



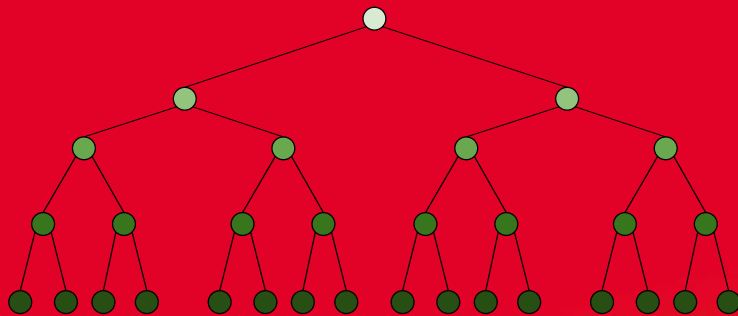
Topology-Aware Data management

Emmanuel Jeannot
Runtime Team
Inria Bordeaux Sud-Ouest

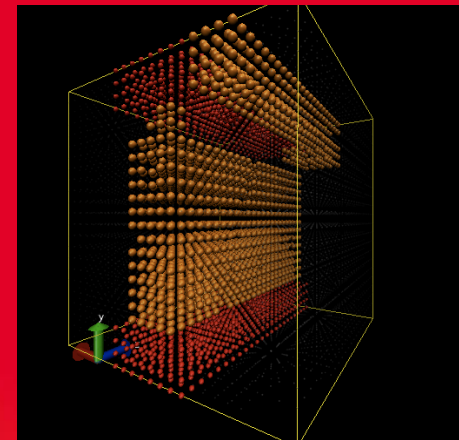
January 16, 2015

INTRODUCTION

Data management for high-performance computing



Topology



Locality



Data



2

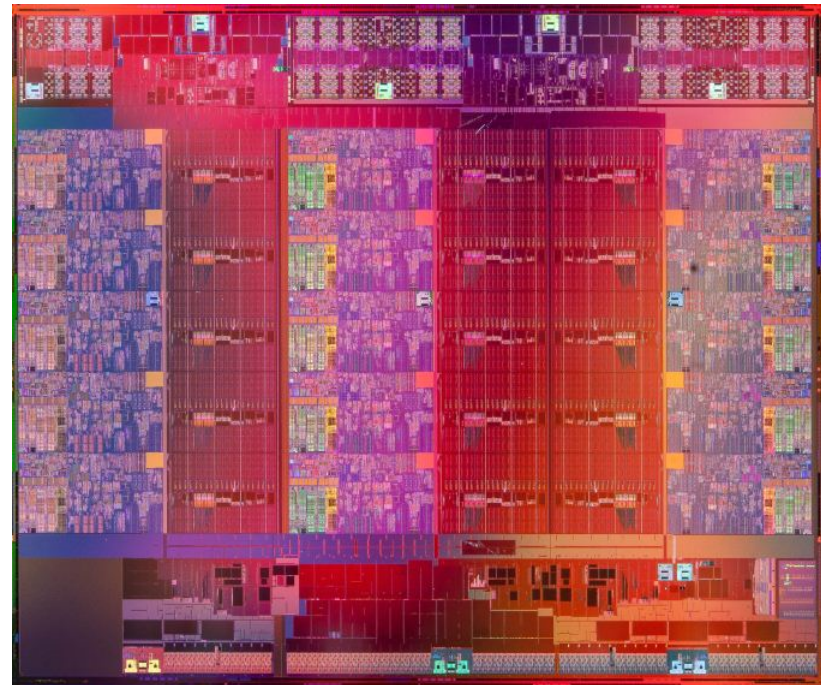
Topology-Aware Data Management

High-performance computing systems more and more complex

Memory hierarchy is deepening:

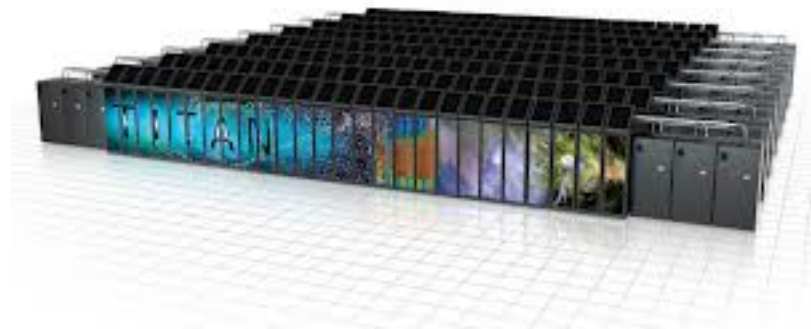
- Cache
- Ram
- NVRAM
- Flash
- etc.

Networks are larger and more intricate.



Computing is easy, accessing data is difficult

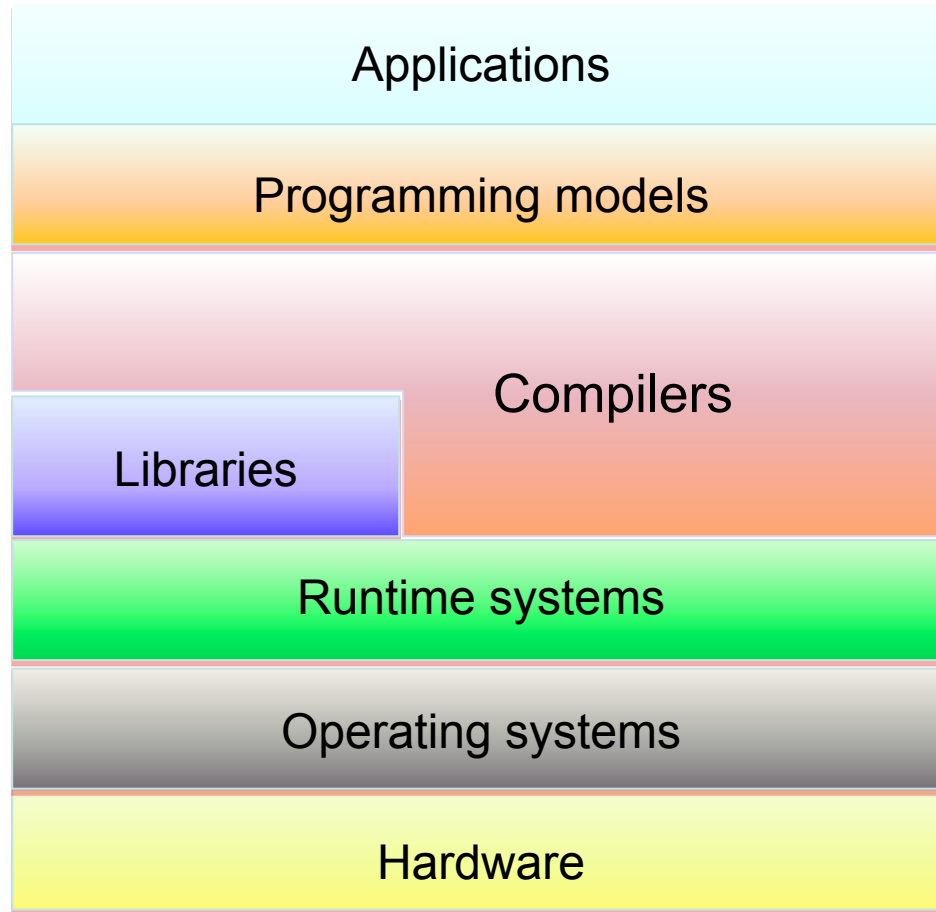
Lot of computing power.



