POLITECNICO DI MILANO





The MIG Framework: Enabling Transparent Process Migration in Open MPI

EuroMPI 2016 - September 26th Edinburgh (SCO)

Federico Reghenzani, Gianmario Pozzi, Giuseppe Massari,
Simone Libutti and William Fornaciari

giuseppe.massari@polimi.it

Outline

Migrating MPI Applications

- Motivations
- Limitations

The MIG framework

- Basic idea
- Design and implementation
- The migration phases

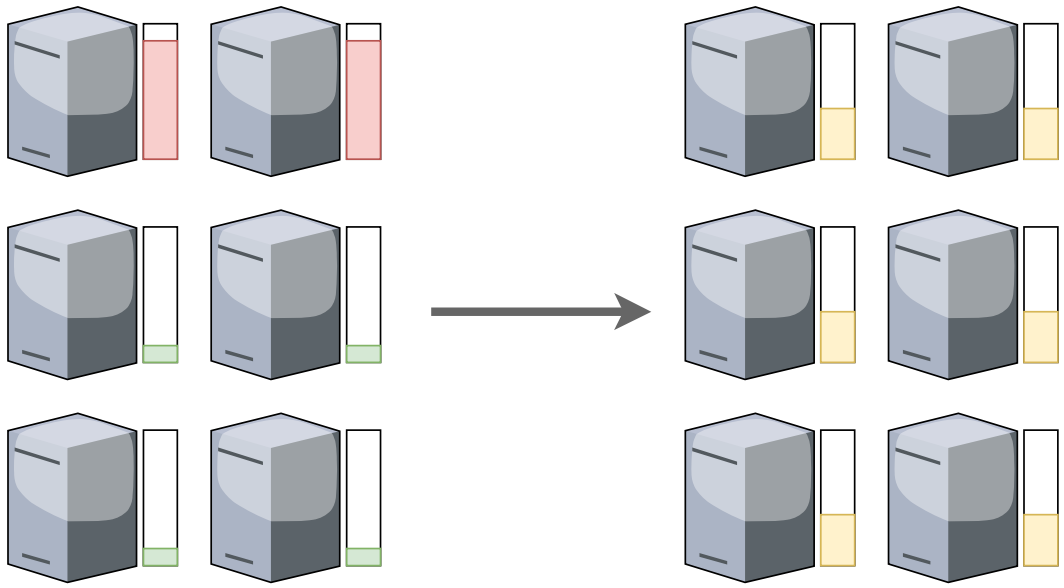
Experimental Results

Conclusions

Migrating MPI Applications

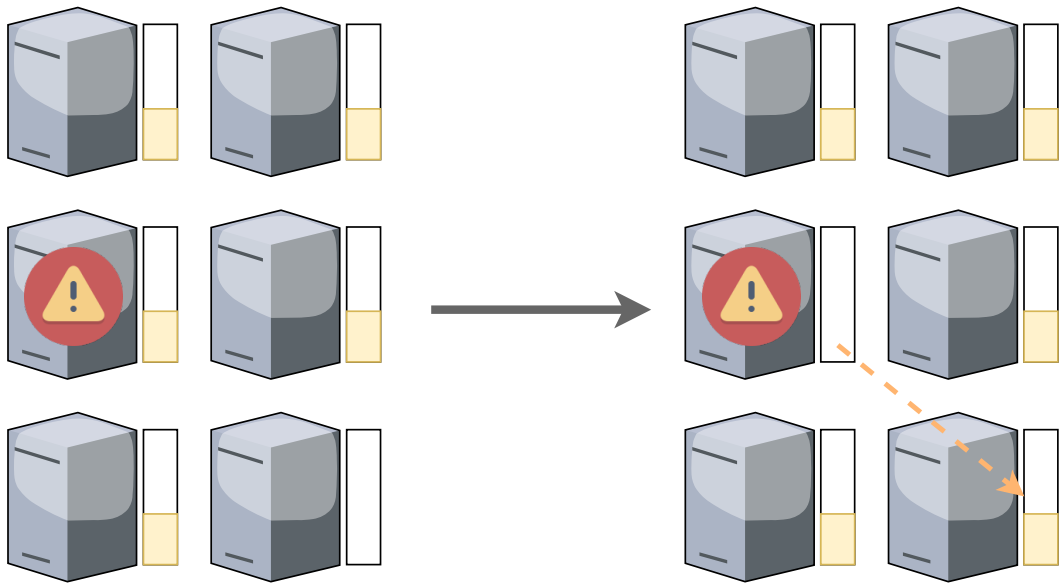
Load balancing

- Dynamic balancing of the workload over the system nodes
  - Performance / thermal management / power consumption



