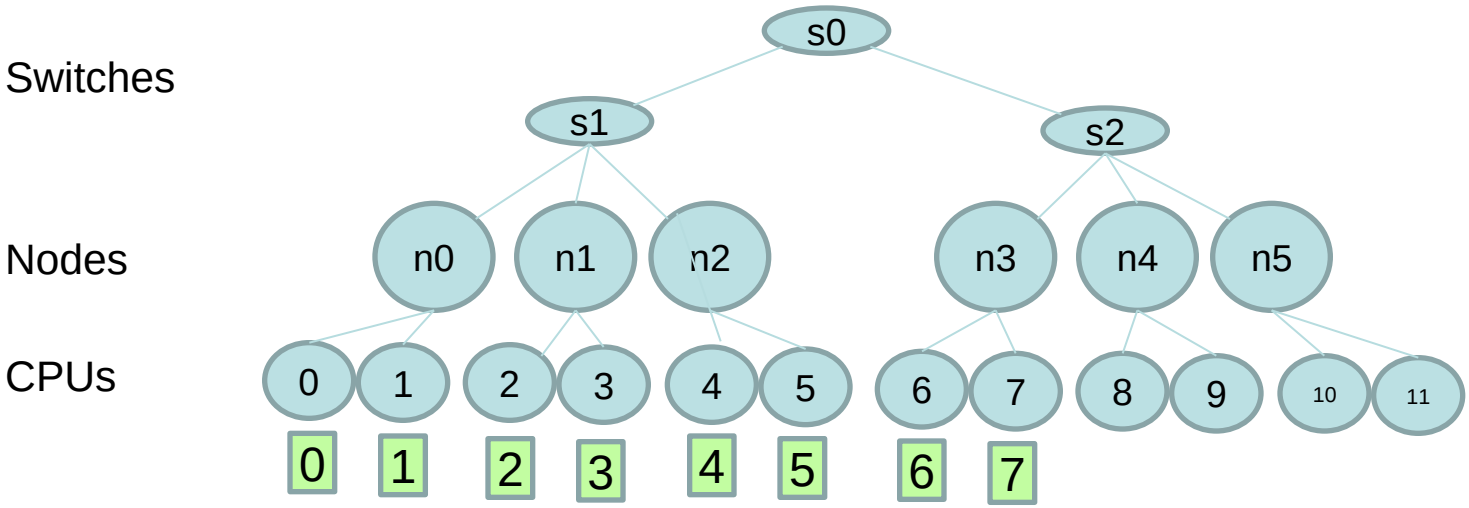


Treematch Example

- A job demands 8 CPUs on 4 nodes

Communication matrix

Proc.	0-1	2-3	4-5	6-7
0-1	0	20	0	**2000**
2-3	20	0	**1000**	0
4-5	0	**1000**	0	10
6-7	**2000**	0	10	0