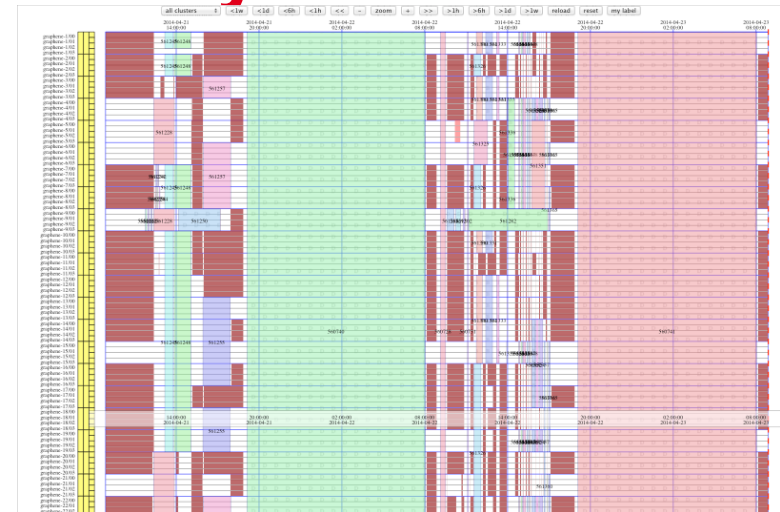
Complex topology + low mem/core :

Bringing data at the **right place** at the **right time** is the challenge.

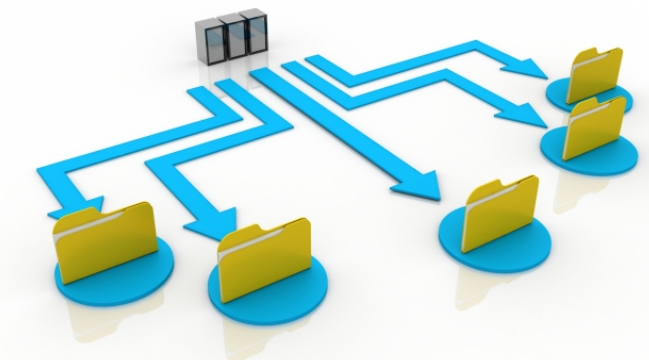
The application and its ecosystem



SW stack



Batch scheduler



Storage

Optimizing execution

Once the application has been written in **nice language** with **performing libraries** and **efficient data layout** there is still room for optimizations:

- Not everything is known at compile time
- allocated resources
- input data

Our goal: take the application as it is and **optimize its execution on its ecosystem** (sw stack + batch scheduler + storage system + ...)

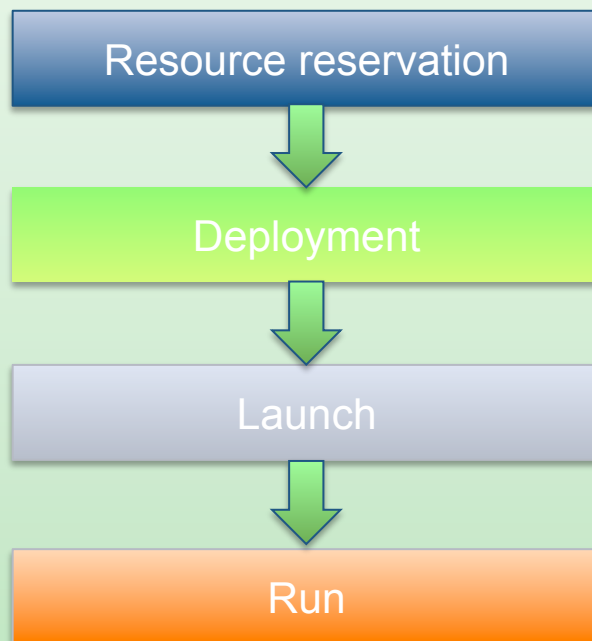
Not that simple...

Execution is not always completely decoupled from application design.

Need for information exchange between application and runtime system

TADaaM: Topology-Aware System-Scale Data Management for High-Performance Computing Applications

