Proteo as a Framework for Dynamic Workload Emulation

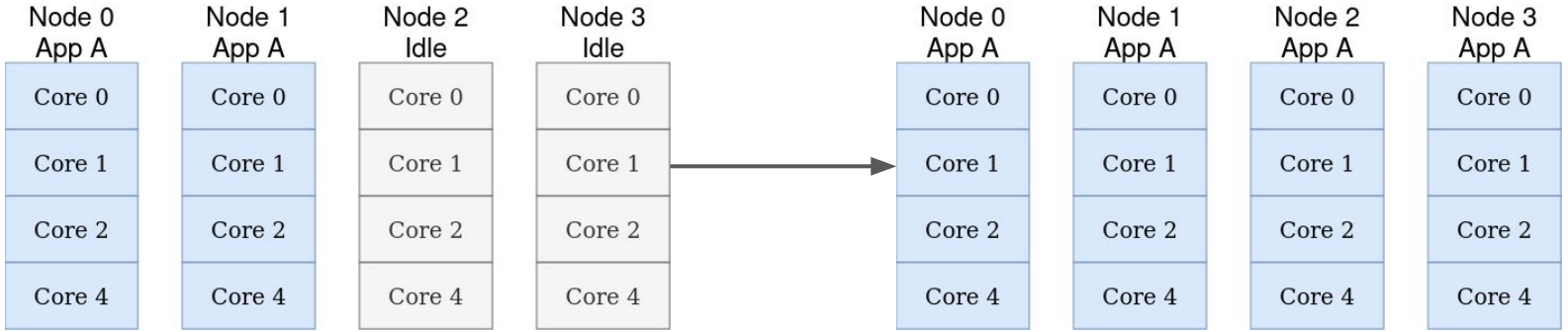
Authors

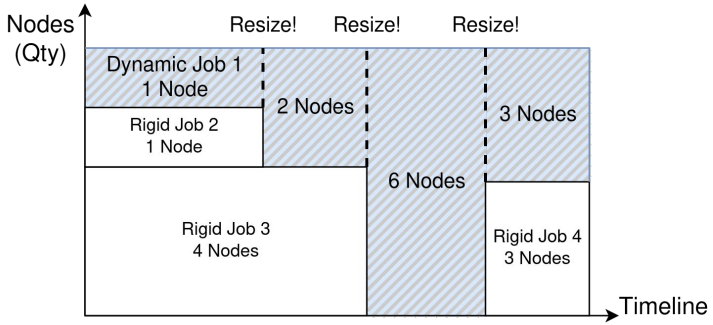
Iker Martín-Álvarez

Maribel Castillo

José I. Aliaga

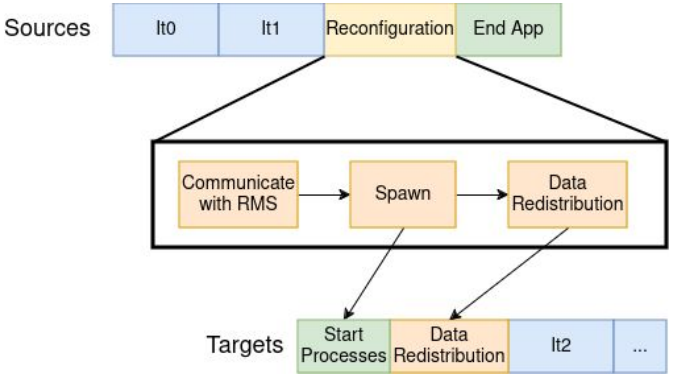
What is Dynamic Resource Management (DRM)?



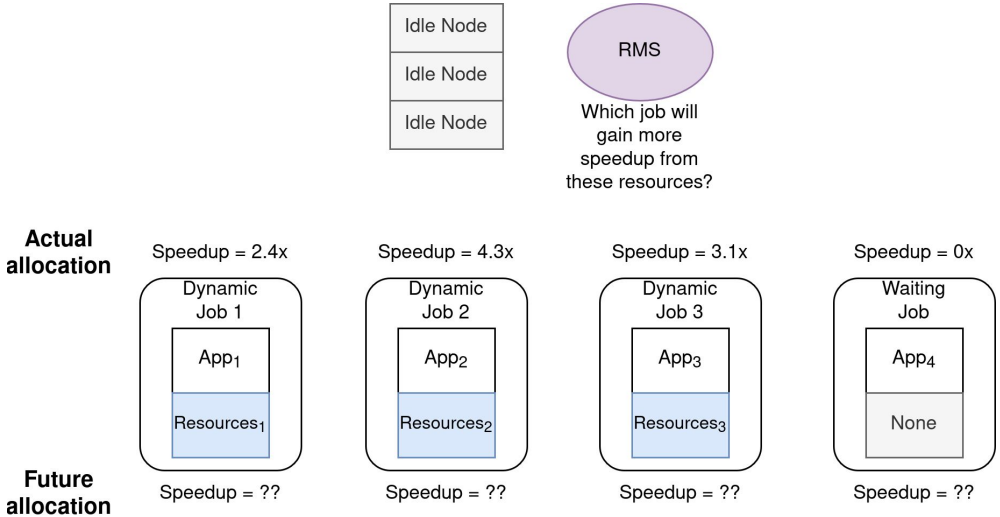


Challenges

- Scheduler must be capable of adapting allocation of executing jobs
- Applications require modifications to introduce reconfigurations
- Testbeds are required to showcase the benefits of DRM.



