

Malleability Tutorial

Alberto Cascajo, Javier Fernández, David E. Singh

October 14-16, 2026, Madrid, Spain

38th IEEE/SBC International Symposium on Computer Architecture and High-Performance Computing

ARCOS
COMPUTER ARCHITECTURE, COMMUNICATIONS AND SYSTEMS RESEARCH GROUP

Contents

- Introduction
- FlexMPI overview
- Application co-design with FlexMPI
- FlexMPI internals
- Use cases

Contents

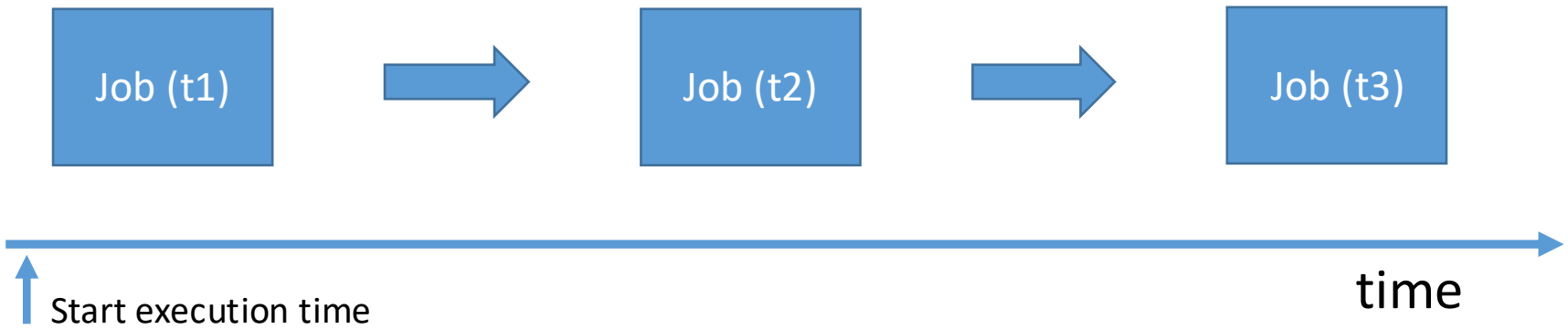
- Introduction
- FlexMPI overview
- Application co-design with FlexMPI
- FlexMPI internals
- Use cases