Application execution phases



Topology-aware data management research

Interaction with the ecosystem

Process placement

Data partitioning

Process reordering

Affinity management

I/O and datapath optimization

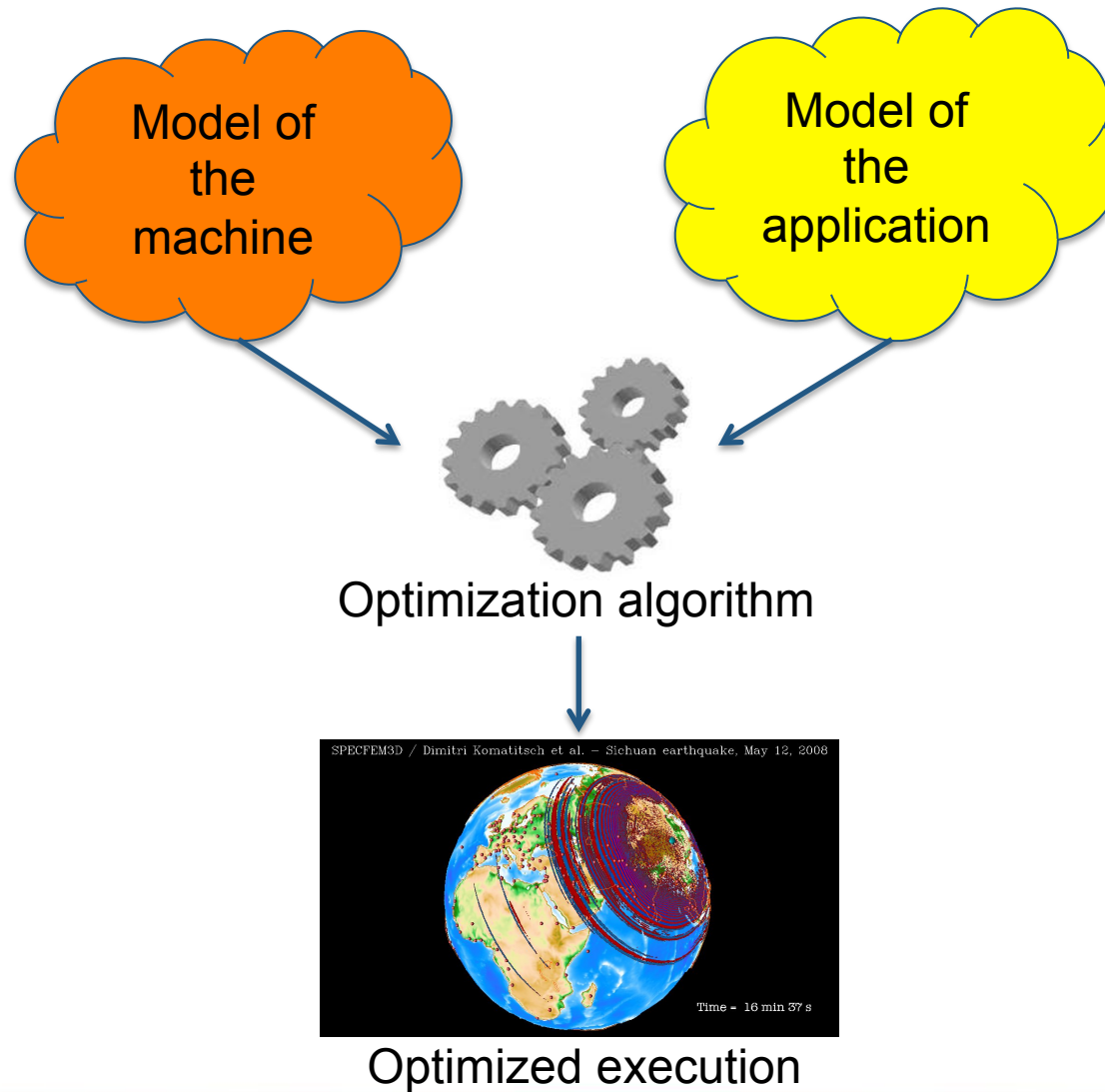
Migration

Partitioning refinement

Data reallocation

Perf. and platform models

Big Picture



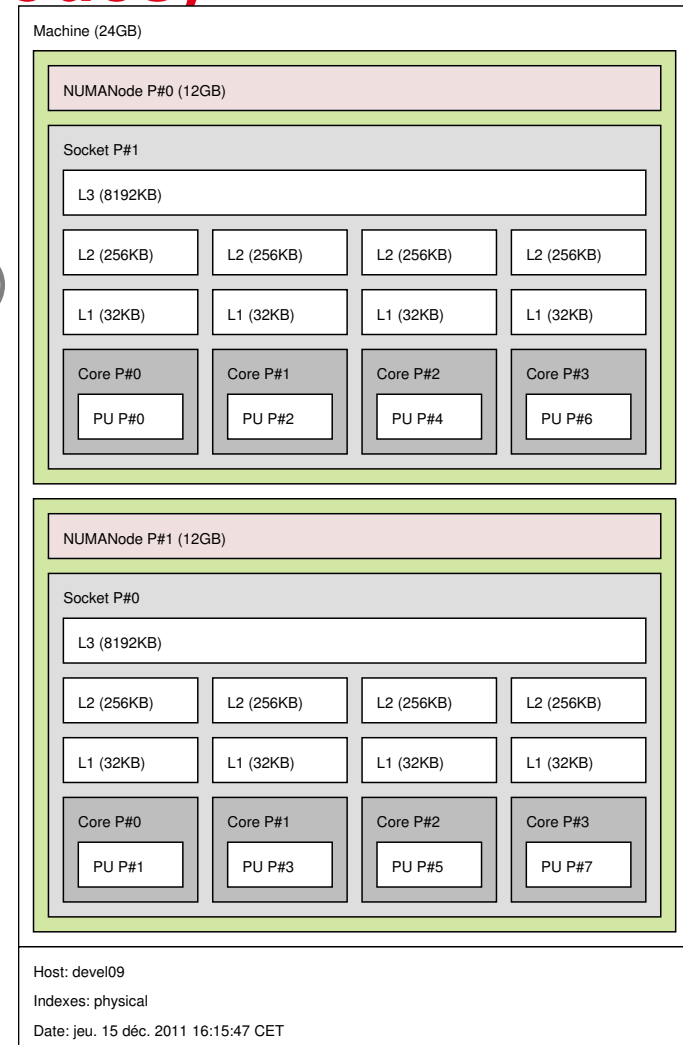
Model of machine (within nodes)

HWLOC (portable hardware locality)

- Runtime and OpenMPI team
- portable abstraction (across OS, versions, architectures, ...)
- Hierarchical topology
- Modern architecture (NUMA, cores, caches, etc.)
- ID of the cores
- C library to play with
- etc.

On going dev. in the team need for:

- dynamic information topology information (within and between nodes)
- API?
- Less bug in the BIOS



Model of the machine (network): Netloc

hwloc companion

Takes care of network topology
and joins hwloc and network information

- Global « map » of your cluster
 - Connects hwloc objects to network edges

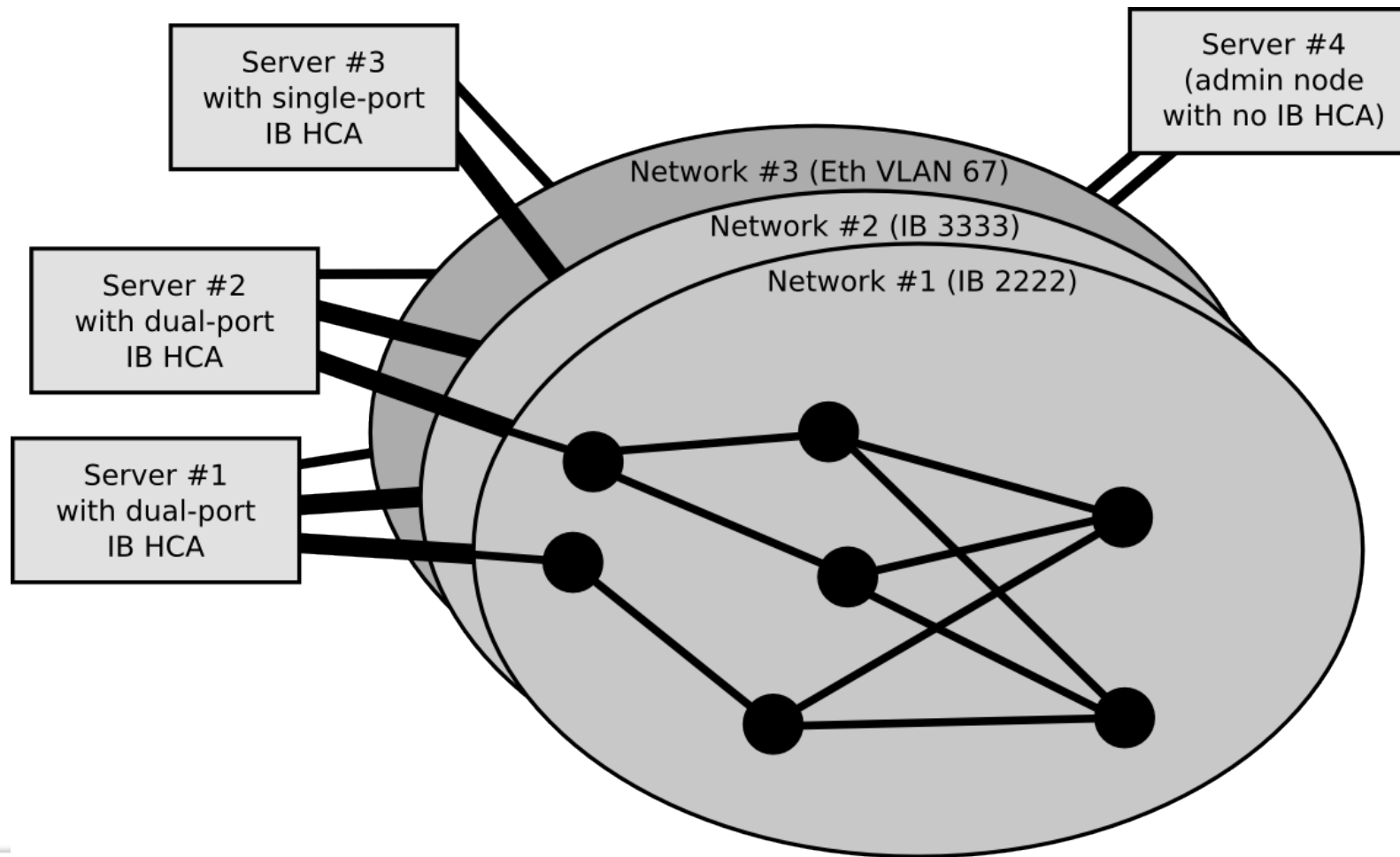
Public API made of

- Network queries (nodes, edges, etc.)
- Global map queries
- hwloc API when looking inside servers