Example for reducing job running time



Outline:

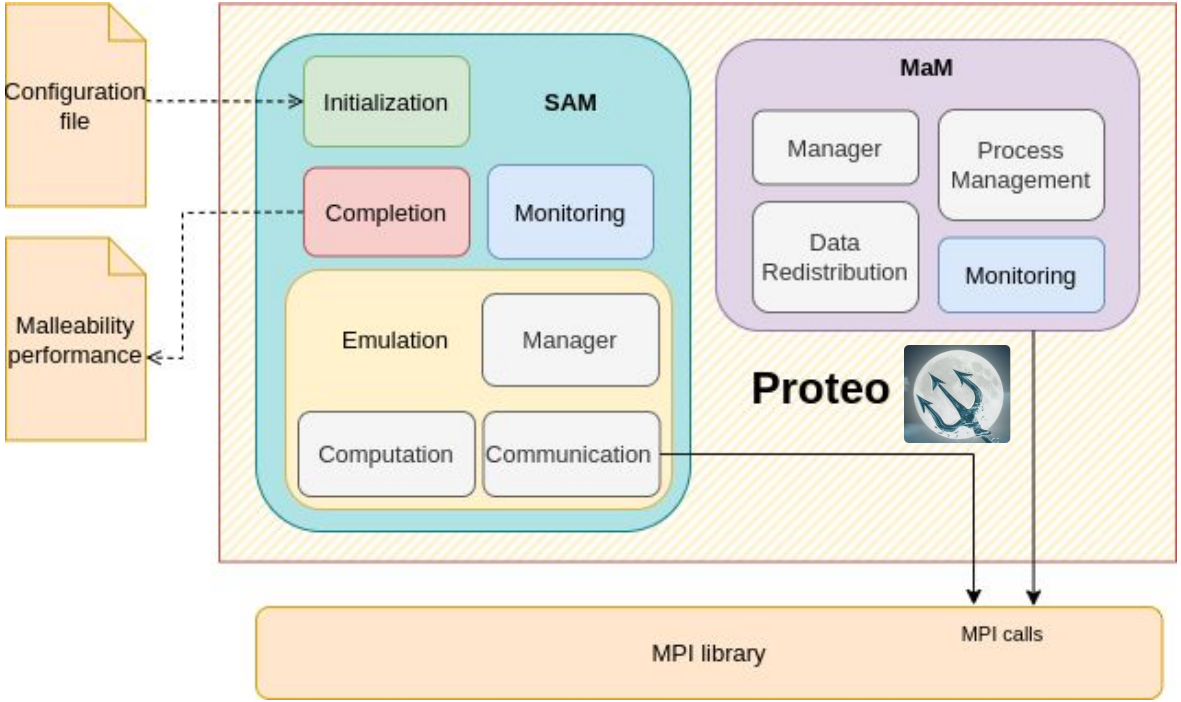
1. Dynamic Resources
2. Proteo
3. Results
4. Closing remarks

Outline:

1. Dynamic Resources
2. Proteo
   - 2.1 Overview
   - 2.2 Goals
   - 2.3 Emulation (SAM)
3. Results
4. Closing remarks

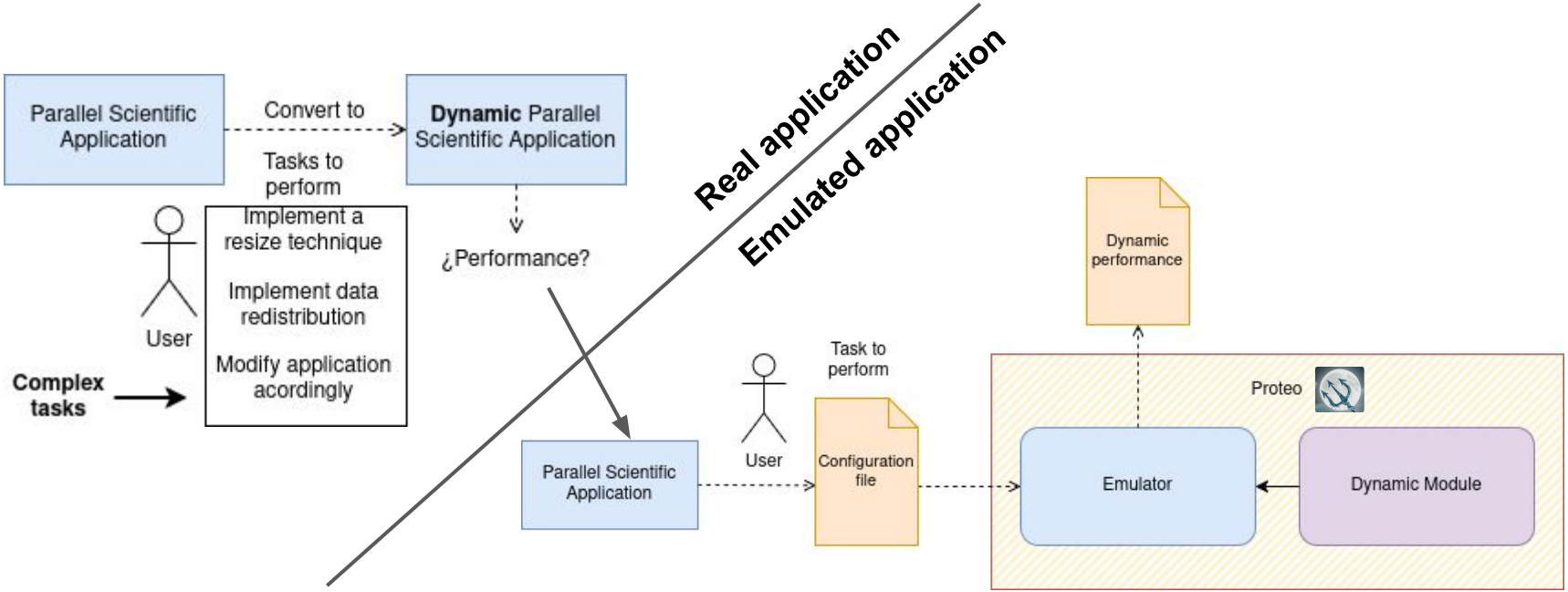
Proteo Overview

- Framework to emulate real applications as dynamic.
- Composed of two modules:
  - SAM: Takes care of emulation.
  - MaM: Takes care of reconfiguring the application.
- Requires a configuration file which describes how to emulate the real application.