Job categories

- Rigid
- Moldable
- Evolving
- Malleable
- Adaptive

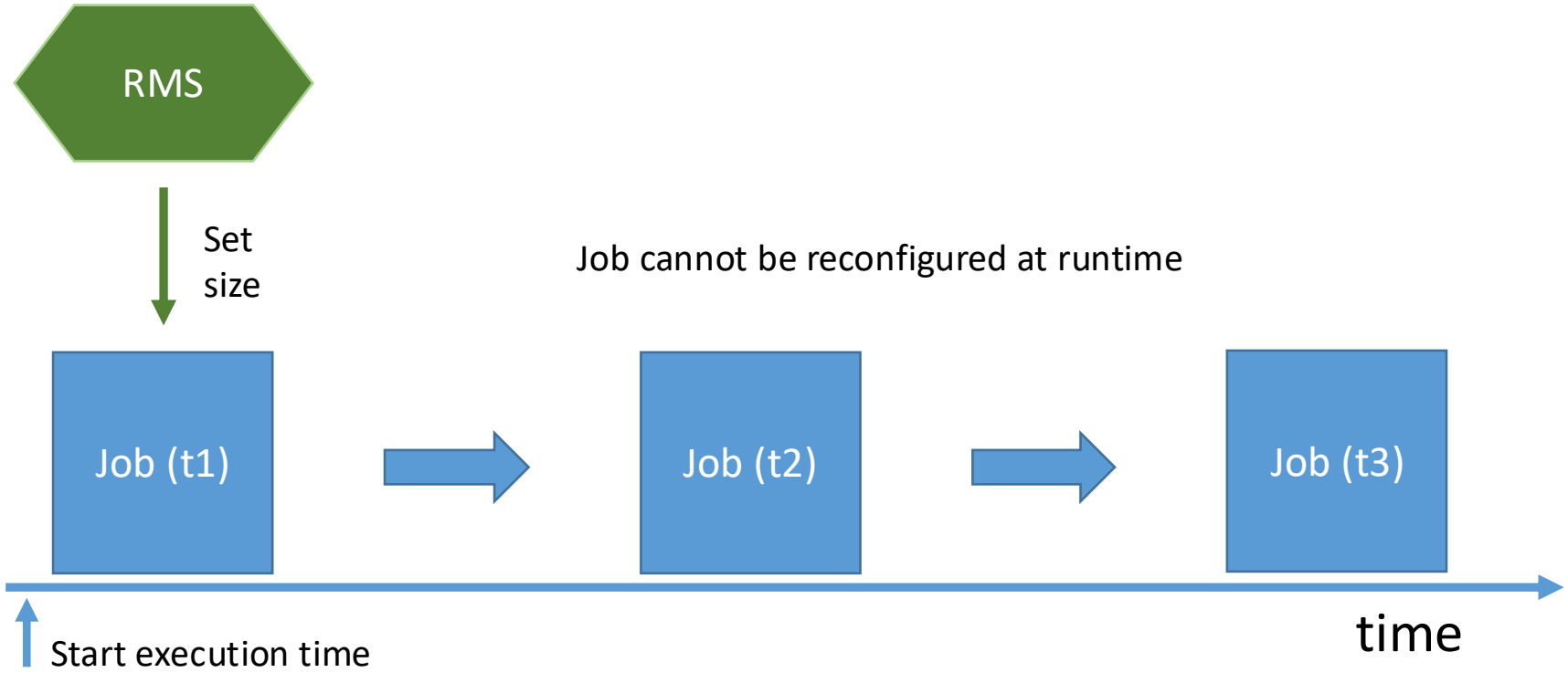
Job categories

- Rigid
- Moldable
- Evolving
- Malleable
- Adaptive



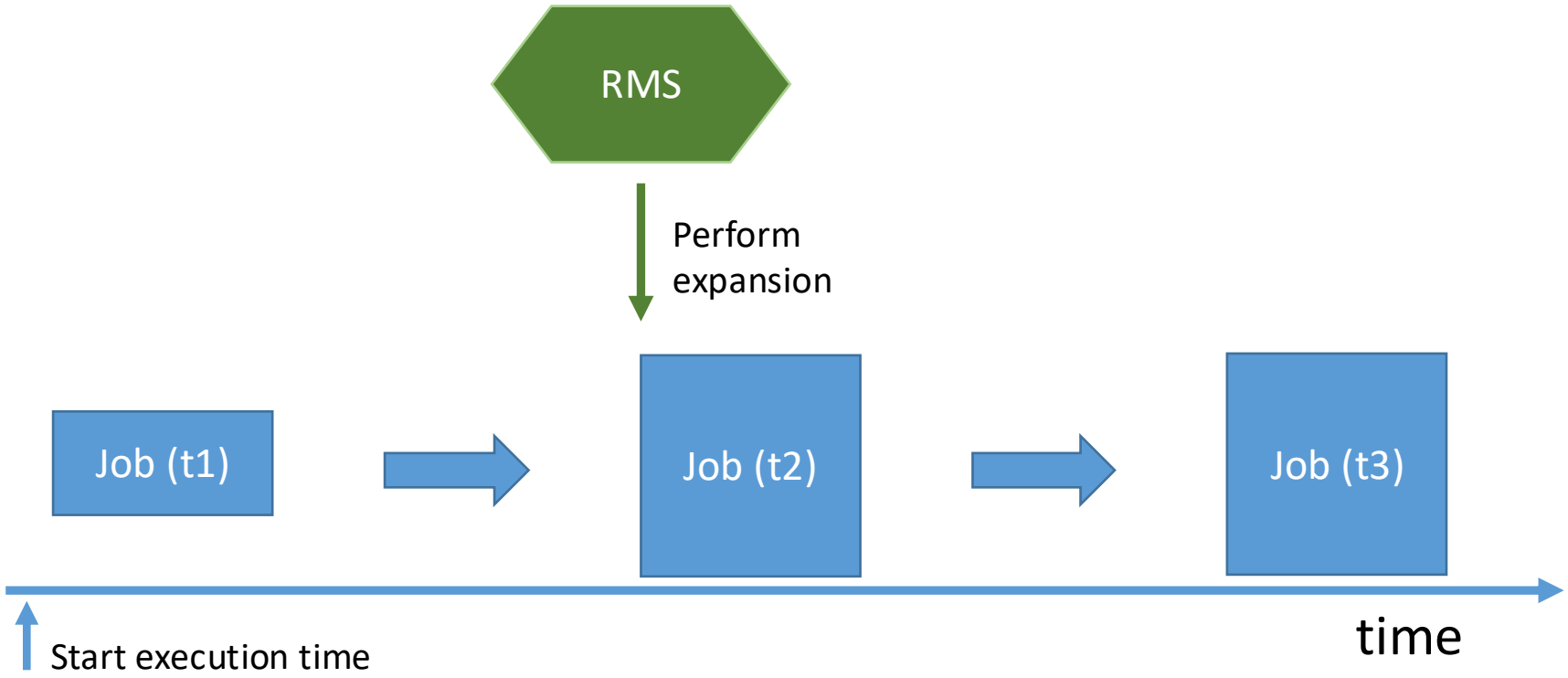
Job categories

- Rigid
- **Moldable**
- Evolving
- Malleable
- Adaptive



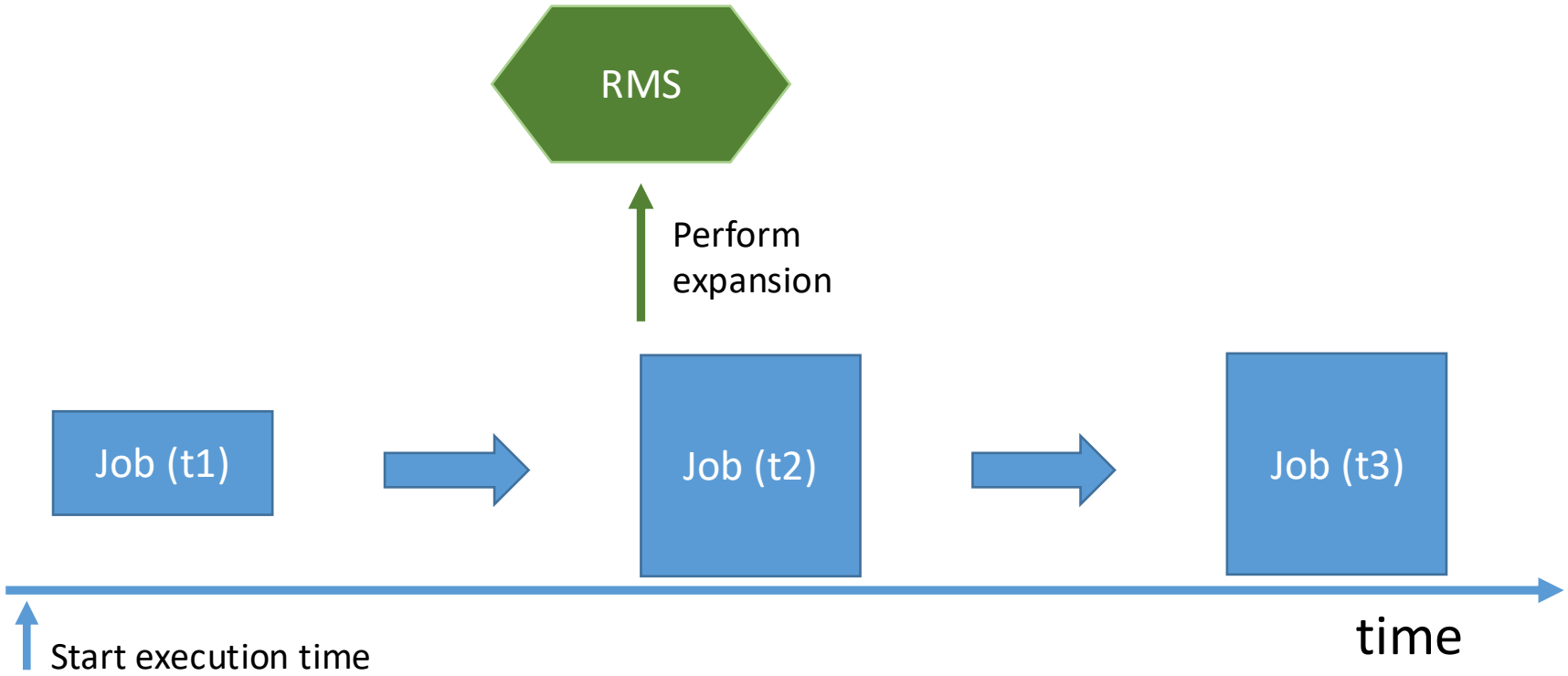
Job categories

- Rigid
- Moldable
- **Evolving**
- Malleable
- Adaptive



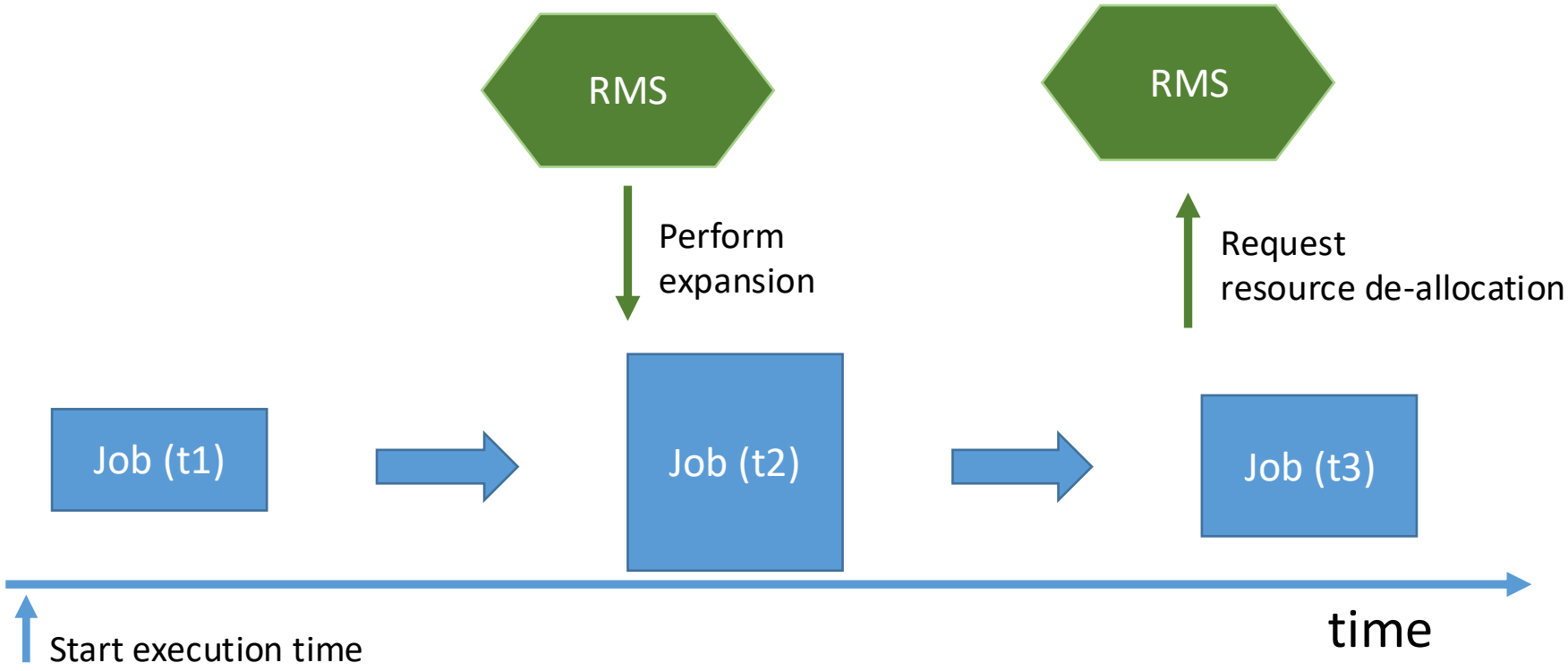
Job categories

- Rigid
- Moldable
- Evolving
- **Malleable**
- Adaptive



Job categories

- Rigid
- Moldable
- Evolving
- Malleable
- **Adaptive**



Malleability policies

- Reactive
  - Based on the application's performance, the runtime can make decisions to expand and reduce the number of processes:

    Example: the FlexMPI runtime detects that the iteration and CPU time increases. It can decide to expand the processes to distribute the load among more processes.
- Proactive
  - Based on the knowledge gathered from the applications and the system, the framework can make proactive decisions to improve performance instead of having to wait for an application event to be triggered.

    Example: Given application *App1*, knowing its execution phases, the system can allocate resources just before an expected expansion, accelerating the resource acquisition.

    Example 2: By using modeling and prediction techniques, performance-enhancing operations can be proactively implemented. This improves the flow of operations between resource allocation and process expansion operations.

Current challenges

- Slurm was designed for static jobs.
  - At certain moment, there were support for dynamic resource allocation.
  - Currently, it does not.
- The applications require support for runtime reconfiguration.
- Monitoring the system is required to perform real-time alloc/release decisions.
- Each user/application must execute the resource alloc/release operations independently: Slurm does not allow executing operations between jobs with different owners.
- Slurm must remain original. In production clusters, there is no choice of installing a custom RMS.

Contents

- Introduction
- FlexMPI overview
- Application co-design with FlexMPI
- FlexMPI internals
- Use cases

FlexMPI overview

- Runtime that provides dynamic load balancing and performance-aware malleability capabilities to MPI applications
- Written in C + MPI (~18,000 lines)
- Used with iterative SPMD parallel applications
- Consists of…
  - A multithreaded library executed within each MPI application
  - An external controller that coordinates the execution of multiple applications (optional)
  - An interface with third-party applications (scheduler, ADMIRE's intelligent controller).
- Source code released
  - https://gitlab.arcos.inf.uc3m.es/desingh/FlexMPI
  - Available under GNU GPL license.



FlexMPI overview

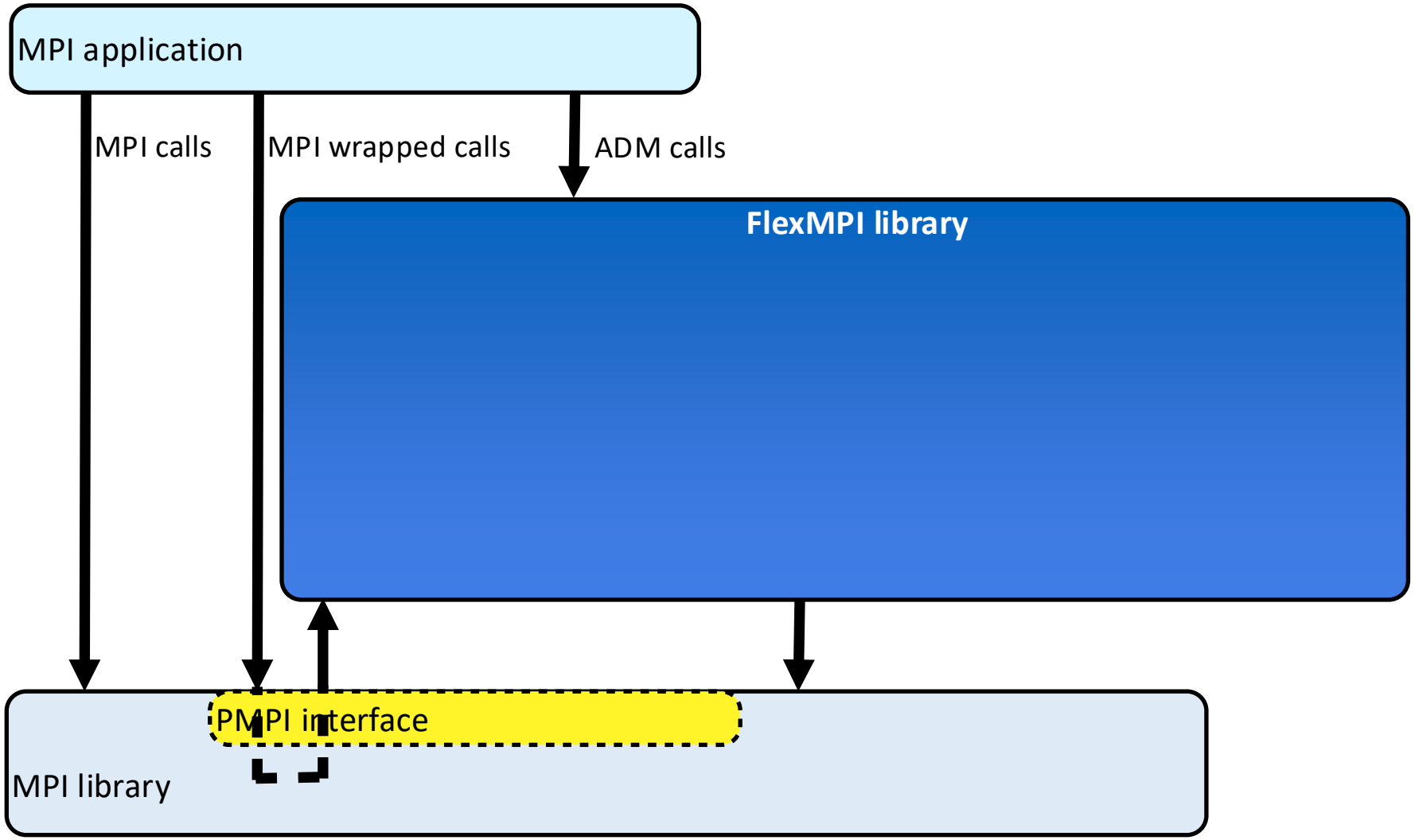
- FLEX-MPI development started in 2012
- Initial version released in 2015