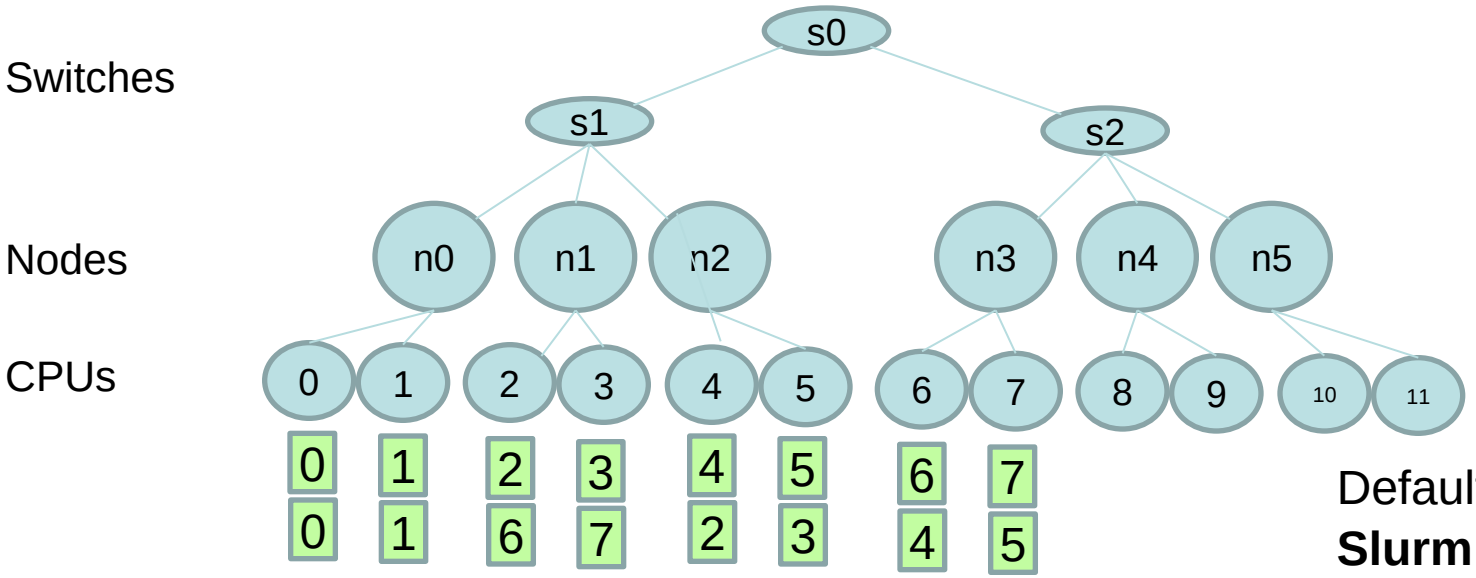
Default Slurm

Treematch Example

- A job demands 8 CPUs on 4 nodes

Communication matrix

Proc.	0-1	2-3	4-5	6-7
0-1	0	20	0	**2000**
2-3	20	0	**1000**	0
4-5	0	**1000**	0	10
6-7	**2000**	0	10	0



Default Slurm

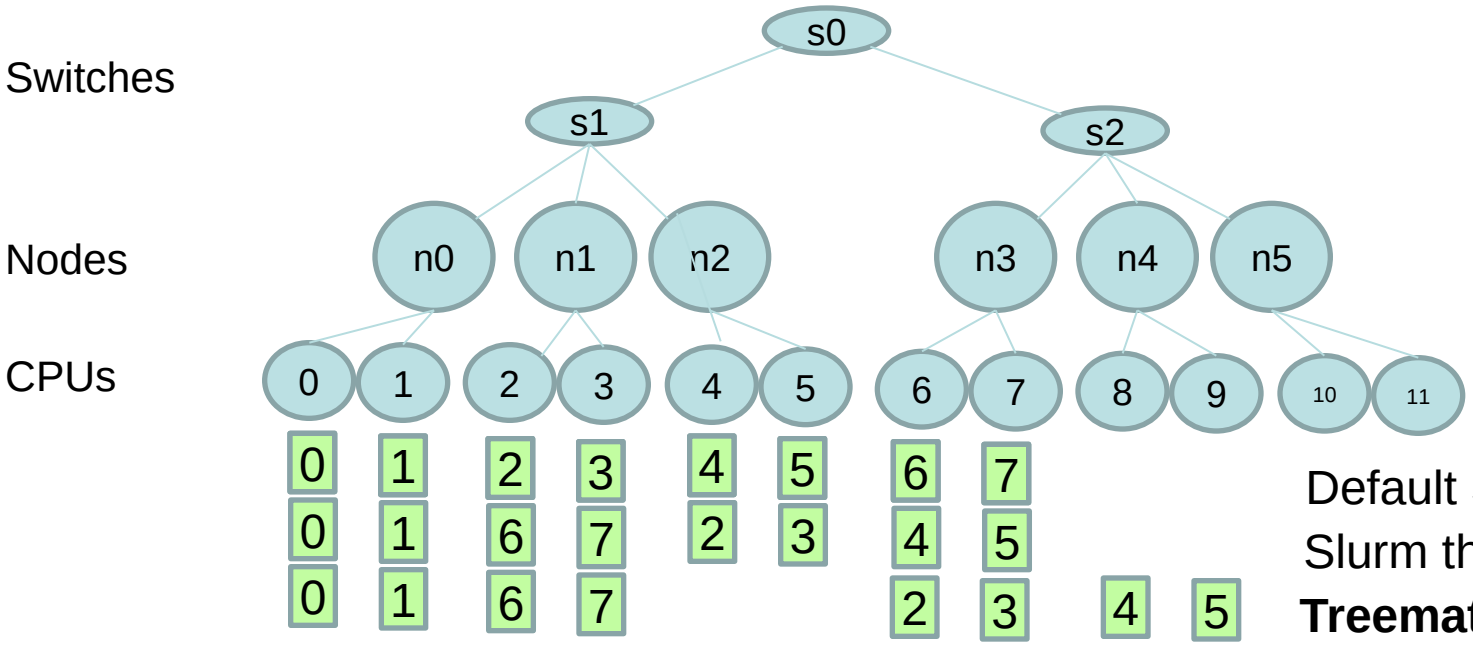
Slurm then Treematch

Treematch Example

- A job demands 8 CPUs on 4 nodes

Communication matrix

Proc.	0-1	2-3	4-5	6-7
0-1	0	20	0	**2000**
2-3	20	0	**1000**	0
4-5	0	**1000**	0	10
6-7	**2000**	0	10	0



Evaluation and current state

- Treematch integration within Slurm:
  - Implementation done by Adele Villiermet in the context of her PhD

- Initial validation and evaluation published in ICDCN 2017:

