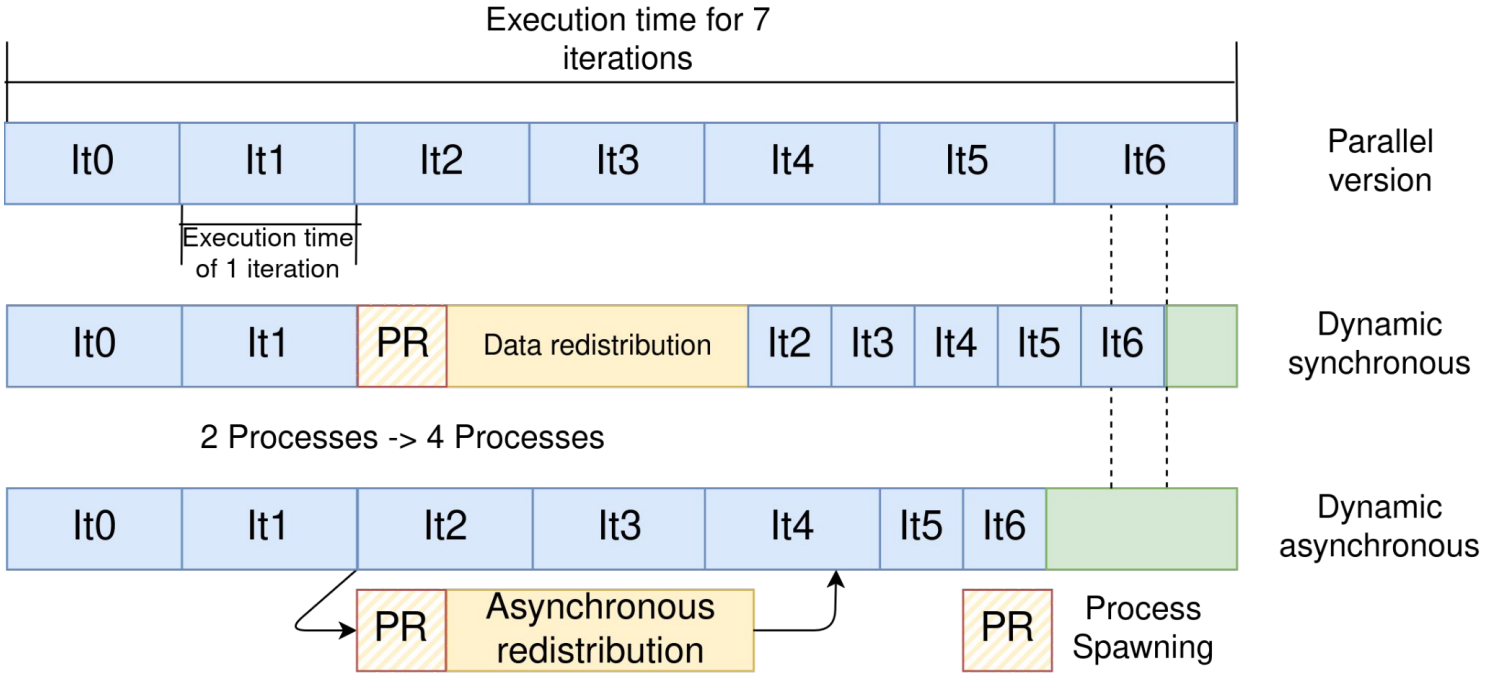
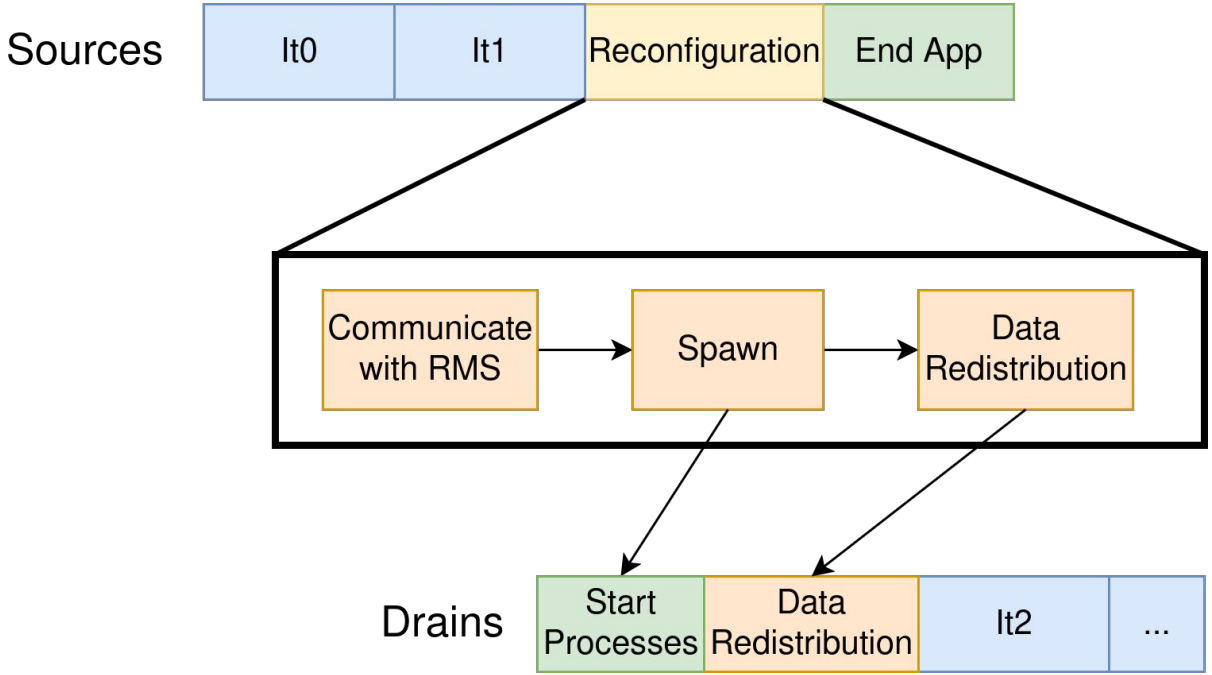
Dynamic reconfiguration for malleable applications using RMA