Fault-tolerance

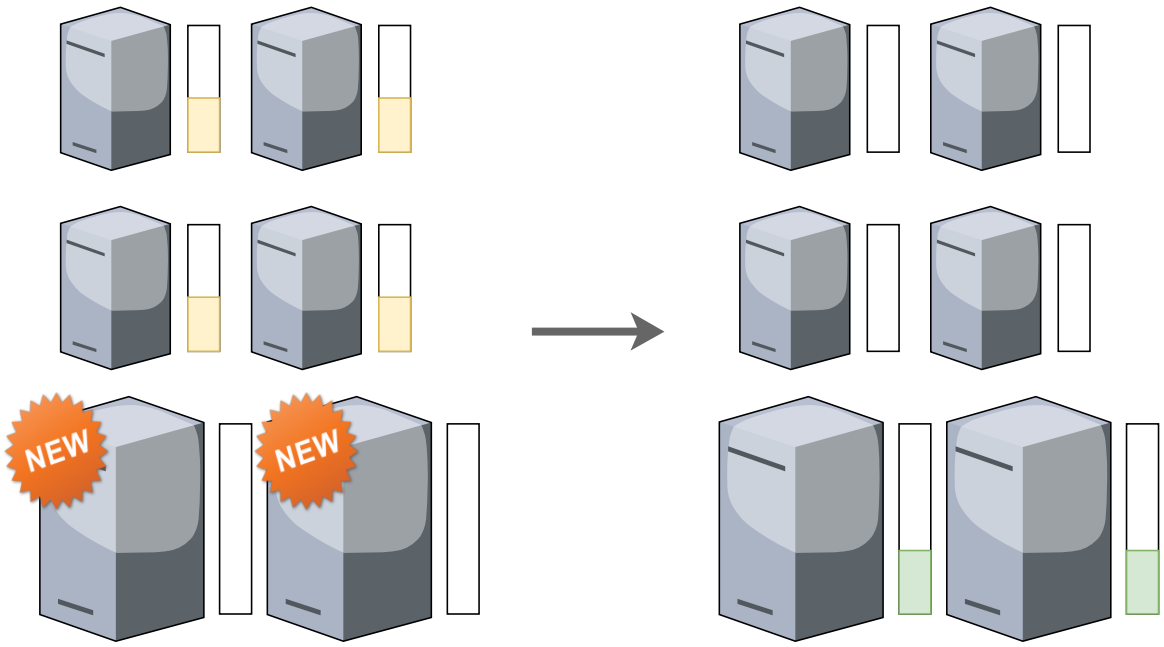
- Faults on a node may require the migration of the workload



Migrating MPI Applications

Performance improvements

- Migration towards new more performing nodes



State of the Art

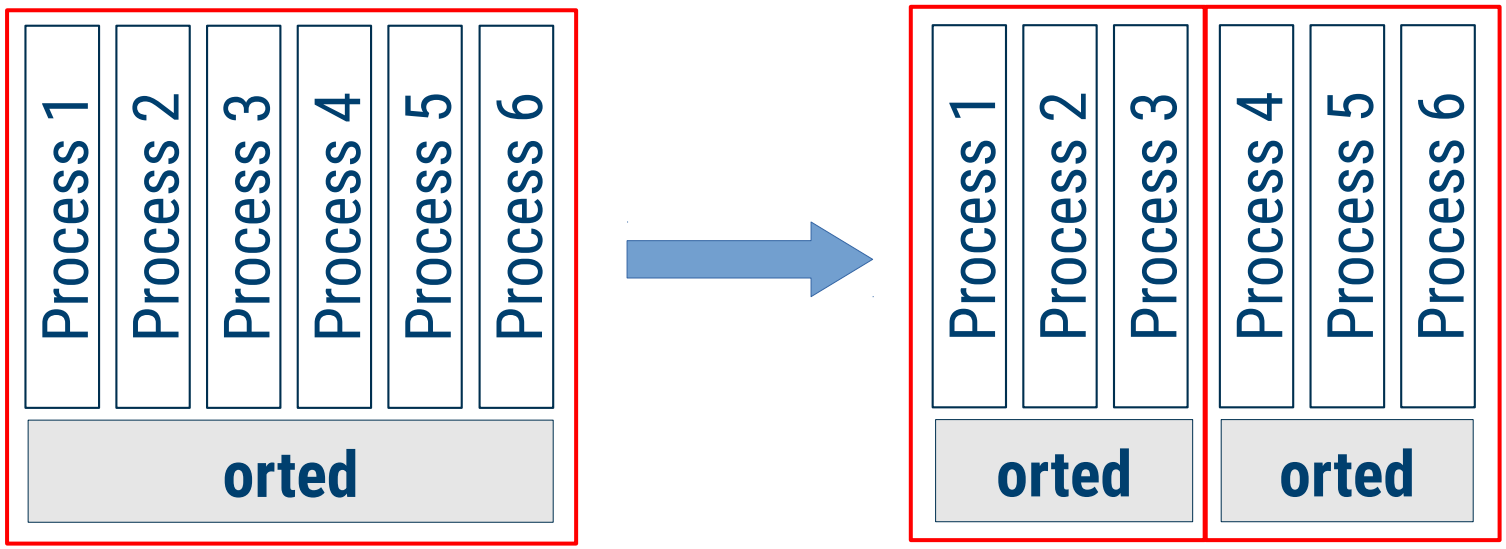
Limitations of already proposed approaches

- Frameworks requiring changes in the application source code
- Coarse-grained migration
  - Entire node workload
  - Virtual Machine or Container granularity
- Not negligible overhead
  - Stop, transfer and resume an entire running system
- Poorly maintanable frameworks
  - Invasive changes introduced in the MPI implementations

MIG: A Open MPI Framework

Basic idea

- Finer grained task migration approach
  - **Group of application processes**
- Open MPI additional framework
- Spread application processes among different ORTE daemons (**orted**)