  Topology-aware resource management for HPC applications

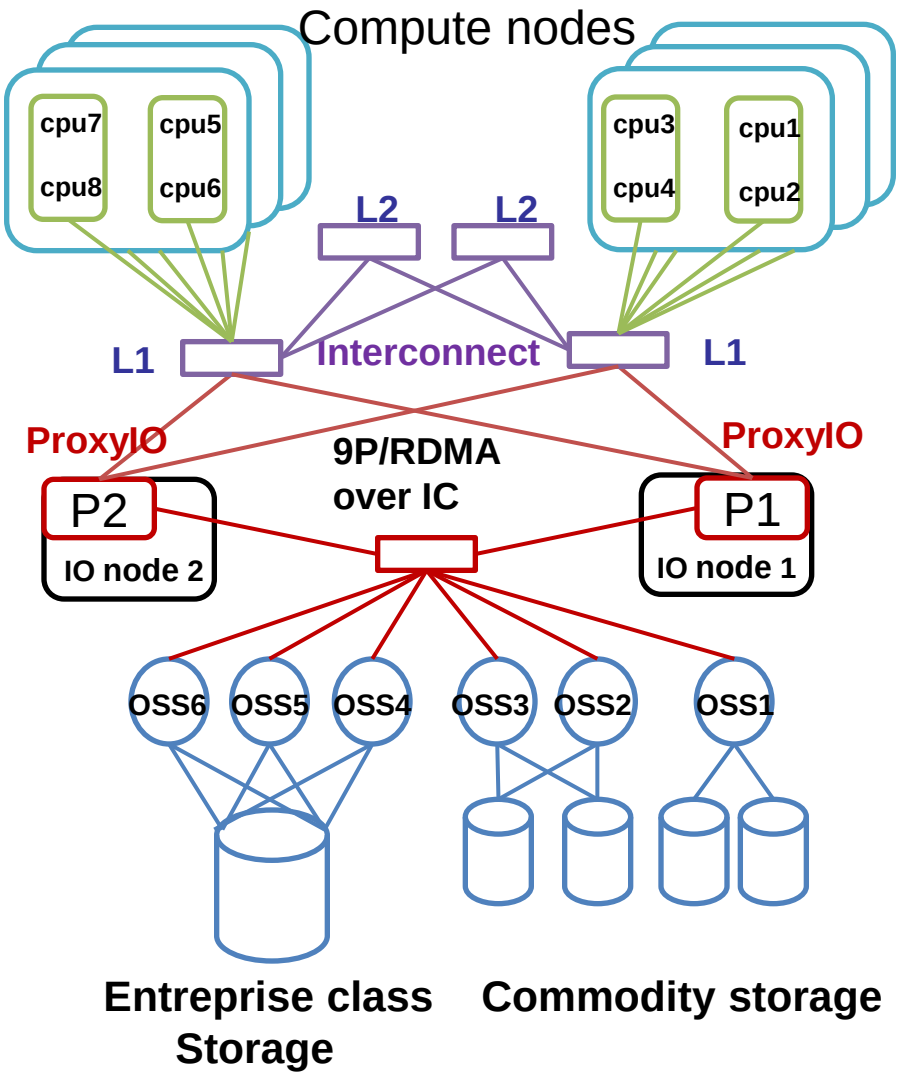
  Yiannis Georgiou, Emmanuelle Jeannot, Adele Villiermet, Guillaume Mercier

- Finalization of the integration of this new feature in the upcoming stable version of Slurm.

- The COLOC consortium
- Slurm: scalable and flexible RJMS
- Extend Slurm to deal with data locality in resource selection:
  - Take into account application communication patterns
  - Take into account data reads/writes
- Conclusions

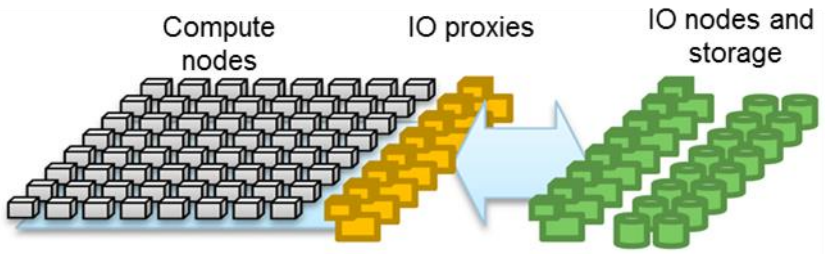
Optimized usage of persistent storage

- Need to quantify IO required by an application
- Need to express this requirement to the RJMS
- In modern architectures, persistent storage will be used through special nodes such as IO proxies



Optimized usage of persistent storage

- Select those IO proxies to be as closer as possible to the compute nodes which will use them.
- Need of new feature of concurrent selection of different types of resources and possible execution of different executables on each group of resources.



