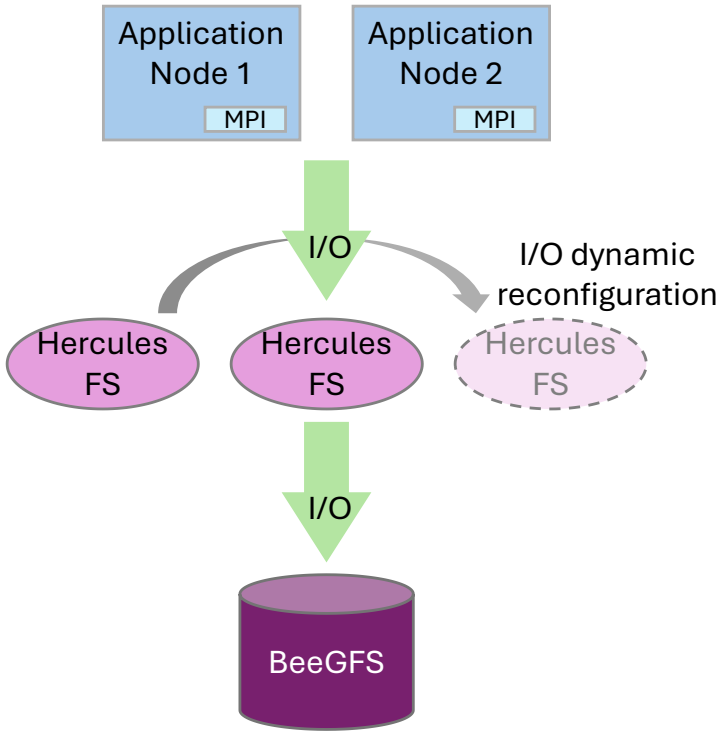
Exploring Combined Dynamic Resource Management in Applications and File Systems

Paula Sanchez-Checa, Genaro Sanchez-Gallegos, Javier García-Blas,
Jesús Carretero and **David E. Singh**


CLINT EASTWOOD
THE GOOD THE BAD and THE UGLY
co-starring
LEE VAN CLEEF
also starring
ELI WALLACH
in the role of TUCO
directed by
SERGIO LEONE

The Good: Hercules

- Ad-hoc in-memory filesystem
- Captures POSIX I/O calls
- I/O and metadata servers
- UCX as communication library
- Dynamic reconfiguration capabilities



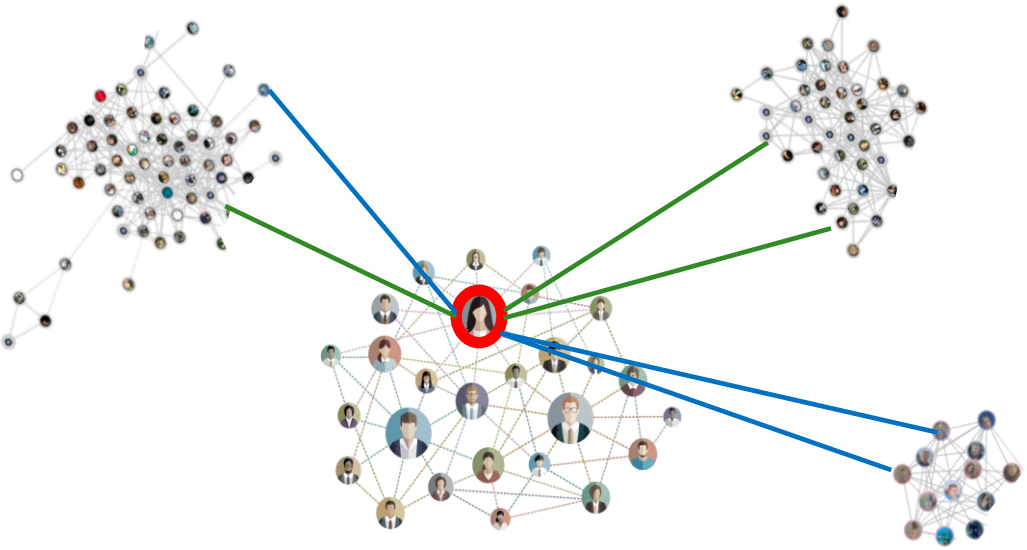
https://gitlab.arcos.inf.uc3m.es/admire/hercules

The Bad: EpiGraph

- Agent-based epidemiologic simulator
- Multiple COVID-19 variants
  Extended to model Influenza and RSV
- Social model built from social networks and contact matrices
- Collaboration with ECDC, Spanish Ministry of Health, and National Epidemiology Center

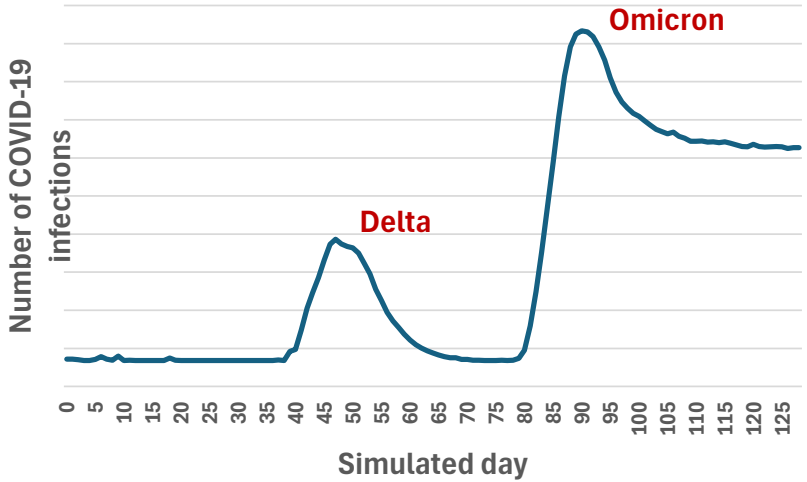
EpiGraph

- EpiGraph simulation scenario
  - SARS-CoV-2 Omicron variant outbreak in Spain in November 2021
  - Madrid city
  - 3.5 million individuals



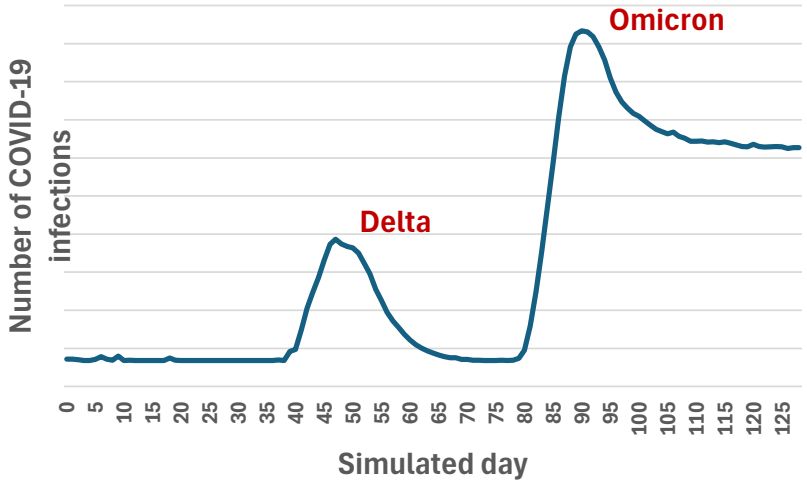
Why EpiGraph is bad?

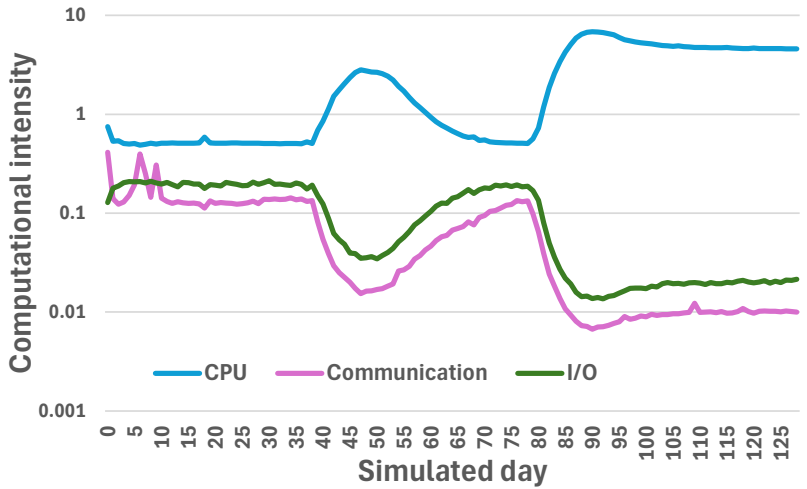
- EpiGraph alternates
  - CPU: 10 minutes (simulated time)
  - Communication: every simulated hour
  - I/O: every simulated hour



Why EpiGraph is bad?

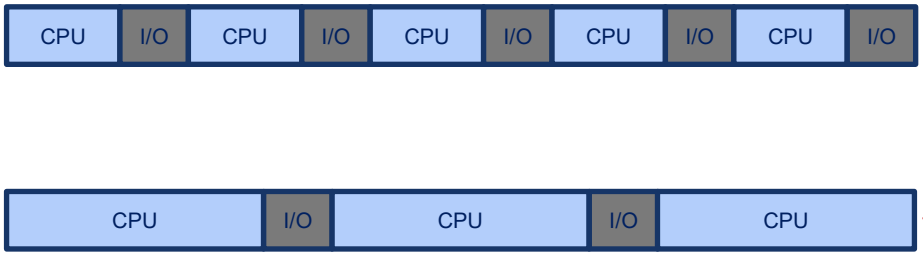
- EpiGraph alternates
  - CPU: 10 minutes (simulated time)
  - Communication: every simulated hour
  - I/O: every simulated hour

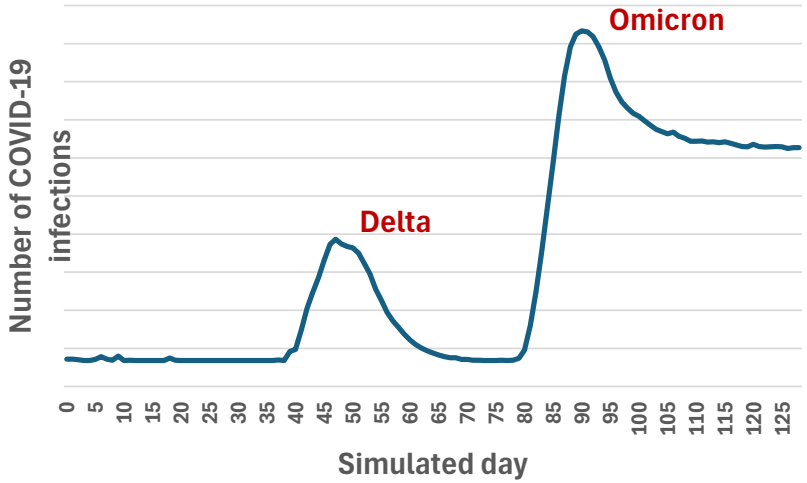


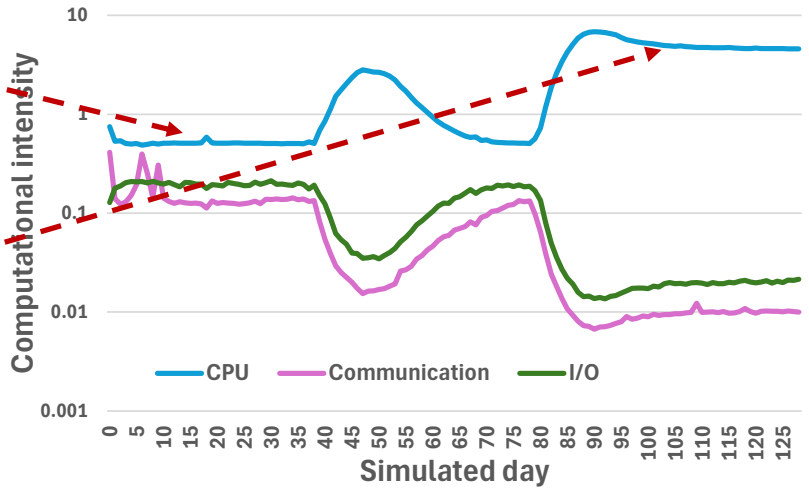


Why EpiGraph is bad?