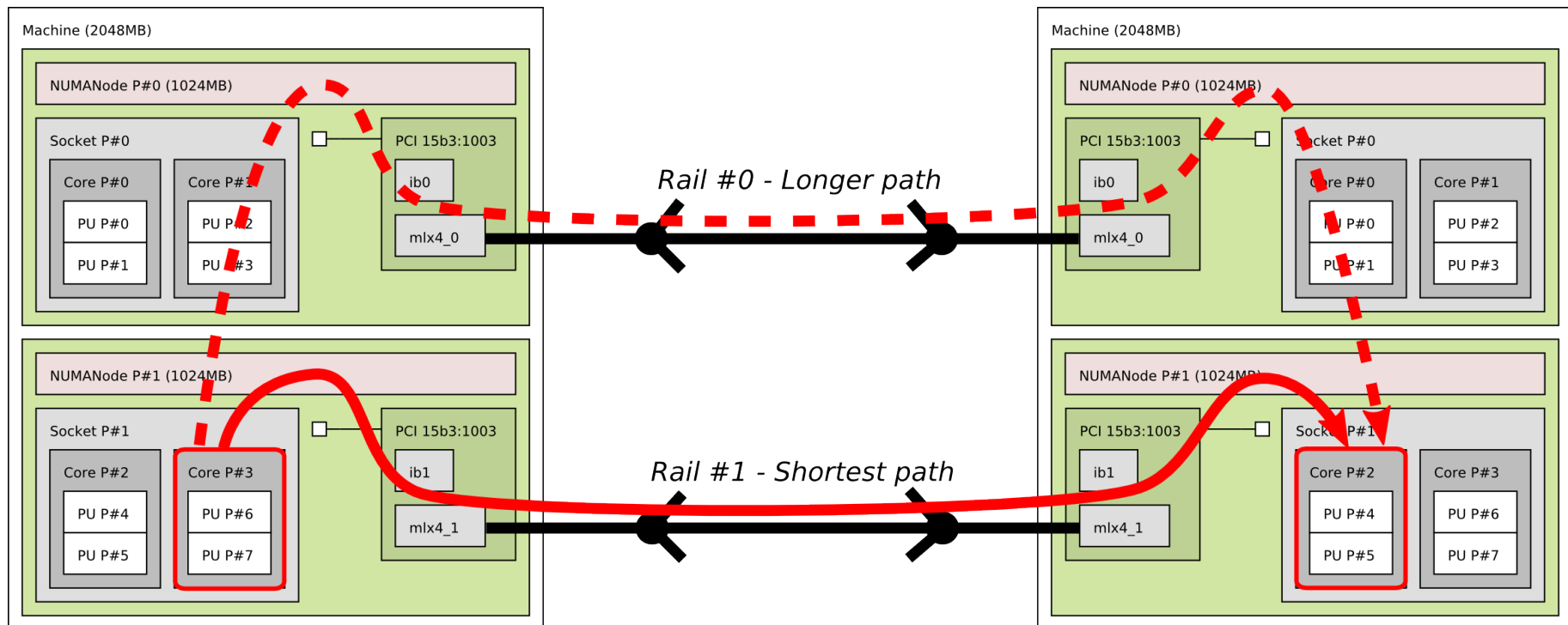
Currently developed by

- University of Wisconsin-LaCrosse (J. Hursey)
- Inria (B. Goglin)
- Cisco (J. Squyres)
- under the umbrella of the Open MPI consortium

Netloc global map



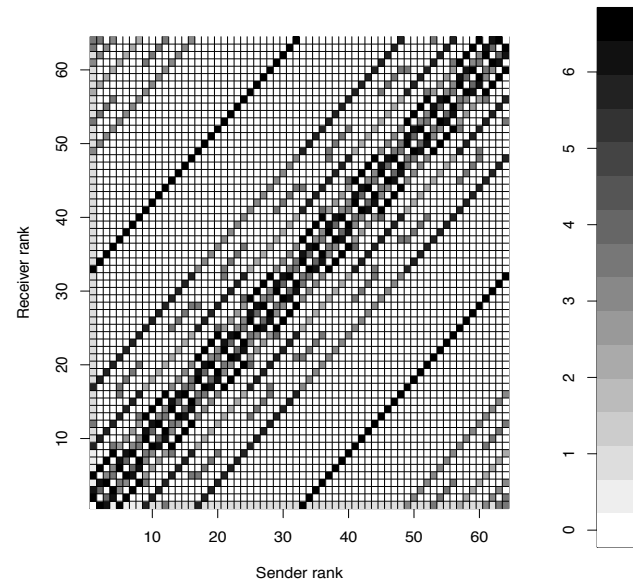
Multirail/multipath Locality



Application model

We target data access.

We need affinity between processing elements:
communication pattern



Building the communication pattern

- Statically (thanks to compiler)
- Dynamic Monitoring (Charm++)
- Blank execution and tracing (OpenMPI)
- After data partitioning (e.g. Scotch)

Software suite: use-case example

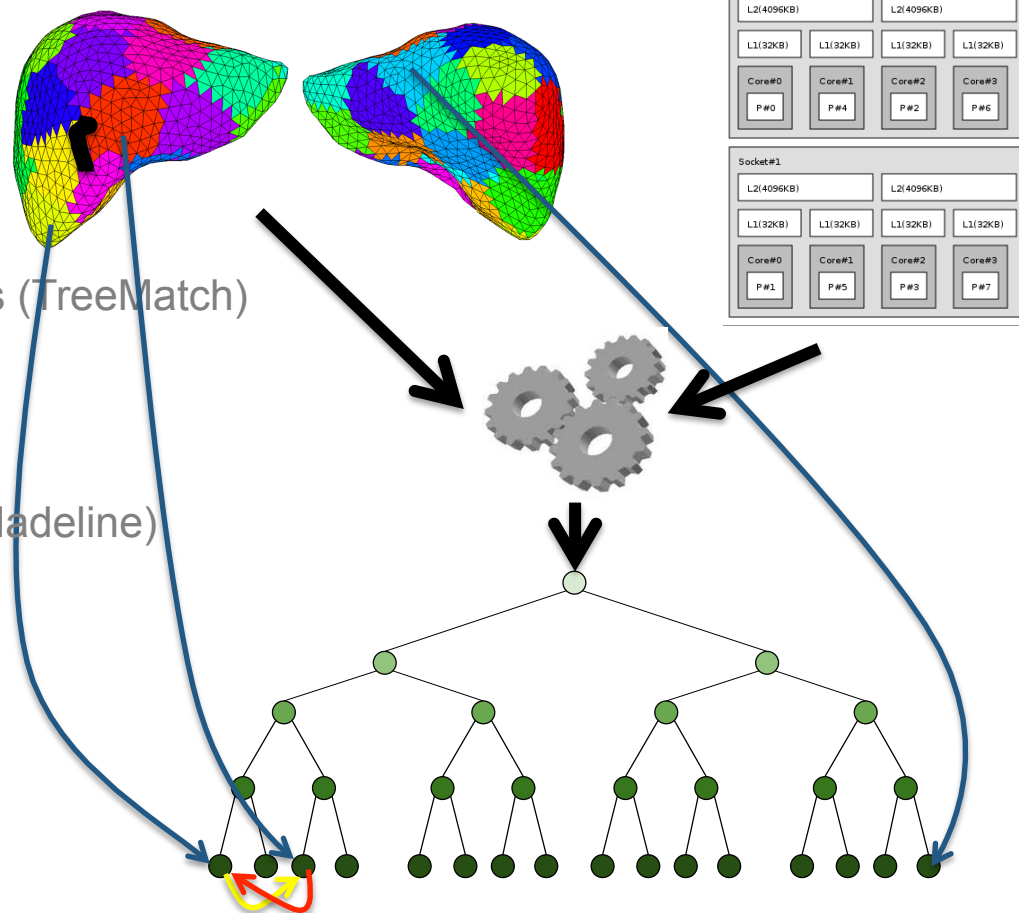
Mesh/graph partitioning (Scotch)