- LIMITLESS – LIght-weight MonItoring Tool for multicriteria scheduling. **Microprocessors and Microsystems.** 2022**.**
- LIMITLESS - LIght-weight MonItoring Tool for LargE Scale Systems. **29th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP 2021).** 2021.
- Performance-aware scheduling of parallel applications on non-dedicated clusters. **Electronics**. 2019.
- Combining malleability and I/O control mechanisms to enhance the execution of multiple applications. **Journal of Systems and Software**. 2019.
- Improving the energy efficiency of MPI applications by means of malleability. 24th **Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP).** 2016.
- Enhancing the performance of malleable MPI applications by using performance-aware dynamic reconfiguration. **Parallel Computing**. 2015.
- FLEX-MPI: an MPI extension for supporting dynamic load balancing on heterogeneous non-dedicated systems. **European Conference on Parallel Computing (EUROPAR).** 2013.
- Runtime support for adaptive resource provisioning in MPI applications. **The 19th European MPI Users' Group Meeting - EuroMPI**. 2012.

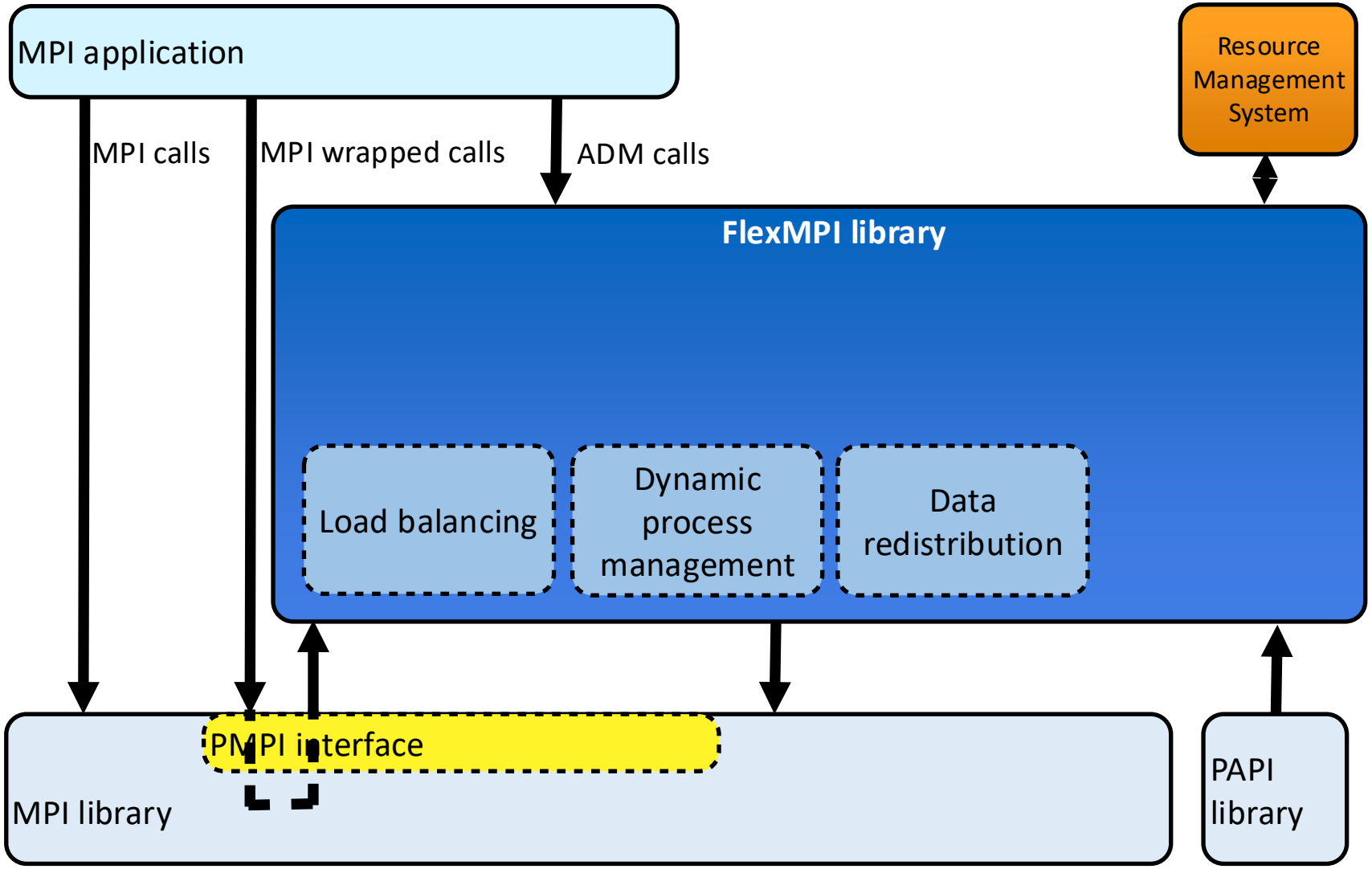
FlexMPI overview

- FlexMPI features
  - Provide malleable capabilities to MPI applications
  - Provide adaptive capabilities to MPI applications with different policies:
    - Strict malleability
    - High-performance
    - Adaptive policies: efficiency, cost, and energy.
    - User-defined actions
  - Load balance for both dedicated and non-dedicated platforms
  - Application run-time monitoring -using hardware counters-
  - Node-level monitoring -in coordination with external monitors (LIMITLESS)-
- Controller features
  - Application scheduling
  - I/O scheduling (with CLARISSE support)