- EpiGraph alternates
  - CPU: 10 minutes (simulated time)
  - Communication: every simulated hour
  - I/O: every simulated hour



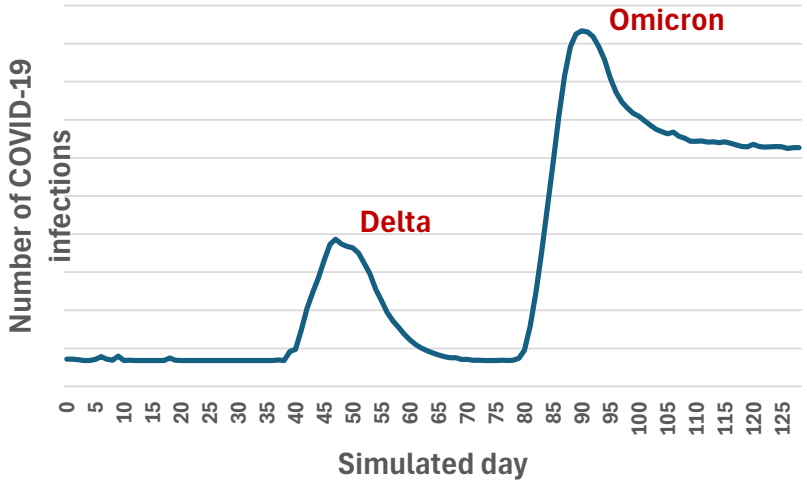


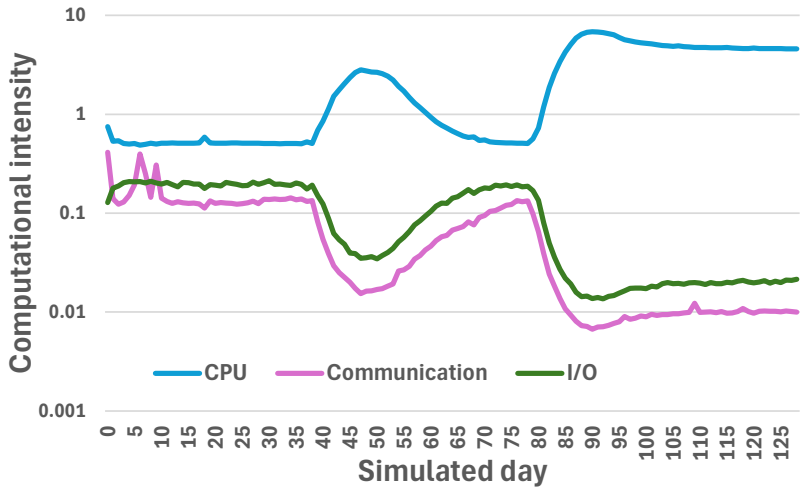


Why EpiGraph is bad?

- EpiGraph alternates
  - CPU: 10 minutes (simulated time)
  - Communication: every simulated hour
  - I/O: every simulated hour

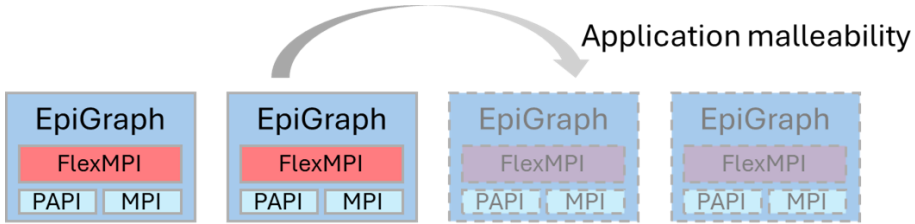
How many computational and I/O resources do we use?





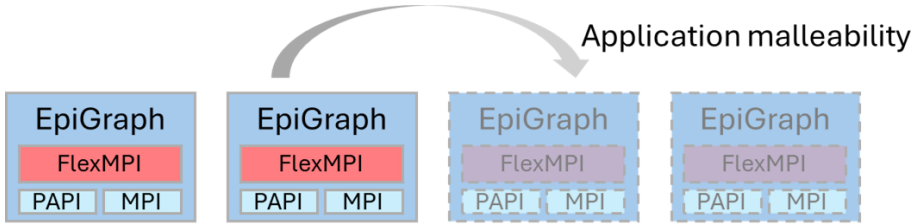
EpiGraph

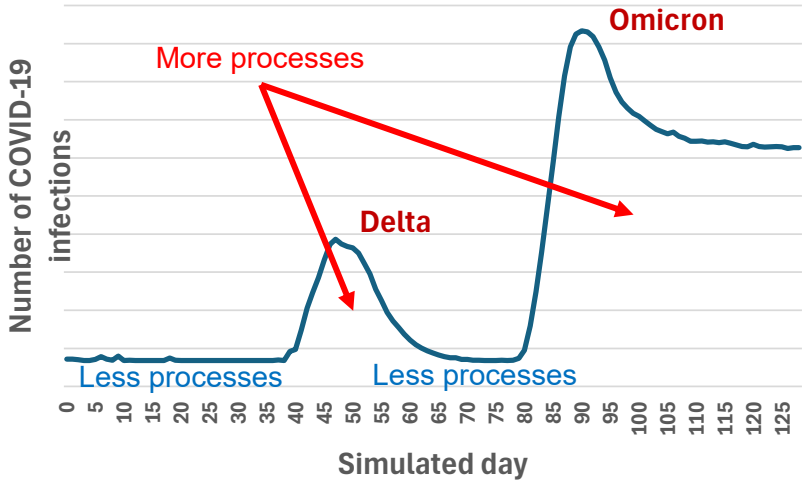
- EpiGraph executed with **FlexMPI**
  - Dynamically increasing or reducing the number of processes
  - Automatic data redistribution



EpiGraph

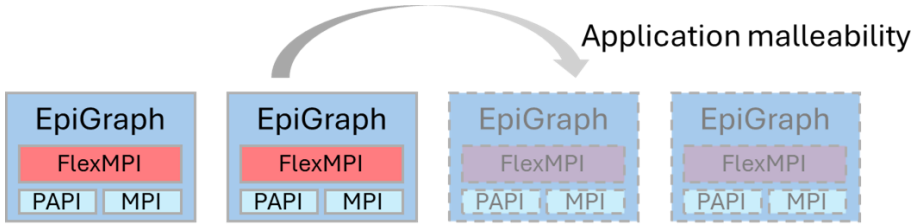
- EpiGraph executed with FlexMPI
  - Dynamically increasing or reducing the number of processes
  - Automatic data redistribution

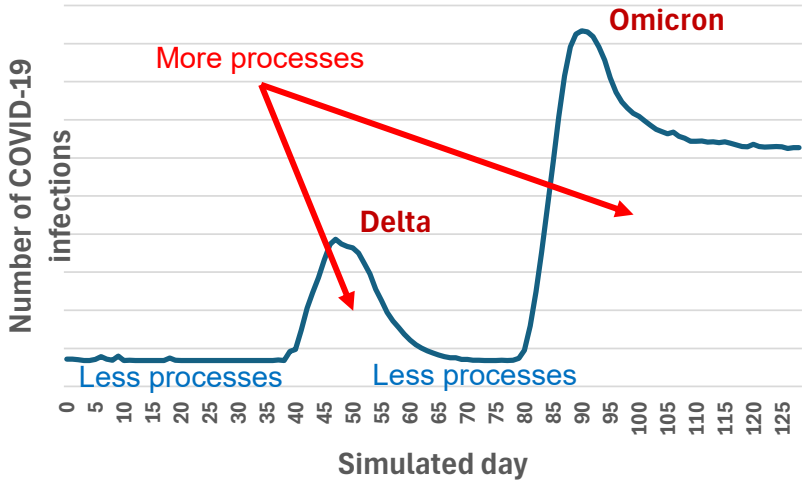


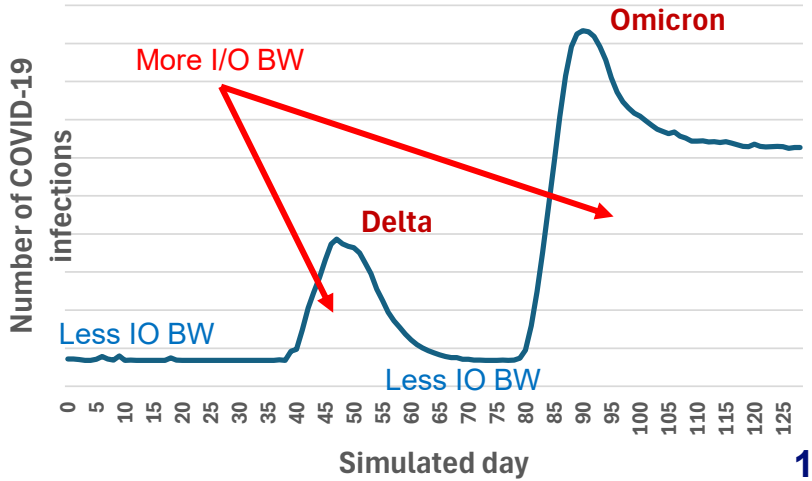


EpiGraph

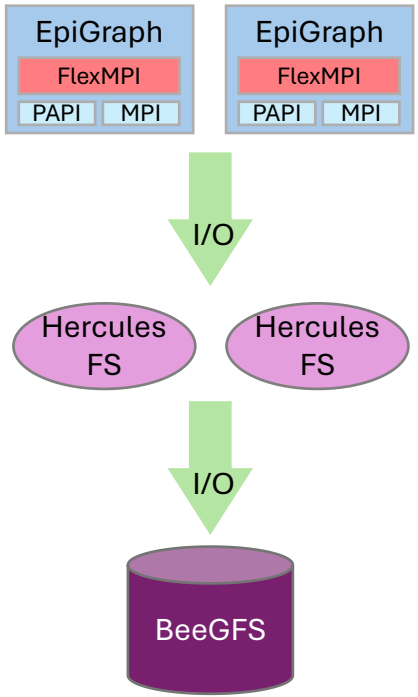
- EpiGraph executed with FlexMPI
  - Dynamically increasing or reducing the number of processes
  - Automatic data redistribution