Platform model (Hwloc)

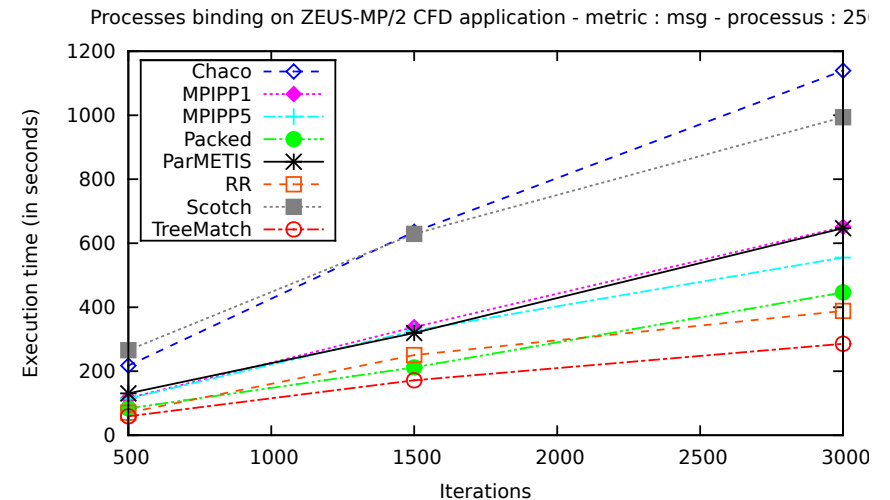
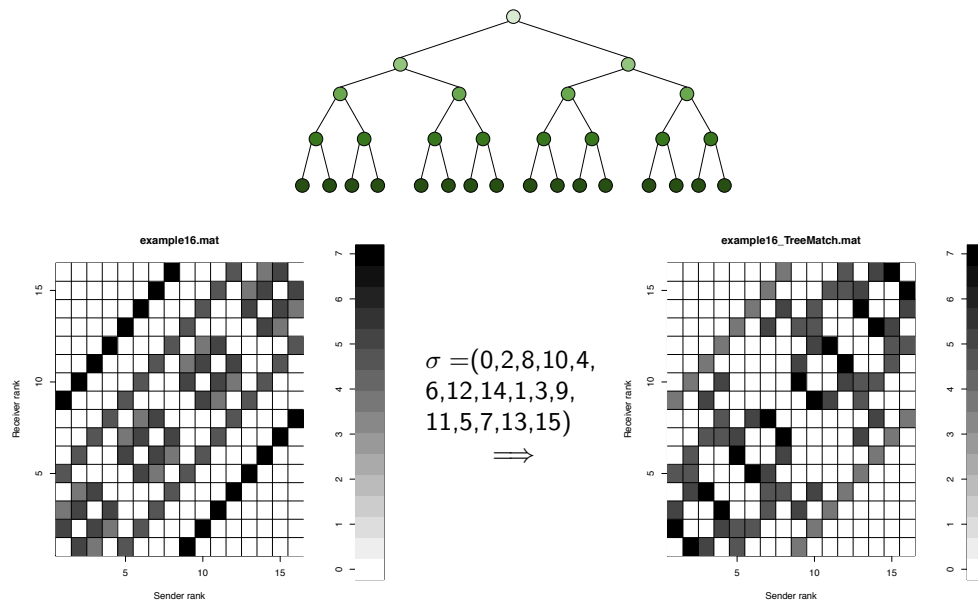
Topology-aware locality mechanisms (TreeMatch)

Parallel mesh adaptation

Communication optimization (New Madeline)



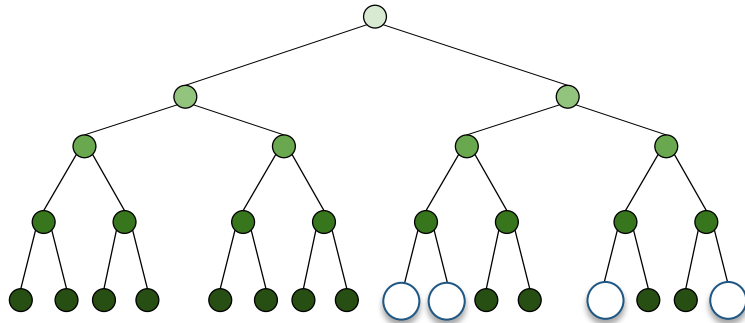
Putting everything together: Process Placement with TreeMatch