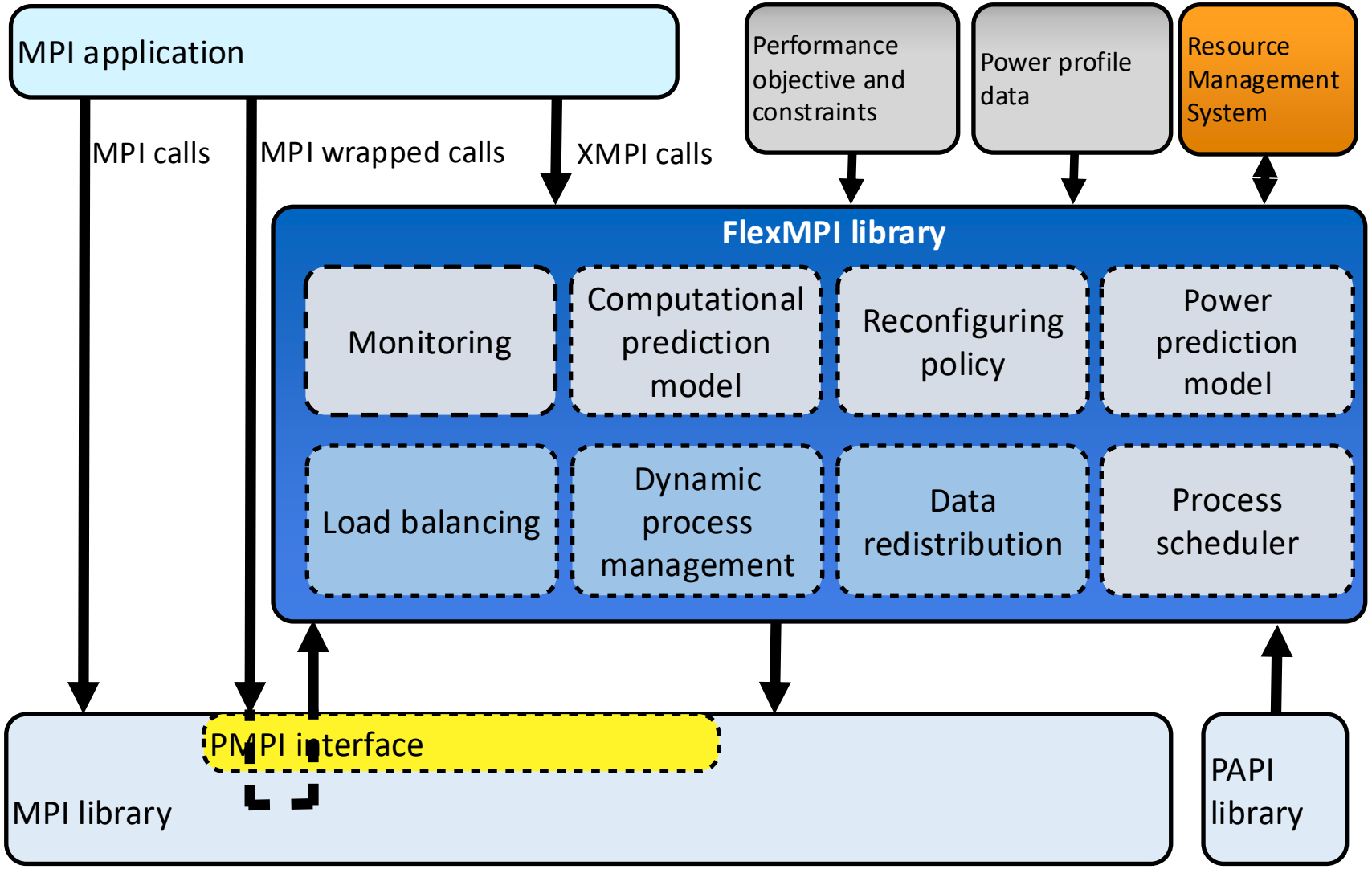
FlexMPI structure



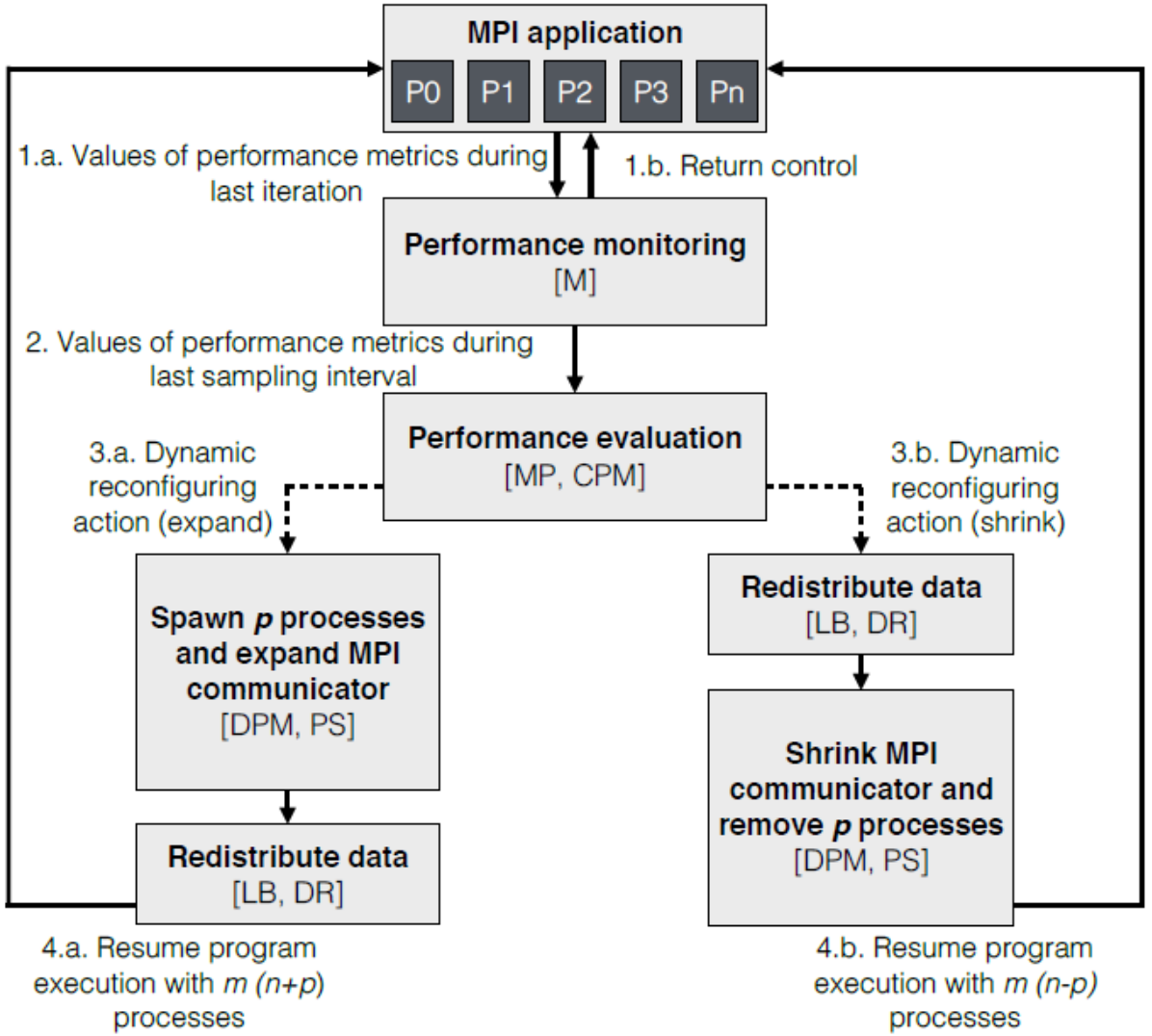
FlexMPI structure: malleable configuration



FlexMPI structure: adaptive configuration

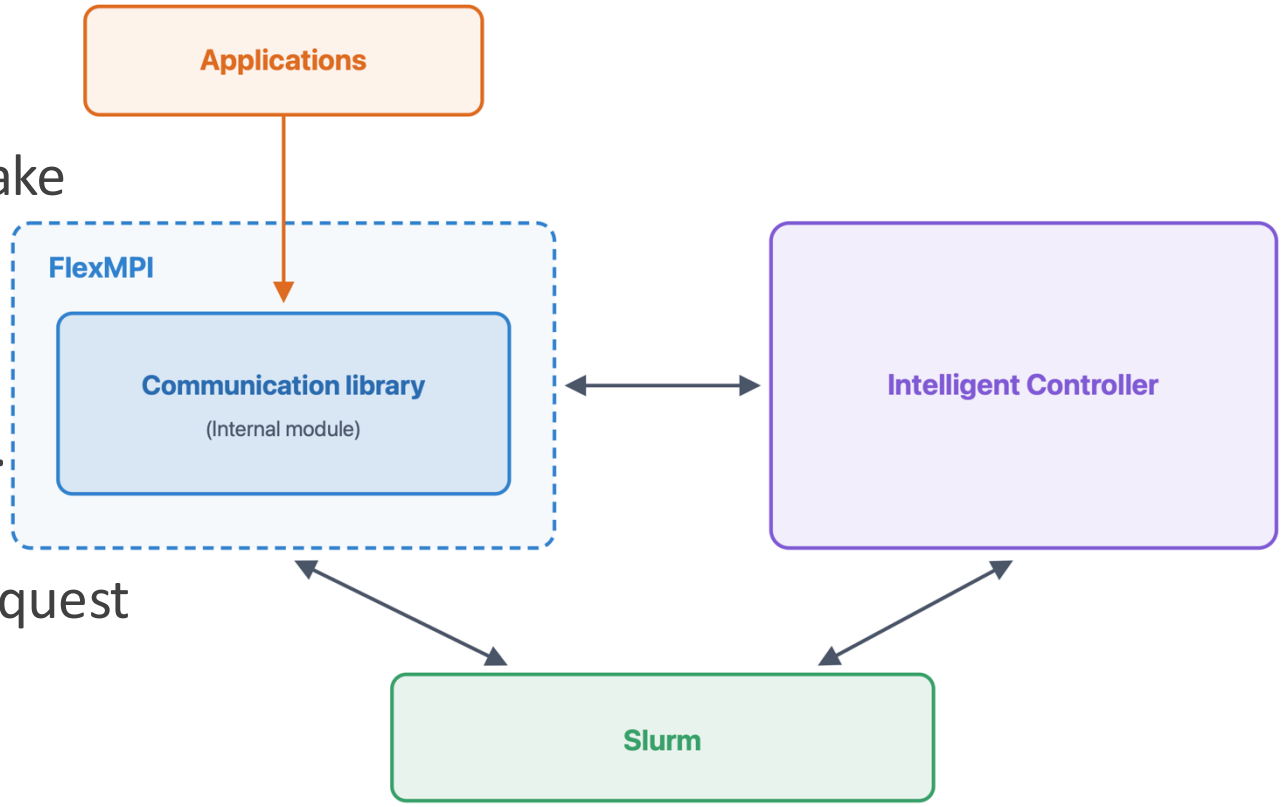


FlexMPI control flow



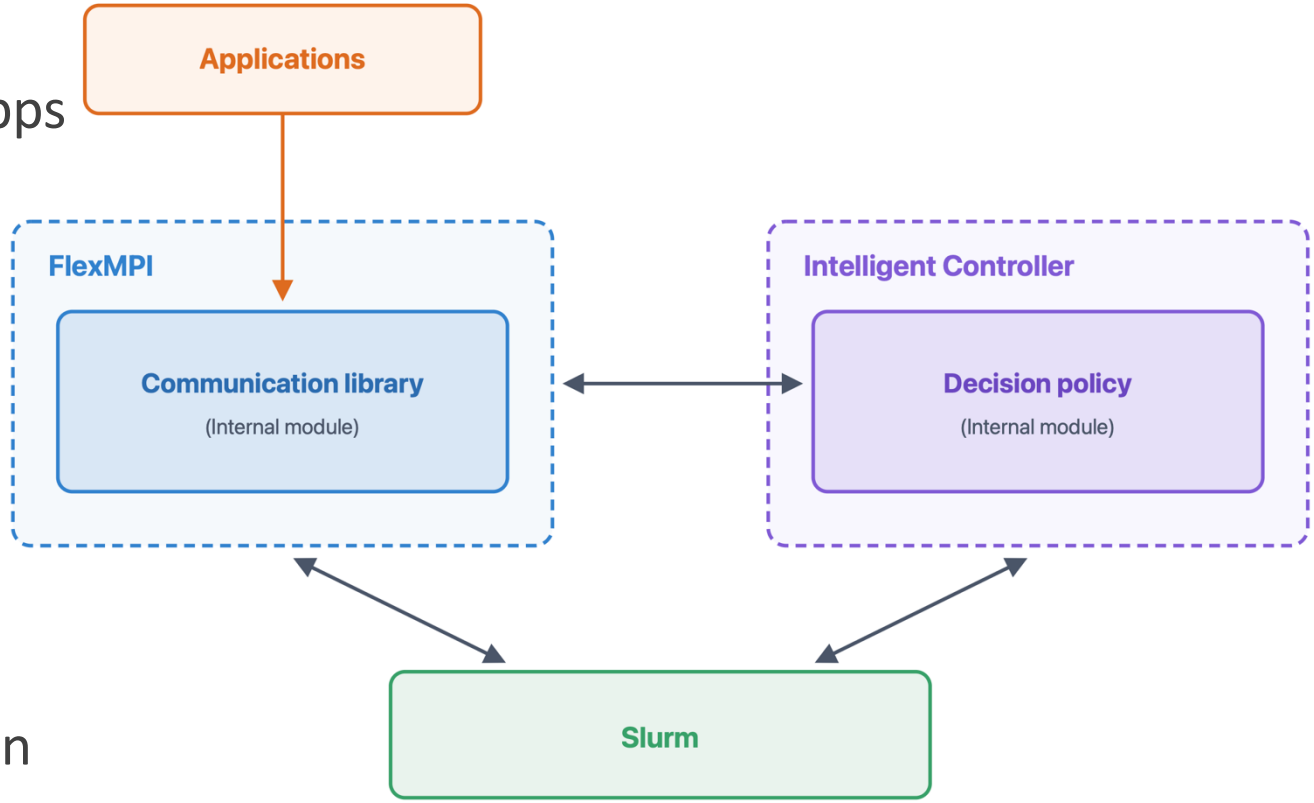
FlexMPI overview

- Intelligent Controller
  - FlexMPI sends resource requests.
  - The IC decides which operations must take place.
    It has an overview of the system status
  - IC responds to FlexMPI with its decision.
  - FlexMPI communicates with Slurm to request whatever is possible.