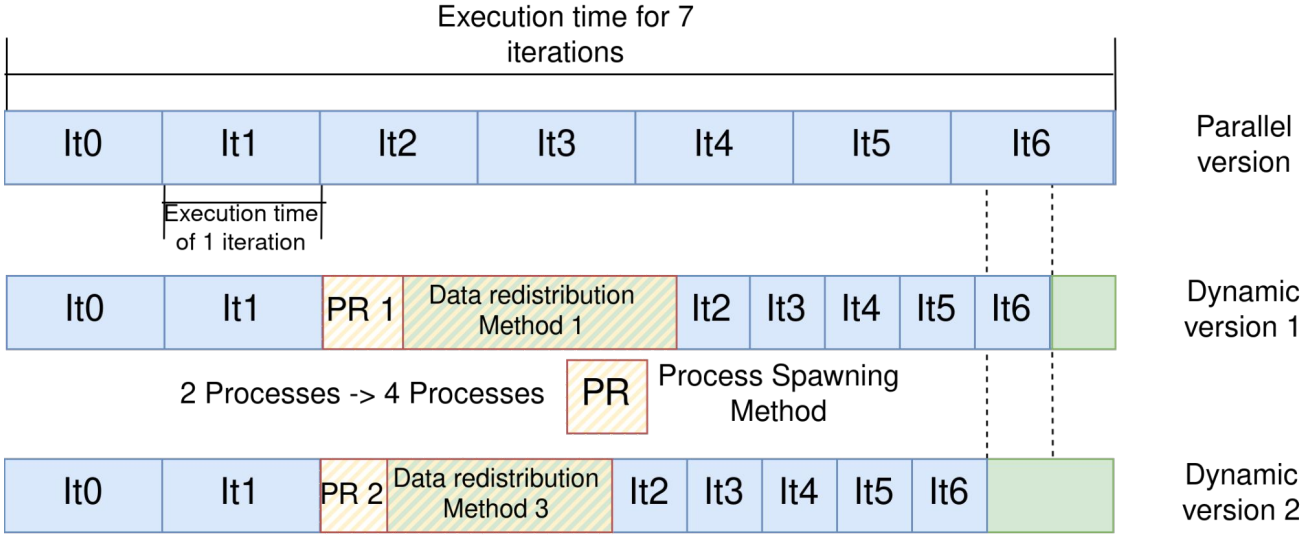


Repository Proteo

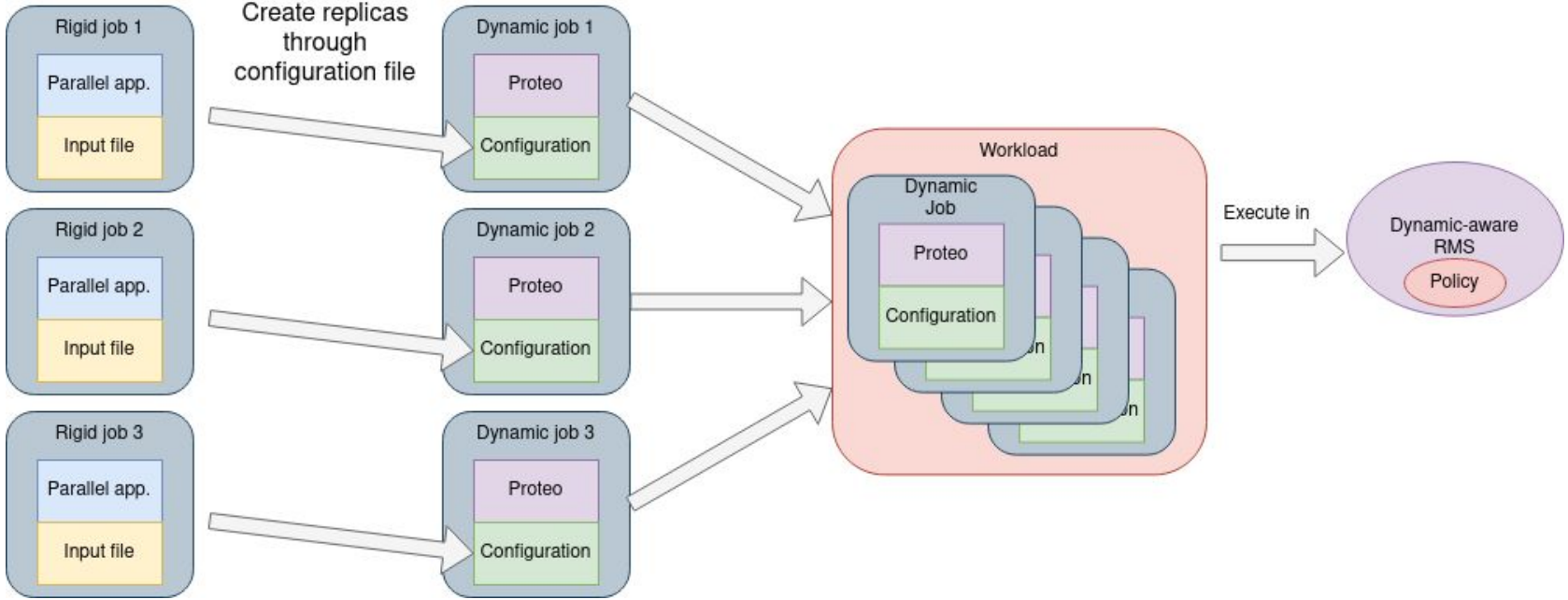
Goal 1: Check dynamic resources benefits