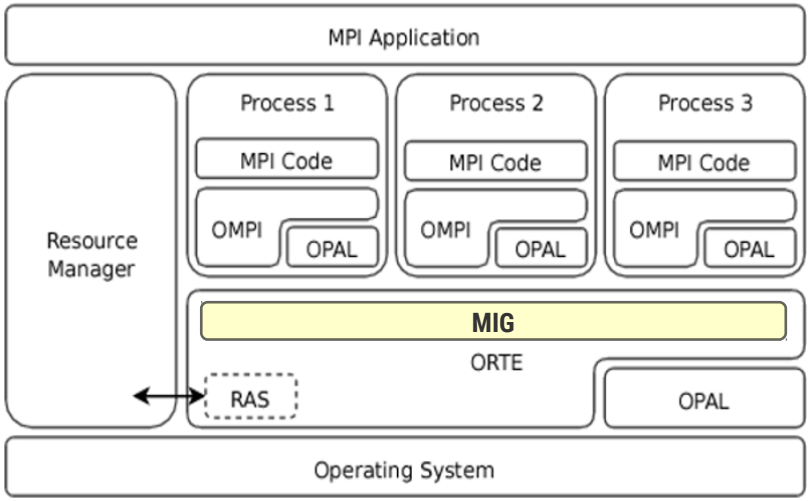
Single orted on a node

Multiple orted on a node

MIG: A Open MPI Framework

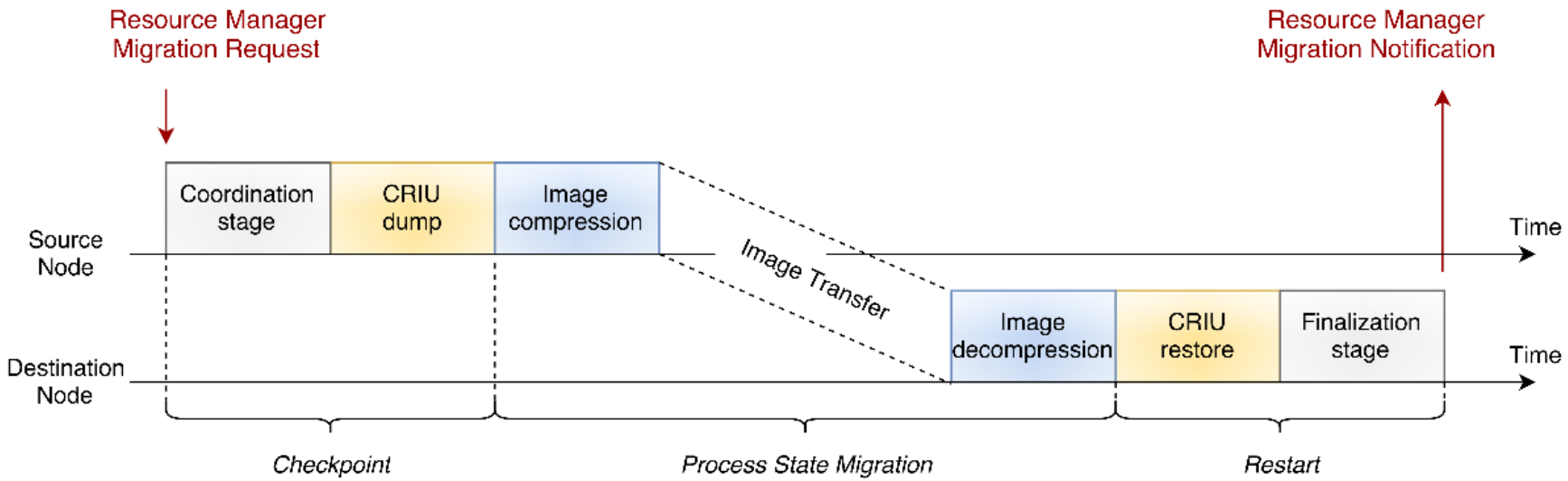
Design and implementation

- Part of the Open MPI ORTE runtime
  - Almost completely self-contained
  - *Byte Transfer Layer* (BTL) (OMPI component) modified to manage the connections
  - *Process Lifetime Management* (PLM) (ORTED component) modified to introduce new message types



- Application processes status saved by using CRIU
  - *Checkpoint/Restore In Userspace*
  - Linux kernel version <= 3.18 required