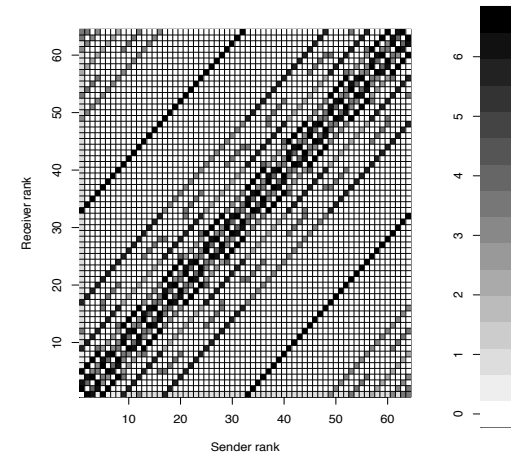
3

Resource selection

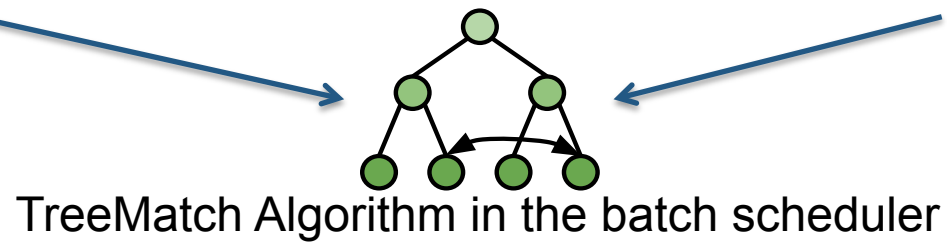
Selecting Resources



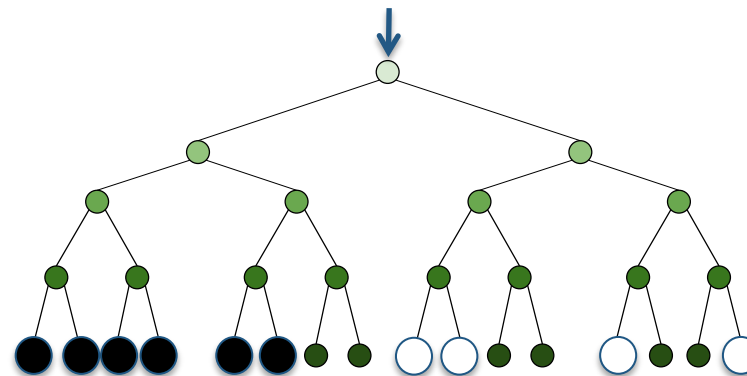
Model of the machine



Model of the application



TreeMatch Algorithm in the batch scheduler

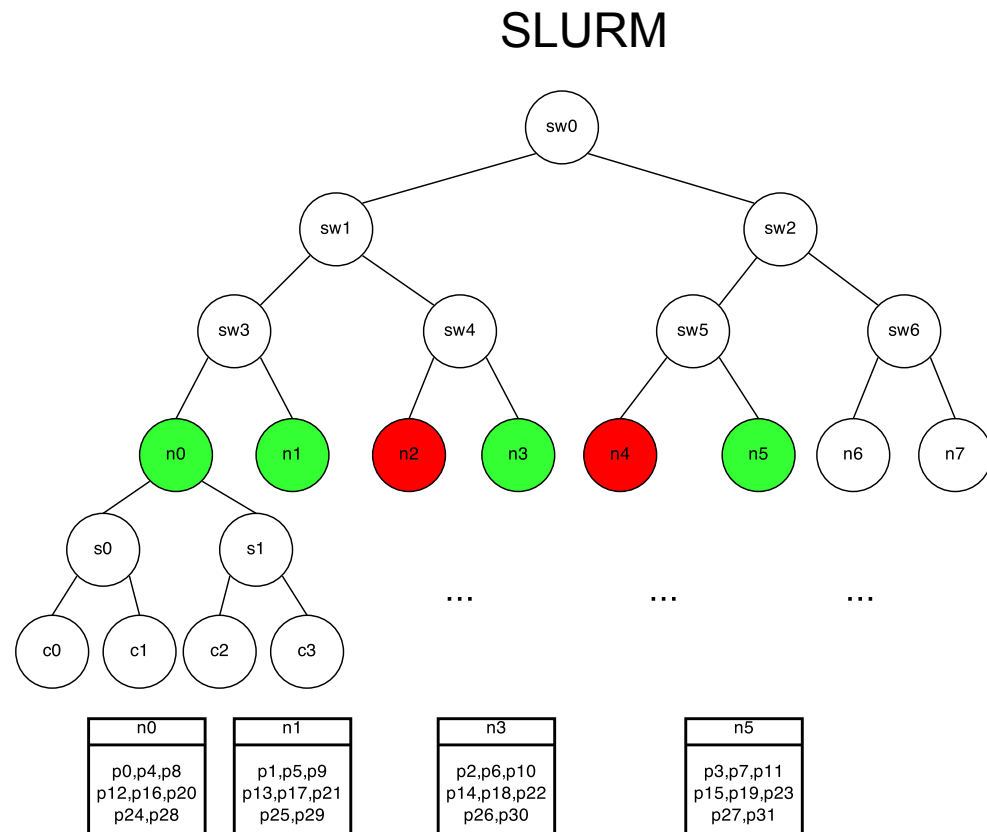


Implementation

- Within SLURM (in collaboration with BULL)
- Plugin
- Resource selection and process placement at the same time

Why topology-aware resource selection could work?

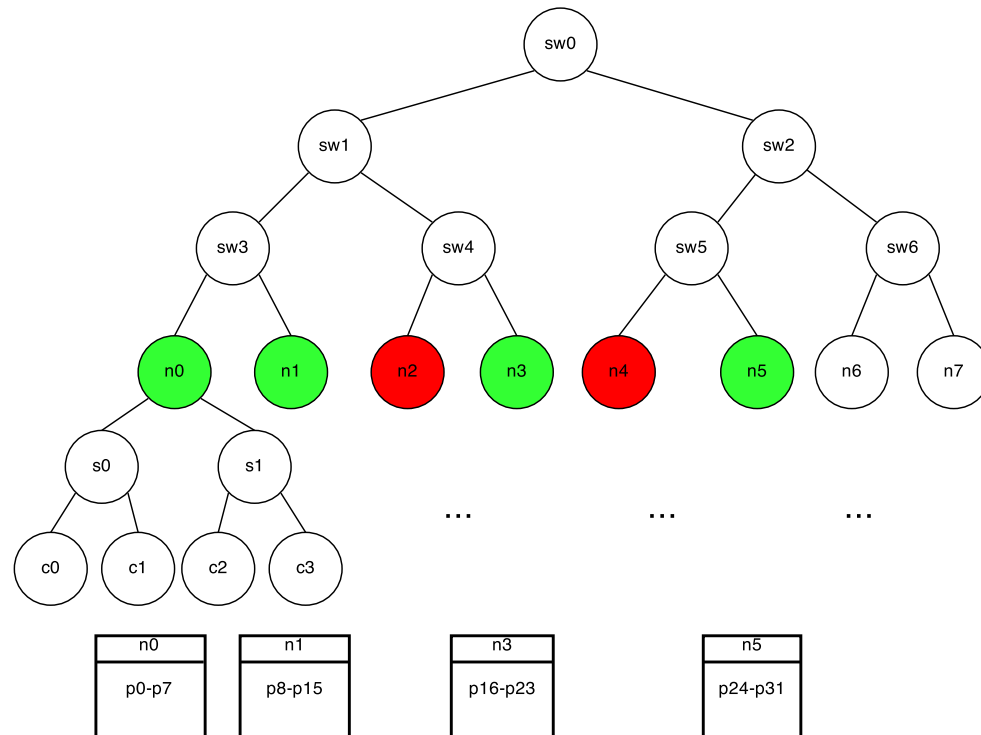
Procs	0-7	8-15	16-23	24-31
0-7	0	1000	0	20
8-15	1000	0	10	0
16-23	0	10	0	1000
24-31	20	0	1000	0



Why topology-aware resource selection could work?