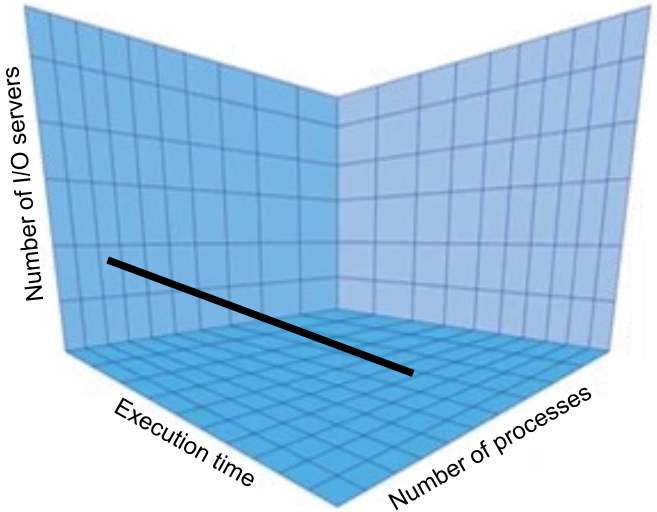




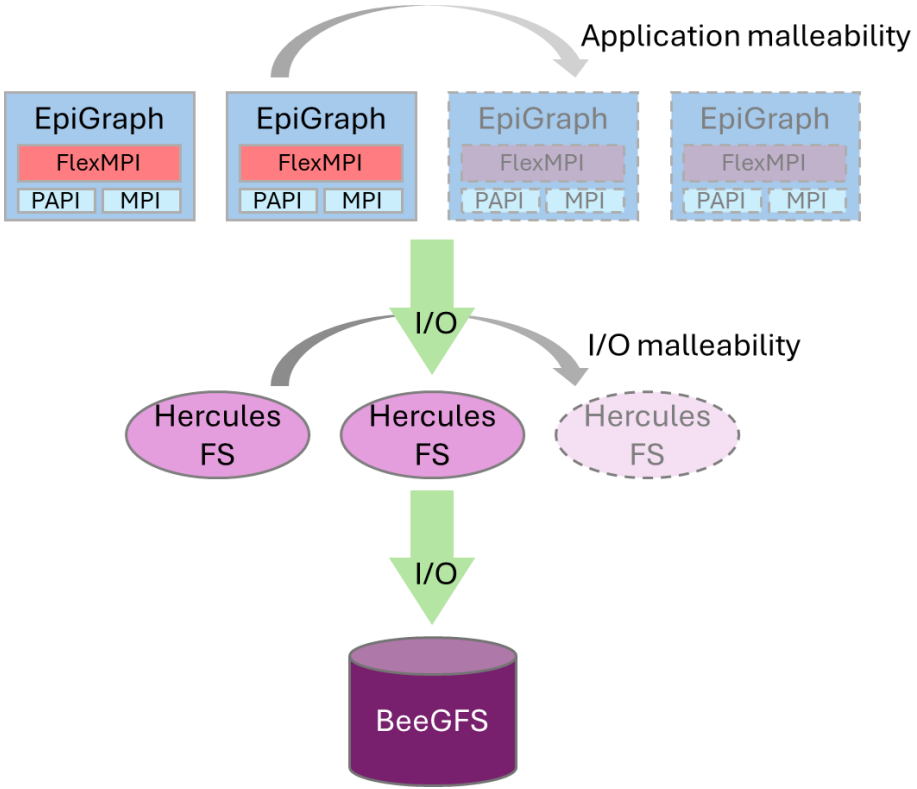


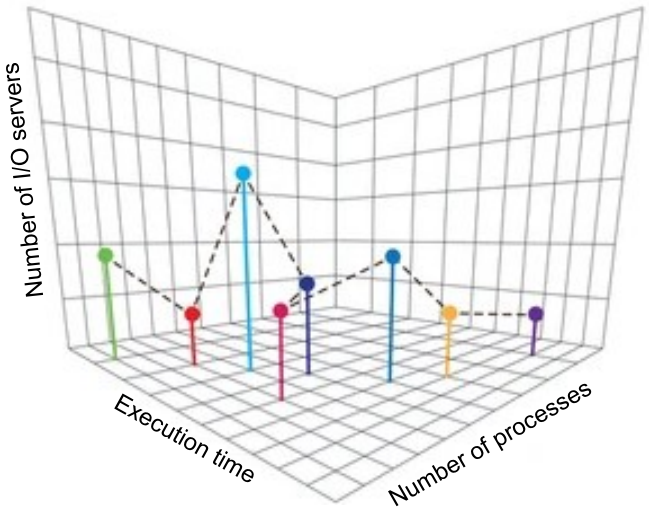
Static execution





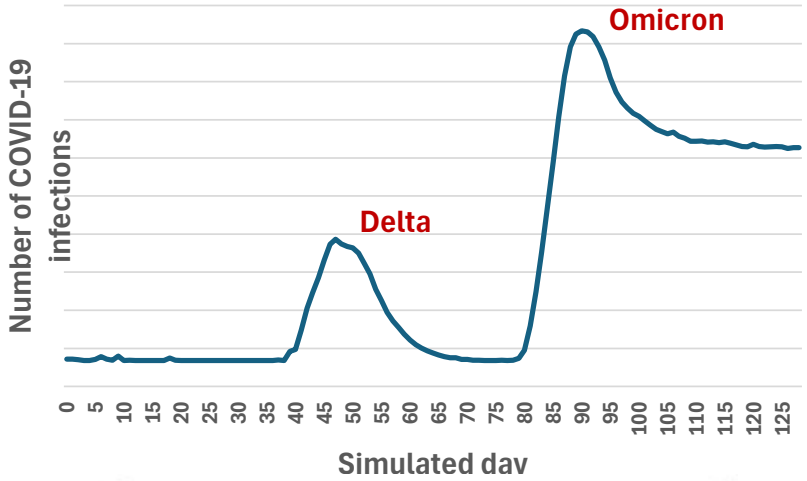
The Ugly: dynamic execution

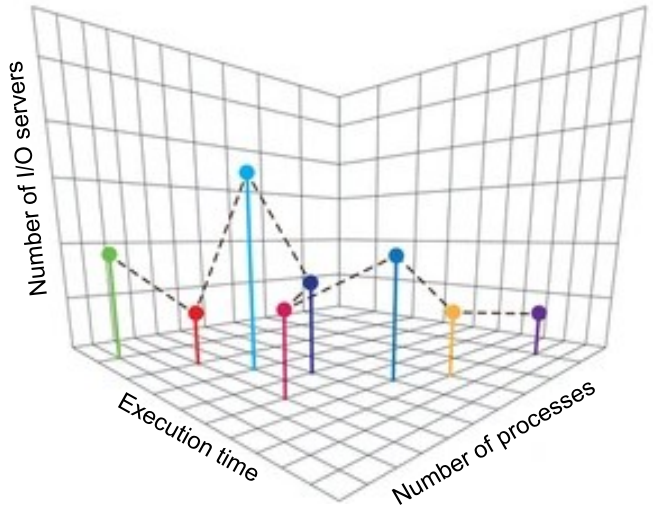




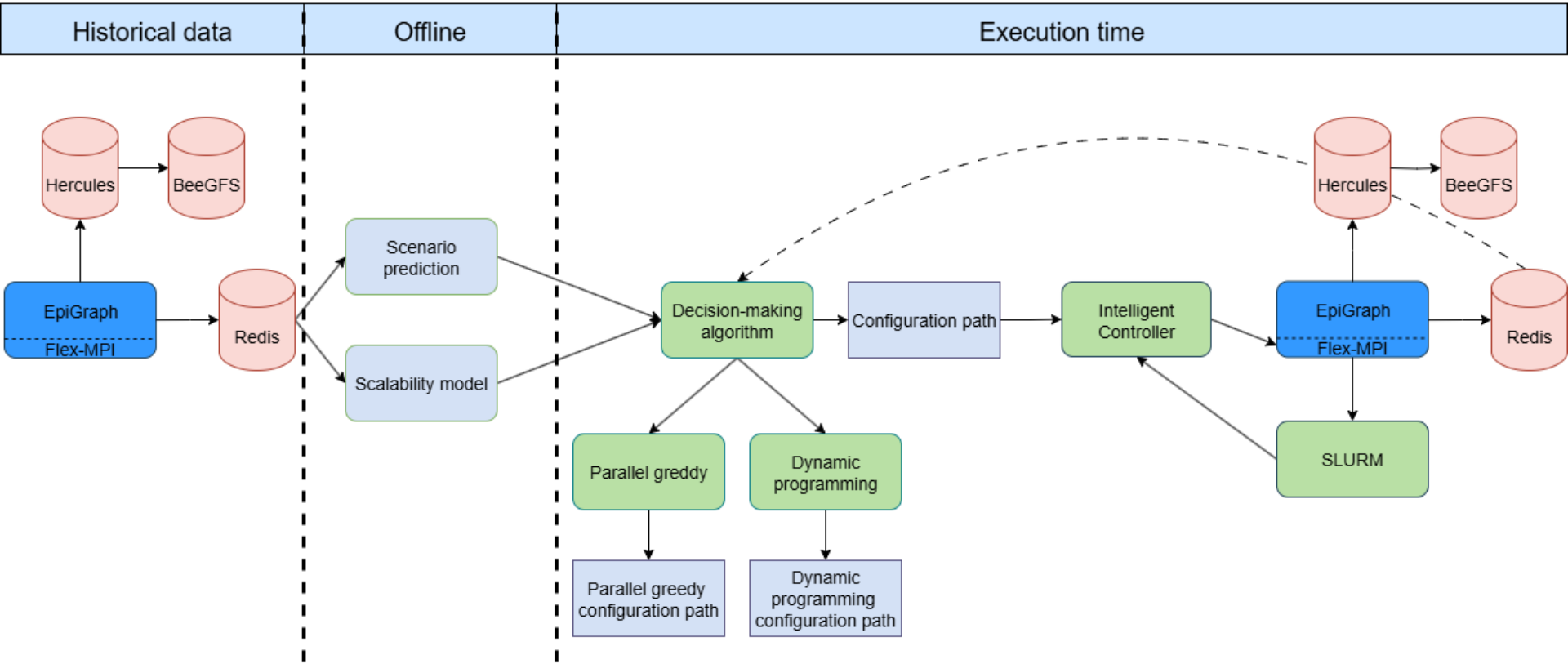
Challenges

- Identify the application behaviour in run-time
- Consider multiple performance metrics
  - CPU, I/O and communication time
  - Resource utilisation
  - Overheads
- Make the decision in run-time
  - Finding the best configuration path