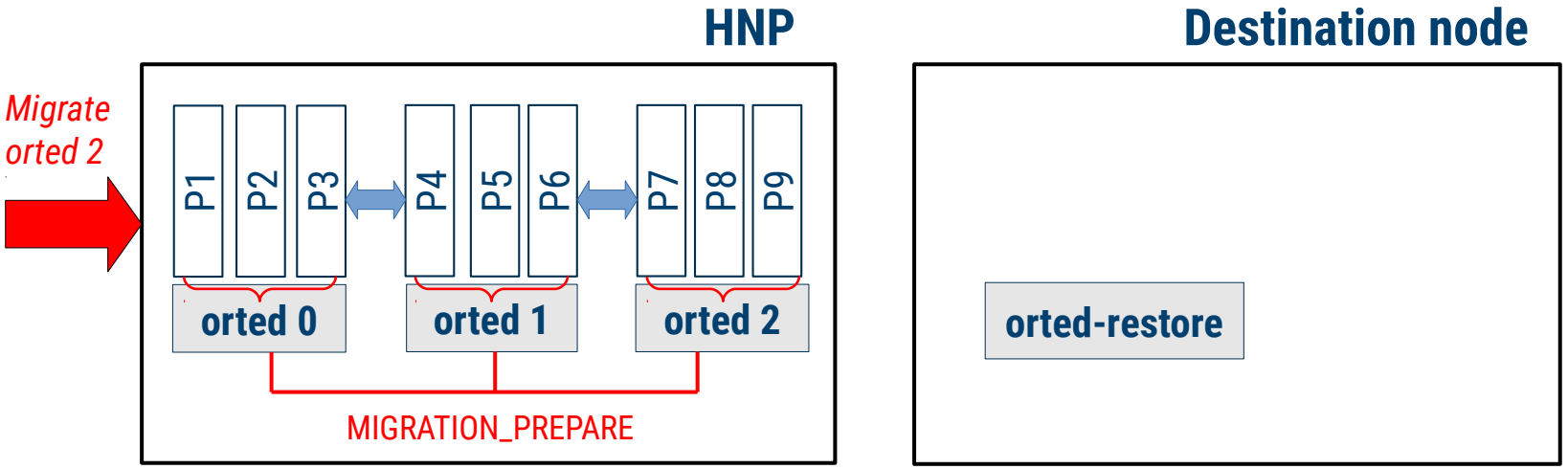
Migration phases overview



Migration Phases

Coordination stage

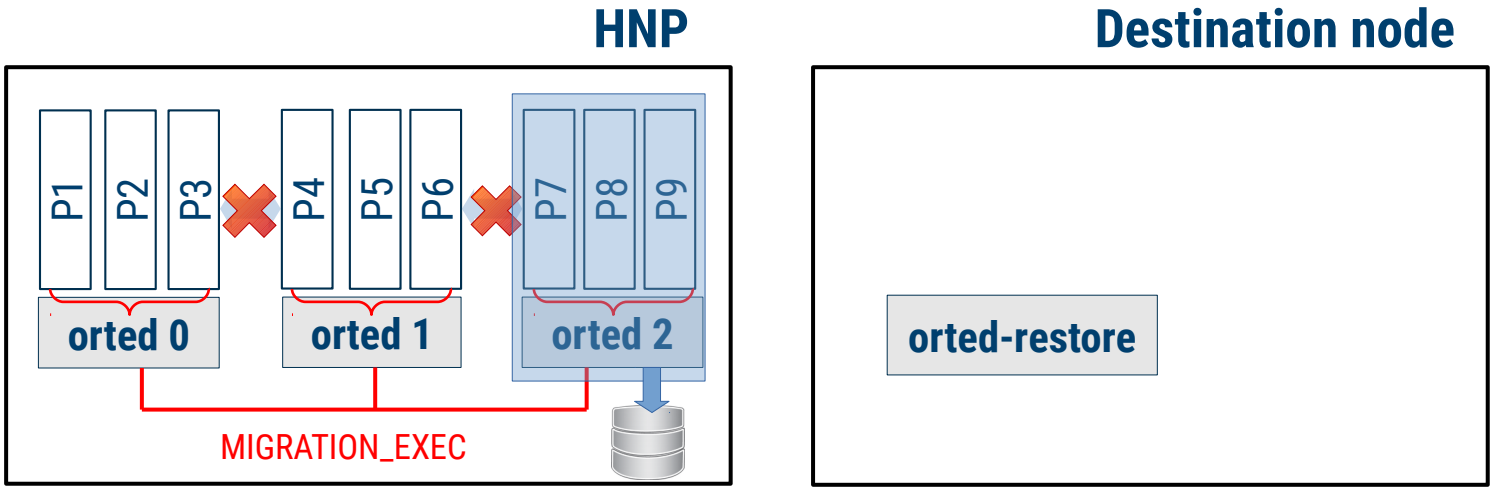
- A migration request is received by HNP (through RAS)
- A **orted-restore** daemon is launched on the destination node
- A MIGRATION_PREPARE is broadcasted to all the **orted** instances (and then to application processes)
- NOT migrating processes stop sending data (TCP) to migrating processes
- HNP is acknowledged



Migration Phases

CRIU dump

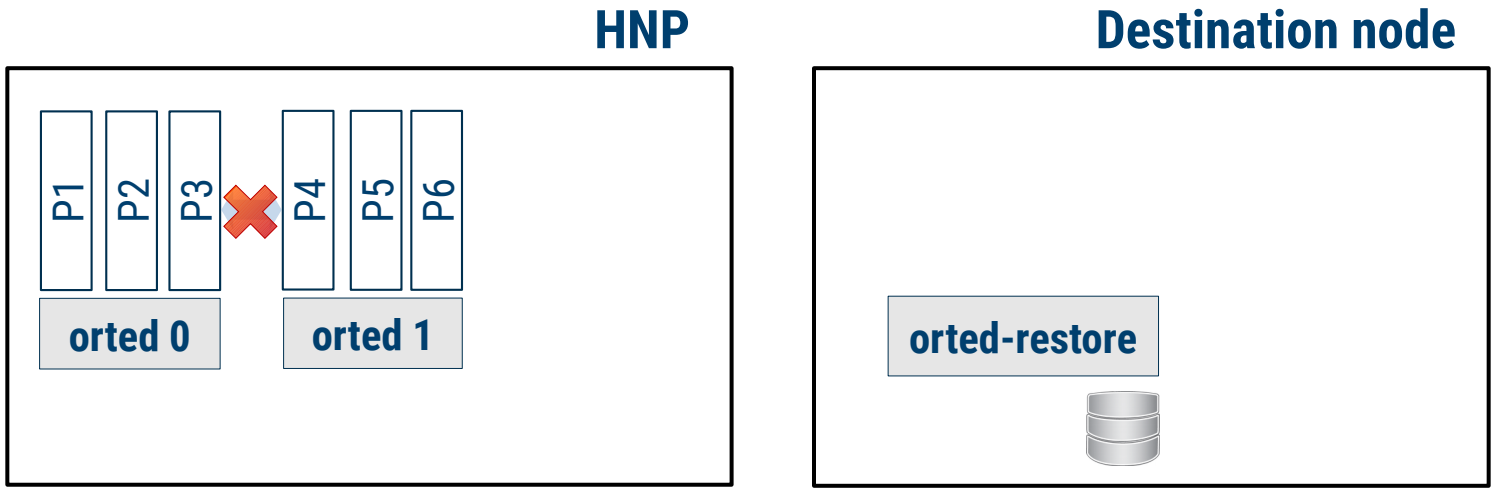
- All the **orted** instances are aware the forthcoming migration
- HNP broadcasts a MIGRATION_EXEC command to all the **orted**
- TCP connections are closed
- Migrating **orted** is acknowledged and the *processes image* is dumped
  - CRIU checkpoint