SLURM Then TreeMatch

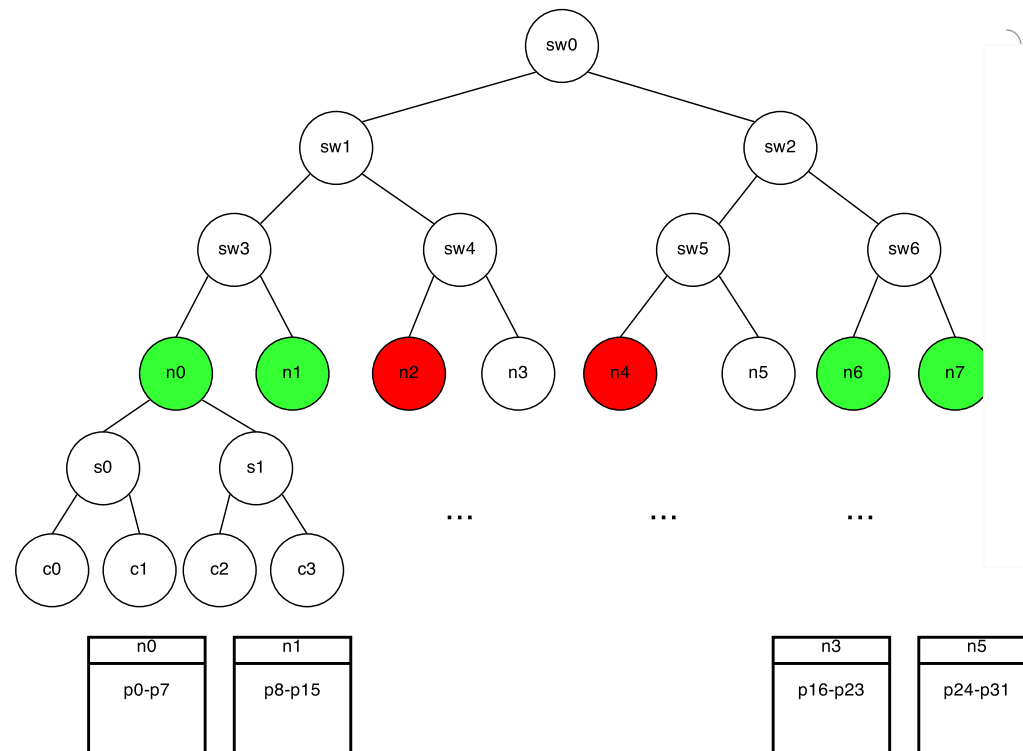
Procs	0-7	8-15	16-23	24-31
0-7	0	1000	0	20
8-15	1000	0	10	0
16-23	0	10	0	1000
24-31	20	0	1000	0



Why topology-aware resource selection could work?

SLURM and TreeMatch

Procs	0-7	8-15	16-23	24-31
0-7	0	1000	0	20
8-15	1000	0	10	0
16-23	0	10	0	1000
24-31	20	0	1000	0



Early experiments

Same protocol as SLURM/Bull team.

Simulation using real traces of the Curie CEA machine:
80640 cores.

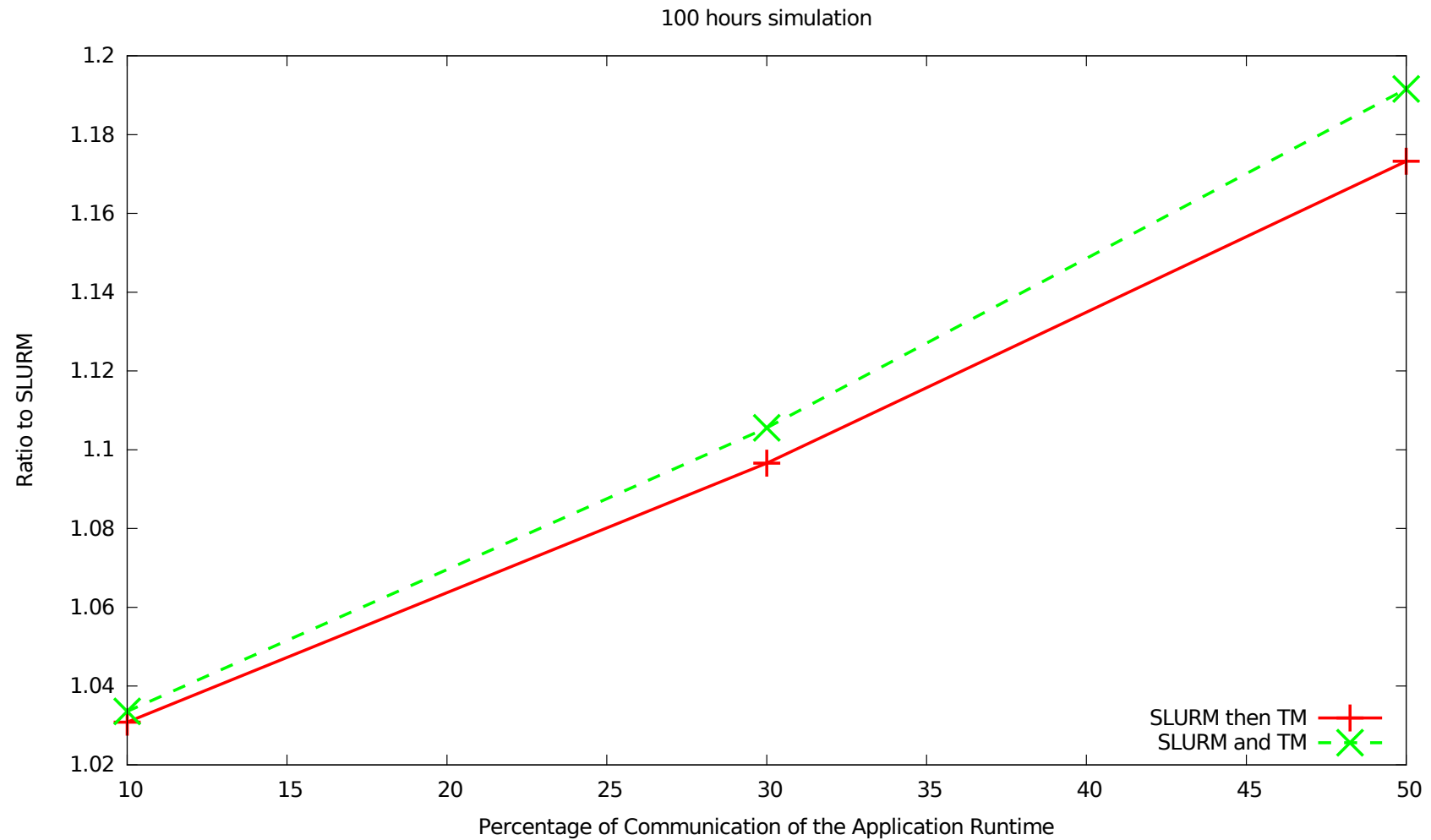
Model of performance gain of TreeMatch depending on the
amount of communication performed by application
(10%, 30%, 50%).

