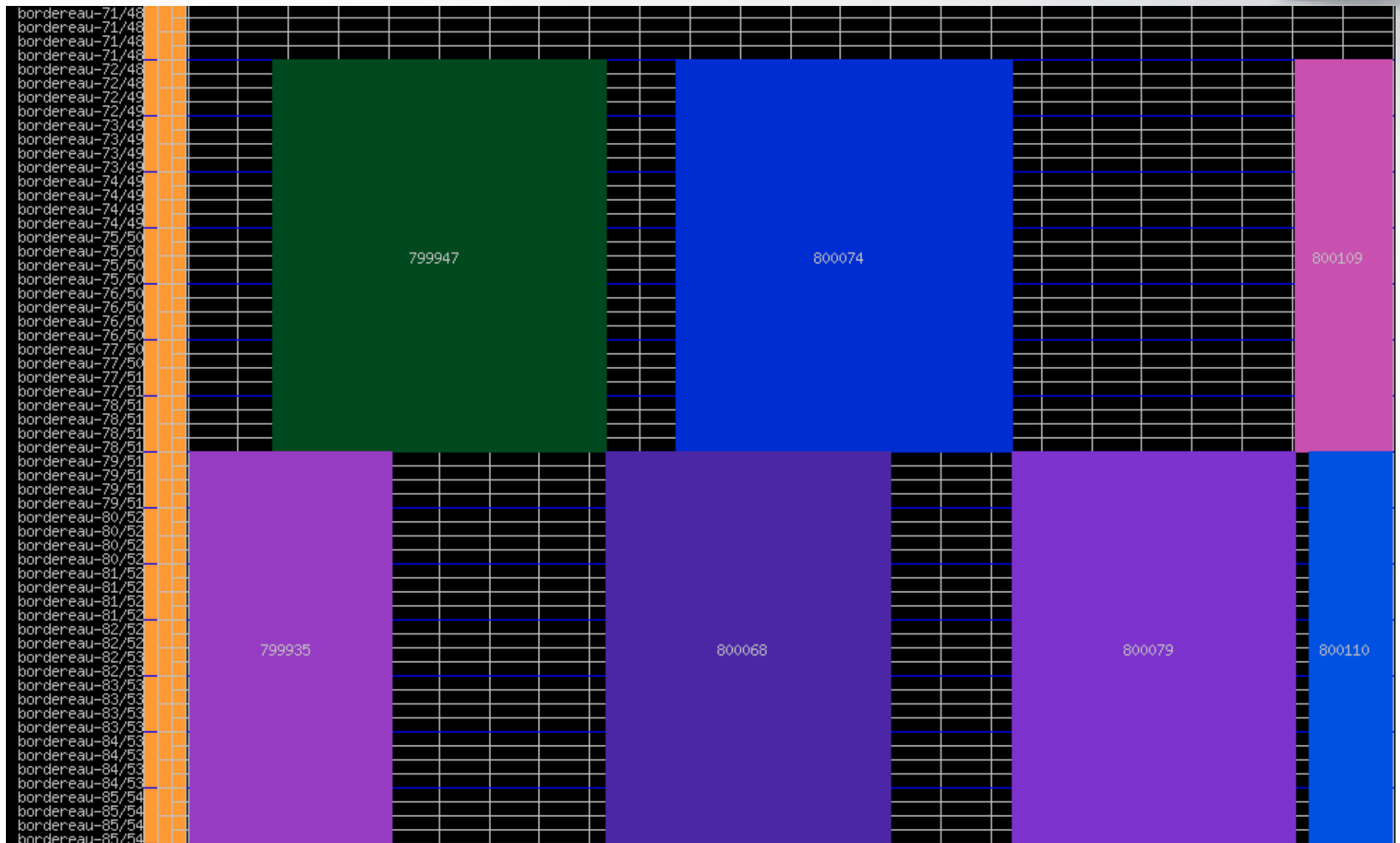




Toward MPI Malleable Applications upon a Scientific Grid Environment