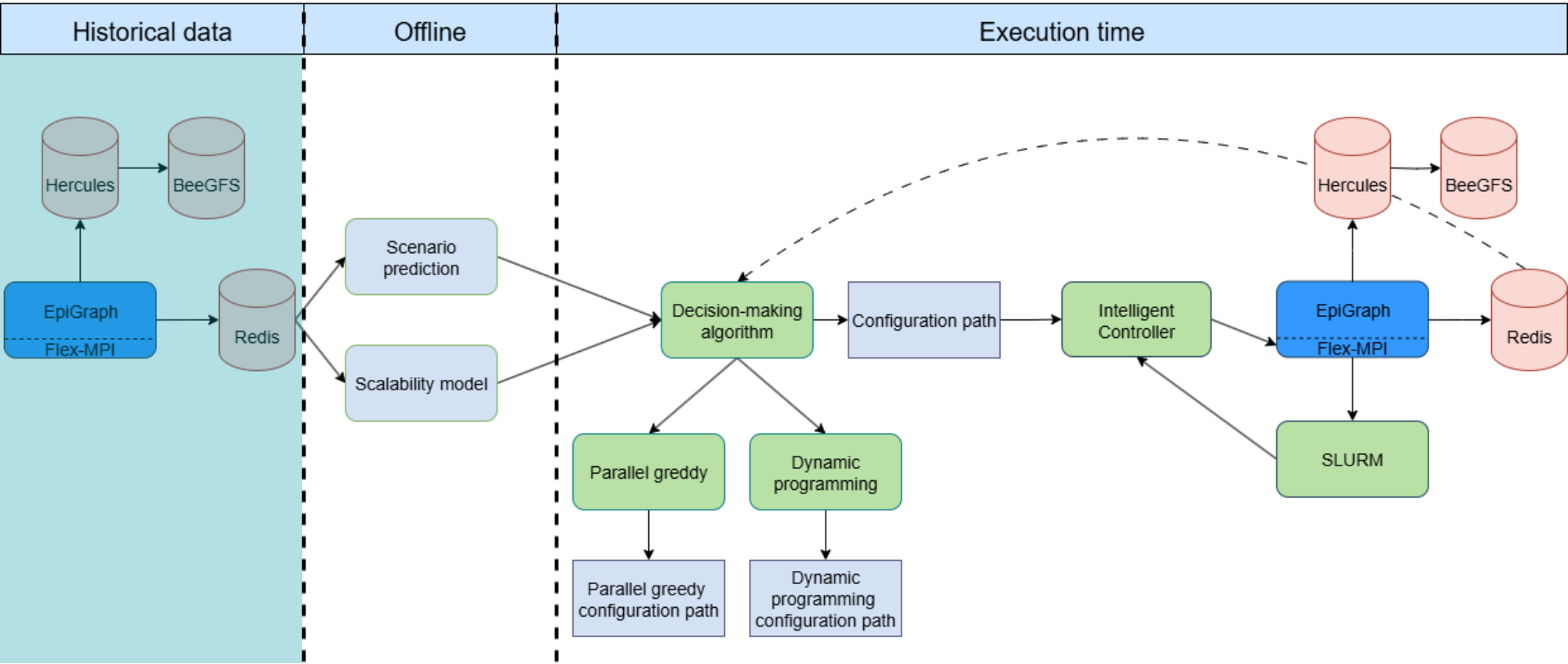




Proposed workflow



Proposed workflow

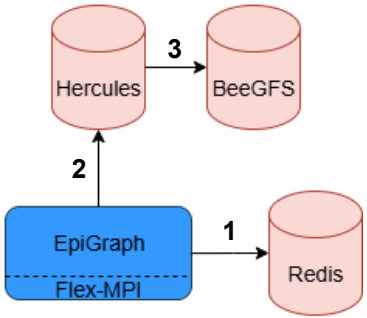


Workflow

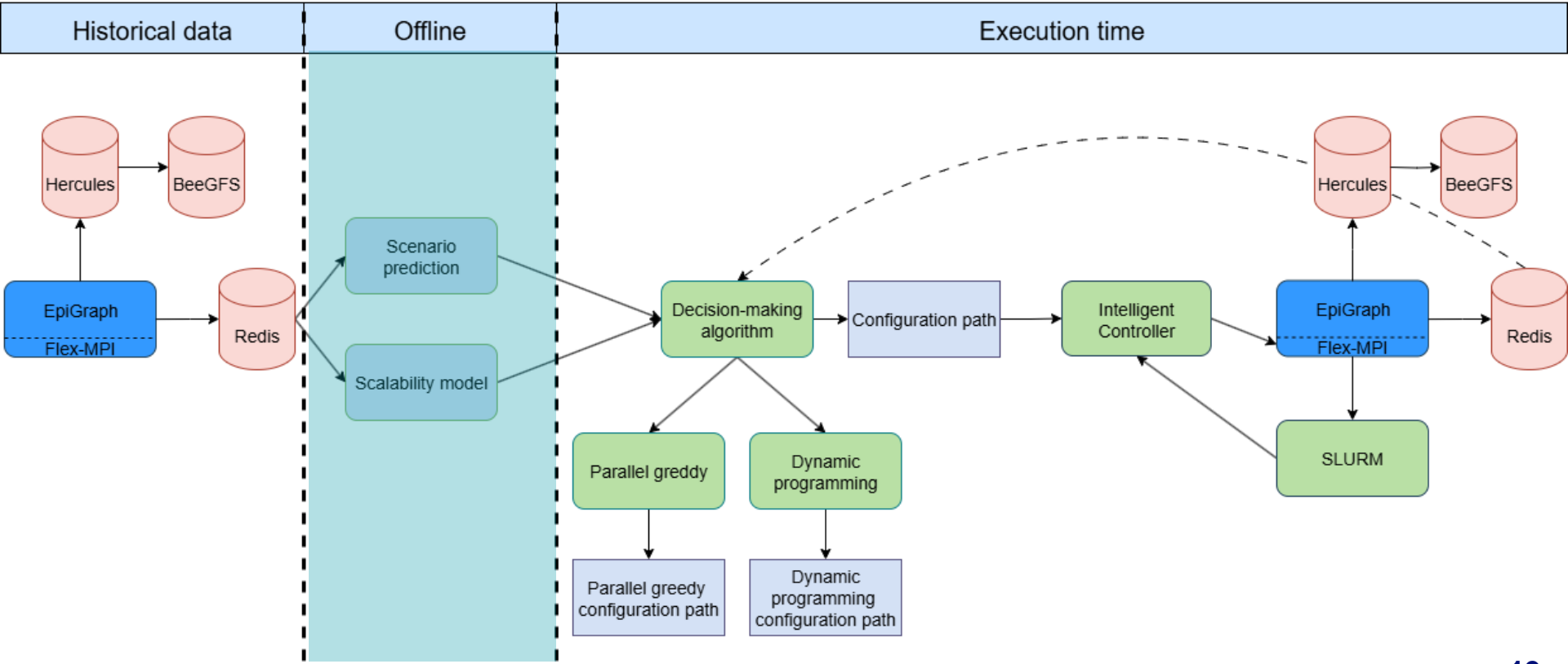
1. Application performance metrics.