Slurm: new job packs feature

- Untill now Slurm provides a SPMD (Single Program Multiple Data) environment.

```
srun -N4 -mem=10000 -gres=gpu ./myapp
```

- All nodes in an allocation have identical resources
  - 4 nodes are allocated, all have 10G of memory, all have a gpu
- All tasks execute the same application
  - myapp launched on all nodes

Introduction to Slurm Job packs

- In some cases it is desirable to have nodes with different characteristics as part of the same step.
  - A node with lots of memory for the serial startup/wrap upphase.
  - Lots of nodes with GPU for the parallel phase.
  - Nodes with Fast I/O to store the results.
  - And these nodes may run different executables that are part of the same MPI_Comm_World supporting MPMD (Multiple Program Multiple Data) model

- Kind of like multiple sruns co-scheduled so that they can run at the same time.
  - We do this by ‘packaging’ a set of jobs, or a ***Job Pack***

Introduction to Slurm Job packs

```
srun -JLdr -pt96big controller : -JMbr1 -N2 -n4 -pt96gpu worker : -JMbr2 -N2 -pt96iopx saver &
squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
             47959   t96iopx     Mbr2    slurm  R       0:06      2 trek[8-9]
             47960    t96gpu     Mbr1    slurm  R       0:06      2 trek[4-5]
             47961    t96big      Ldr    slurm  R       0:06      1 trek7
```

- One task executing **controller** on trek7 (the pack_leader)
- Two tasks executing **worker** on both trek[4-5] (member_1)
- One task executing **saver** on both trek[8-9] (member_2)

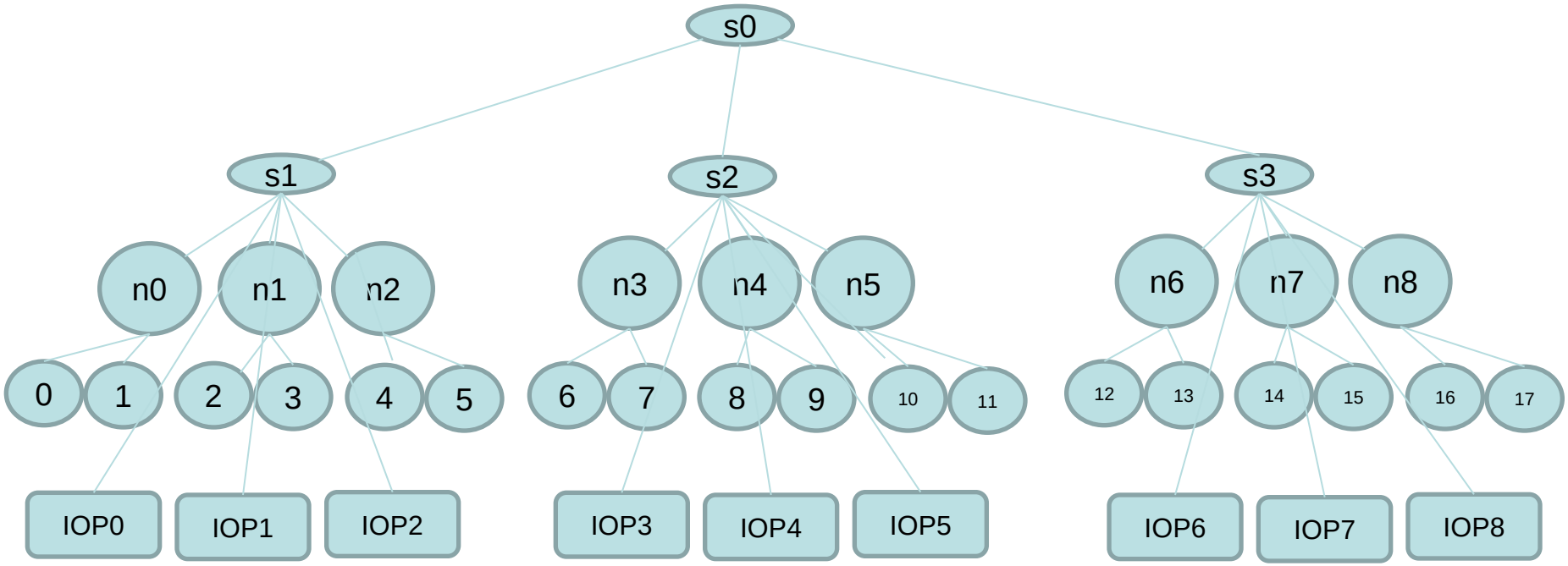
Introduction to Slurm Job packs

```
srun -JLdr -pt96big controller : -JMbr1 -N2 -n4 -pt96gpu worker : -JMbr2 -N2 -pt96iopx saver &
squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
             47959   t96iopx     Mbr2    slurm  R       0:06      2 trek[8-9]
             47960    t96gpu     Mbr1    slurm  R       0:06      2 trek[4-5]
             47961    t96big      Ldr    slurm  R       0:06      1 trek7
```