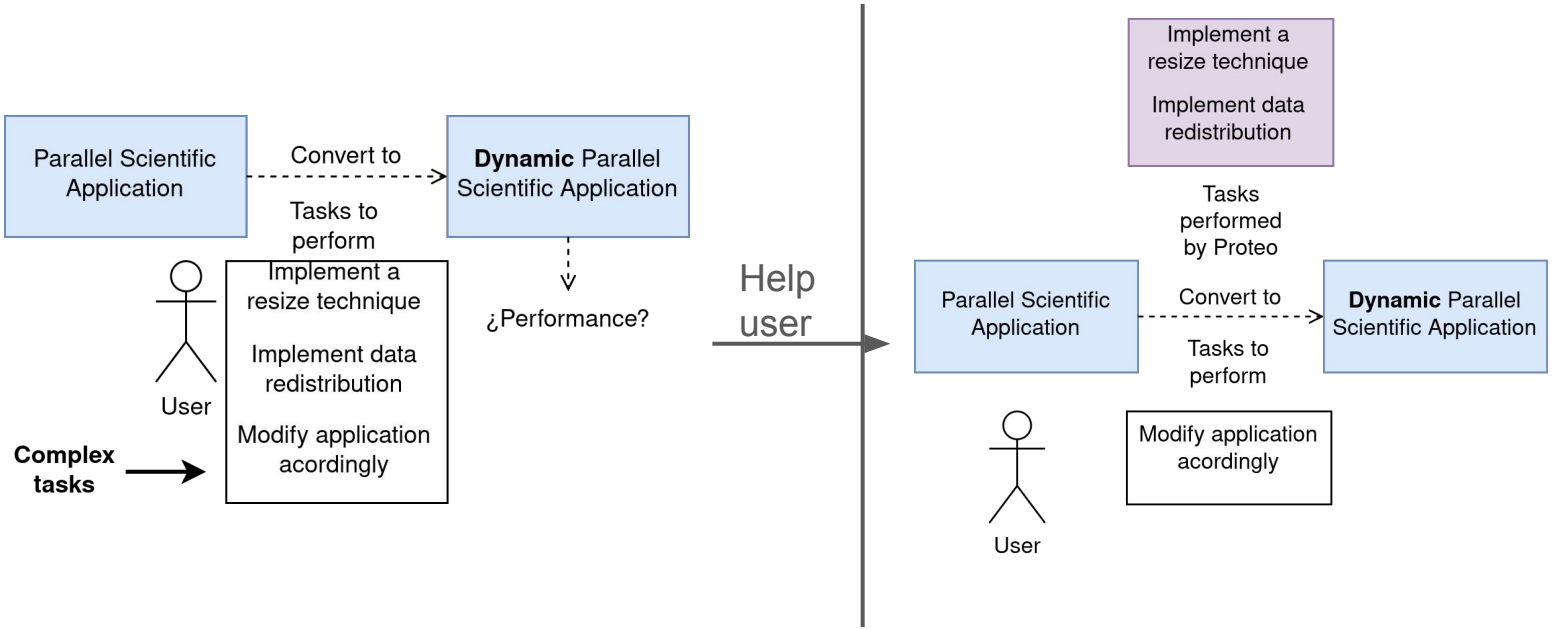
Goal 2: Reduce reconfiguration costs



Goal 3: Create dynamic workloads



Goal 4: Help convert real applications into dynamic



Outline:

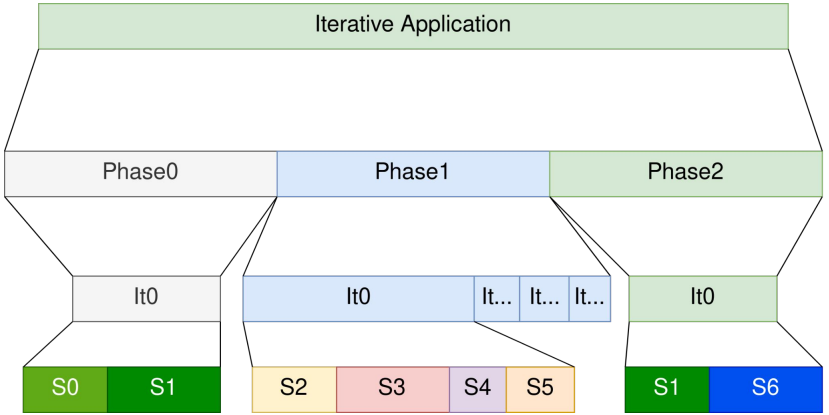
1. Dynamic Resources
2. Proteo
   - 2.1 Overview
   - 2.2 Goals
   - 2.4 Emulation (SAM)
3. Results
4. Closing remarks

Proteo - Synthetic Application Module (SAM)