2. I/O operations on Hercules.

3. Persistent storage of simulation results.



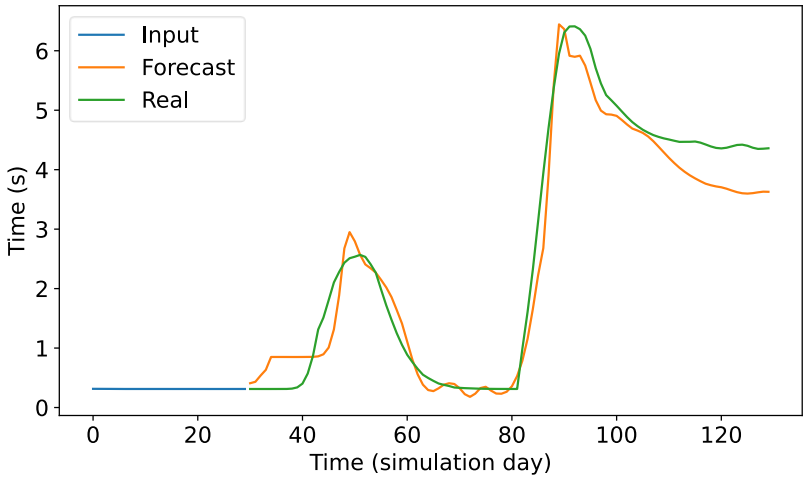
Proposed workflow



Scenario prediction

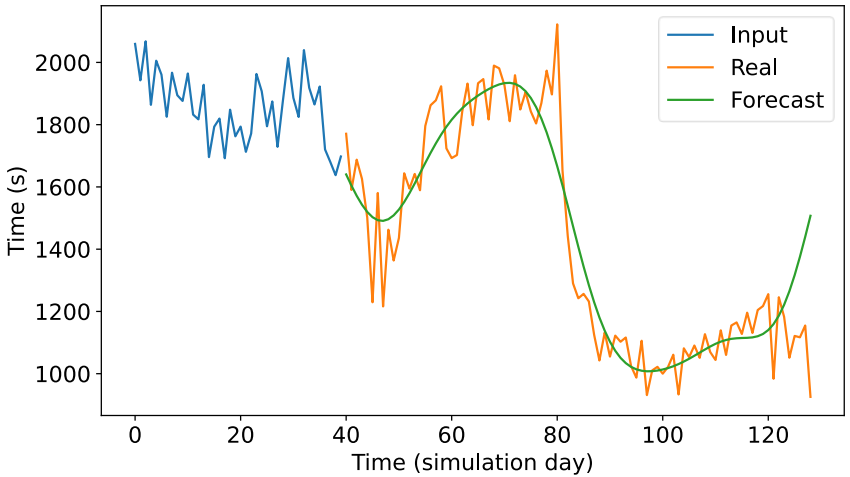
CPU time

- Long Short-Term Memory NN
- RMSE: 5,5



Communication time

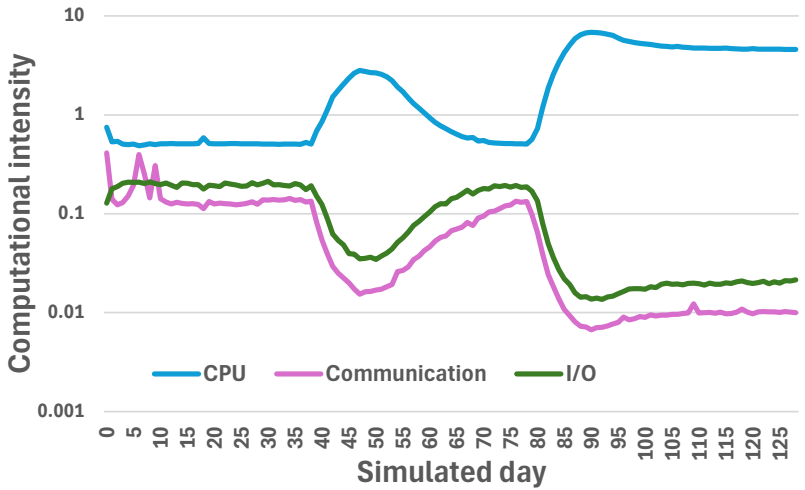
- Low-pass filter
- RMSE: 1,3



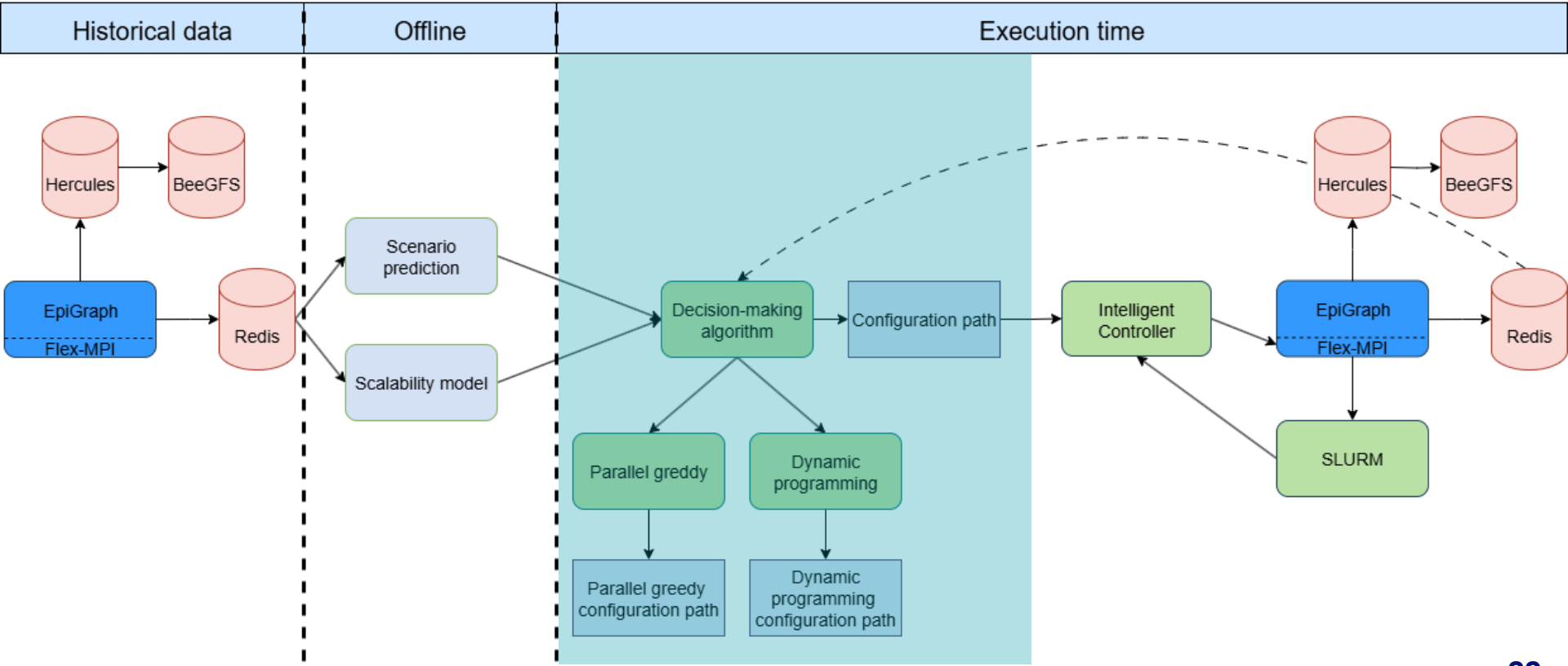
Scalability model

- Performance metrics related to different configurations
  - Number of processes (EpiGraph)
  - Number of I/O nodes (Hercules)

- Numerical fitting (like Extra-P)
  - Considering multiple configuration paths
  - Considering different EpiGraph execution phases

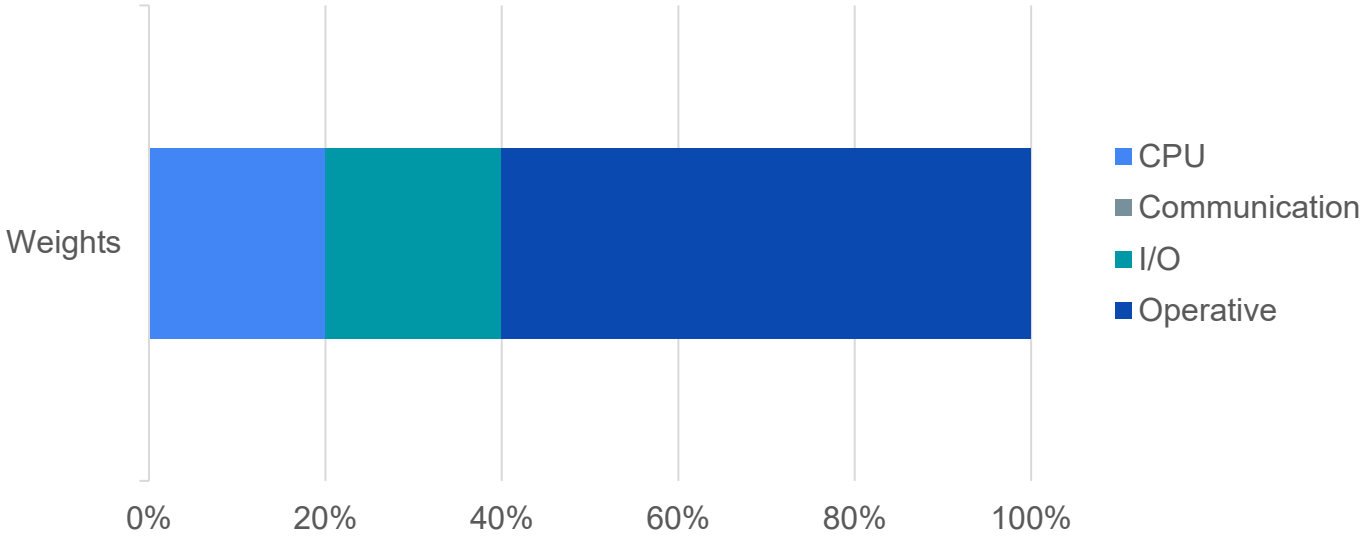


Proposed workflow



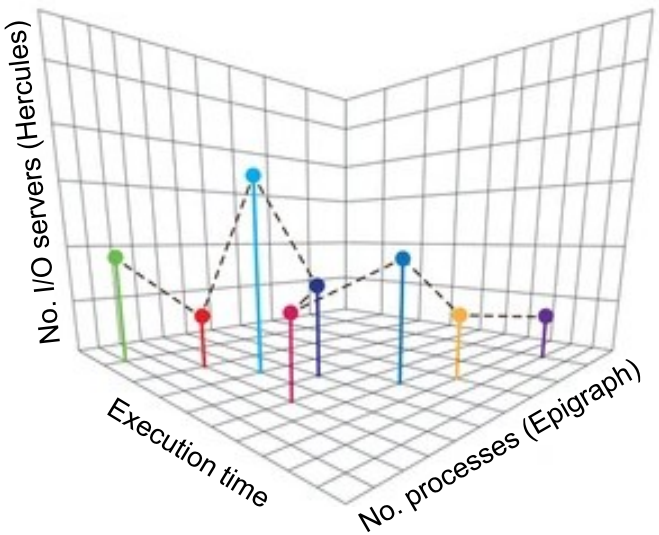
Optimization cost

$$\begin{cases} c_t^i = \alpha \cdot t_{cpu} + \beta \cdot t_{com} + \gamma \cdot t_{io} + \delta \cdot t_{op} \\ \alpha + \beta + \gamma + \delta = 1 \end{cases}$$