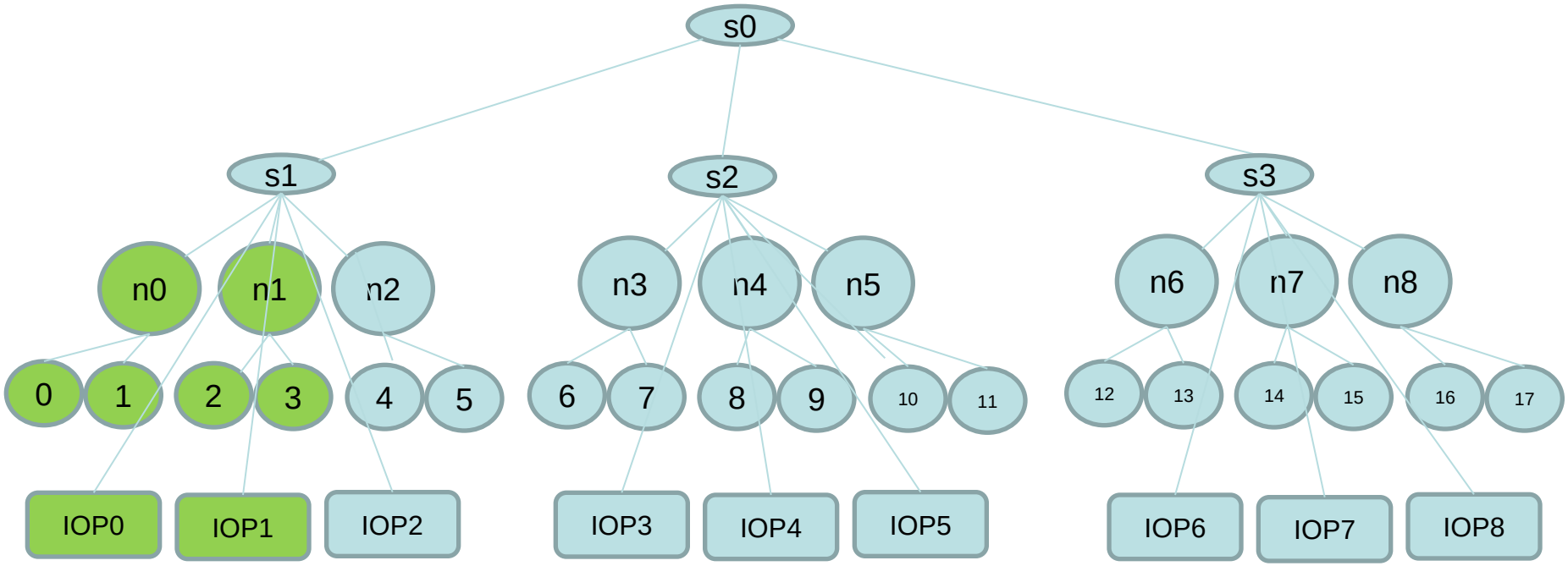
- One task executing **controller** on trek7 (the pack_leader)
- Two tasks executing **worker** on both trek[4-5] (member_1)
- One task executing **saver** on both trek[8-9] (member_2)
- Example with new description of job packs
  - a job asking for compute resources
  - a job asking for IO proxies

```
sbatch -p Compute -N1000 -n16000 ./exec.sh : -p IO -N 100 -n1600 ./exec_IO.sh
```