Migration Phases

Process state migration

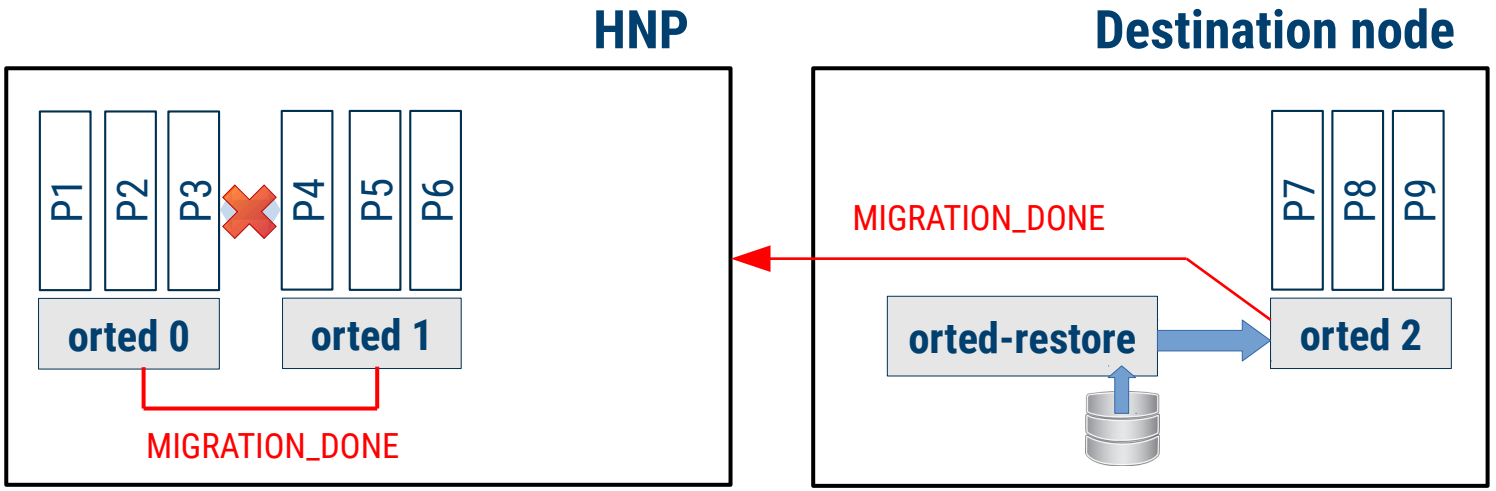
- Processes image (CRIU checkpoint files) packaged into an archive
- The archive is compressed [optional]
- The archive is sent to the destination node



Migration Phases

CRIU restore

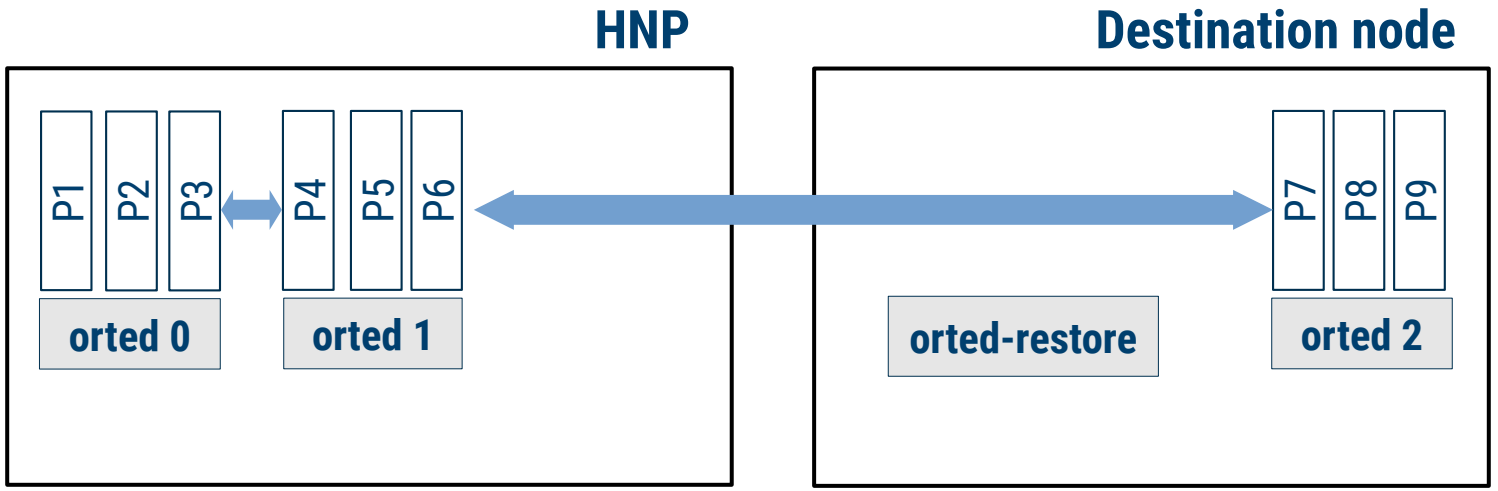
- The archive is received by the destination node
- The archive is uncompressed [optional]
- **orted-restore** restarts the migrated **orted** using CRIU
- Restored **orted** reconnects to HNP and sends a MIGRATION_DONE message
- HNP broadcasts MIGRATION_DONE to the others **orted**