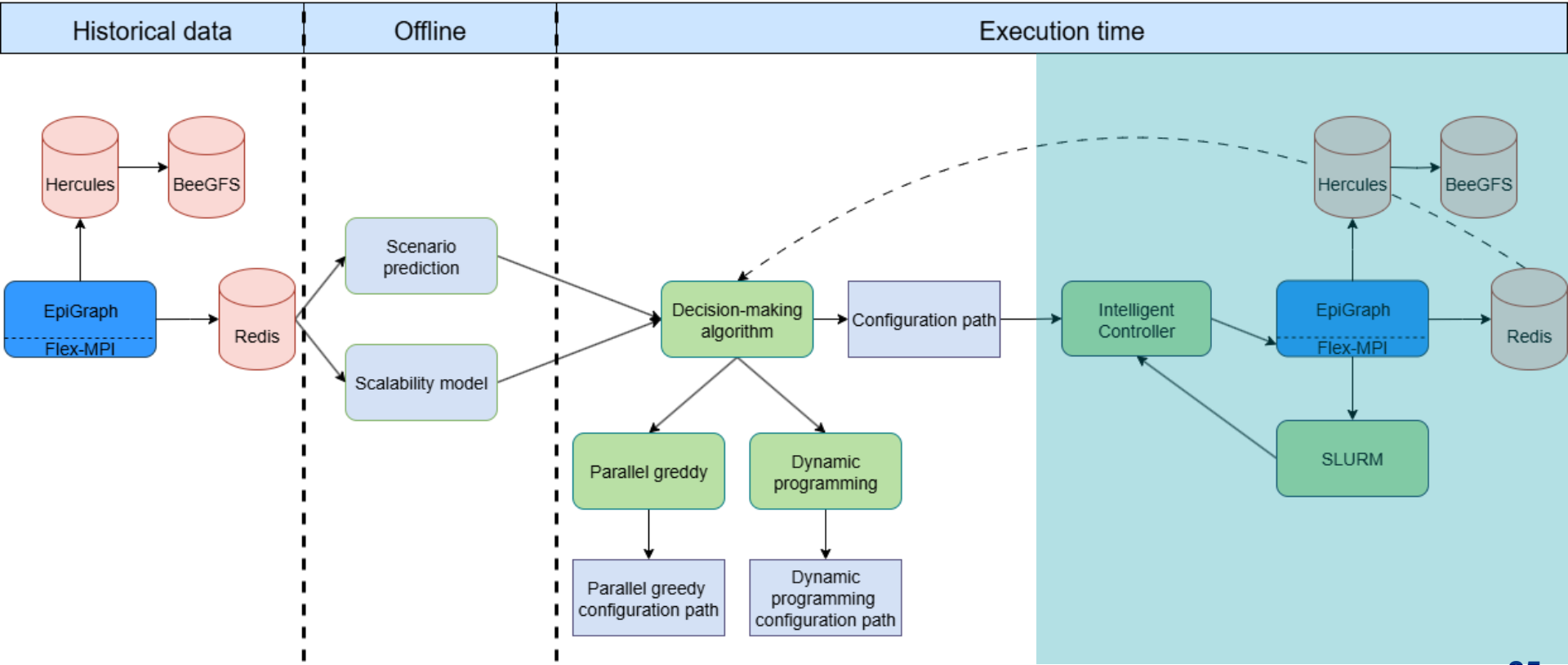


Configuration path

- Two algorithms
  - Parallel greedy algorithm
  - Dynamic programming search



Proposed workflow



Evaluation

- HPC4AI Laboratory, Universidad di Torino

Id	Initial configuration	(α,	β,	γ,	δ)	Execution time (s)	Operational time (s)	Reconfigurations
1	**(1, 2)**	**(-,**	**-,**	**-,**	**-)**	**507**	**2.447**	———
2	(1, 2)	(1,	0,	0,	0)	338	3.000	$(1, 2) \xrightarrow{40} (8, 4)$
3	(1, 2)	(0,	0,	1,	0)	338	3.000	$(1, 2) \xrightarrow{40} (8, 4)$
4	(1, 2)	(0.4,	0,	0.4,	0.2)	338	3.000	$(1, 2) \xrightarrow{40} (8, 4)$
5	(1, 2)	(0.2,	0,	0.2,	0.6)	325	1.651	$(1, 2) \xrightarrow{40} (5, 4) \xrightarrow{84} (5, 2)$
6	(1, 2)	(0.1,	0,	0.1,	0.8)	294	1.547	$(1, 2) \xrightarrow{41} (5, 2)$
7	**(2, 4)**	**(-,**	**-,**	**-,**	**-)**	**483**	**2.835**	———
8	(2, 4)	(1,	0,	0,	0)	329	3.145	$(2, 4) \xrightarrow{40} (8, 4)$
9	(2, 4)	(0,	0,	1,	0)	329	3.145	$(2, 4) \xrightarrow{40} (8, 4)$
10	(2, 4)	(0.4,	0,	0.4,	0.2)	329	3.145	$(2, 4) \xrightarrow{40} (8, 4)$
11	(2, 4)	(0.2,	0,	0.2,	0.6)	321	1.747	$(2, 4) \xrightarrow{42} (5, 4) \xrightarrow{84} (5, 2)$
12	(2, 4)	(0.1,	0,	0.1,	0.8)	305	1.633	$(2, 4) \xrightarrow{42} (3, 4) \xrightarrow{84} (5, 2)$
13	**(8, 4)**	**(-,**	**-,**	**-,**	**-)**	**379**	**4.210**	———
14	(8, 4)	(1,	0,	0,	0)	379	4.210	(8, 4)
15	(8, 4)	(0,	0,	1,	0)	379	4.210	(8, 4)
16	(8, 4)	(0.4,	0,	0.4,	0.2)	379	4.210	(8, 4)
17	(8, 4)	(0.2,	0,	0.2,	0.6)	383	2.472	$(8, 4) \xrightarrow{40} (5, 4) \xrightarrow{84} (5, 2)$
18	(8, 4)	(0.1,	0,	0.1,	0.8)	382	2.495	$(8, 4) \xrightarrow{40} (3, 4) \xrightarrow{82} (5, 2)$

Evaluation: minimizing I/O time

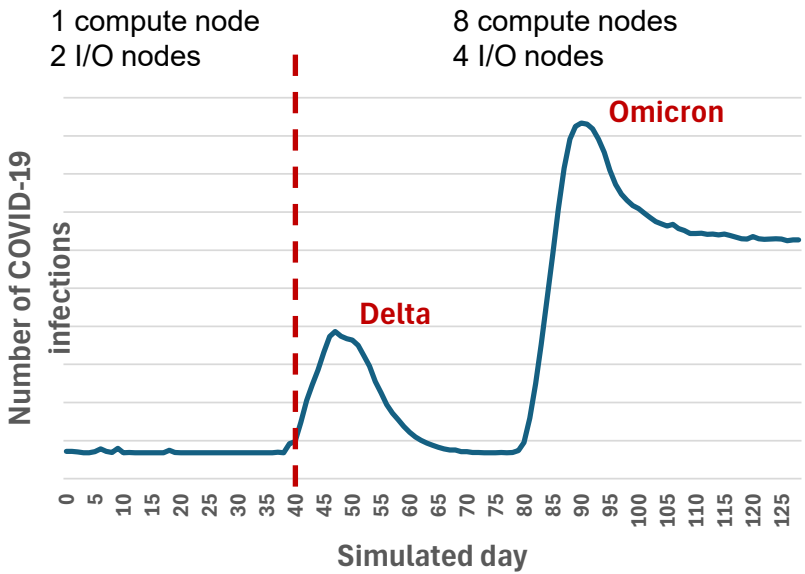
- HPC4AI Laboratory, Universidad di Torino

Id	Initial configuration	$(\alpha,$	$\beta,$	$\gamma,$	$\delta)$	Execution time (s)	Operational time (s)	Reconfigurations
1	**(1, 2)**	**(-,**	**-,**	**-,**	**-)**	**507**	**2.447**	——
2	(1, 2)	(1,	0,	0,	0)	338	3.000	$(1, 2) \xrightarrow{40} (8, 4)$
3	(1, 2)	(0,	0,	1,	0)	338	3.000	$(1, 2) \xrightarrow{40} (8, 4)$
4	(1, 2)	(0.4,	0,	0.4,	0.2)	338	3.000	$(1, 2) \xrightarrow{40} (8, 4)$
5	(1, 2)	(0.2,	0,	0.2,	0.6)	325	1.651	$(1, 2) \xrightarrow{40} (5, 4) \xrightarrow{84} (5, 2)$
6	(1, 2)	(0.1,	0,	0.1,	0.8)	294	1.547	$(1, 2) \xrightarrow{41} (5, 2)$
7	**(2, 4)**	**(-,**	**-,**	**-,**	**-)**	**483**	**2.835**	——
8	(2, 4)	(1,	0,	0,	0)	329	3.145	$(2, 4) \xrightarrow{40} (8, 4)$
9	(2, 4)	(0,	0,	1,	0)	329	3.145	$(2, 4) \xrightarrow{40} (8, 4)$
10	(2, 4)	(0.4,	0,	0.4,	0.2)	329	3.145	$(2, 4) \xrightarrow{40} (8, 4)$
11	(2, 4)	(0.2,	0,	0.2,	0.6)	321	1.747	$(2, 4) \xrightarrow{42} (5, 4) \xrightarrow{84} (5, 2)$
12	(2, 4)	(0.1,	0,	0.1,	0.8)	305	1.633	$(2, 4) \xrightarrow{42} (3, 4) \xrightarrow{84} (5, 2)$
13	**(8, 4)**	**(-,**	**-,**	**-,**	**-)**	**379**	**4.210**	——
14	(8, 4)	(1,	0,	0,	0)	379	4.210	(8, 4)
15	(8, 4)	(0,	0,	1,	0)	379	4.210	(8, 4)
16	(8, 4)	(0.4,	0,	0.4,	0.2)	379	4.210	(8, 4)
17	(8, 4)	(0.2,	0,	0.2,	0.6)	383	2.472	$(8, 4) \xrightarrow{40} (5, 4) \xrightarrow{84} (5, 2)$
18	(8, 4)	(0.1,	0,	0.1,	0.8)	382	2.495	$(8, 4) \xrightarrow{40} (3, 4) \xrightarrow{82} (5, 2)$

Evaluation: minimizing I/O time

$$(1,\ 2) \xrightarrow{40} (8,\ 4)$$

	Static execution	Dynamic execution
Execution time	507 s	338 s
I/O time	21s	12s
Operative time	2.447 s	3.000 s



Evaluation: balance CPU, I/O and OT

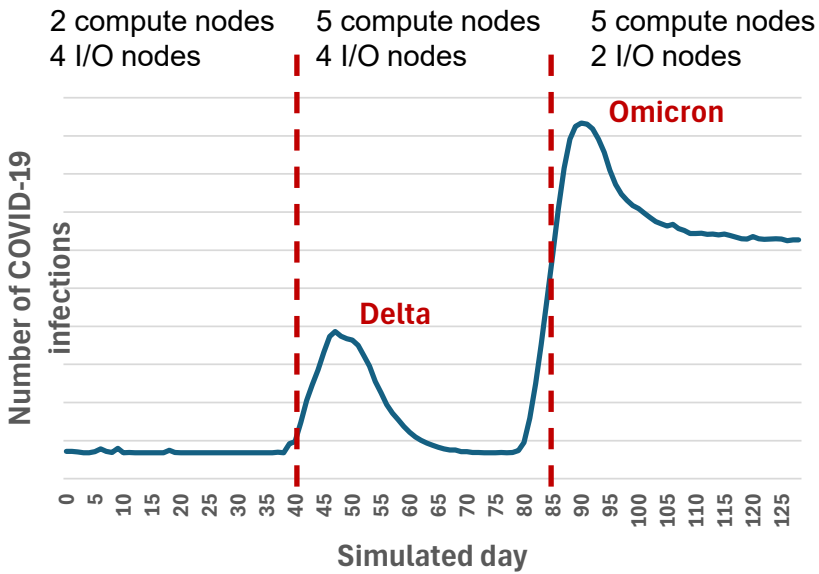
- HPC4AI Laboratory, Universidad di Torino

Id	Initial configuration	(α,	β,	γ,	δ)	Execution time (s)	Operational time (s)	Reconfigurations
1	**(1, 2)**	**(-,**	**-,**	**-,**	**-)**	**507**	**2.447**	——
2	(1, 2)	(1,	0,	0,	0)	338	3.000	$(1, 2) \xrightarrow{40} (8, 4)$
3	(1, 2)	(0,	0,	1,	0)	338	3.000	$(1, 2) \xrightarrow{40} (8, 4)$
4	(1, 2)	(0.4,	0,	0.4,	0.2)	338	3.000	$(1, 2) \xrightarrow{40} (8, 4)$
5	(1, 2)	(0.2,	0,	0.2,	0.6)	325	1.651	$(1, 2) \xrightarrow{40} (5, 4) \xrightarrow{84} (5, 2)$
6	(1, 2)	(0.1,	0,	0.1,	0.8)	294	1.547	$(1, 2) \xrightarrow{41} (5, 2)$
7	**(2, 4)**	**(-,**	**-,**	**-,**	**-)**	**483**	**2.835**	——
8	(2, 4)	(1,	0,	0,	0)	329	3.145	$(2, 4) \xrightarrow{40} (8, 4)$
9	(2, 4)	(0,	0,	1,	0)	329	3.145	$(2, 4) \xrightarrow{40} (8, 4)$
10	(2, 4)	(0.4,	0,	0.4,	0.2)	329	3.145	$(2, 4) \xrightarrow{40} (8, 4)$
11	(2, 4)	(0.2,	0,	0.2,	0.6)	321	1.747	$(2, 4) \xrightarrow{42} (5, 4) \xrightarrow{84} (5, 2)$
12	(2, 4)	(0.1,	0,	0.1,	0.8)	305	1.633	$(2, 4) \xrightarrow{42} (3, 4) \xrightarrow{84} (5, 2)$
13	**(8, 4)**	**(-,**	**-,**	**-,**	**-)**	**379**	**4.210**	——
14	(8, 4)	(1,	0,	0,	0)	379	4.210	(8, 4)
15	(8, 4)	(0,	0,	1,	0)	379	4.210	(8, 4)
16	(8, 4)	(0.4,	0,	0.4,	0.2)	379	4.210	(8, 4)
17	(8, 4)	(0.2,	0,	0.2,	0.6)	383	2.472	$(8, 4) \xrightarrow{40} (5, 4) \xrightarrow{84} (5, 2)$
18	(8, 4)	(0.1,	0,	0.1,	0.8)	382	2.495	$(8, 4) \xrightarrow{40} (3, 4) \xrightarrow{82} (5, 2)$