Selection of compute and IO nodes with Job packs

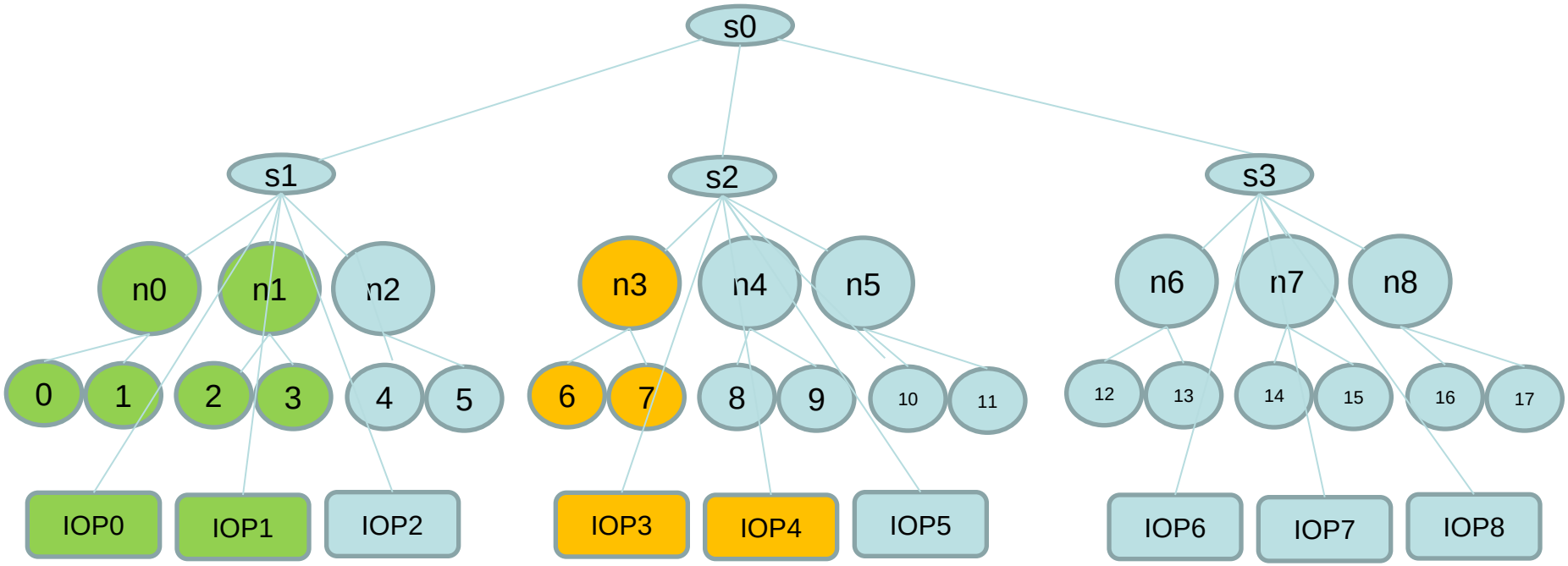


Selection of compute and IO nodes with Job packs




```
srun -N2 -p Compute ./app : -N2 -p IO ./appIO
```

Optimized selection of compute and IO nodes to improve their proximity

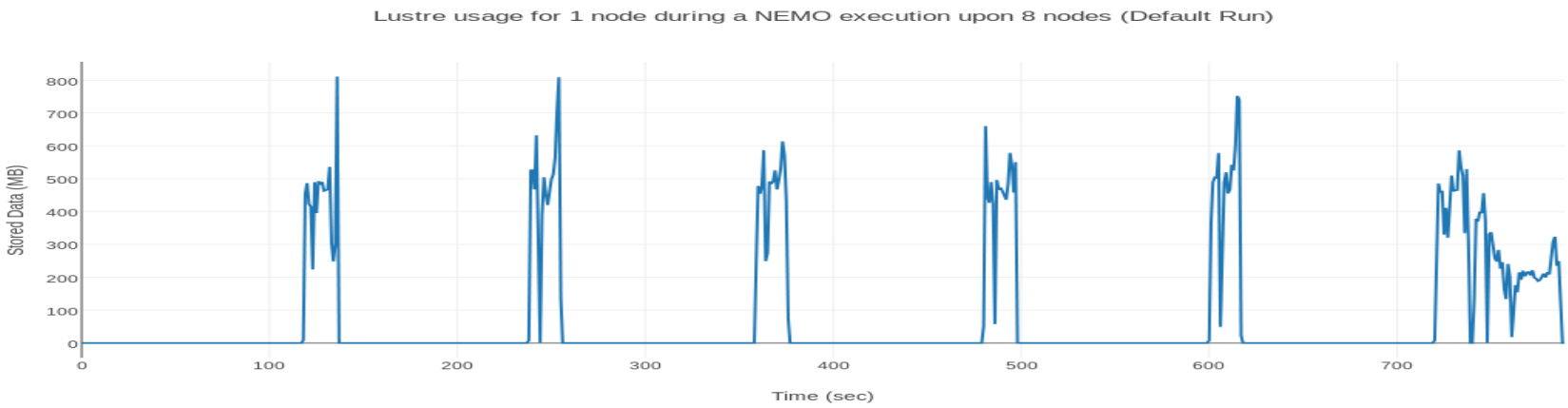



```
srun -N2 -p Compute ./app : -N2 -p IO ./appIO
```

```
srun -N1 -p Compute ./app : -N2 -p IO ./appIO
```

Current State

- Slurm job packs feature in Slurm version from Bull
- Discussions with SchedMD opened
- Compute and IO nodes proximity optimization feature currently under development
  - Algorithm based upon the layouts framework for resource management
- IO profiling from slurm to help setting the parameters for selection of IO-proxies



- The COLOC consortium
- Slurm: scalable and flexible RJMS
- Extend Slurm to deal with data locality in resource selection:
  - Take into account application communication patterns
  - Take into account data reads/writes
- Conclusions