Migration Phases

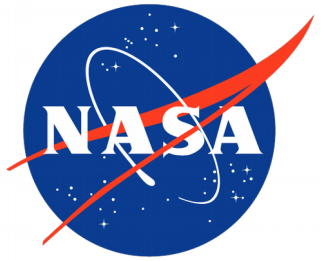
Finalization stage

- The migrated application processes re-open the TCP connections towards other processes (only if needed)



Experimental Results

Setup



- NAS benchmarks (by NASA)
  - **IS, MG** kernels
  - **BT, SP, LU** pseudo-applications (system solvers)
  - Different balancing of *computation* and *communication*
  - Different problem sizes
    - Classes *B*, *C*, *D*

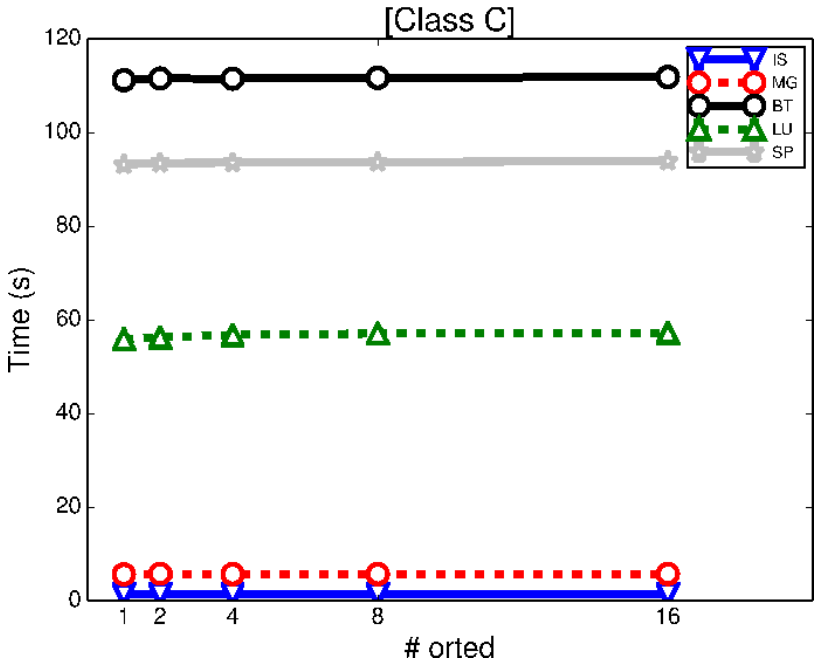
- 2 node distributed system held @ *IT4I (Ostrava, Czech Republic)*
  - Intel Xeon E5-2640 8 core (x 2)
    - Hyper-Threading disabled
  - CentOS 6.7 with Linux kernel v3.18
  - Ethernet interconnection

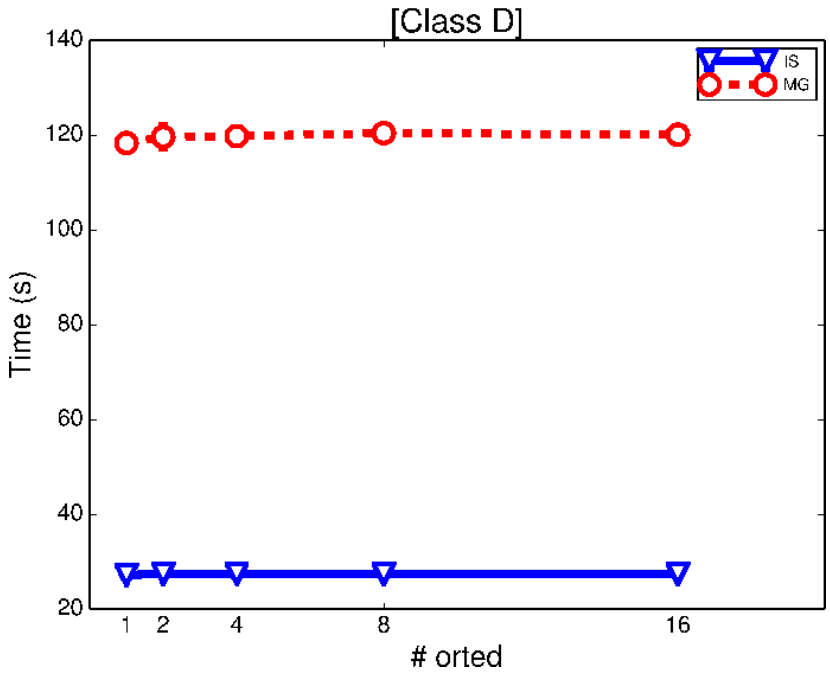
IT4Innovations
national01$#&0
supercomputing
center@#01%101

Experimental Results

Orted granularity overhead

- How much splitting the application processes into multiple **orted** does affect performance?
  - Each application spawned 16 processes
  - Execution time increased by [0.6 – 5]%