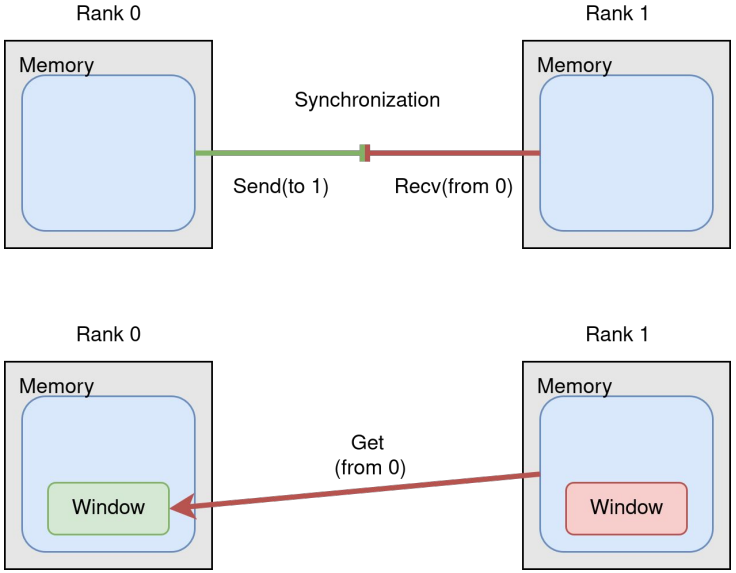
Authors

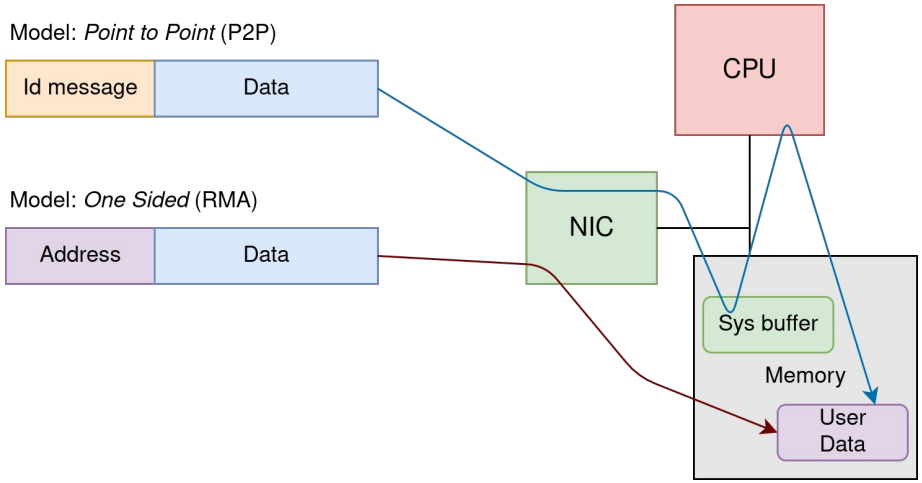
Iker Martín-Álvarez, José I. Aliaga, Maribel Castillo,


Execution time for 7 iterations
It0
It1
It2
It3
It4
It5
It6
Parallel version
Execution time of 1 iteration
It0
It1
PR
Data redistribution
It2
It3
It4
It5
It6
Dynamic synchronous
2 Processes -> 4 Processes
It0
It1
It2
It3
It4
It5
It6
Dynamic asynchronous
PR
Asynchronous redistribution
PR
Process Spawning


Sources
It0
It1
Reconfiguration
End App
Communicate with RMS
Spawn
Data Redistribution
Drains
Start Processes
Data Redistribution
It2
...

MPI *One-Sided* (RMA) benefits

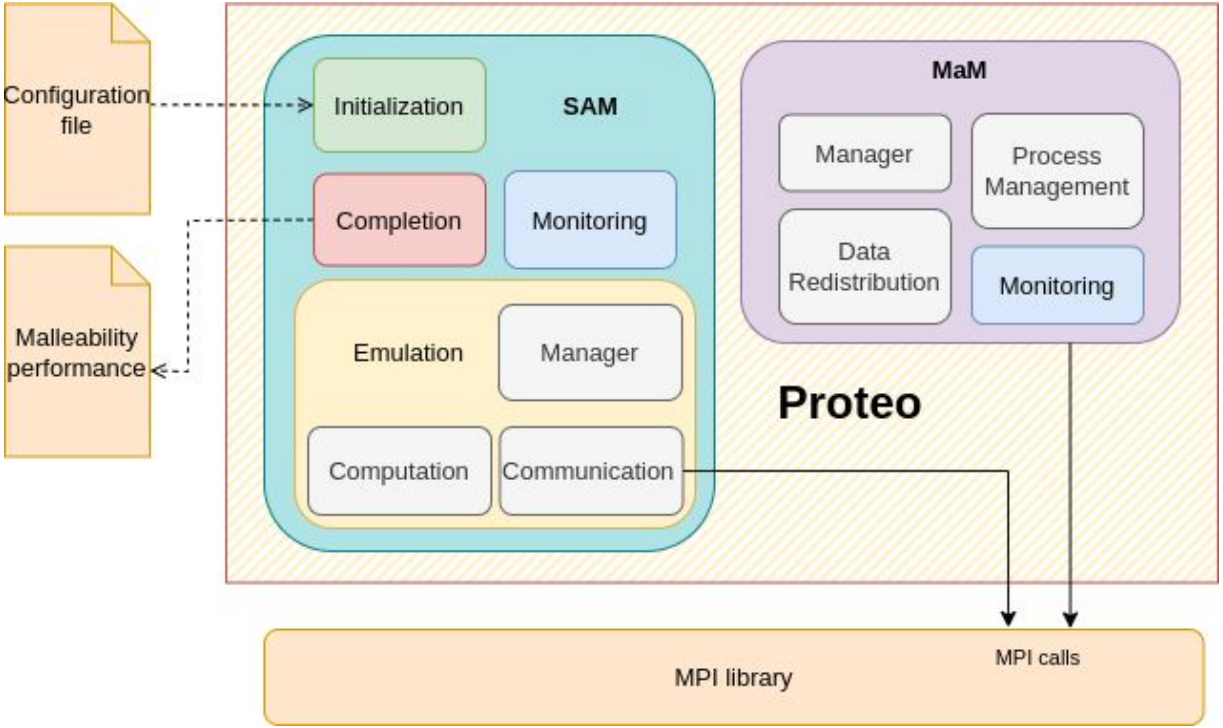




Index:

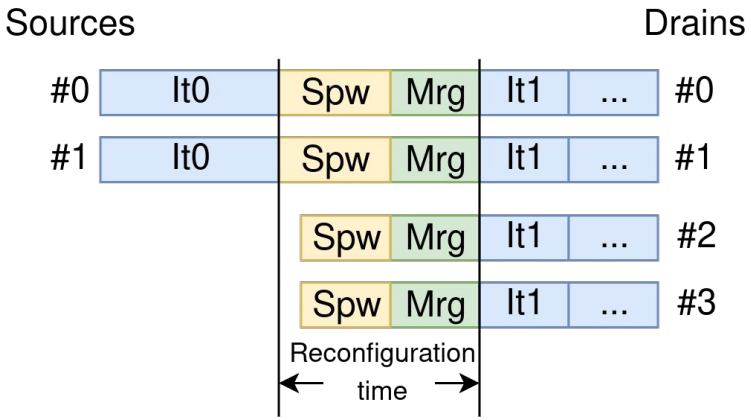
Summary of Proteo

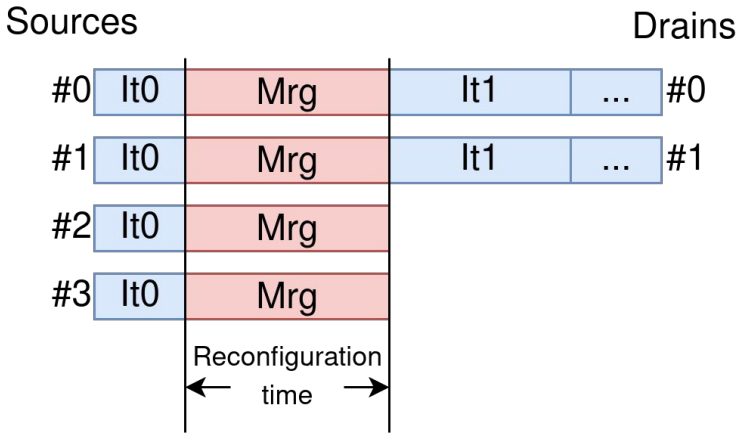
- Framework to emulate real applications.
- Composed by two modules:
  - SAM: Performs the emulation.
  - MaM: Takes cares of reconfigurations.
- Requires a configuration file to emulate the real application.