FlexMPI overview

- Intelligent Controller
  - It can accept every request from the apps
  - It can set a policy based on different variables:
    - Arrival time
    - Priorities
    - Speedup
    - SJF
  - It takes the decision, but the application apply it.



Contents

- Introduction
- FlexMPI overview
- **Application co-design with FlexMPI**
- FlexMPI internals
- Use cases

FlexMPI internals: code example (I)

Legacy code (MPI)

```
MPI_Init(...);

MPI_Comm_rank(...);
MPI_Comm_size(...);


for (it=0; it < itmax; it++) {

        for (i=0; i < count; i ++) {
                //Parallel computation
        }

   MPI_Allreduce (...);
}

MPI_Finalize ();
```

Instrumented code (FlexMPI)

```
MPI_init (...);
MPI_Comm_rank(...);
MPI_Comm_size(...);

ADM_GetSysAttributesInt
("ADM_GLOBAL_PROCESS_TYPE", &proctype);
if (proctype == ADM_NATIVE) {
     … original initialization…
}
else{
     ADM_GetSysAttributesInt
("ADM_GLOBAL_ITERATION", &it);
}
for (it; it < itmax; it++) {
        ADM_MalleableRegion(START);
        for (i=despl; i < count; i ++) {
                //Parallel computation
        }
        MPI_Allreduce (...);

        statuss = ADM_MalleableRegion(STOP);

        if (status != ADM_ACTIVE) break;
}
MPI_Finalize ();
```

FlexMPI internals: code example (II)

Legacy code (MPI)

```
MPI_Init(...);

MPI_Comm_rank(...);
MPI_Comm_size(...);
A = init();

for (it=0; it < itmax; it++) {

        for (i=0; i < count; i ++) {
                //Parallel computation
                ...A[i]...
        }

   MPI_Allreduce (...);
}

MPI_Finalize ();
```

Instrumented code with data structures

```
MPI_init (...);
MPI_Comm_rank(...);
MPI_Comm_size(...);

EMPI_Register (“matrix”,A, EMPI_DISJOINT);
ADM_GetSysAttributesInt ("ADM_GLOBAL_PROCESS_TYPE",
                                        &proctype);
if (proctype == ADM_NATIVE) {
      A = init();
}
else{
      ADM_GetSysAttributesInt ("ADM_GLOBAL_ITERATION", &it);
      A = EMPI_Get_addr("matrix");
}
for (it; it < itmax; it++) {
          ADM_MalleableRegion(START);
          for (i=despl; i < count; i ++) {
                  //Parallel computation
                  …A[i]…
          }
          MPI_Allreduce (...);

          status = ADM_MalleableRegion(STOP);

          if (status != ADM_ACTIVE) break;
          A = EMPI_Get_addr("matrix");
}
MPI_Finalize ();
```

Contents

- Introduction
- FlexMPI overview
- Application co-design with FlexMPI
- **FlexMPI internals**
- Use cases

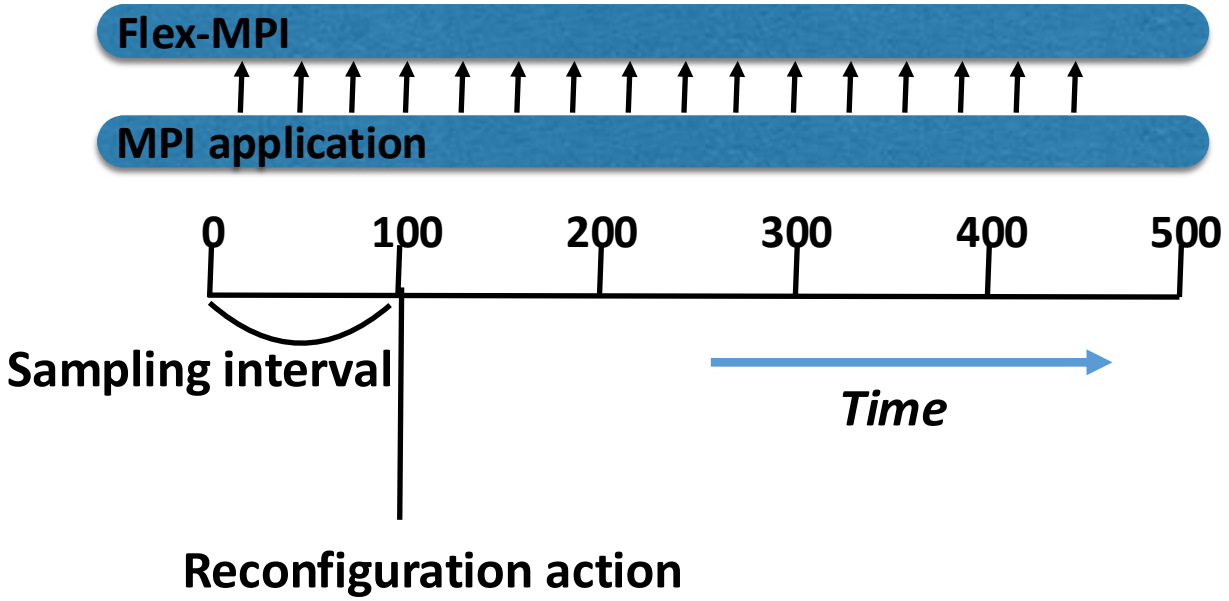
Flex-MPI low-level interfaces

Interface	Description
XMPI_Monitor_iter_begin	Starts iteration monitoring
XMPI_Monitor_iter_end	Stops iteration monitoring
XMPI_Spawn	Spawns dynamic processes
XMPI_Remove	Removes dynamic processes
XMPI_Get_wsize	Retrieves current data partition
XMPI_Register_vector	Registers a vector
XMPI_Register_dense	Registers a dense data structure
XMPI_Register_sparse	Registers a sparse data structure
XMPI_Redistribute_data	Data redistribution

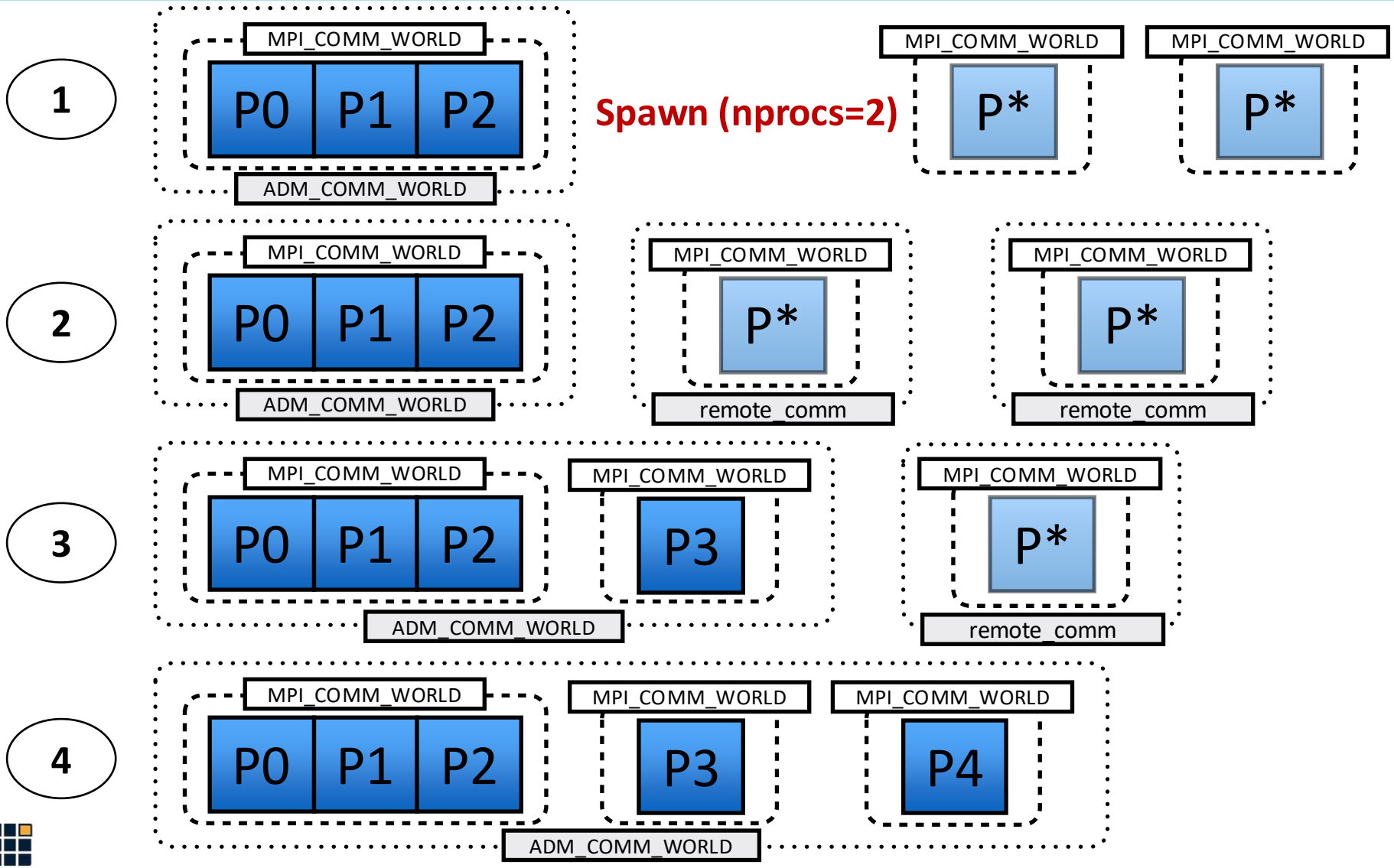
- Apart from functional abstractions, users can utilize FlexMPI's low-level functions.