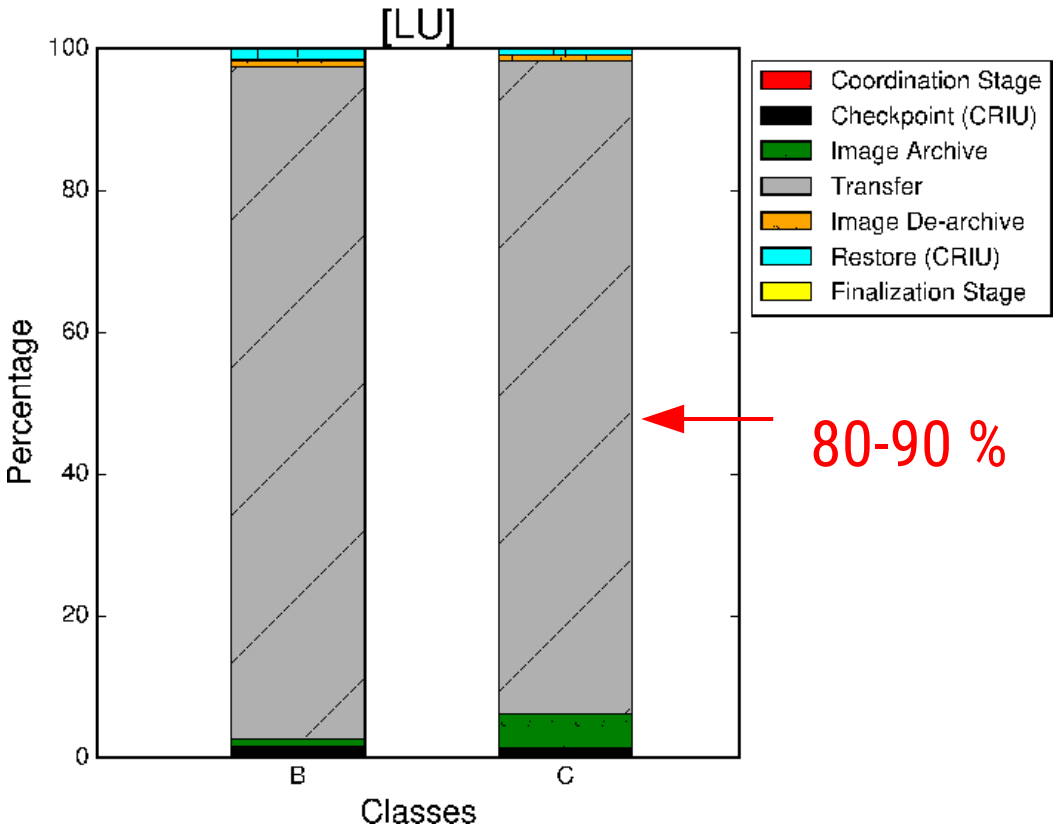




Experimental Results

Migration overhead breakdown

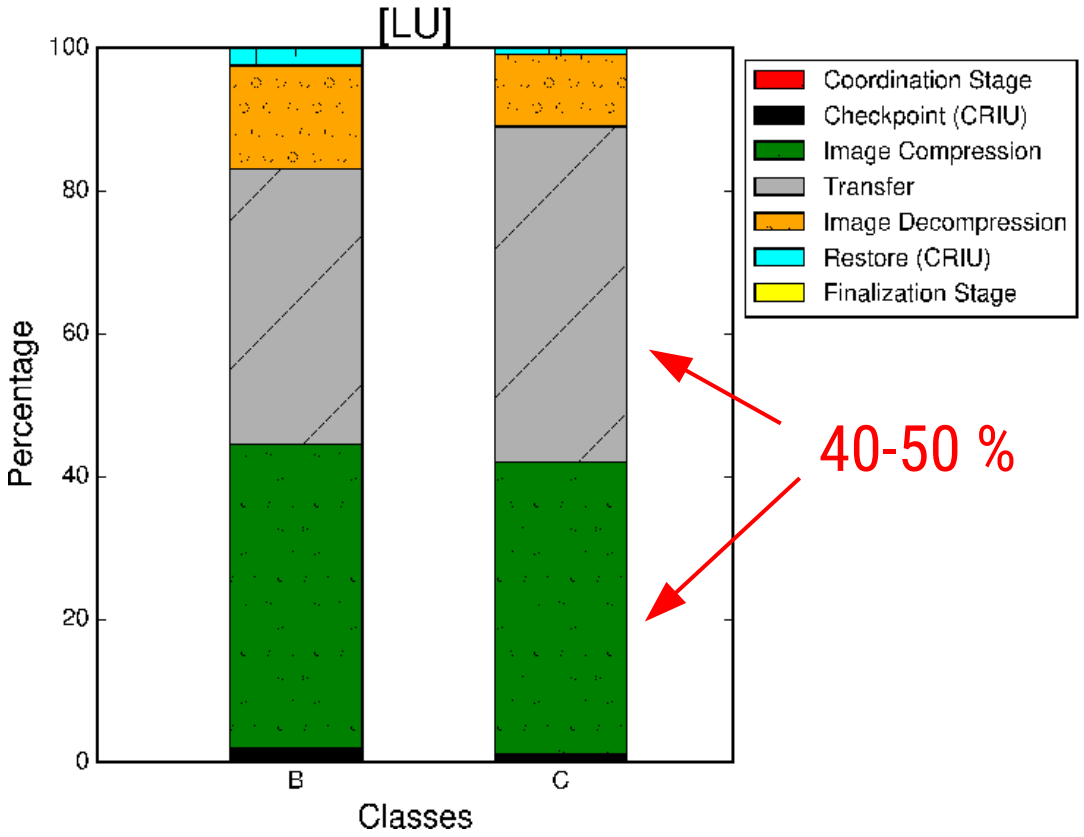
- Impact of each single phase on the overall migration time?
  - Checkpoint image transfer dominates