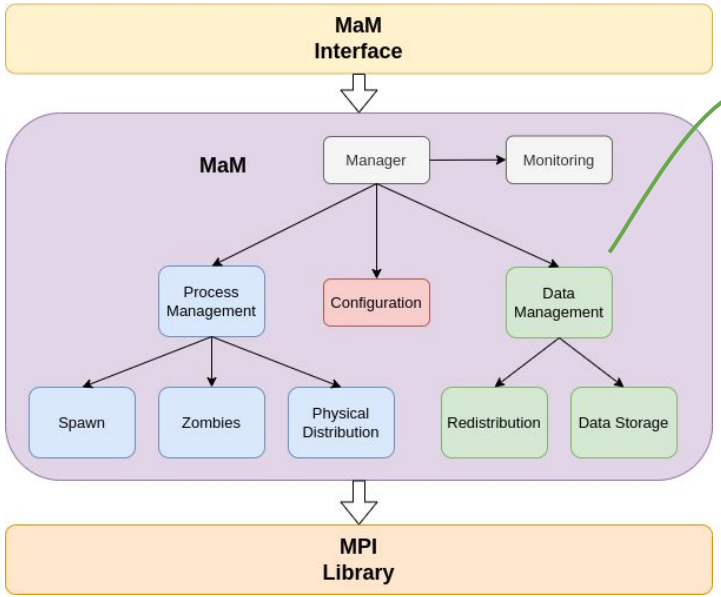
Proteo Repository

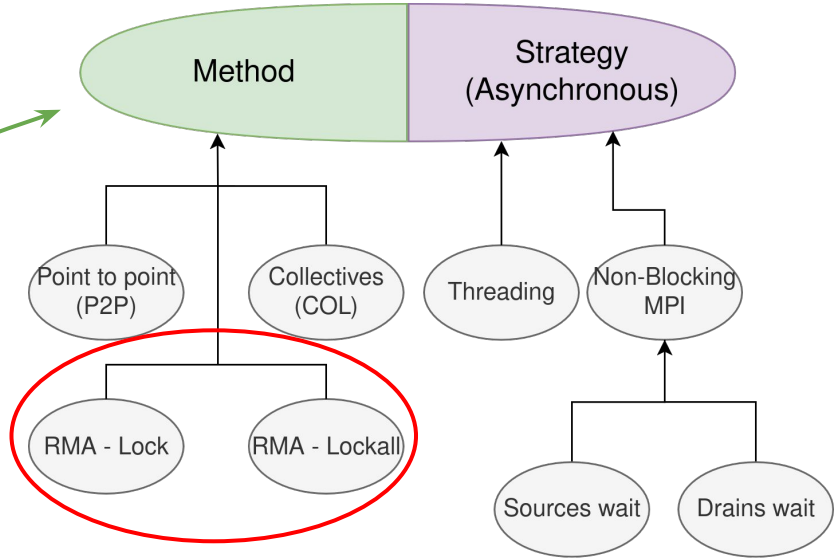
Spawn Methods - Merge





Data redistribution methods

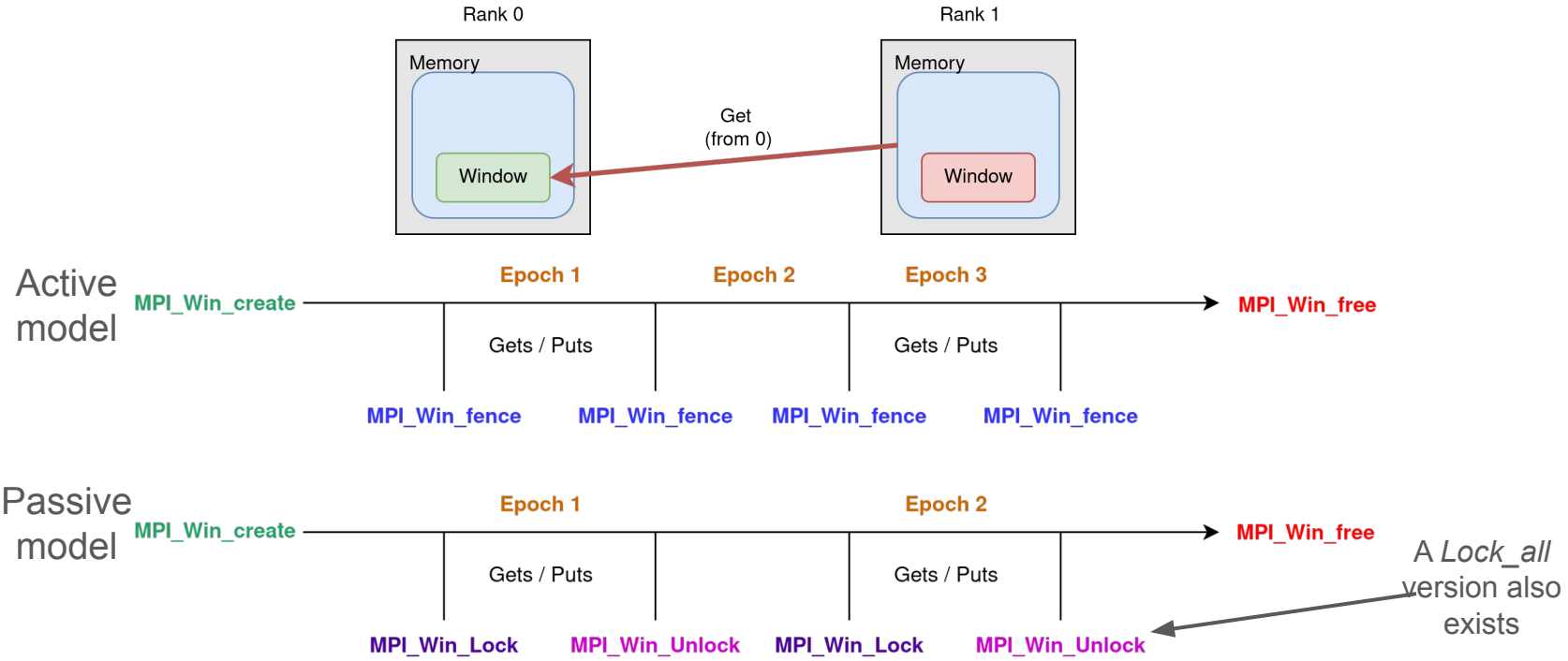




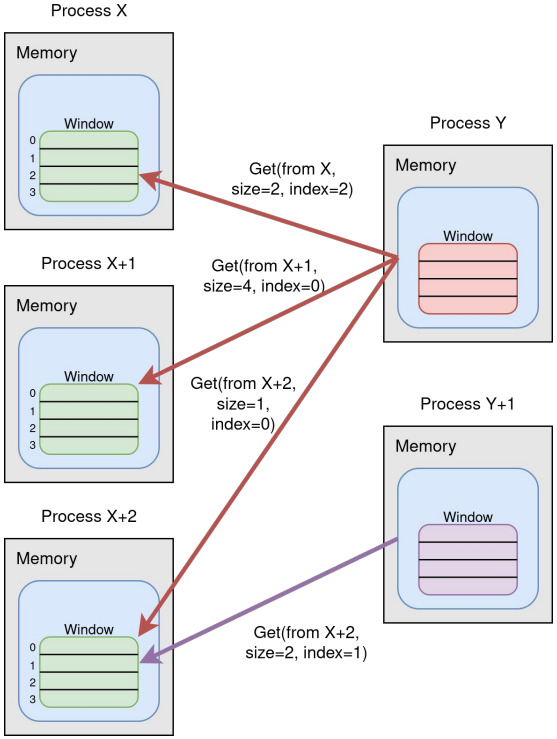
Index:

1. Objective
2. Proteo
3. RMA implementation
4. Asynchronous Implementation
5. Results
6. Conclusions

Basics


Rank 0
Rank 1
Memory
Memory
Window
Window
Get
(from 0)
Active
model
MPI_Win_create
Epoch 1
Epoch 2
Epoch 3
MPI_Win_free
Gets / Puts
Gets / Puts
MPI_Win_fence
MPI_Win_fence
MPI_Win_fence
MPI_Win_fence
Passive
model
MPI_Win_create
Epoch 1
Epoch 2
MPI_Win_free
Gets / Puts
Gets / Puts
MPI_Win_Lock
MPI_Win_Unlock
MPI_Win_Lock
MPI_Win_Unlock
A Lock_all
version also
exists

Proposed methods

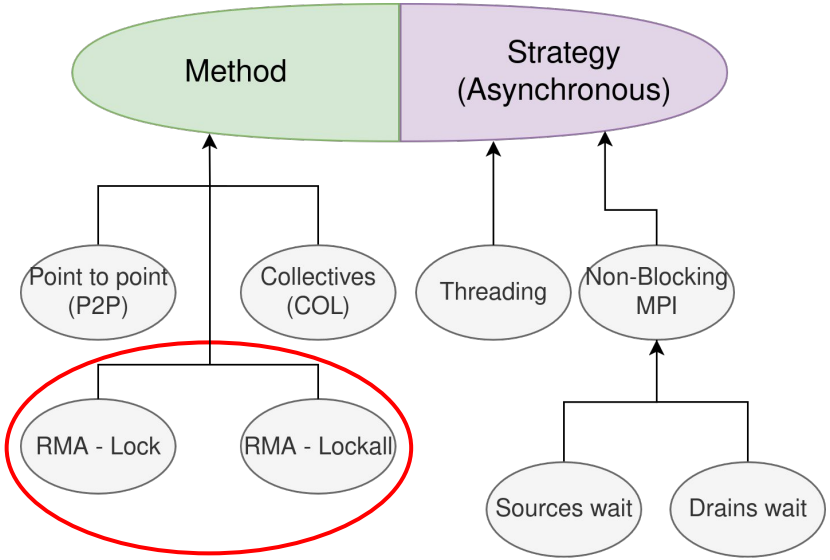


First method:

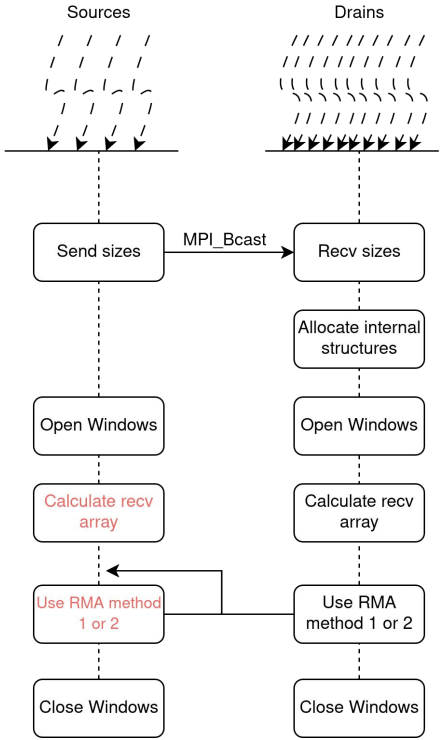
1. Loop Lock(x) + Get(x)
2. Loop Unlocks

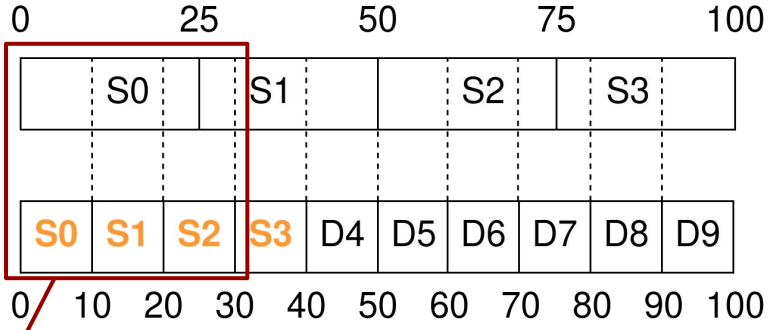
Second method:

1. Lock_all
2. Loop Gets
3. Unlock_all

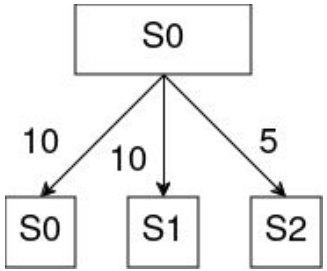


Merge - Steps





Distribution for 100 elements between 4 Sources and 10 Drains.