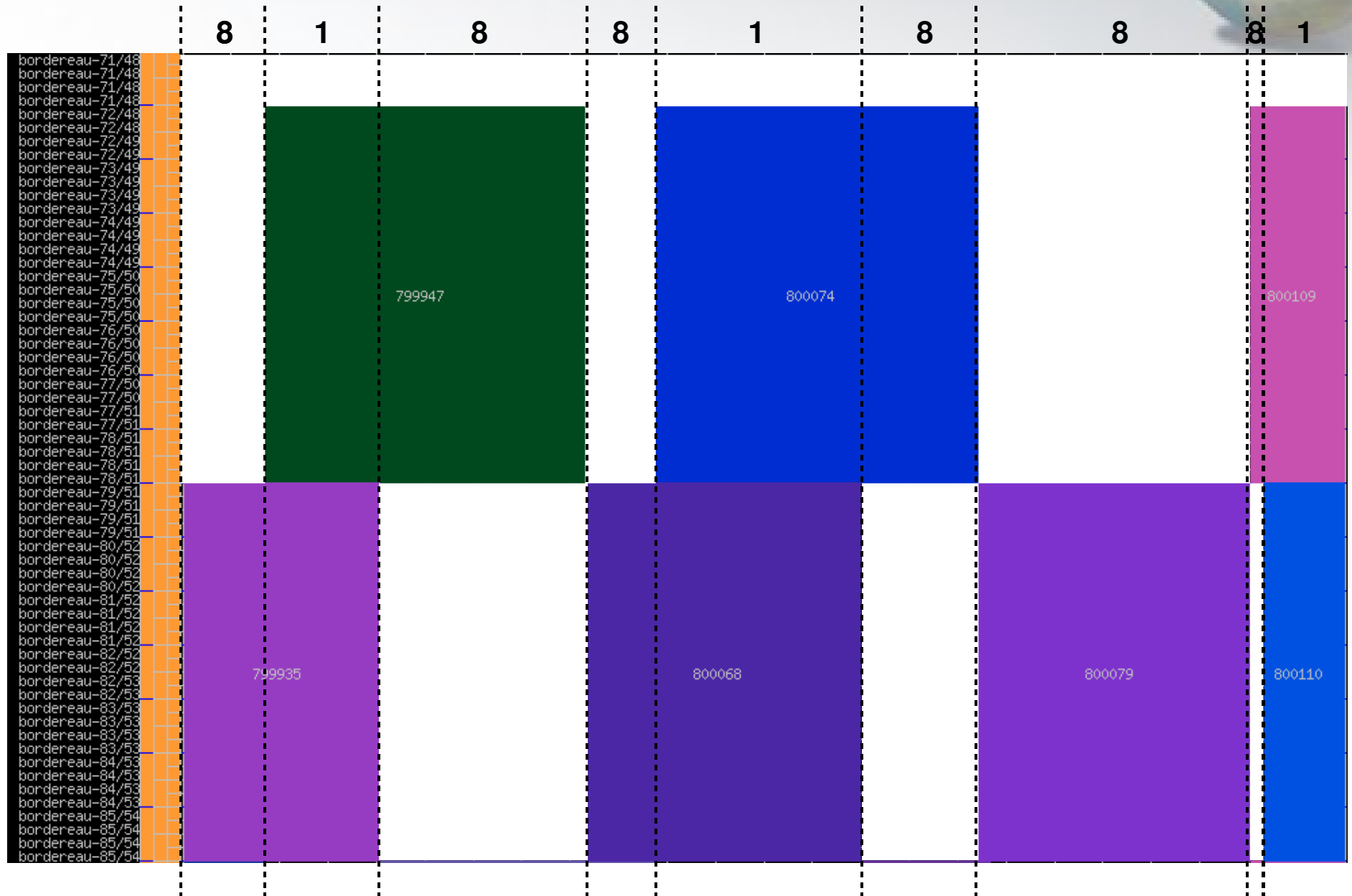
Márcia Cristina Cera, Nicolas Maillard and Philippe O. A. Navaux