Experimental Results

Migration overhead breakdown

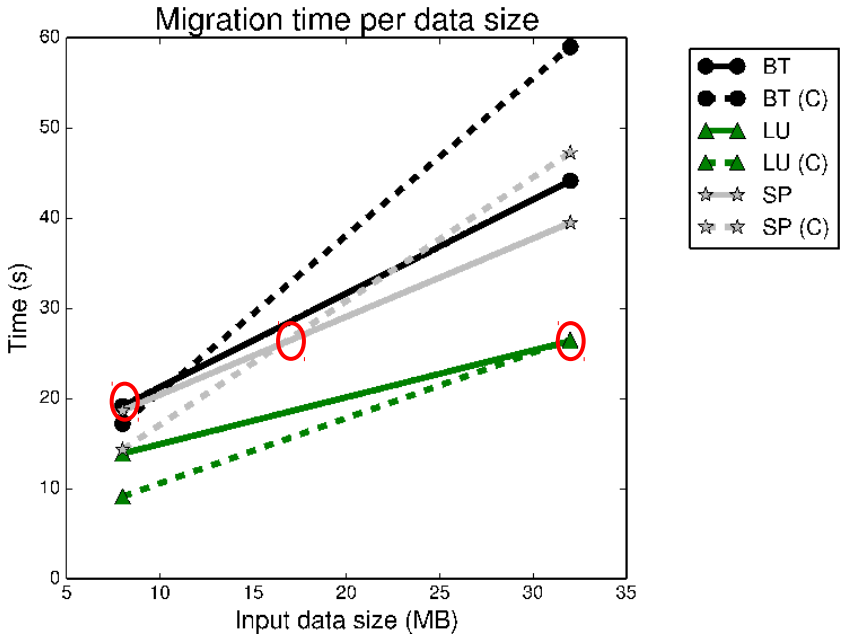
- What if compression/decompression is applied before transfer?
  - Compression/decompression itself requires time

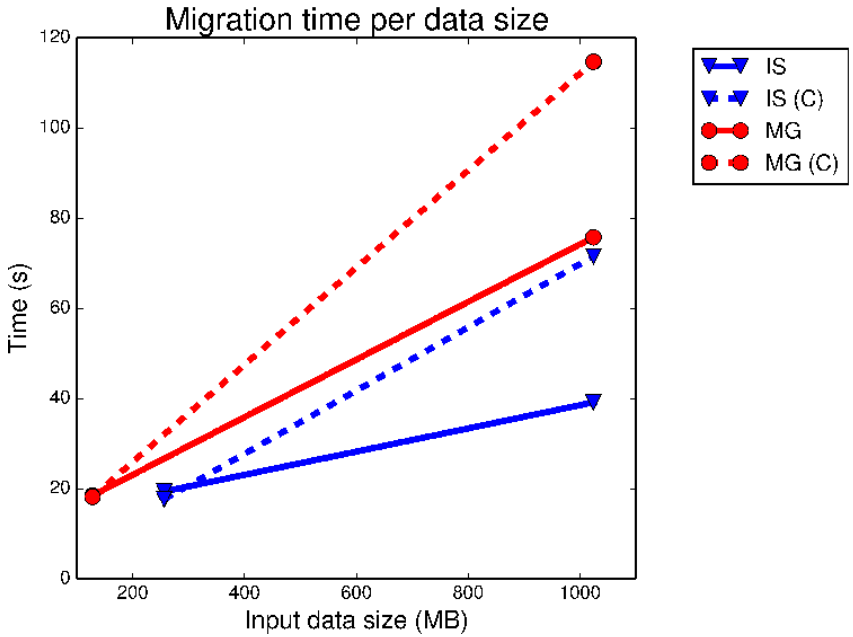


Experimental Results

Input data size dependency

- How much input data affect the migration time?
- To compress or not?
  - *Application* and *data* dependent, but in most cases not





Conclusions

MIG Advantages

- *Finer grained task migration*
  - Group of application processes instead of VMs
- *Completely transparent to the application*
  - No need to introduce specific function calls
- *Maintainability*
  - A mostly self-contained code
- *Portability*
  - It depends only on CRIU (userspace tool)

Future developments

- Rebase on top of Open MPI version 2.0
- InfiniBand support development
- Several under-the-hood optimizations
- Integration with the BarbequeRTRM

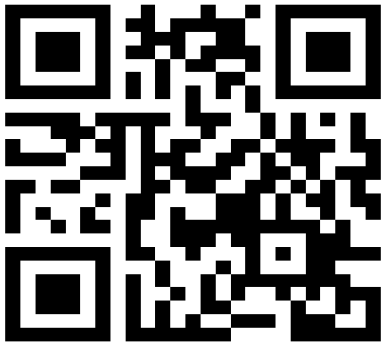
Questions?

Contacts

BarbequeRTRM Open-Source Project (BOSP)

http://bosp.dei.polimi.it

https://twitter.com/BarbequeRTRM



giuseppe.massari@polimi.it