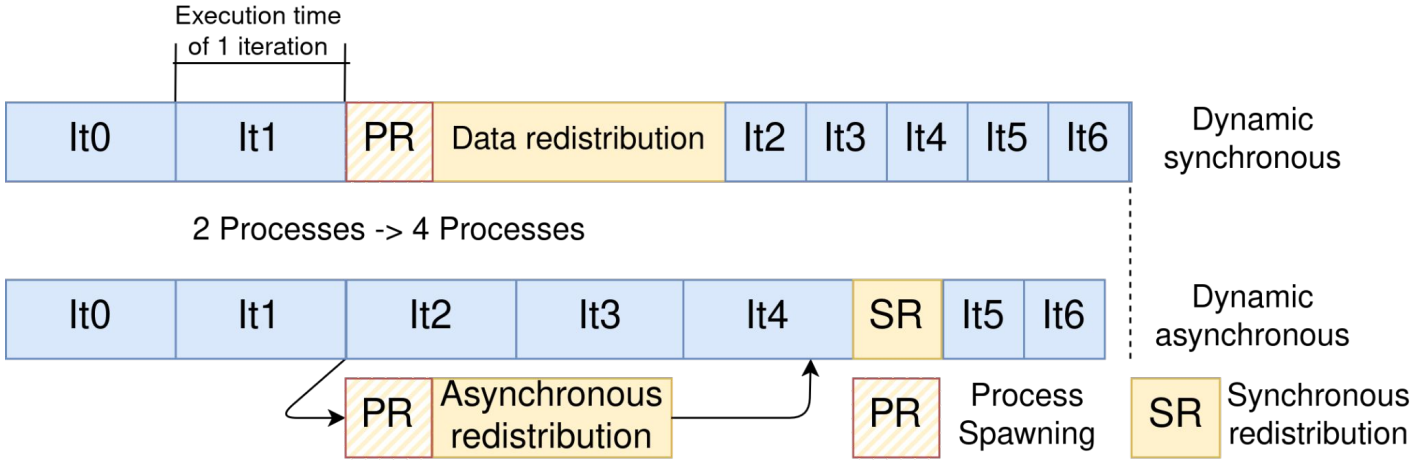
Elements distribution from Source 0 to Drains 0, 1 and 2.

Index:

Asynchronous benefits

Three strategies to redistribute data:

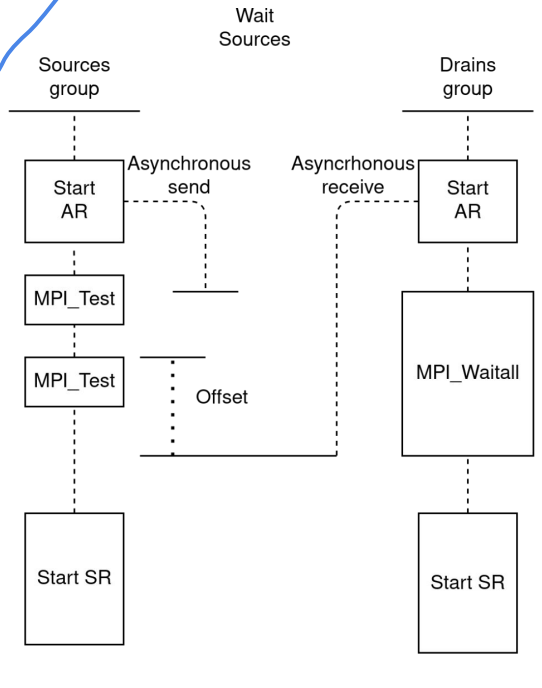
- Non-Blocking
  - Sources Wait
  - Drains Wait
- Threading

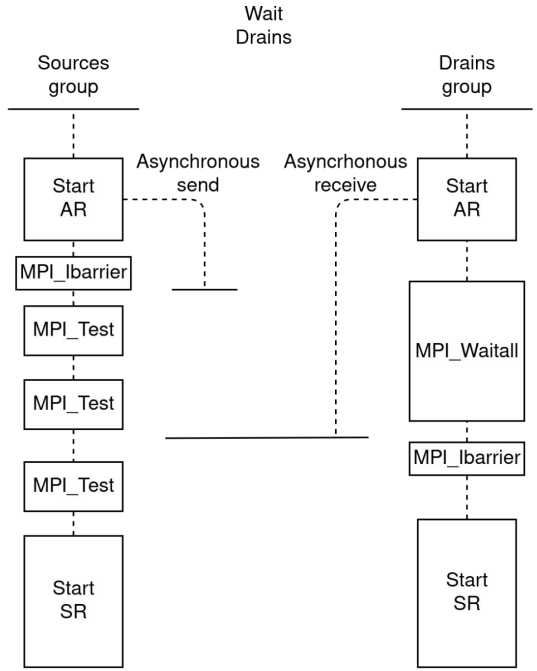


Asynchronous strategies

Three strategies to redistribute data:

- **Non-Blocking - Sources Wait**
- **Non-Blocking - Drains Wait**
- Threading

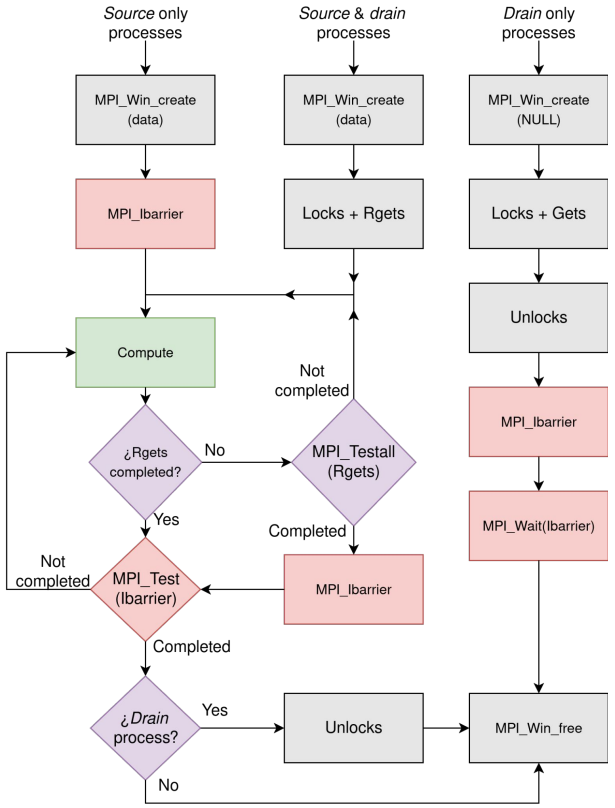




Operating flowchart

Steps for non-blocking model:

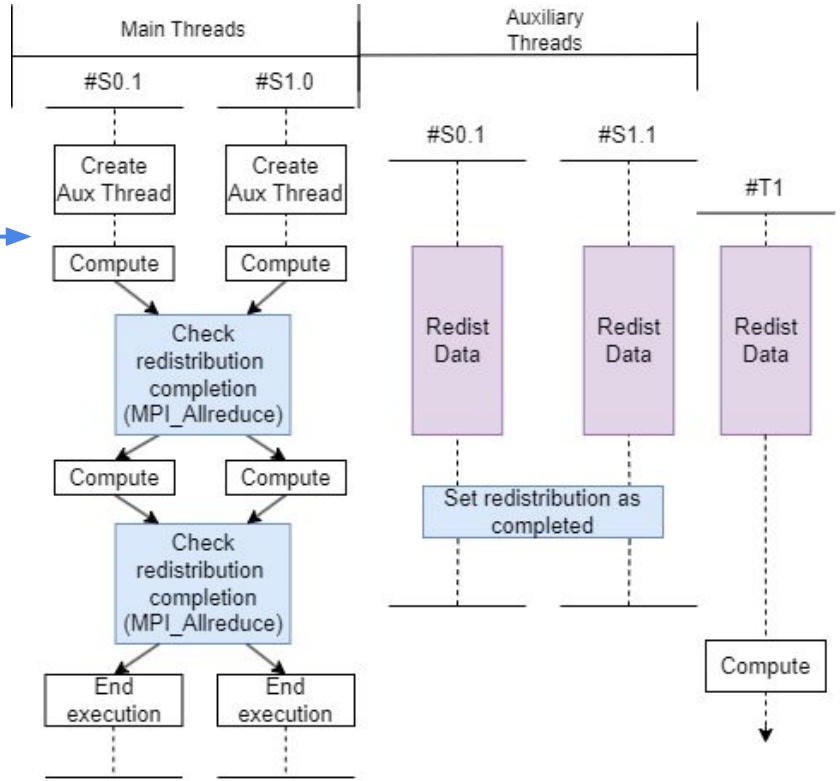
- Initialization
- Start readings
- Check local readings
- Check global readings
- Free windows



Asynchronous strategies

Three strategies to redistribute data:

- Non-Blocking - Sources Wait
- Non-Blocking - Drains Wait
- **Threading**



Index:

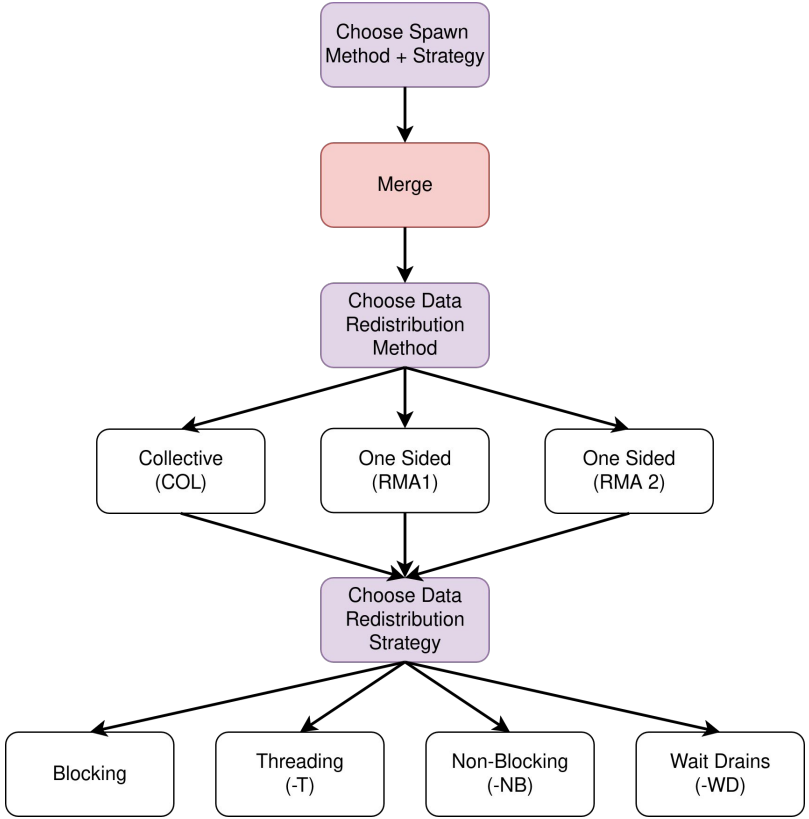
Testbed

Machine description:

- Compute nodes (x8): 2x Intel Xeon 4210 (10-core)
- Total of 160 *cores*
- Network *Infiniband EDR 100GB/s*
- MPICH 4.2.0 CH4:OFI - fi_provider=verbs

Emulation description:

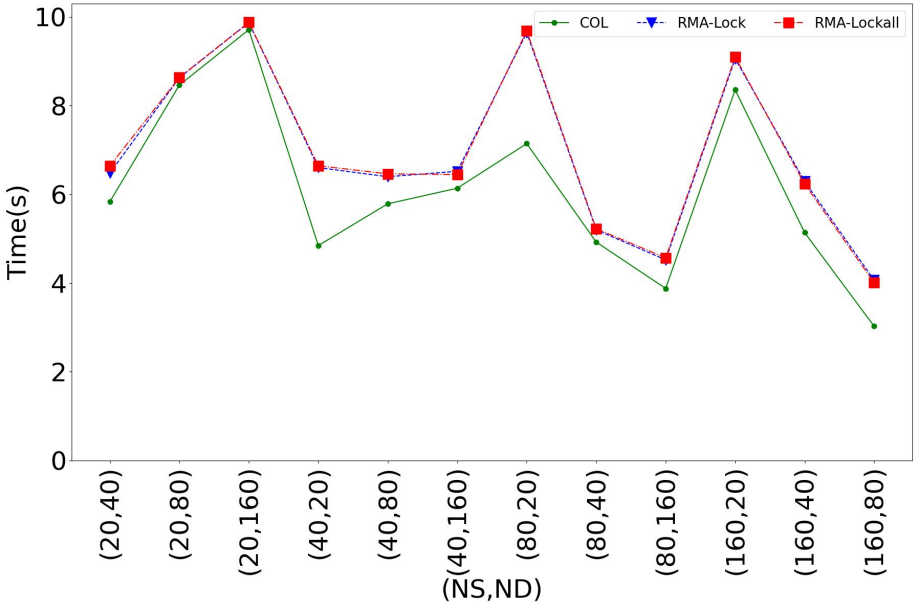
- Proteo emulates the Conjugate Gradient.
- One reconfiguration per execution.
- Data redistribution of 64GB.
- Median of 20 executions.