Motivation



Grid 5000: Bordereau Cluster

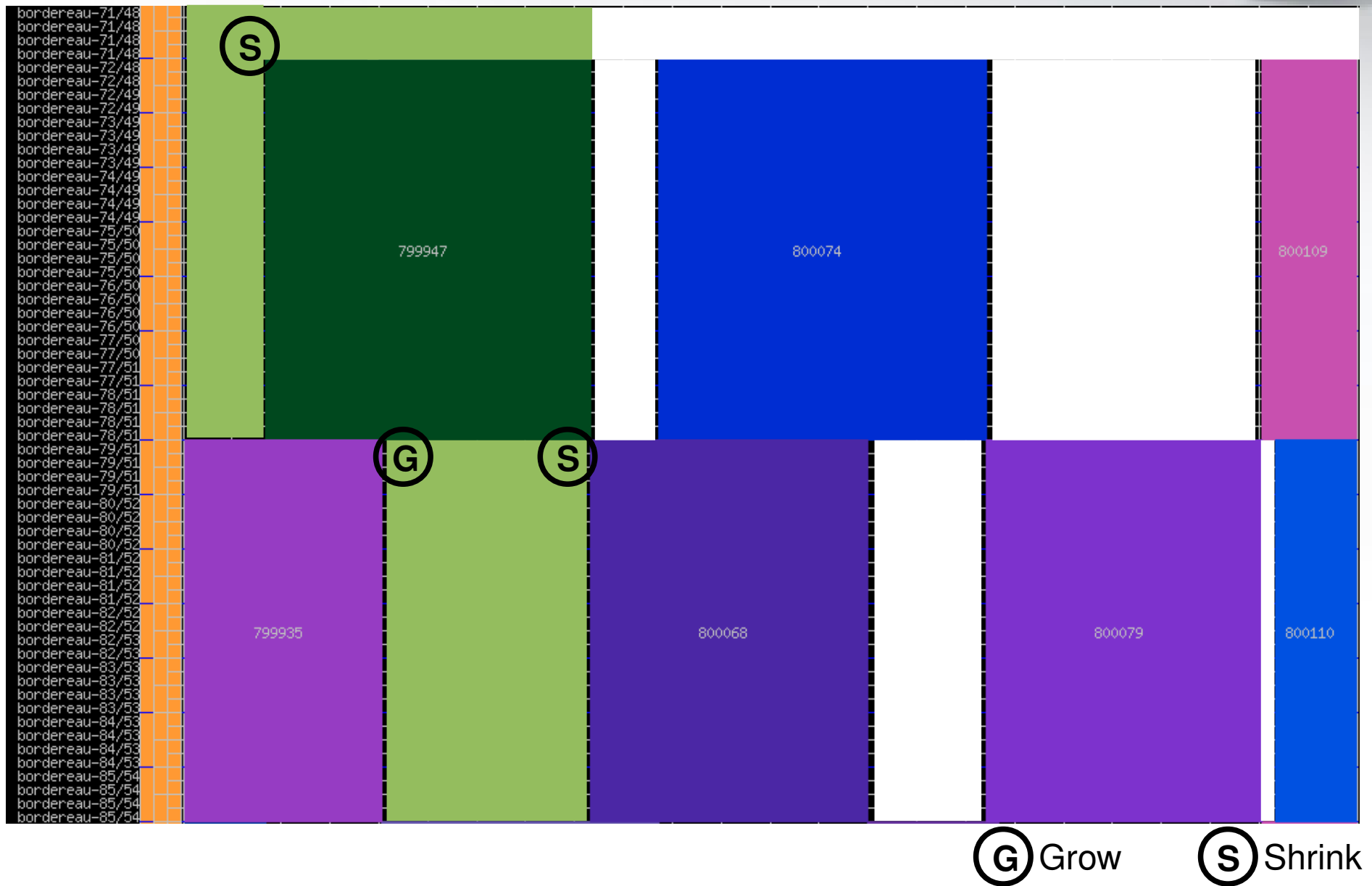
Motivation



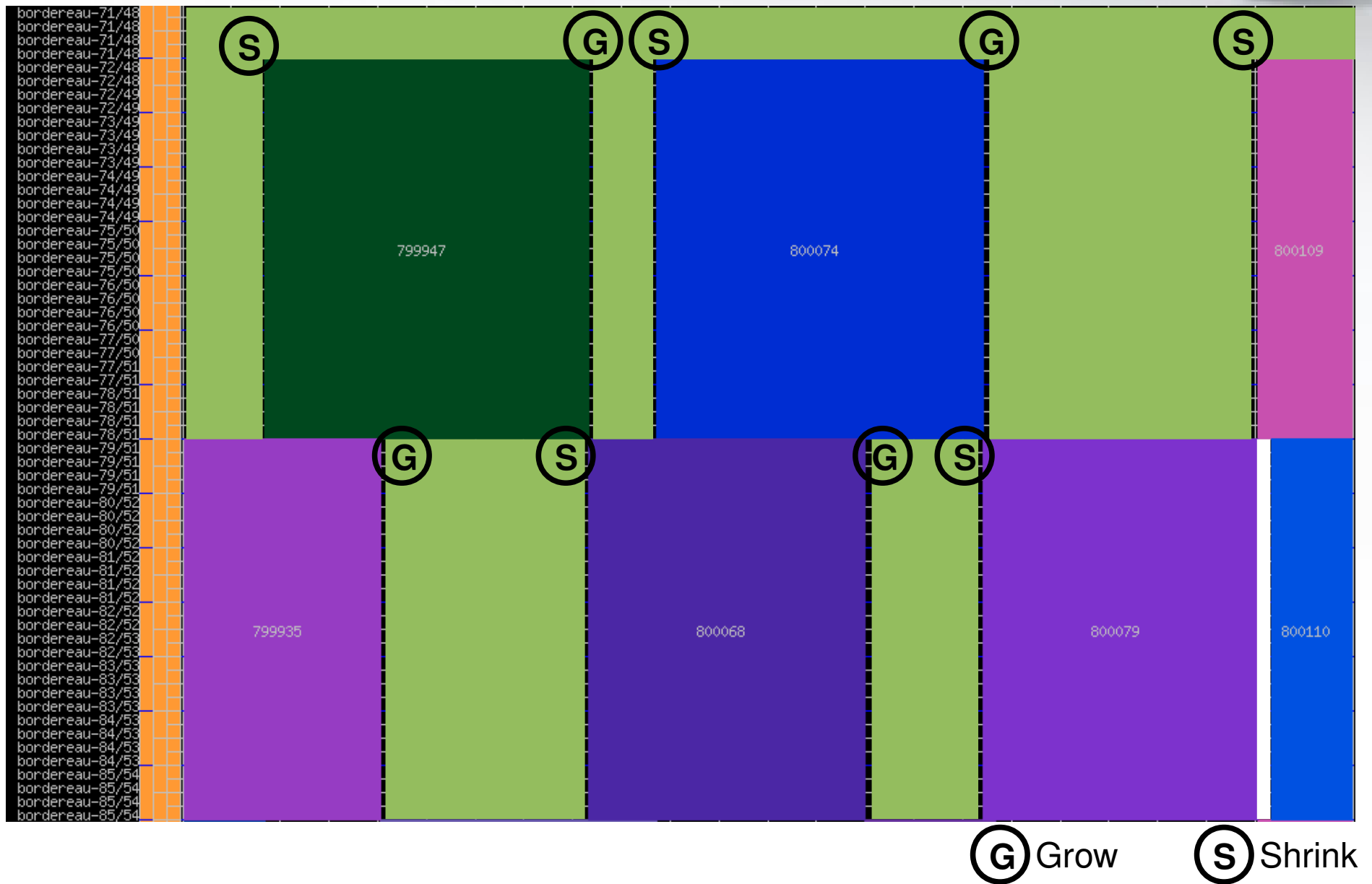
Motivation



Motivation



Motivation



Toward Malleability



- ◆ **Runtime system support**
 - ◆ Dynamic resources management
 - ◆ Which resources are available?
 - ◆ How much time?
 - ◆ Who will receive them?

- ◆ **Programming environment support**
 - ◆ Enabling malleable operations
 - ◆ *Growing*: identify and load new resources
 - ◆ *Shrinking*: ensure application execution while resources are released

Toward Malleability



♦ **Runtime system support**

- ♦ Dynamic resources management
 - ♦ Which resources are available?
 - ♦ How much time?
 - ♦ Who will receive them?

OAR

♦ **Programming environment support**

- ♦ Enabling malleable operations
 - ♦ *Growing*: identify and load new resources
 - ♦ *Shrinking*: ensure application execution while resources are released

MPI-2

Dynamic Resources Management



- ♦ **OAR resource management system**
 - ♦ Cooperation with *Laboratoire d'Informatique de Grenoble*
 - ♦ Modular and flexible
- ♦ *Best Effort* jobs
 - ♦ Harness idle resources and can be killed at any time
- ♦ OAR malleable jobs composed by two parties:
 - ♦ **Rigid** – minimum amount of resources required by the application
 - ♦ Ensures the safe application execution
 - ♦ **Best Effort** – the flexible part of the job
 - ♦ As large as the amount of resources available
 - ♦ Growing: further submissions of Best Effort jobs
 - ♦ Shrinking: kill of the Best Effort jobs