Future Works

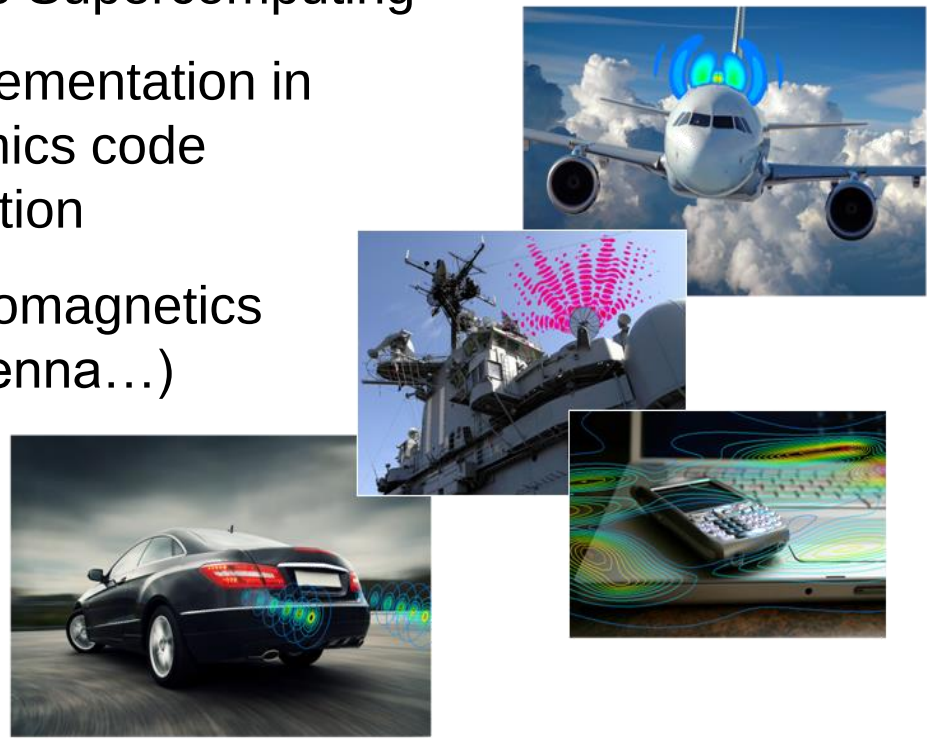
- Network Topology Aware Placement
  - Improve data locality
  - Usage of behavior from past executions within the RJMS
  - Get the communication pattern by profiling with Slurm
  - Use accounting/profiling data for optimal job parameters

- Job Pack
  - Optimization to improve the proximity of compute and IO nodes
  - Develop simple way to describe the needs of IO nodes

    --IO-usage =[low,medium,maximum]

Main Achievements as of Today

- **HWLOC-NETLOC: Modeling tools** → 6000 nodes Curie Supercomputer
- **Multicriteria placement framework** and **new IO proxies architecture**:
  → a significant step forward Exascale Supercomputing
- **Divide & Conquer mechanism** implementation in industrial Computational Fluid Dynamics code developed by UVSQ & Dassault Aviation
- On going optimization steps in Electromagnetics codes for large scale simulation (Antenna…)
- New releases of ESI Electromagnetics simulation tools