Application as a set of phases

Emulate each phase with a set of iterations

Emulate each iteration as a combination of stages

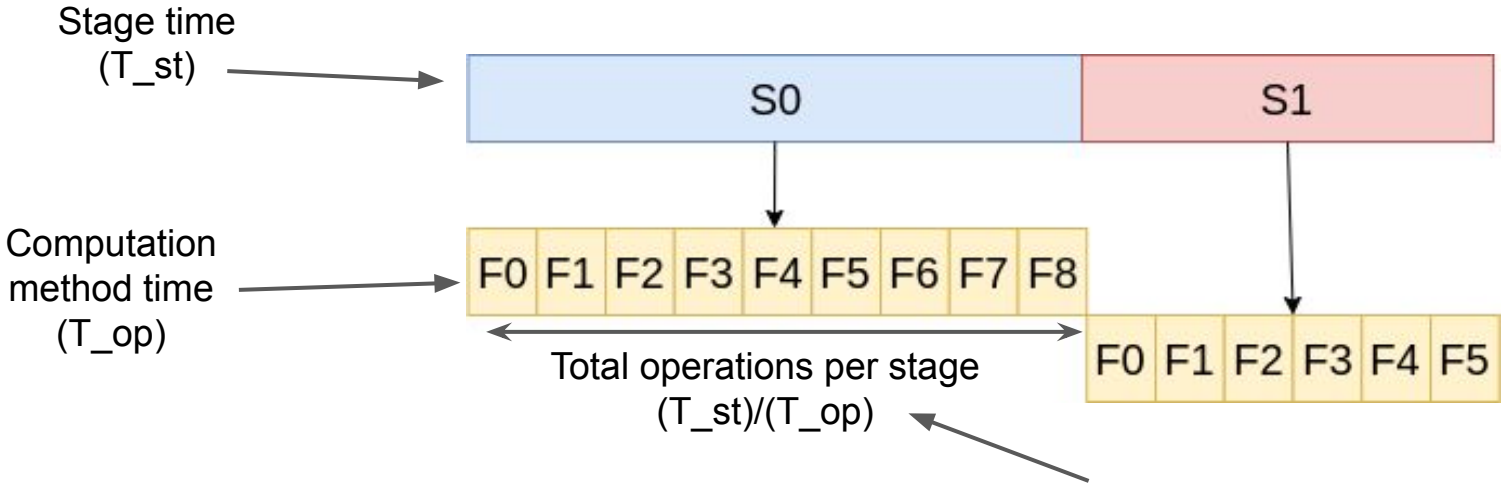


Each phase defines different stages

The cost of the stage Sx is constant between iterations of the same phase

Each stage can have a different cost

SAM - Computation and I/O methods



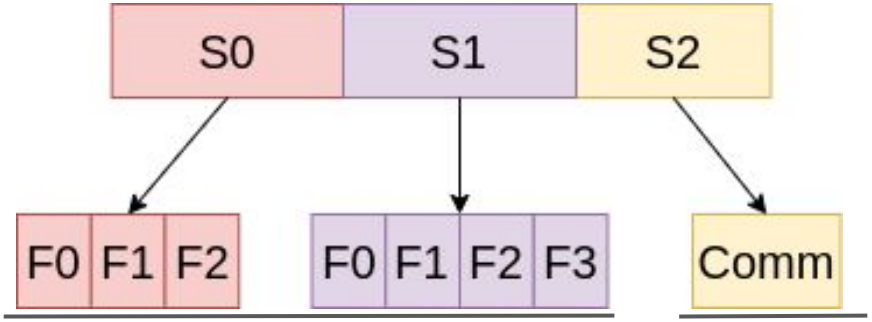
Computation methods:

- Compute-bound: Montecarlo method to calculate PI
- Memory-bound: Matrix-vector multiplication
- Write: A single process or all processes writes for T_st time or a given number of bytes.
- Read: A single process or all processes reads for T_st time or a given number of bytes.

SAM - Communication methods

Communication methods:

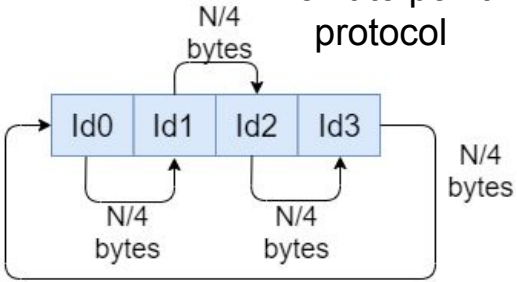
- Point to point (MPI_Send/Recv)
- Non-Blocking Point to Point (MPI_Isend/Irecv)
- Collective one to all (MPI_Bcast)
- Collective all to all (MPI_Allgatherv)
- Reduction (MPI_Reduce)
- Reduction all to all (MPI_Allreduce)



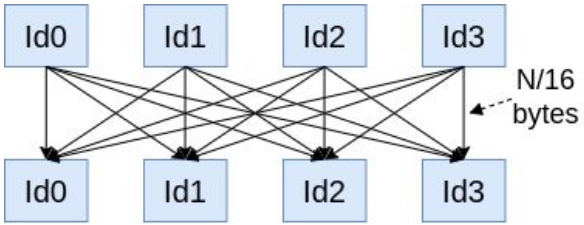
Total operations per stage

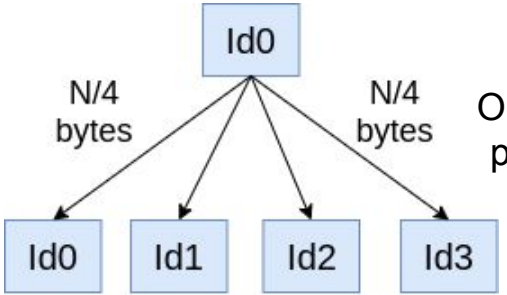
Single operation of N bytes

Point to point protocol



All to all protocol





One to all protocol

SAM - Configuration

Configuration file in INI format. Declares emulation and reconfigurations characteristics.