FlexMPI internals

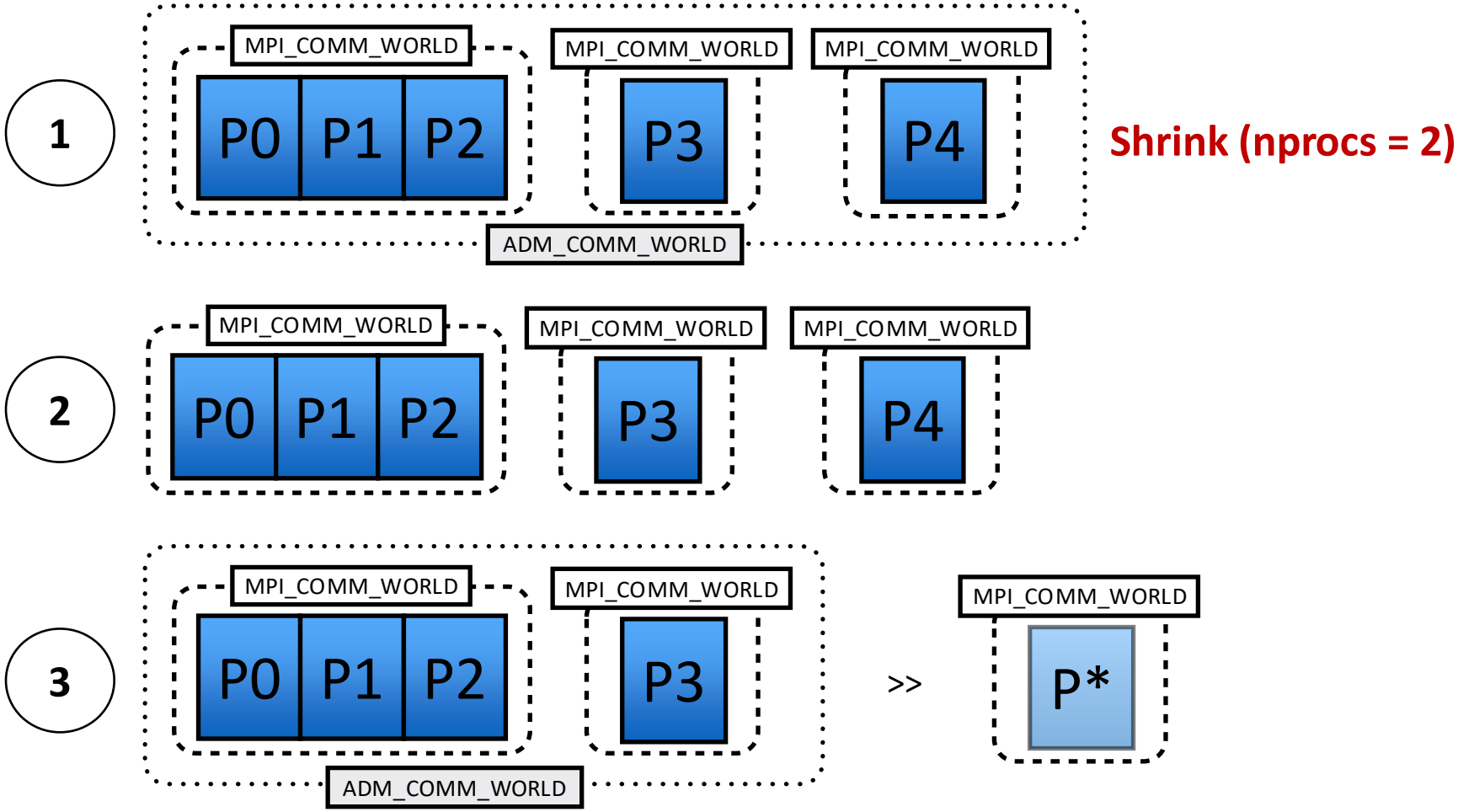
- FlexMPI was designed for
  - Iterative SPMD MPI applications



Communicator management



Communicator management



Contents

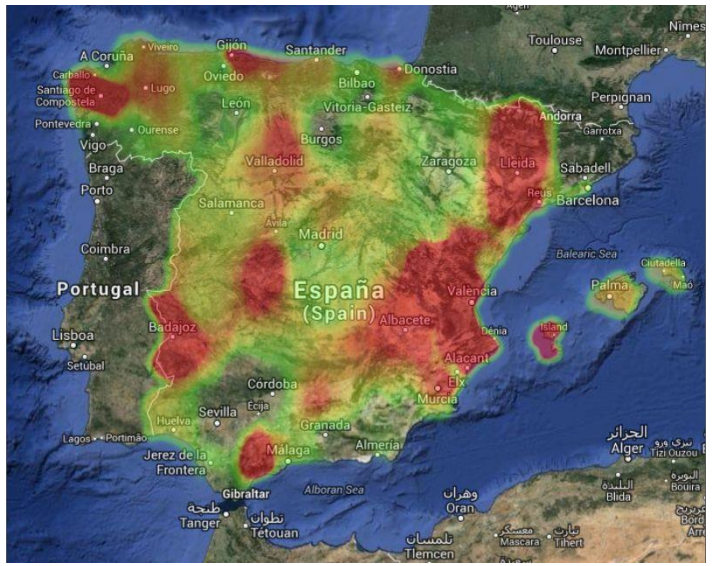
- Introduction
- FlexMPI overview
- Application co-design with FlexMPI
- FlexMPI internals
- Use cases

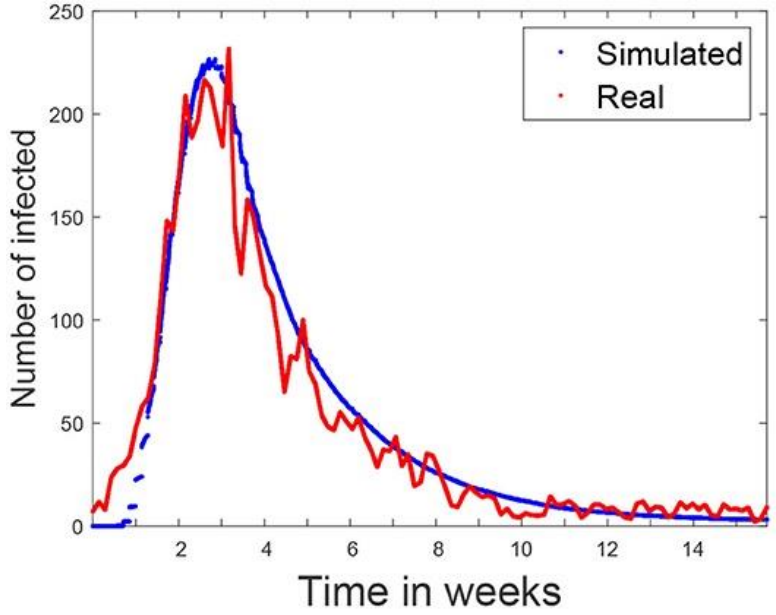
Applications (I)

- Kernels
  - Jacobi-method kernel
    - CPU intensive, MPI communications, no I/O
    - Dense matrices
    - Good data locality
  - Conjugate Gradient kernel
    - CPU intensive, MPI communications, no I/O
    - Sparce matrices
    - Poor data locality
  - Synthetic kernel
    - Alternates CPU phases, communication and I/O phases
    - Phase intensities are parametrized
    - I/O access pattern can be parametrized (use of MPI views)

Applications (II)

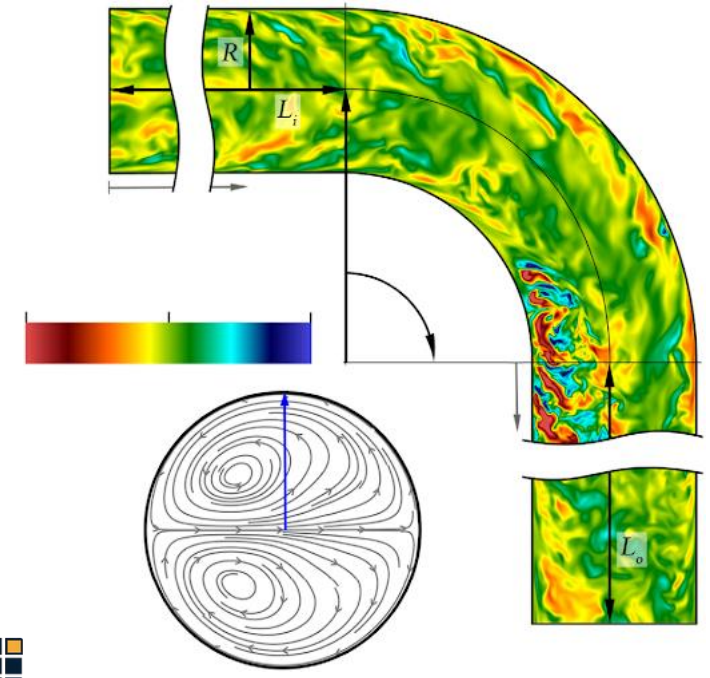
- EpiGraph
  - Influenza and COVID-19 propagation simulator
  - https://www.arcos.inf.uc3m.es/epigraph/
  - Communication-intensive application
  - Sparse data structures
  - CPU and communication phases have variable intensities