Same starting workflow:

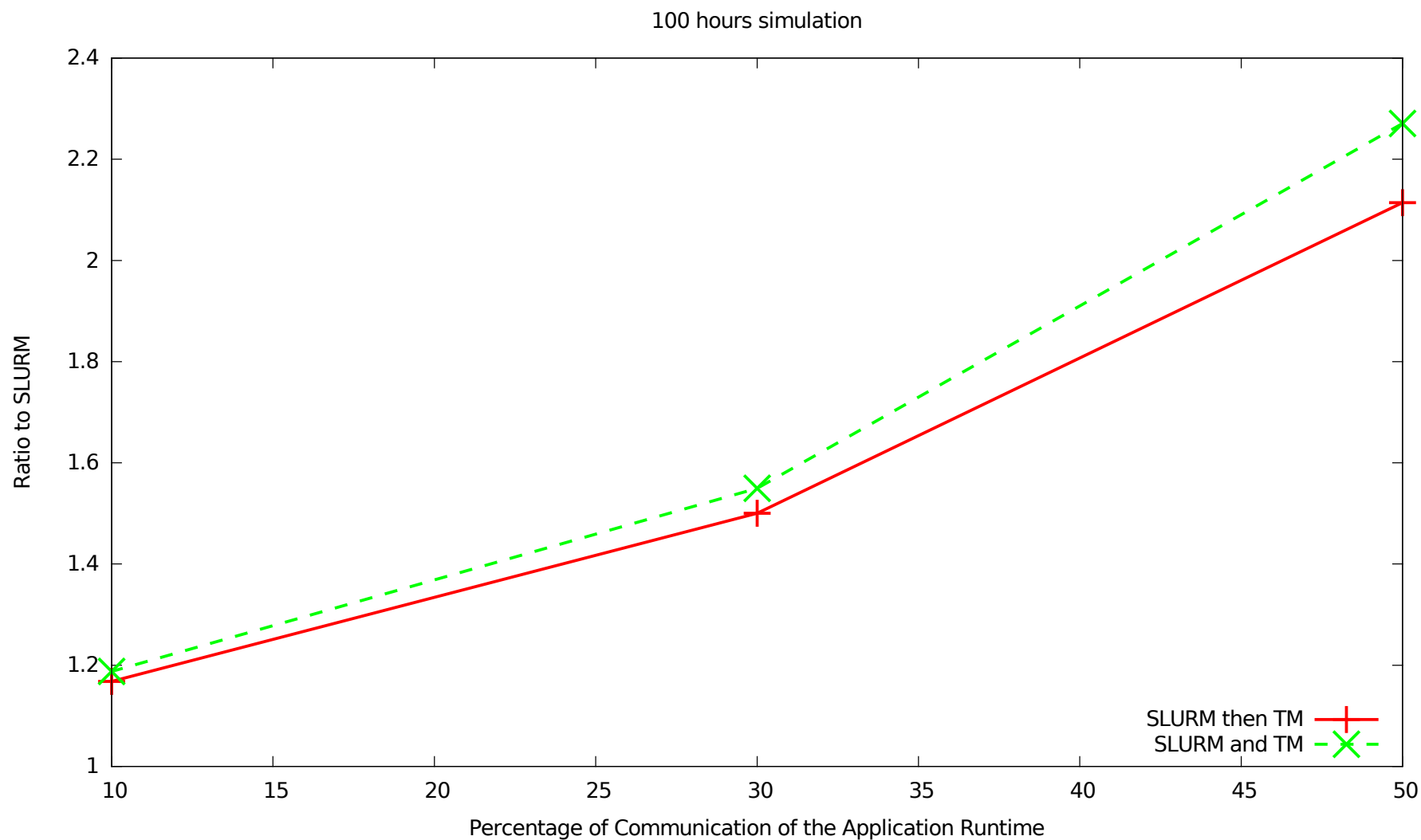
- 130 running jobs
- 26 queued jobs
- 372 submitted jobs (1 hour)

Evaluation on the difference of the submitted jobs.

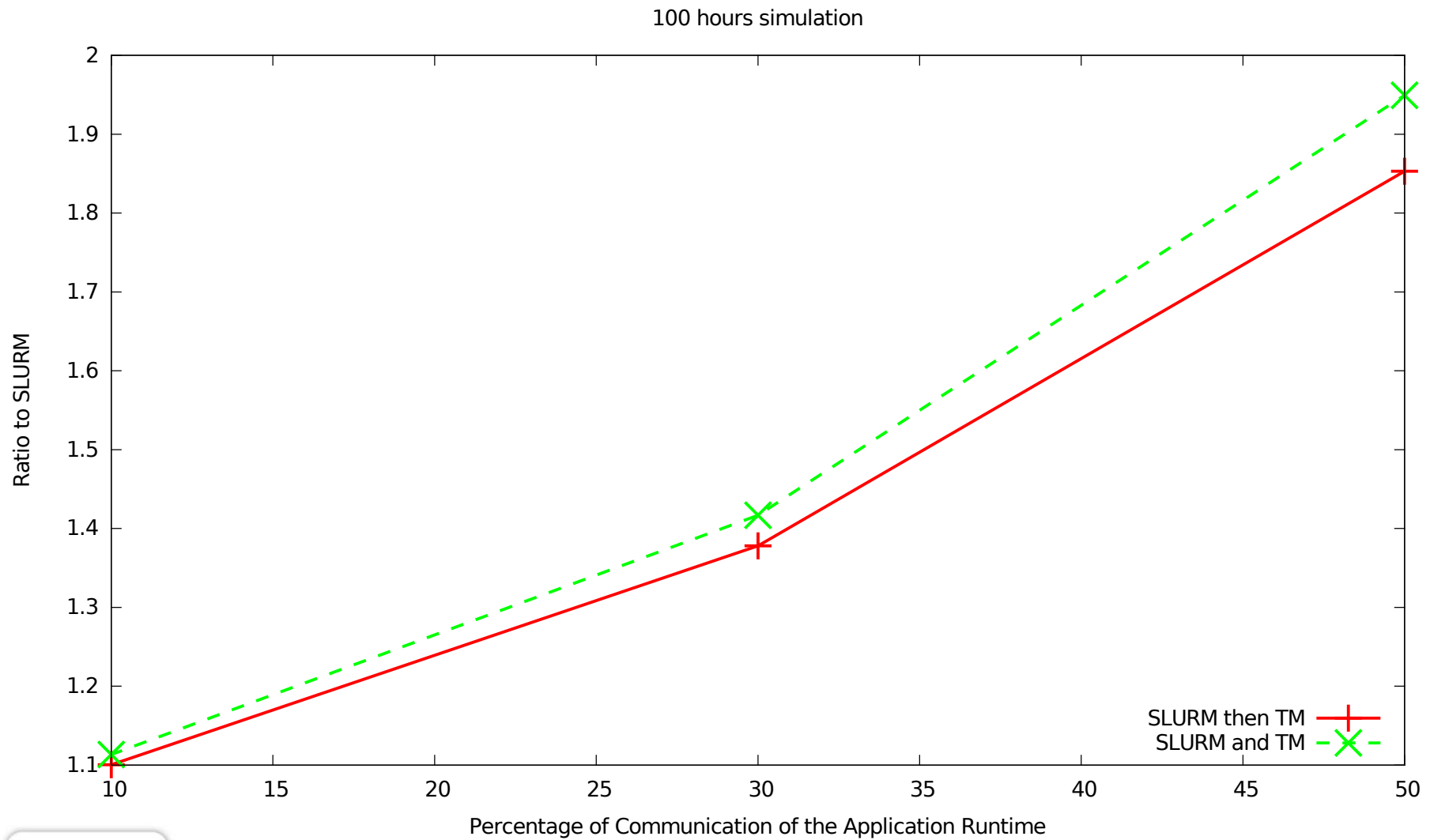
Simulation: makespan



Simulation: average stretch



Simulation: average flow



Conclusion

Take Away Message

Locality! Bytes are more important than flops

Not everything can be optimized statically at compile time

Need for runtime topology-aware data management

Need to take into consideration the whole application ecosystem such as the storage or the batch scheduler