Towards MPI Malleable Applications

- ♦ **MPI-2 dynamic process creation**

- ♦ MPI_Comm_spawn

- ♦ **Malleability with MPI-2**

- ♦ *Growing*: application spawns processes upon new resources loading them
 - ♦ *Shrink*: tasks are stopped and set to be re-started in the future before the releasing of a required resource

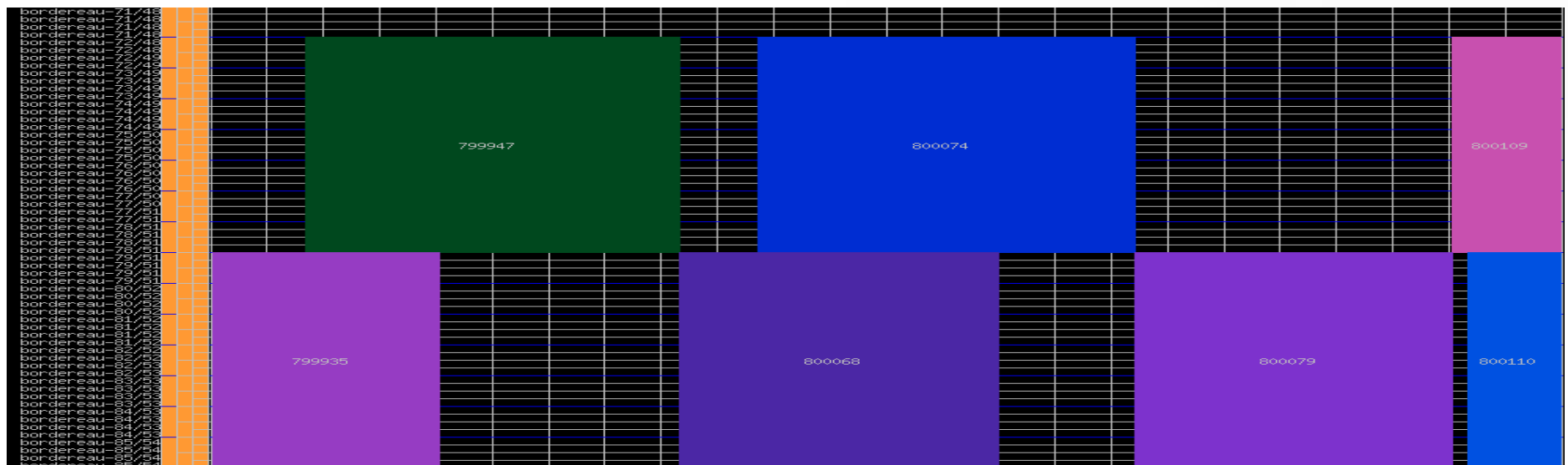
- ♦ *libDynamicMPI*

- ♦ Inform the physical location of spawning tasks
 - ♦ Decide new locations according to the resources' availability
 - ♦ Interact with OAR to be updated about resources' availability