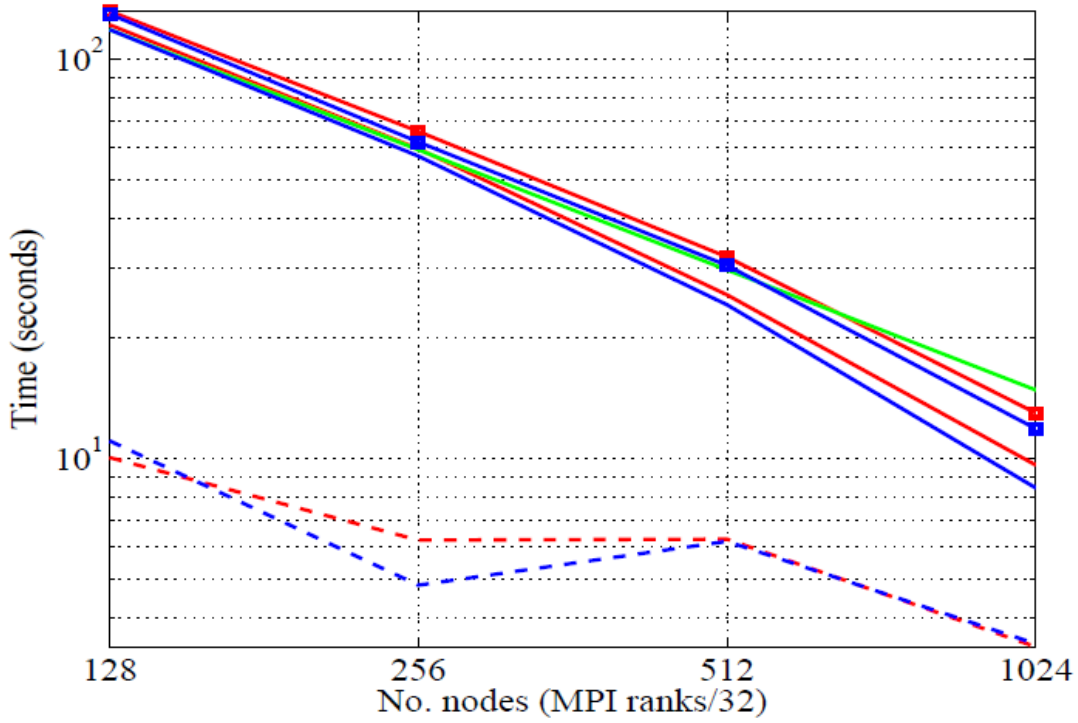




Applications (III)

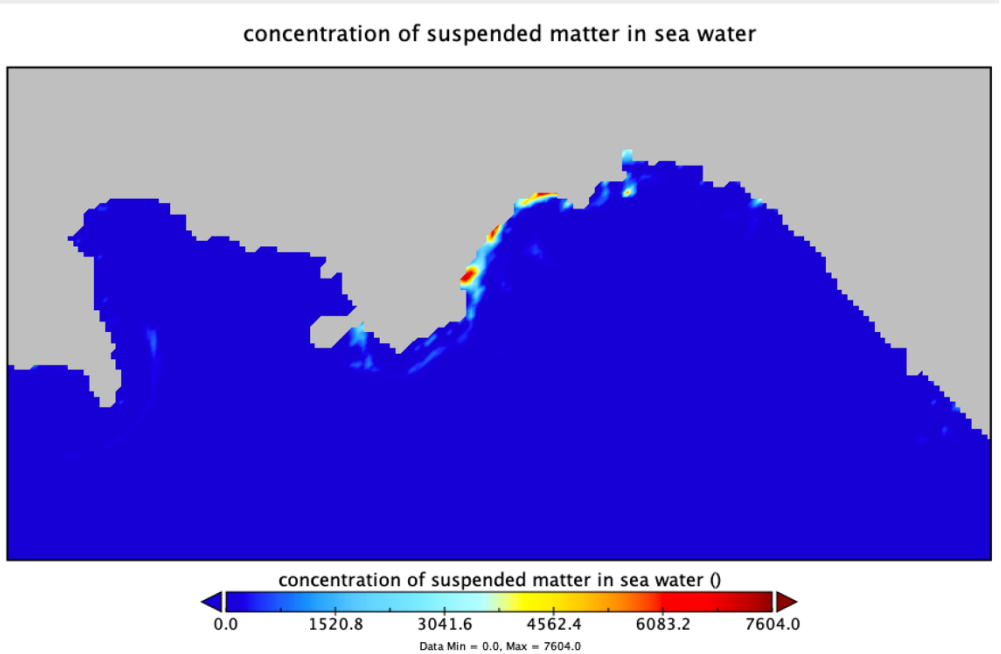
- NEK5000
  - Computational Fluid Dynamics solver based on Spectral Element Method.
  - https://github.com/Nek5000/Nek5000
  - This use case performs a study of the turbulent flows generated in bent pipes.
  - The computation processes 204.800.000 grid points with multiple values in double precision.

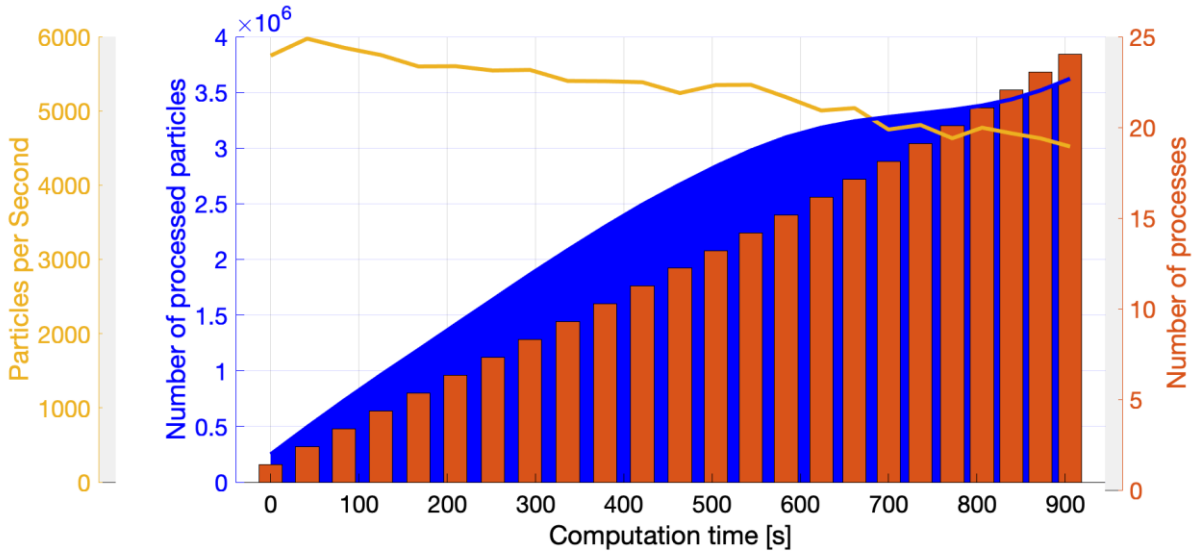




Applications (IV)

- WaCoMm++
  - Three-dimensional Lagrangian model implemented as an evolution of the LAMP3D.
  - https://github.com/CCMMMA/wacommplusplus
  - WaComM++ is used for pollutants transport and diffusion at the Center for Monitoring and Modelling Marine and Atmosphere applications (CMMMA) run by the Department of Science and Technologies of the University of Naples.





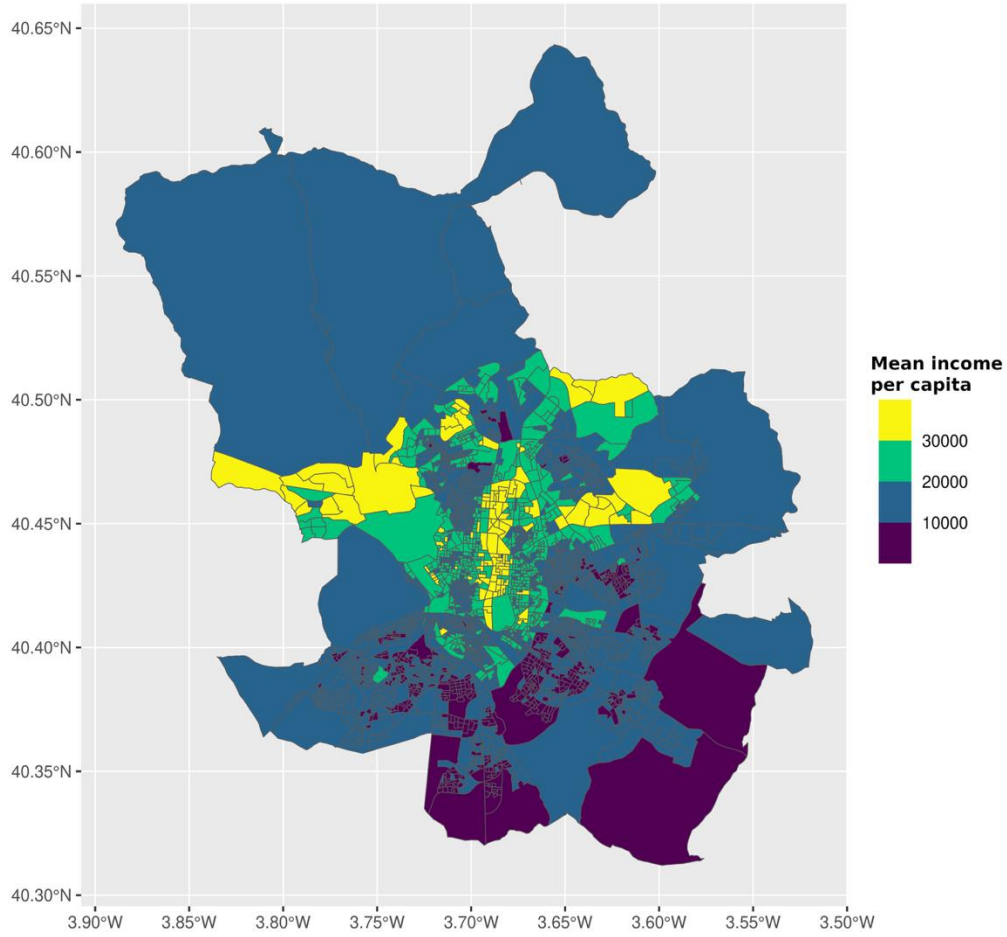
Contents

- Introduction
- FlexMPI overview
- Application co-design with FlexMPI
- FlexMPI internals
- **Use cases**
  - EpiGraph

EpiGraph

- EpiGraph is a parallel epidemiological simulator
  - Coded in C programming language + MPI
  - Agent-based
  - Models the virus propagation (influenza and COVID-19)

- Active collaboration with:
  - European Union collaboration via the European Centre for Disease Prevention and Control
  - Spanish Ministry of Health
  - Health authorities of Comunidad de Madrid