For each phase, declare stages with expected time or bytes to use.

Each reconfiguration declares number of processes and how many iterations to perform before a new reconfiguration.

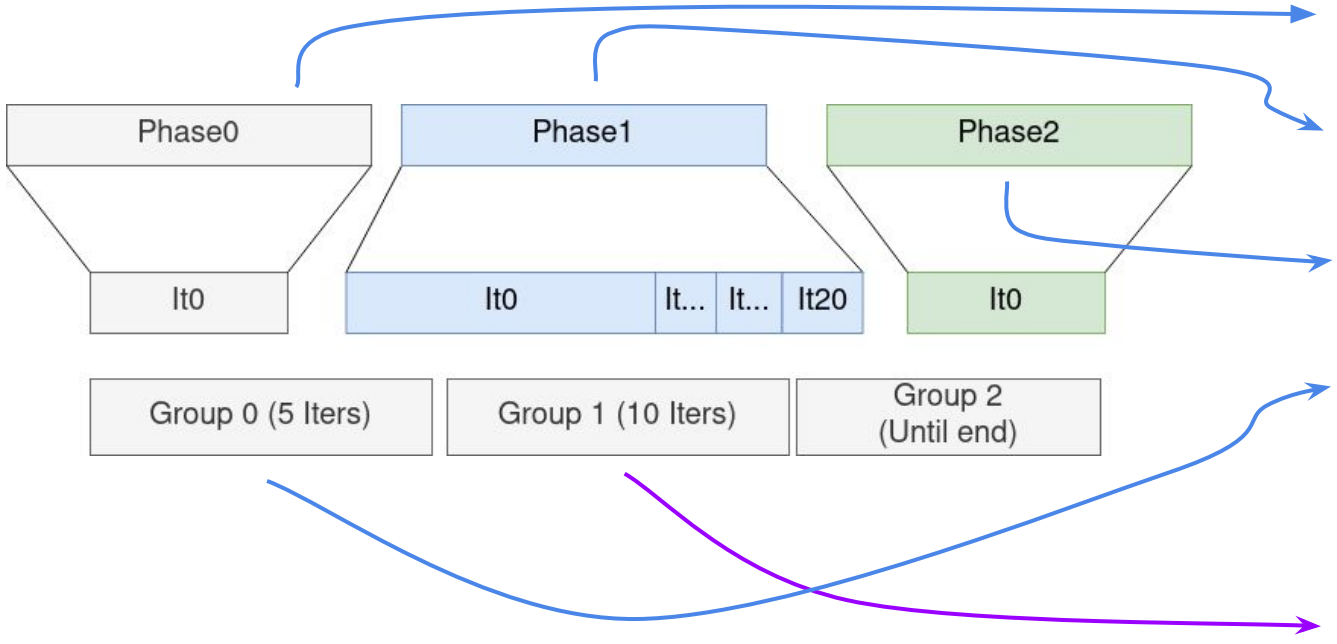
General and Phases

```
[general]
Total_Resizes=2
Total_Phases=3
SDR=500000.0
ADR=500000.0
Rigid=0
Capture_Method=0
;end [general]
[phase0]
Total_Iters=1
Total_Stages=3
;end [phase0]
[phase0.stage0]
Stage_Type=10
Stage_Bytes=0
Stage_Time=4.0
;end [phase0.stage0]
[phase0.stage1]
Stage_Type=5
Stage_Bytes=1000
Stage_Time=0
;end [phase0.stage1]
```

Resizes

```
[resize0]
Iters=5
Procs=2
FactorS=1
Dist=compact
Redistribution_Method=0
Redistribution_Strategy=0
Spawn_Method=0
Spawn_Strategy=0
;end [resize0]
[resize1]
Iters=10
Procs=4
FactorS=1
Dist=compact
Redistribution_Method=0
Redistribution_Strategy=0
Spawn_Method=1
Spawn_Strategy=0
;end [resize1]
[resize2]
Iters=5
```