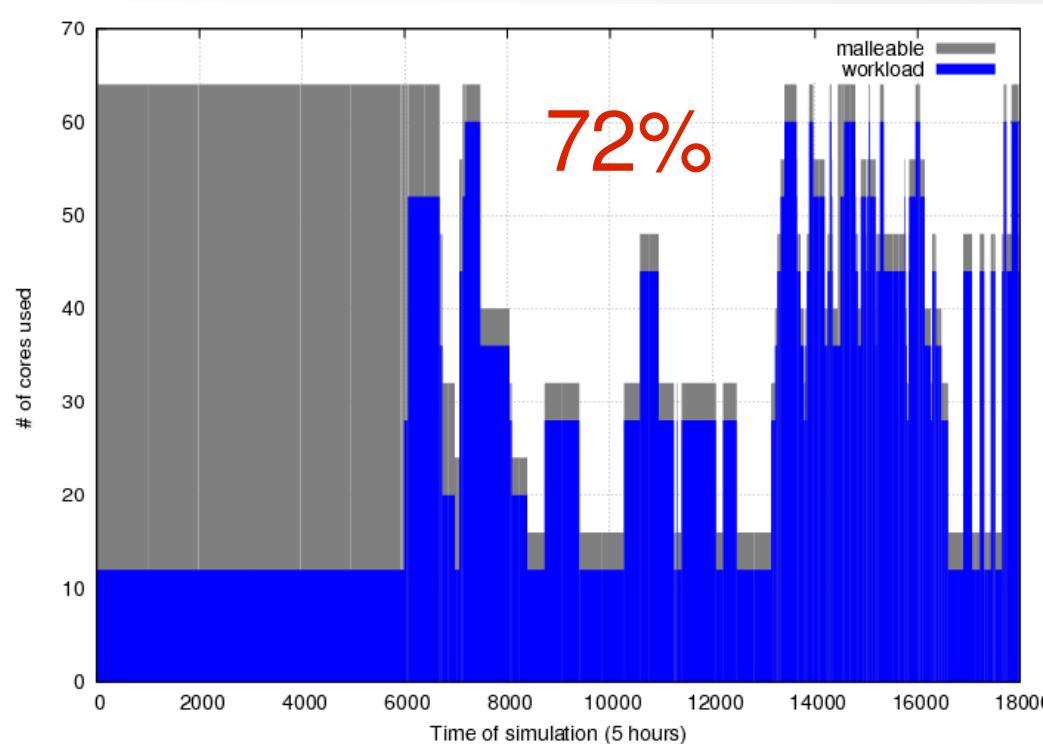
Testbed Environment

- ◆ DAS2 Workload of the production clusters
- ◆ Workload of one cluster with 40% of utilization is injected into OAR
- ◆ The resource utilization of **Moldable-*Best Effort*** and **Malleable** applications are compared