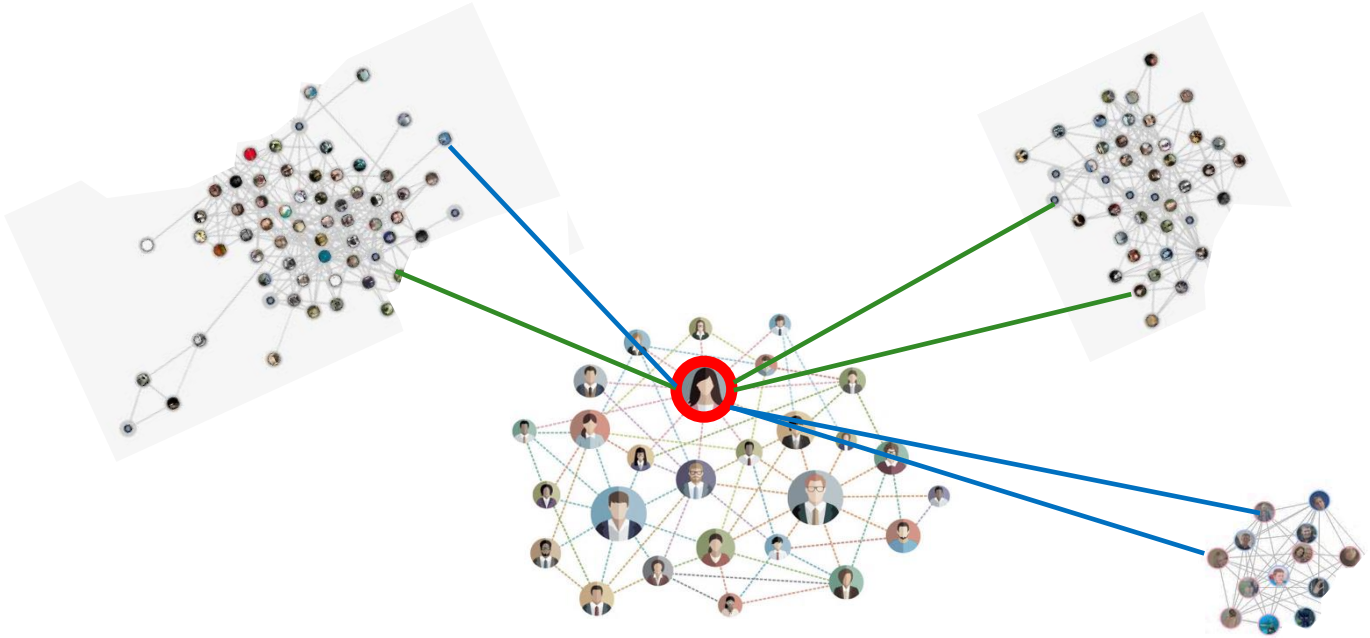
EpiGraph

Social interaction model

- Combine contact info from social networks with contact matrices to reproduce individual interactions
- Different kind of connections according to the interaction nature
- Connections take into account social particularities (day of week, time of day, profession, etc.)

Simulator scalability

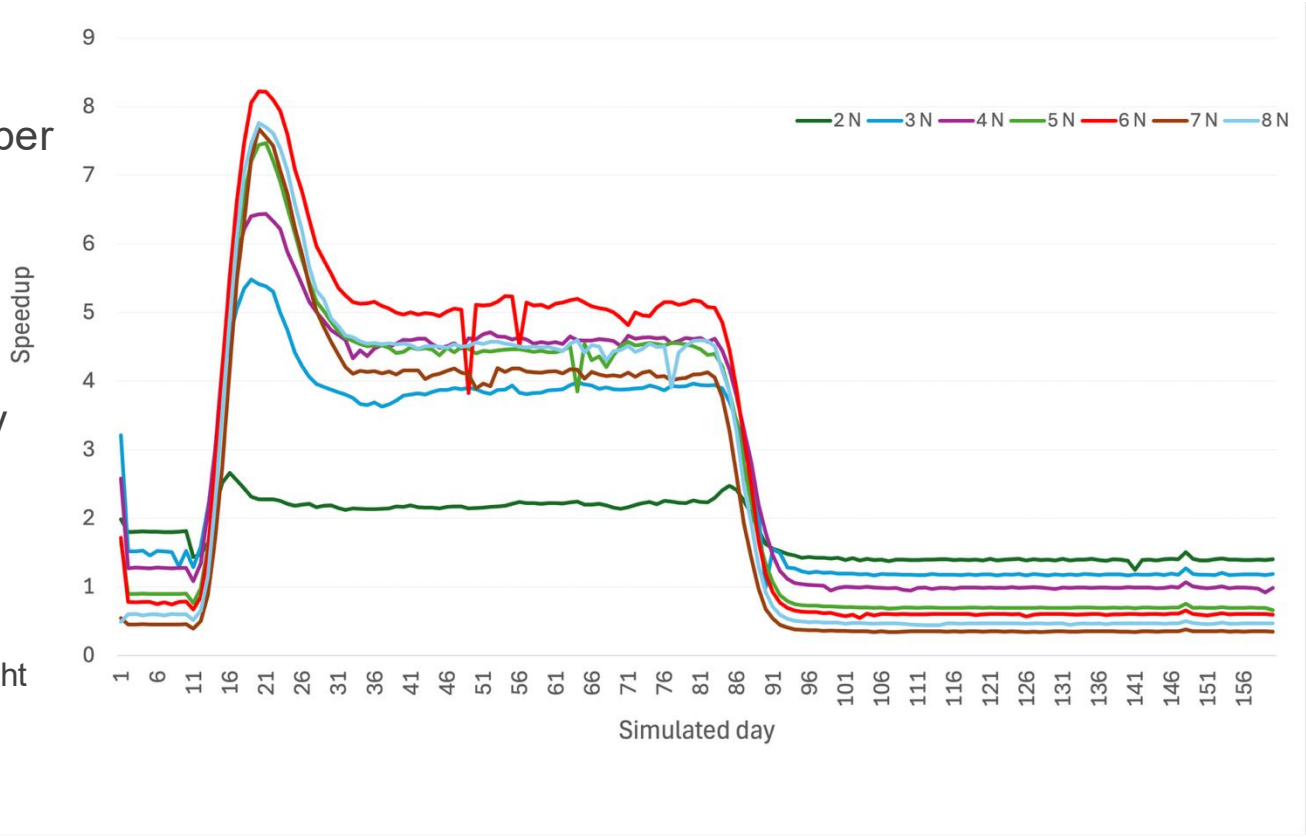
- Allow simulations across wide areas
- Madrid urban area: 5.2M individuals
- Spain: 20M individuals



EpiGraph

Performance behavior

- EpiGraph combines CPU phases with Communication and IO phases.
- The intensity of the CPU phases depends on the number of infected individuals → peaks of infected individuals generate peaks of CPU consumption.
- On the contrary, Communication phases appear when the population is healthy.
- With malleability, EpiGraph can save costs and energy because it adapts the number of resources allocated depending on the CPU phases:
  - If infected individuals increase but FLOPS remain constant, the processes are overloaded → **EXPAND**
  - If FLOPS fall below a certain threshold, the processes are under a light load → **SHRINK**.

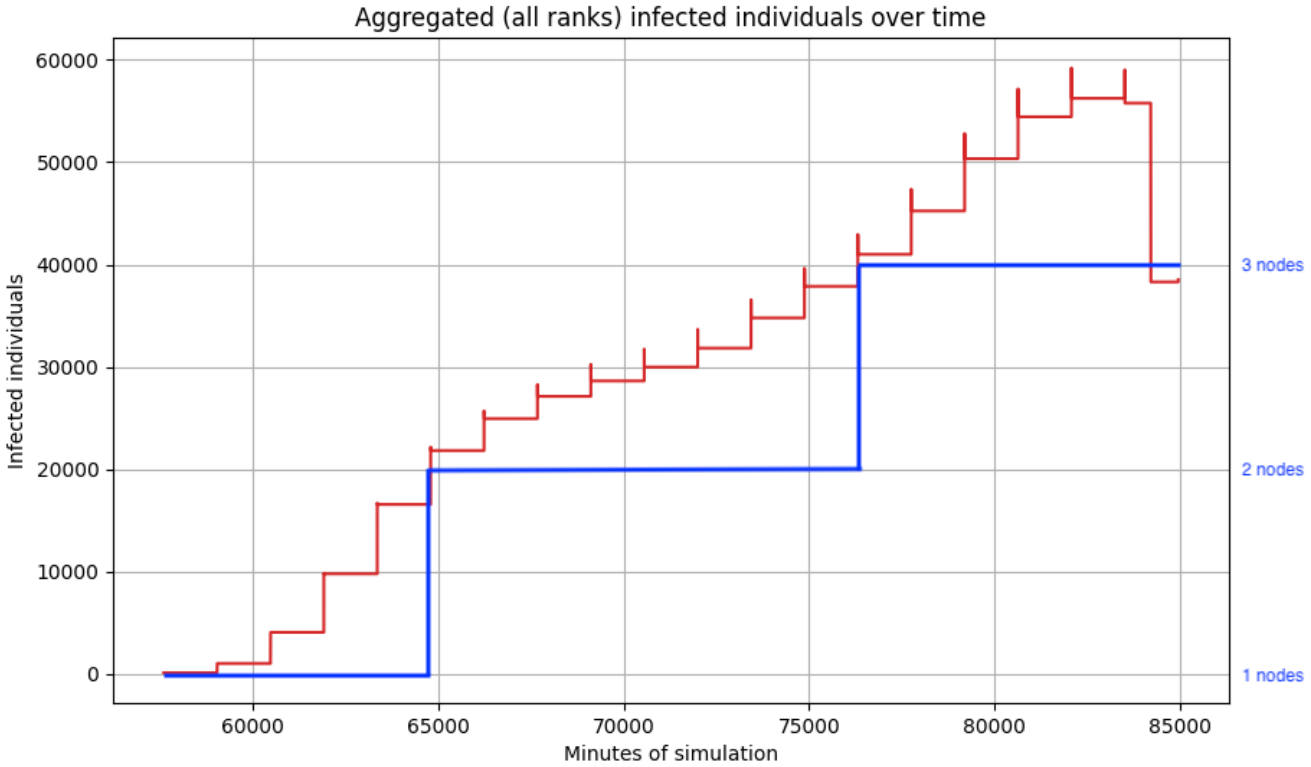


Objective: Adjust resources to CPU demands

EpiGraph

Performance behavior

- If the number of infected people exceeds a threshold, **expand** to keep the processing rate
- Above the threshold, the nodes are overloaded.



Objective: Adjust resources to CPU demands

Unlocking malleability in HPC applications

Malleability Tutorial (part 1)

Alberto Cascajo, Javier Fernández, David E. Singh

October 14-16, 2026, Madrid, Spain

38th IEEE/SBC International Symposium on Computer Architecture and High-Performance Computing

ARCOS
COMPUTER ARCHITECTURE, COMMUNICATIONS AND SYSTEMS RESEARCH GROUP