Evaluation: balance CPU, I/O and OT

$$(2,\ 4) \xrightarrow{42} (5,\ 4) \xrightarrow{84} (5,\ 2)$$

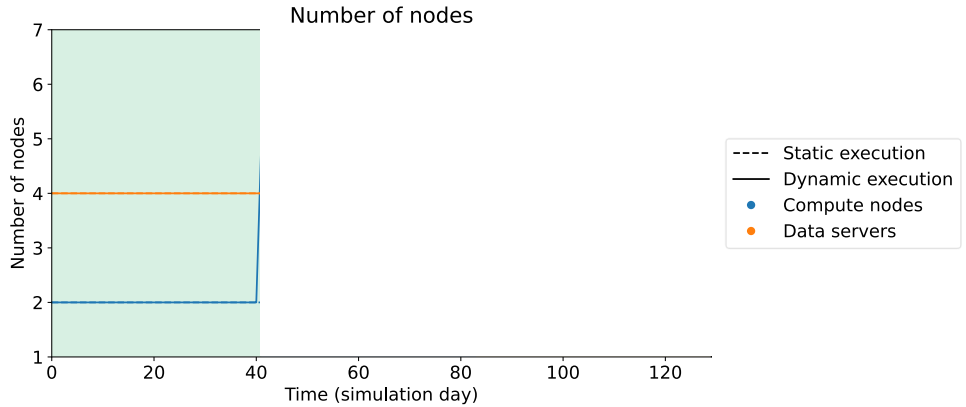
	Static execution	Dynamic execution
Execution time	483 s	321 s
Operative time	2.835 s	1.747 s