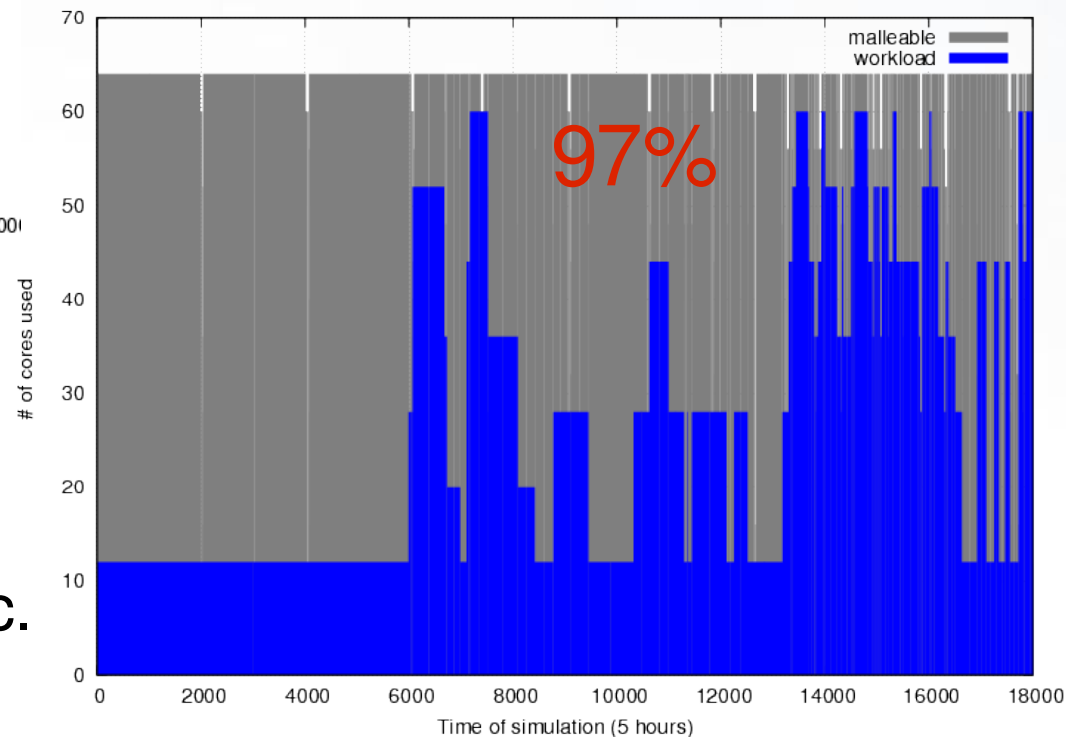




Experimental Results



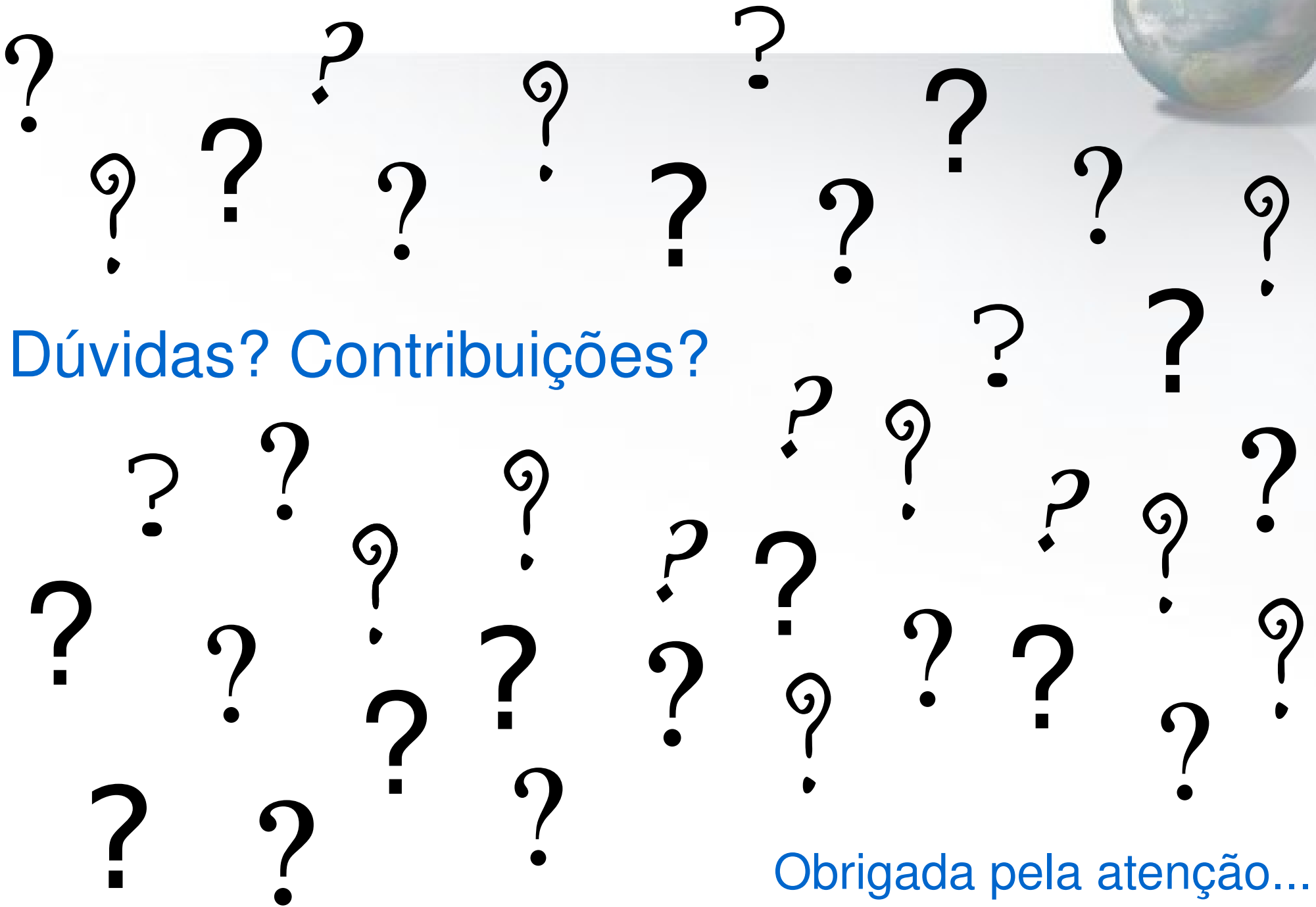
Idle resources used:	32%	;	57%
Jobs Terminated:	4	;	8
Jobs Errors:	5	;	0
Response Time:	8 sec.	;	44 sec.



Conclusion



- MPI malleable applications are enabled by:
 - **Application programming support:** MPI-2 features and fault tolerance procedure
 - **Dynamic resources management:** OAR provides dynamic resources taken advantage of the unused resources
- Malleability can improve the resource utilization
 - Our experiences shown a **gain of 26%** when compared to a Moldable-*Best Effort* approach
- Future works
 - Increase the range of cluster workloads tested
 - Execution of multiple malleable jobs
 - Defining of OAR Scheduling Policies
 - Improve *libDynamicMPI* and OAR communication



Obrigada pela atenção...