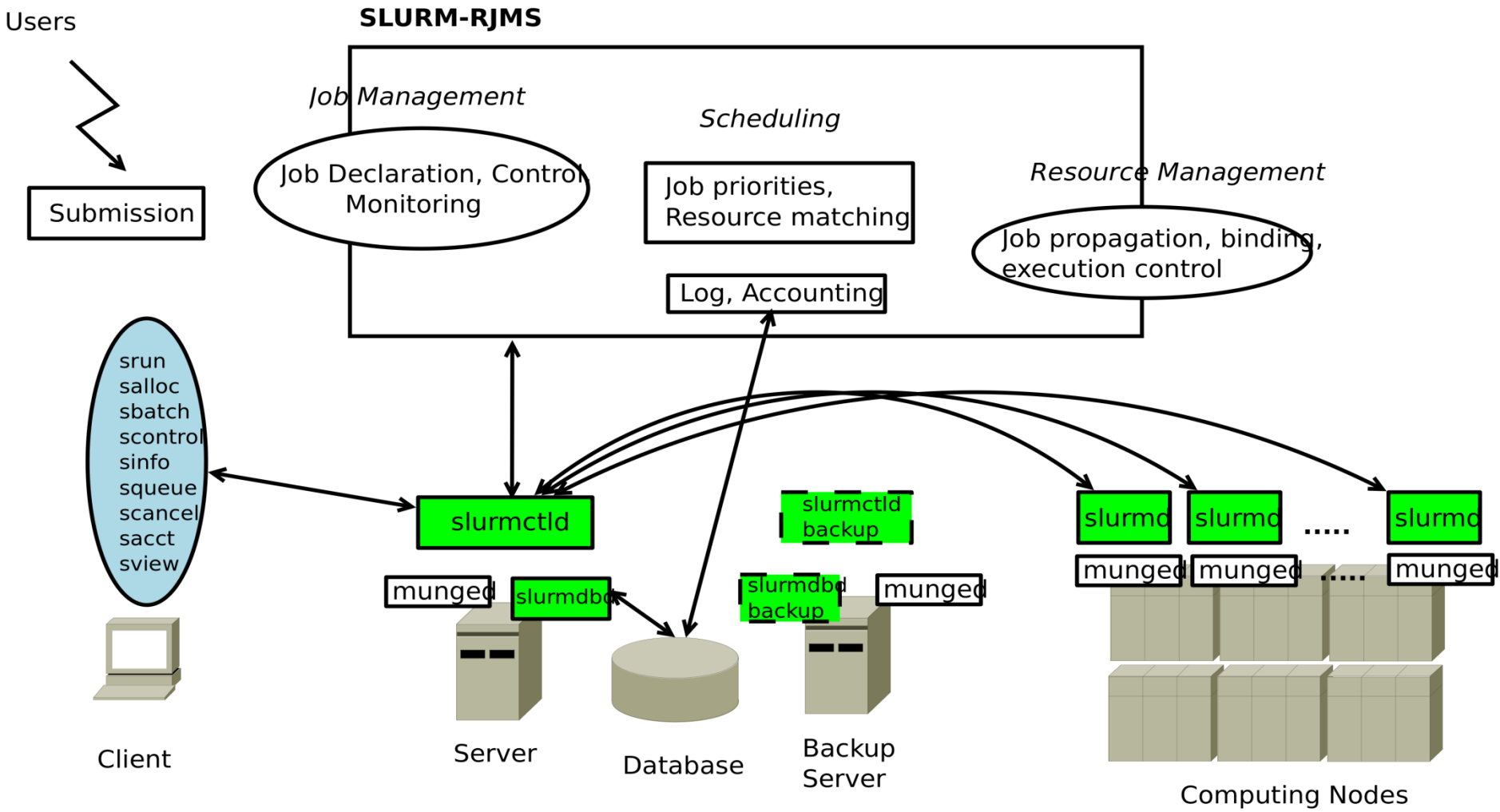


ITEA 2
INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

Bull
atos technologies

Thank you for your attention

Slurm Architecture



Architecture Treematch within Slurm

- Implemented a new selection option for the select/cons_res plugin of Slurm
  - The communication matrix is provided at job submission time through a new distribution option

```
srun –m TREEMATCH=/comm/matrix/path
```

  - The topology as needed by Treematch is provided by a new parameter in the configuration file

```
TreematchTopologyFile=/topology/file/path
```

  - The availability of resources is retrieved through the node and core bitmaps data structures
    - Slurm local CPU ids are translated to Treematch CPU ids in order to calculate the process permutation.
    - The selected list of CPUs as done by Treematch is then translated back to bitmaps for Slurm to use

Motivations

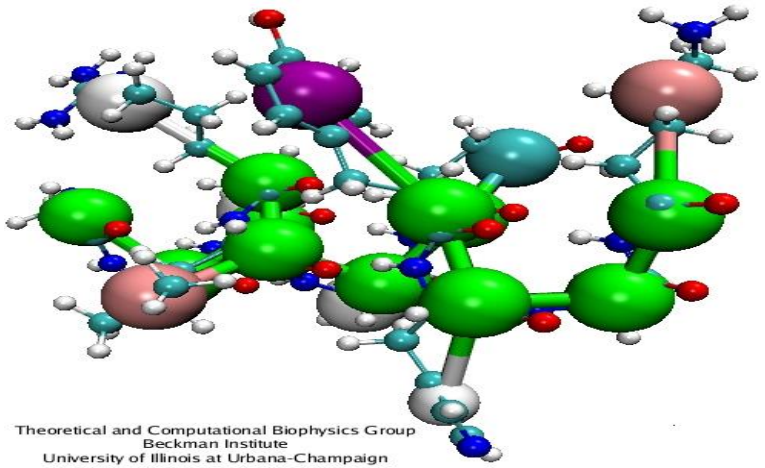
- More and more scientific fields rely on High Performance Computing for simulations

- Harnessing the power of supercomputers remains challenging
  - Scalability
  - Heterogeneity
  - Architecture complexity
    - Network Topology
    - Multi-core nodes
    - Memory hierarchies

- How to **improve application performance** ?

Motivations

- Consider the **application behavior** in terms of **data locality** and deploy it on the supercomputer accordingly
  - different resources needs for different phases of execution,
  - communication pattern,
  - memory accesses pattern,
  - data store needs



Slurm scalable and flexible RJMS

- **Scalability**: Designed to operate in a heterogeneous cluster with up to tens of millions of processors.
- **Performance**: Can accept 1,000 job submissions per second and fully execute 500 simple jobs per second (depending upon hardware and system configuration).
- **Free and Open Source**: Its source code is freely available under the GNU General Public License.
- **Portability**: Written in C with a GNU autoconf configuration engine. While initially written for Linux, Slurm has been ported to a diverse assortment of systems.
- **Power Management**: Job can specify their desired CPU frequency and power use by job is recorded. Idle resources can be powered down until needed.
- **Fault Tolerant**: It is highly tolerant of system failures, including failure of the node executing its control functions.
- **Flexibility**: A plugin mechanism exists to support various interconnects, authentication mechanisms, schedulers, etc.