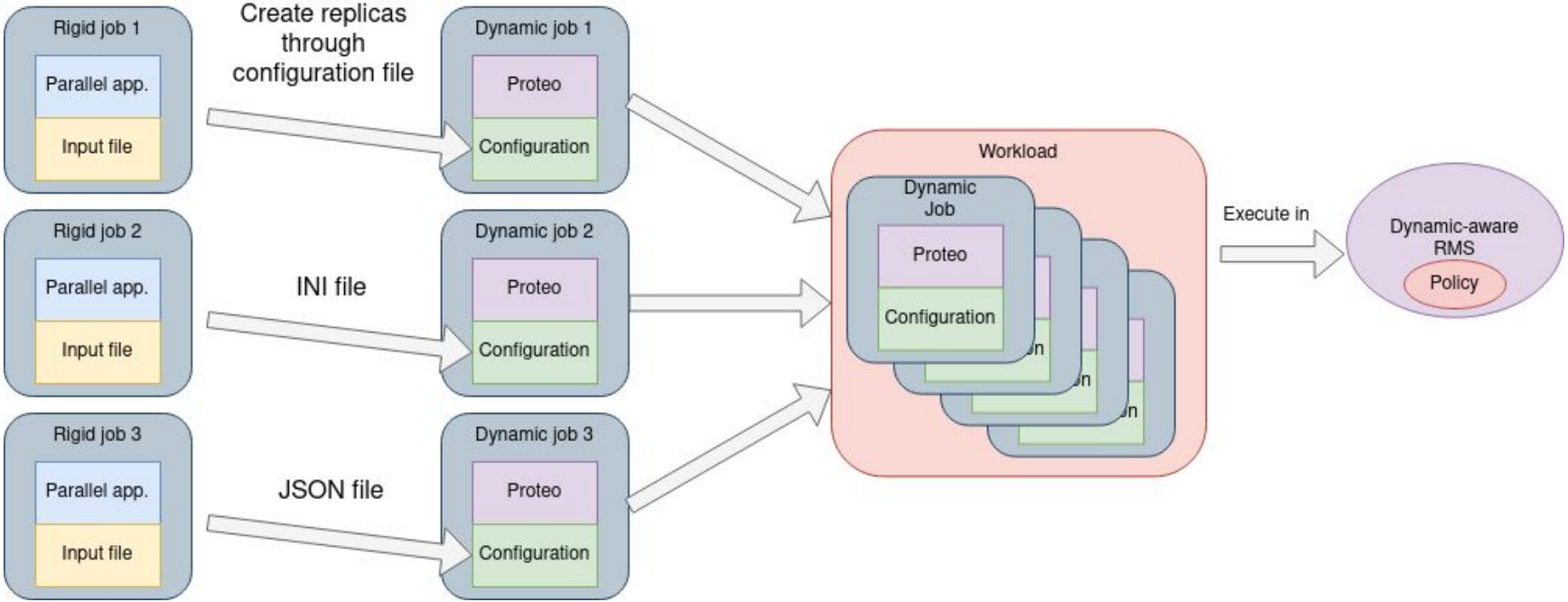
SAM - Configuration



Resizes

```
[phase0]
Total_Iters=1
Total_Stages=3
;end [phase0]
; stages info
[phase1]
Total_Iters=22
Total_Stages=4
;end [phase1]
; stages info
[phase2]
Total_Iters=1
Total_Stages=1
;end [phase2]
; stages info
[resize0]
Iters=5
Procs=2
FactorS=1
Dist=compact
Redistribution_Method=0
Redistribution_Strategy=0
Spawn_Method=0
Spawn_Strategy=0
;end [resize0]
[resize1]
Iters=10
Procs=4
```

Proteo - Execution



Outline:

1. Dynamic Resources
2. Proteo
3. Results
   - 3.1 Testbed
   - 3.2 Emulated CG
   - 3.3 Emulated vs Real CG
4. Closing remarks

Description

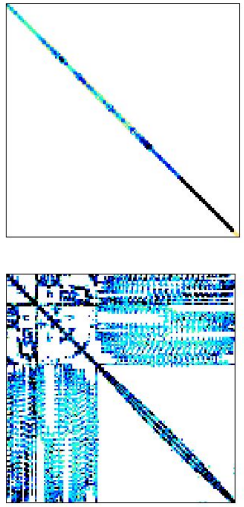
Machines:

- System_1: Two 10-core processor per Node, EDR infiniband (100 GB/s).
- System_2: Two 10-core processor per Node, EDR infiniband (100 GB/s).
- System_3: Two 16-core processor per Node, EDR infiniband (100 GB/s).

Goal: Test if Proteo (SAM) can be configured to behave as a Conjugate Gradient (CG) application.

Testbed:

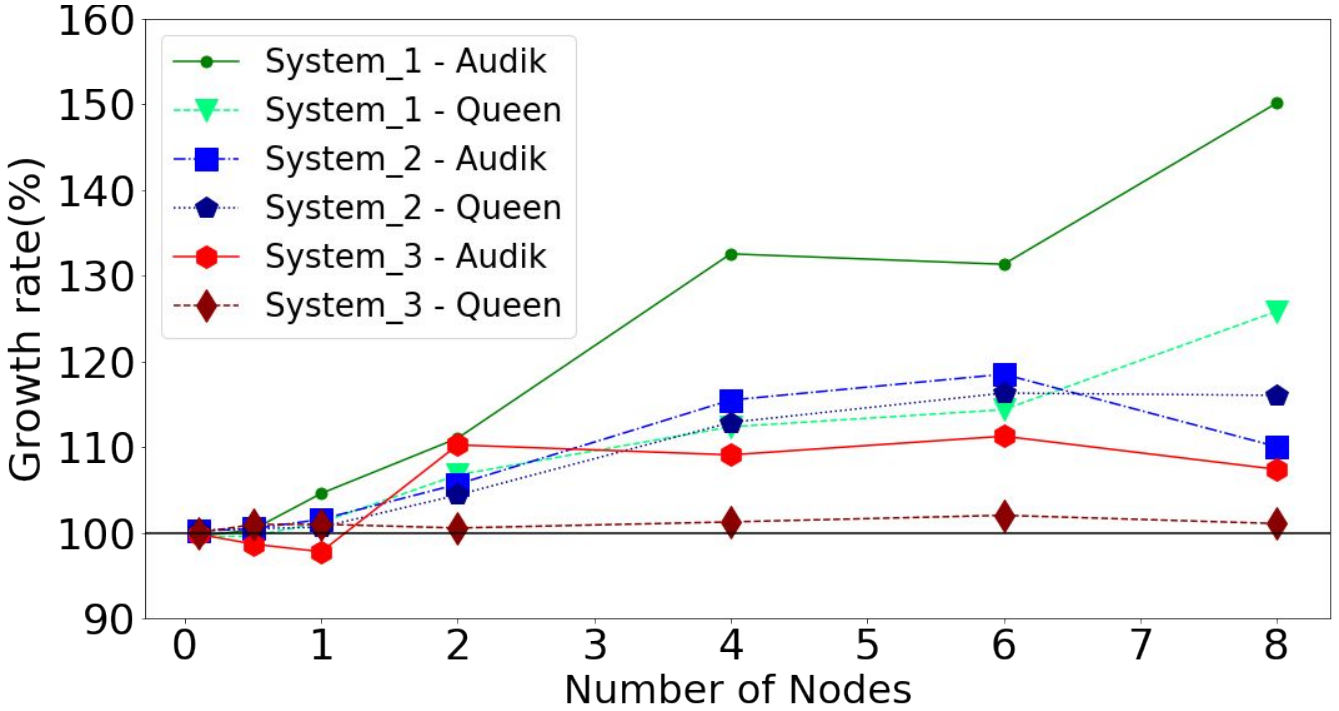
- CG executions with:
  - Balanced workload (Queen).
  - Unbalanced workload (Audik).
- System 1 & 2: 2, 10, 20, 40, 80, 120, 160 processes
- System 3: 2, 16, 32, 64, 128, 192, 256 processes.
- Average of 10 executions.
- Study divided in:
  - Computation.
  - Communication.
  - Whole execution.