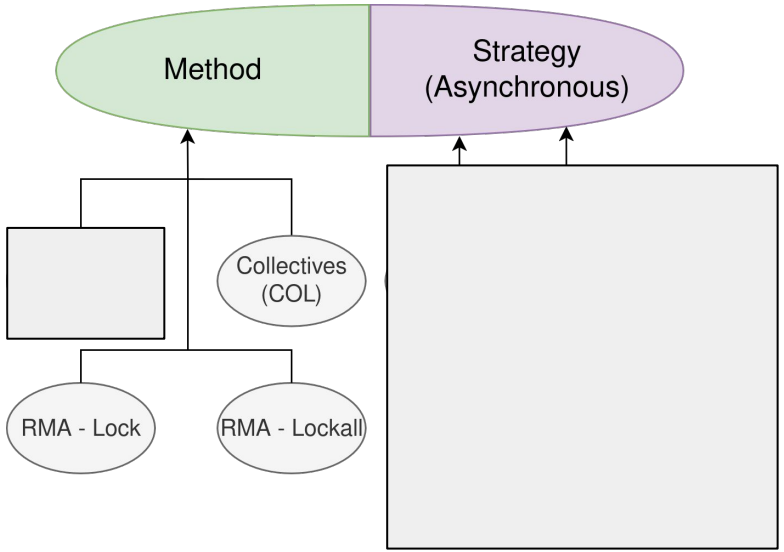


Generated Dataset

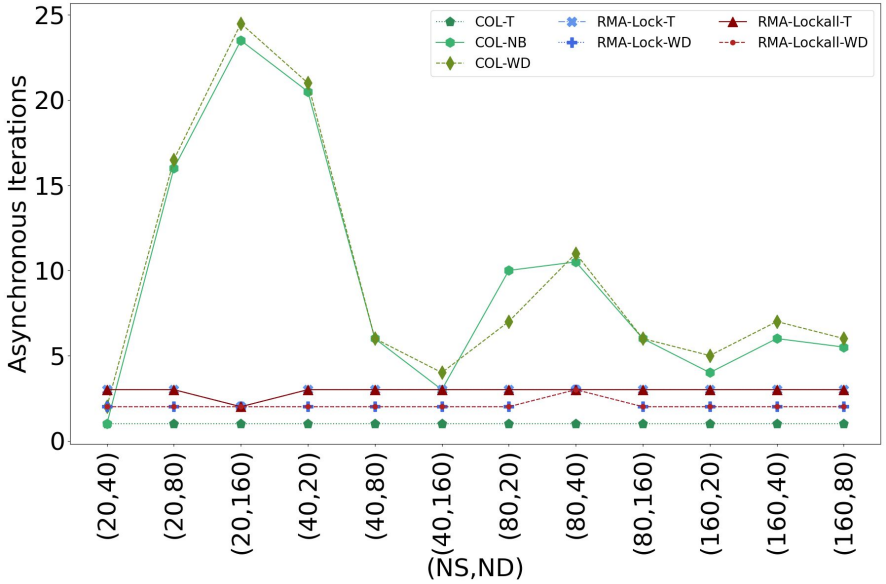
Proteo Repository

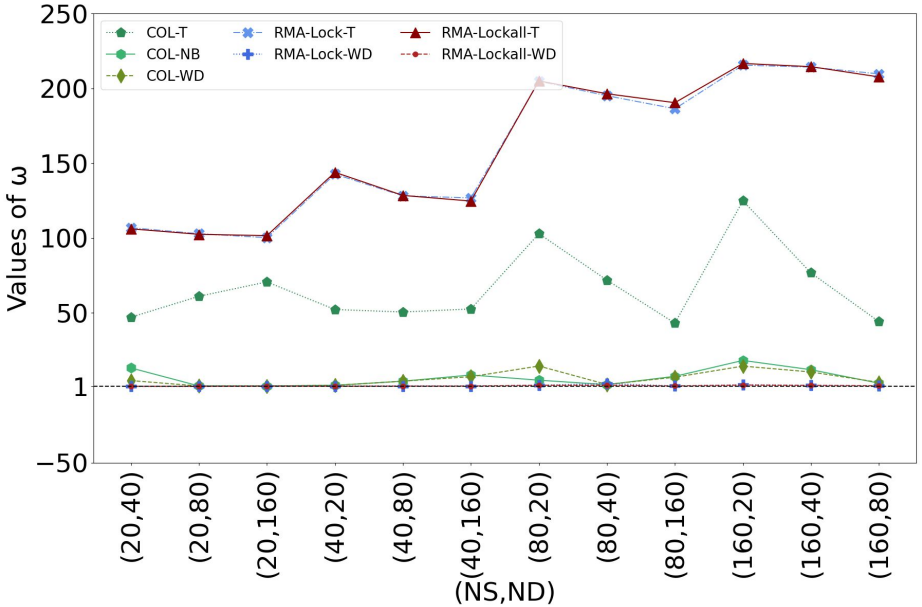
Synchronous times - Data redistribution





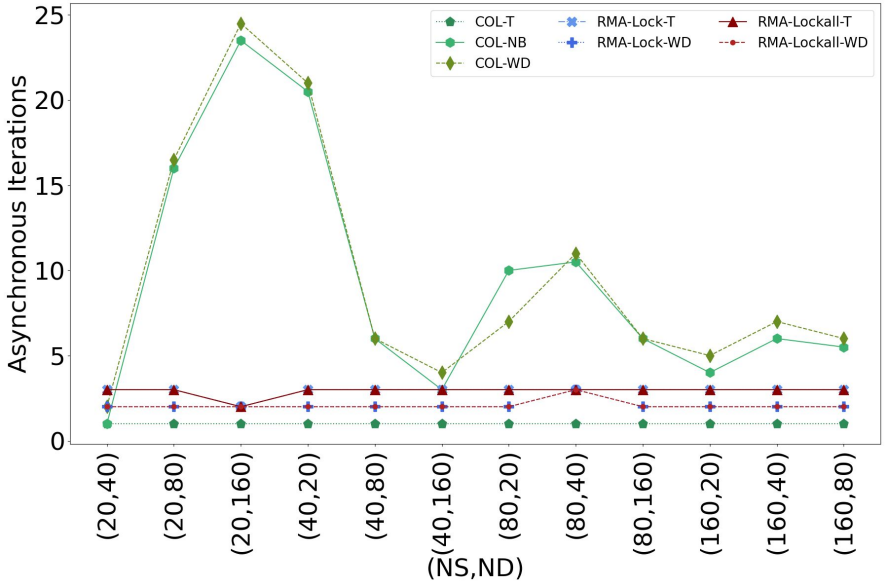
Asynchronous times - Internal parameters

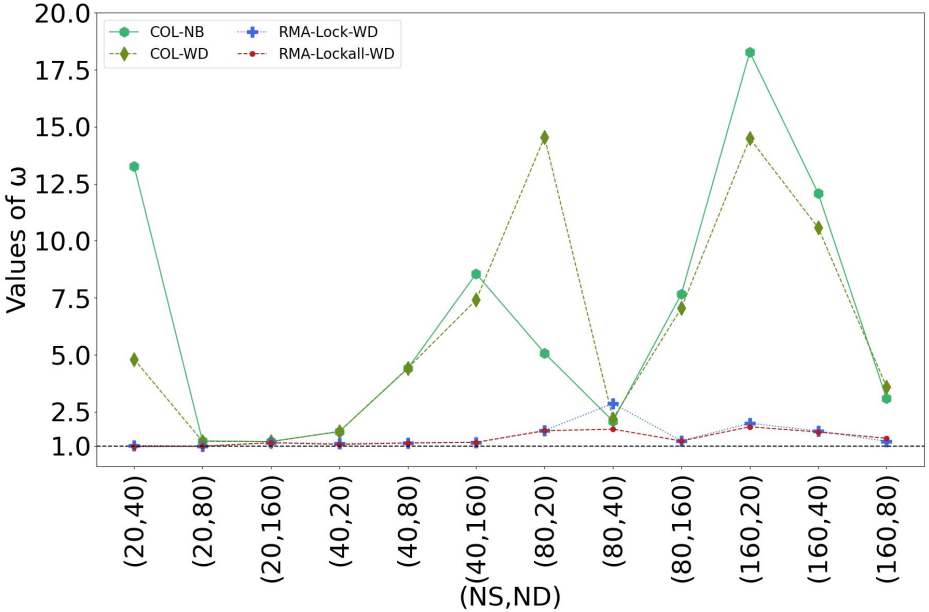




ω: Iteration overcost while performing a background reconfiguration

Asynchronous times - Internal parameters



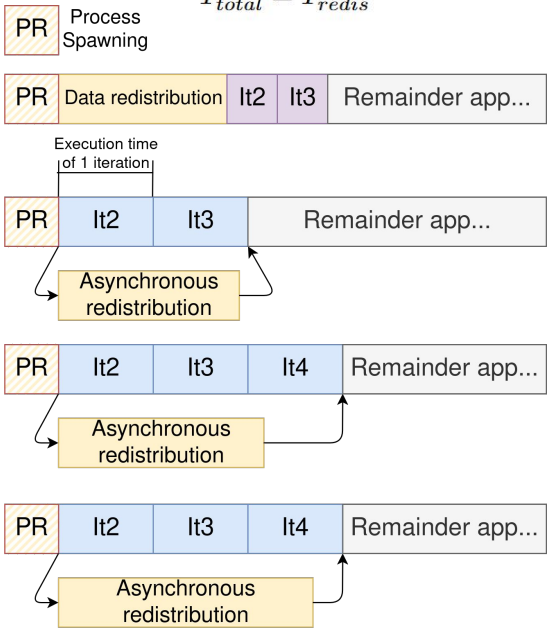


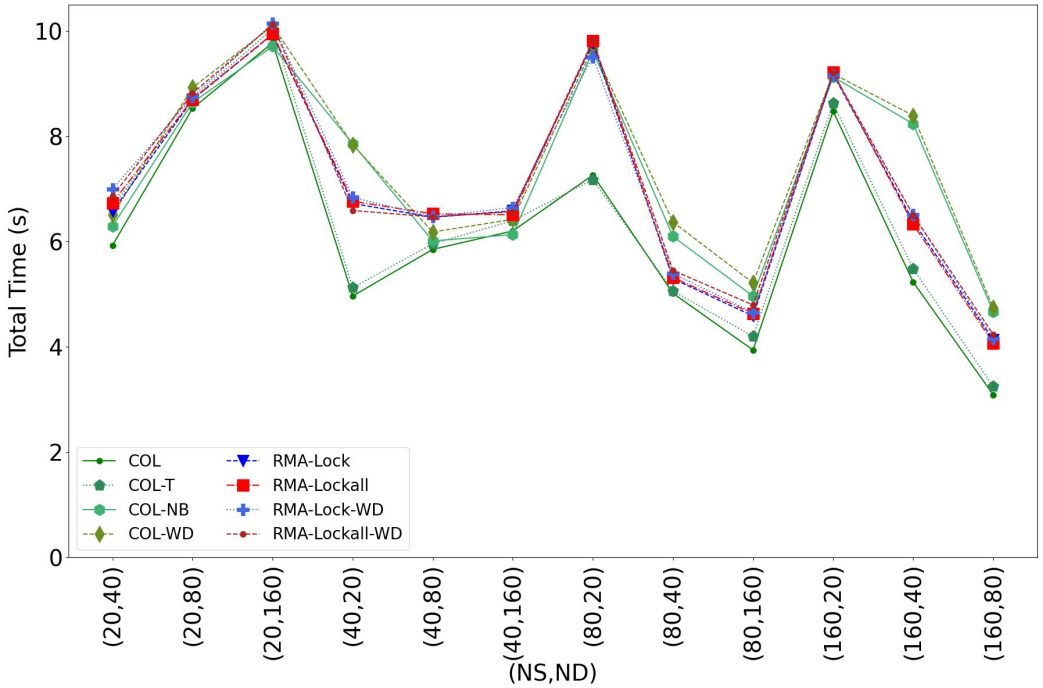
ω: Iteration overcost while performing a background reconfiguration

Asynchronous times - Data redistribution

$$T^{Bl}_{total} = T^{Bl}_{redis} + T^{NT}_{it} * \min_{variantes}(N^{NS \to NT}_{it})$$

$$T^{SP}_{total} = T^{SP}_{redis}$$





Index:

MaM has been extended with One-Sided communications for data redistribution

The asynchronous One-Sided model for data redistribution does not impact the performance of the application

Yet, the proposed model does not outstand the collective model because the initialization of the One-Sided windows

Dynamic reconfiguration for malleable applications using RMA

Authors

Iker Martín-Álvarez (martini@uji.es),

José I. Aliaga, Maribel Castillo,