CG Operations

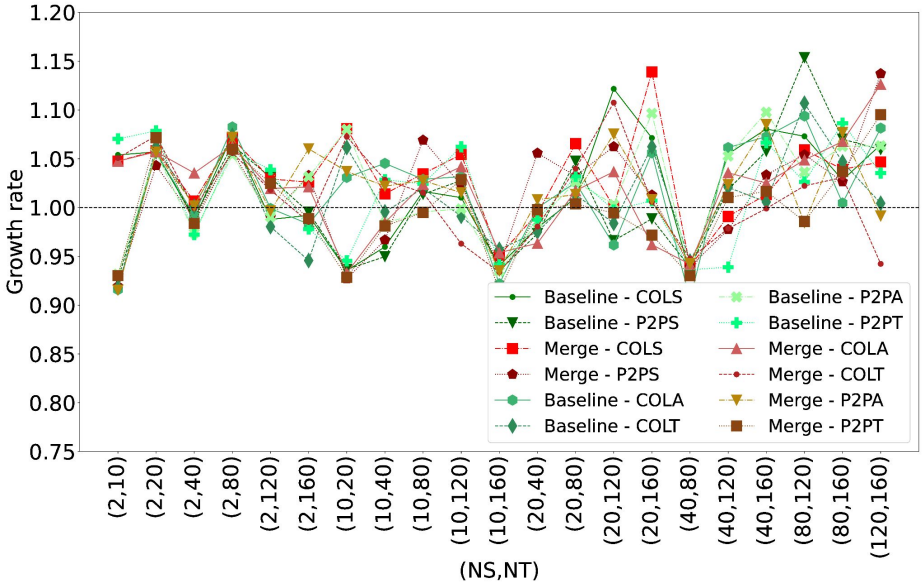
Computation	Communication
SPMV	MPI_Allgatherv(n)
DOT	MPI_Allreduce(1)
AXPY	
AXPY	
DOT	MPI_Allreduce(1)
XPAY	

CG execution study

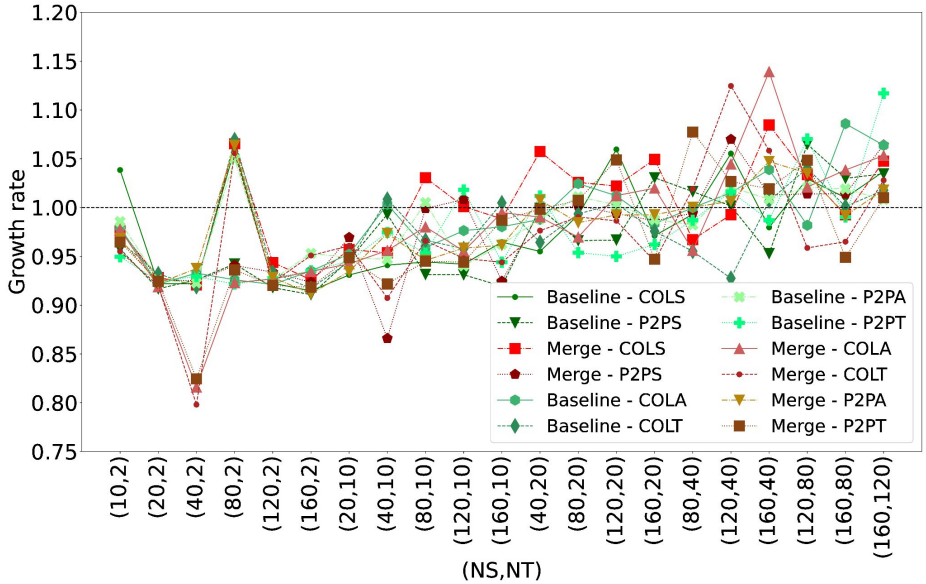


Dynamic-aware CG execution study

Expansion (Queen)



Shrinkage (Queen)



Outline:

1. Dynamic Resources
2. Proteo
3. Results
4. Closing remarks

Proteo as a framework to emulate parallel applications as dynamic-aware.

The framework can be used to study which reconfiguration mechanisms are most beneficial depending on the state of the application.

Proteo can be used to generate a workload of dynamic and emulated applications.

Simplify characterization of applications by using LIMITLESS and TALP.

Add more complex stages as GPU functions or I/O related

Communicate with the RMS. Preferably FLUX.

Proteo as a Framework for Dynamic Workload Emulation

Authors

Iker Martín-Álvarez (martini@uji.es)

Maribel Castillo

José I. Aliaga