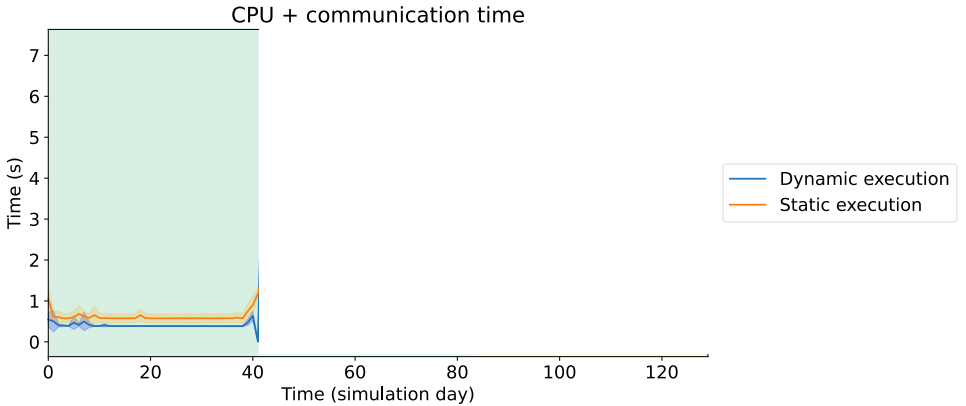


Evaluation

CPU vs communication times

$$(2,\ 4) \xrightarrow{42} (5,\ 4) \xrightarrow{84} (5,\ 2)$$

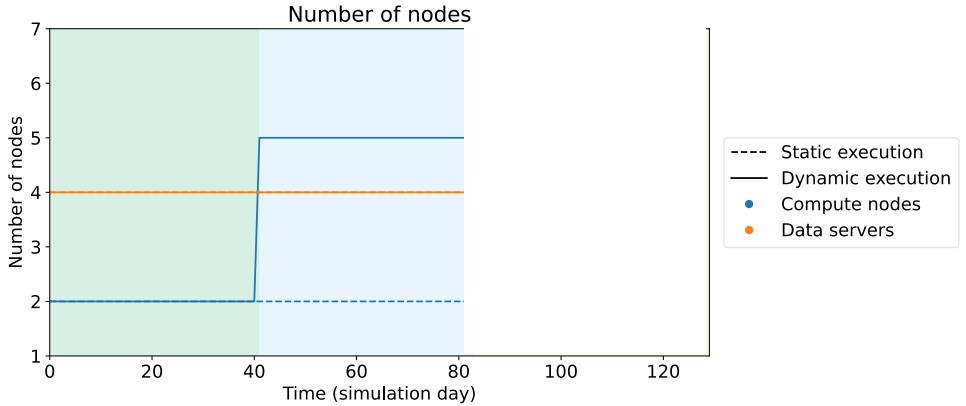


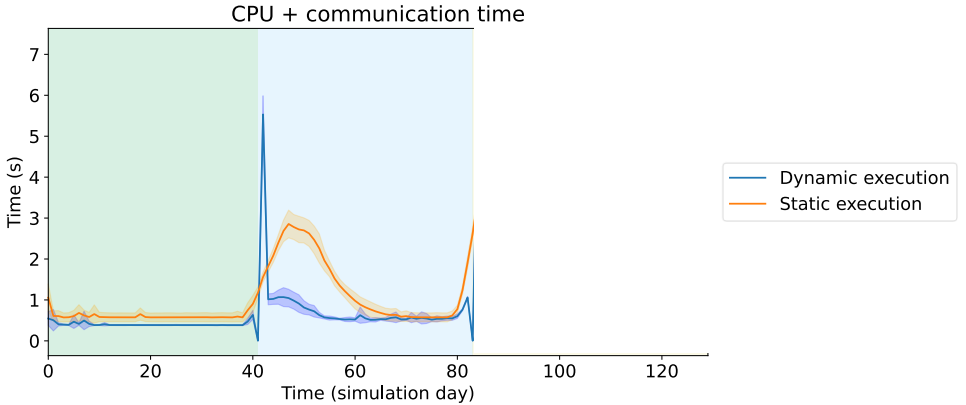


Evaluation

CPU vs communication times

$$(2,\ 4) \xrightarrow{42} (5,\ 4) \xrightarrow{84} (5,\ 2)$$

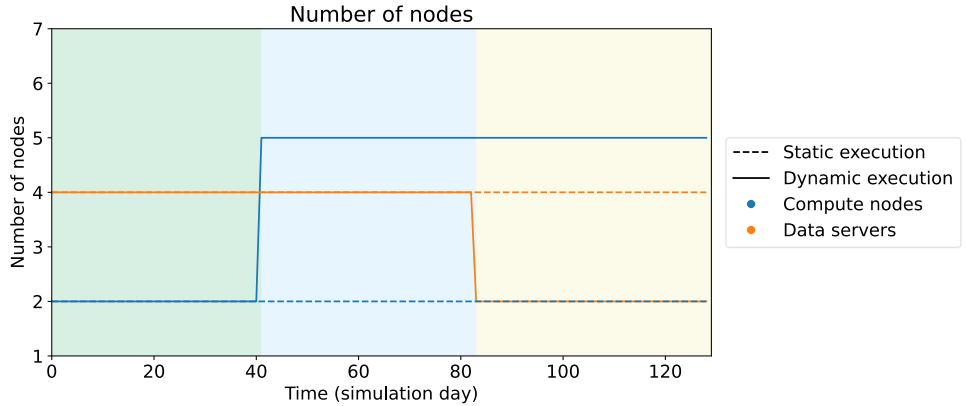


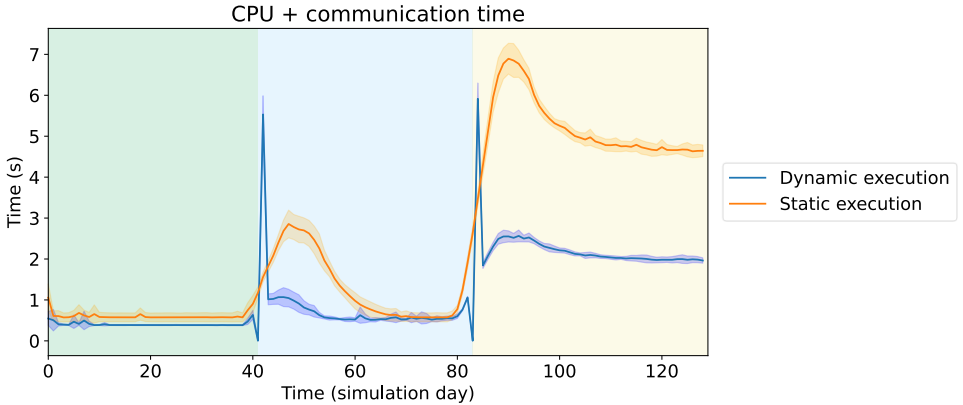


Evaluation

CPU vs communication times

$$(2,\ 4) \xrightarrow{42} (5,\ 4) \xrightarrow{84} (5,\ 2)$$

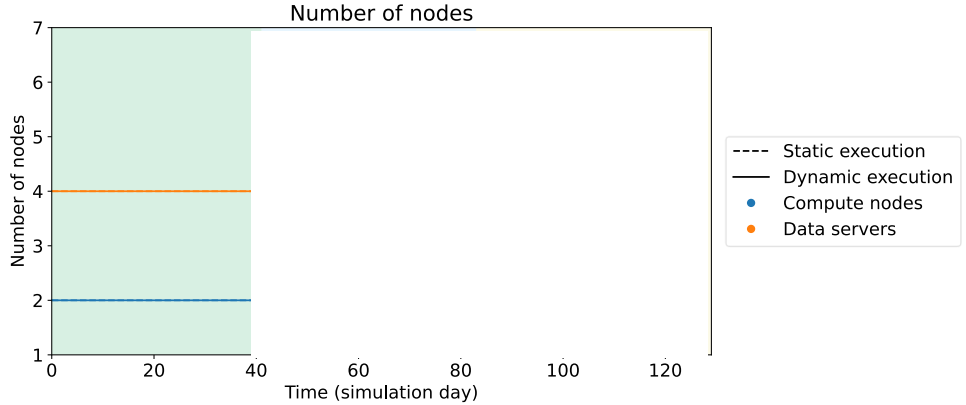


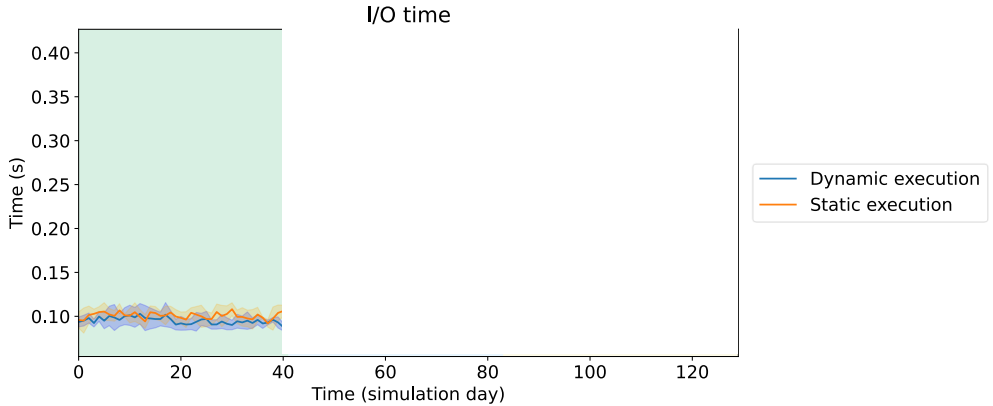


Evaluation

CPU vs I/O time

$$(2,\ 4) \xrightarrow{42} (5,\ 4) \xrightarrow{84} (5,\ 2)$$

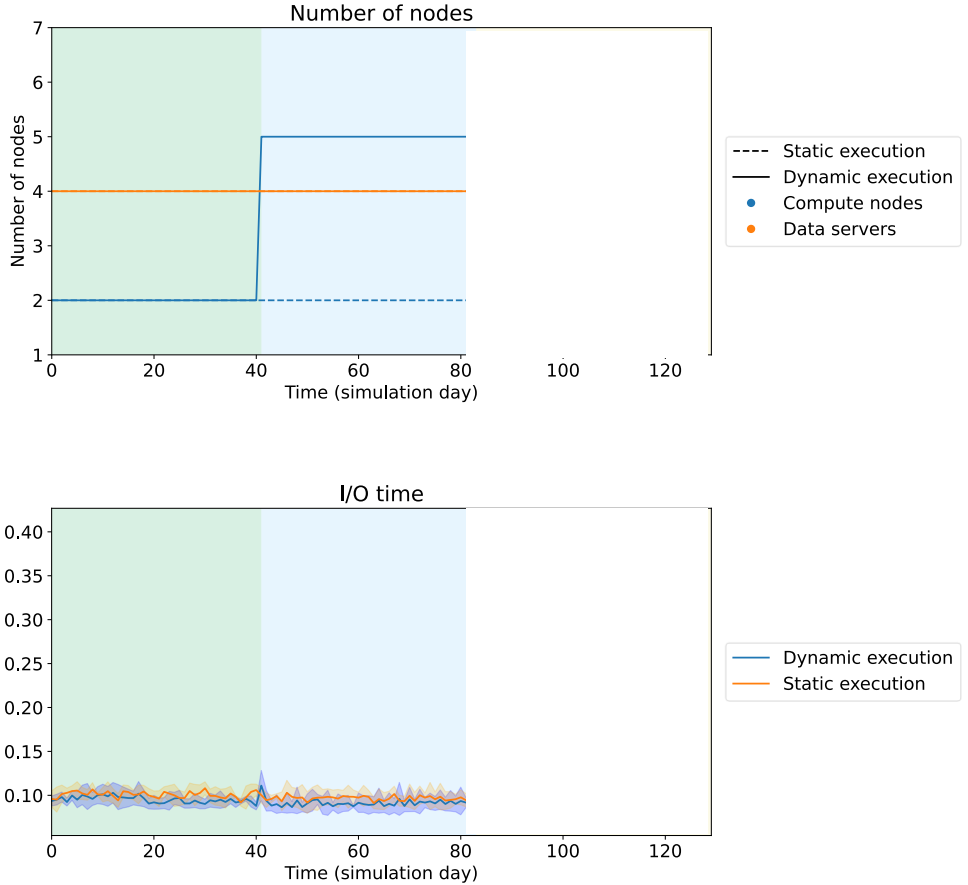




Evaluation

CPU vs I/O time

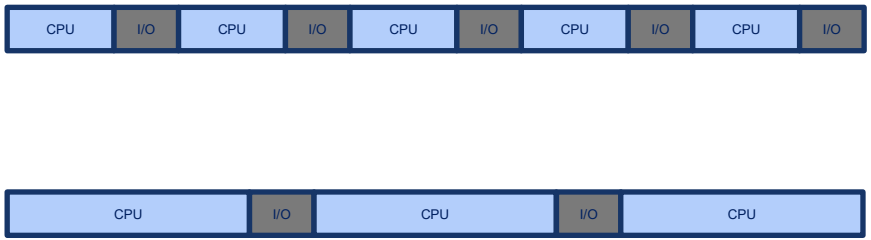
$$(2,\ 4) \xrightarrow{42} (5,\ 4) \xrightarrow{84} (5,\ 2)$$

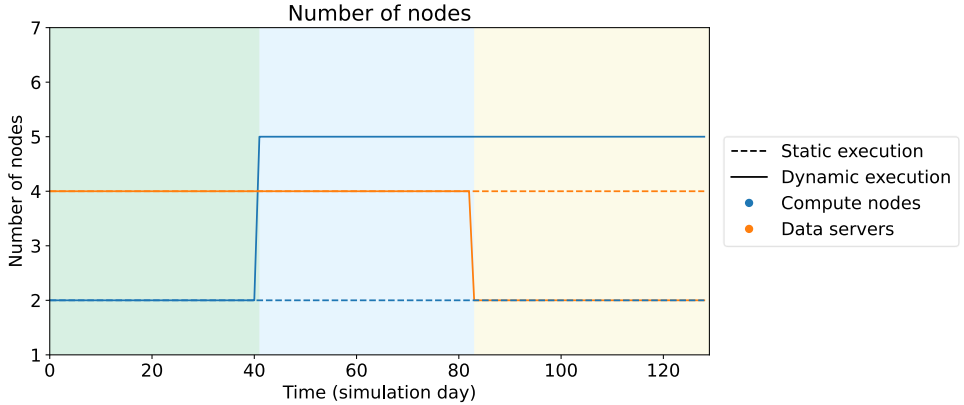


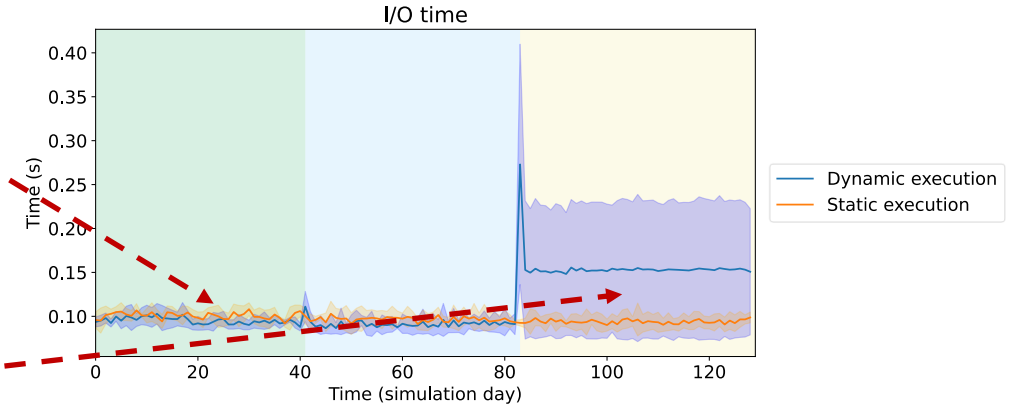
Evaluation

CPU vs I/O time

$$(2,\ 4) \xrightarrow{42} (5,\ 4) \xrightarrow{84} (5,\ 2)$$



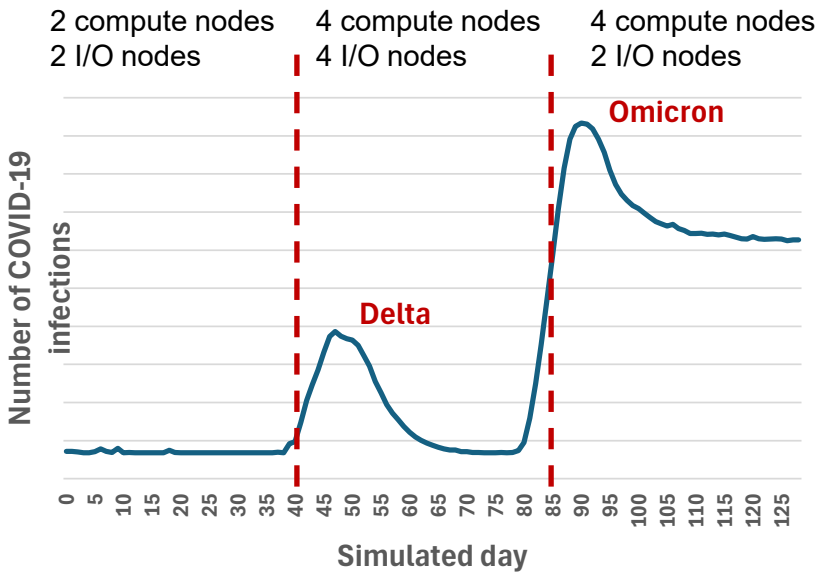




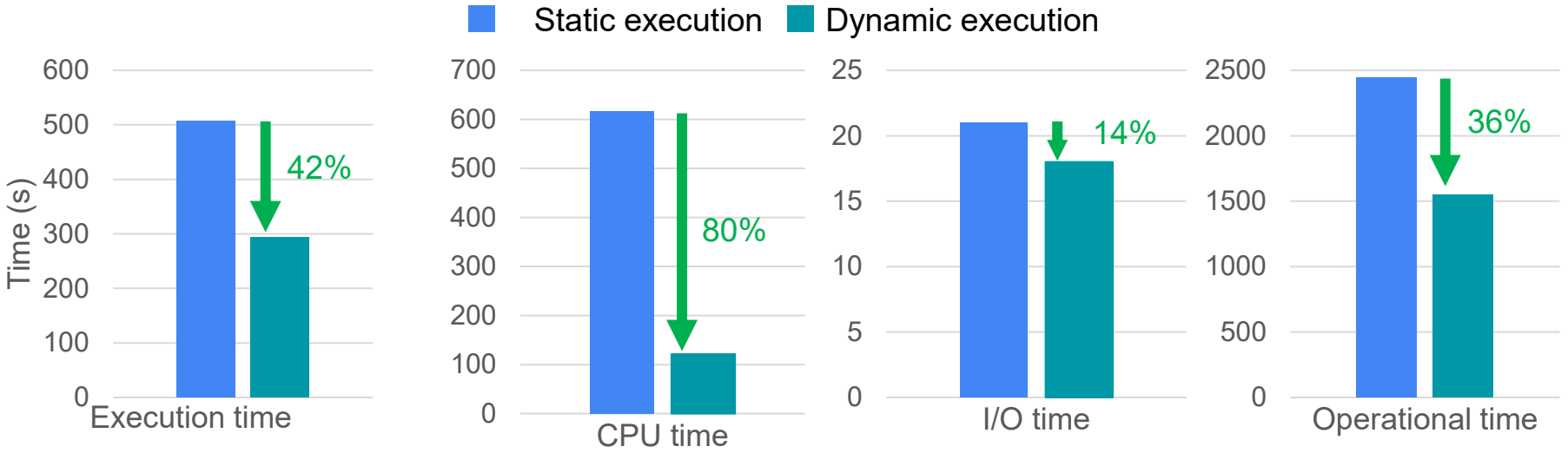
Evaluation: balance CPU, I/O and OT

$$(2,\ 4) \xrightarrow{42} (5,\ 4) \xrightarrow{84} (5,\ 2)$$

	Static execution	Dynamic execution
Execution time	483 s	321 s
Operative time	2.835 s	1.747 s



Evaluation


Static execution
Dynamic execution
Time (s)
600
500
400
300
200
100
0
42%
Execution time
700
600
500
400
300
200
100
0
80%
CPU time
25
20
15
10
5
0
14%
I/O time
2500
2000
1500
1000
500
0
36%
Operational time

Conclusion

- Optimizing applications with varying performance is a challenge.
- Dynamic resource management (CPU and I/O) adds additional dimensions to the optimization problem.
- There is significant room for improvement!
- The approach can be extended to other applications and file systems.

Credits (in order of appearance)

- Hercules. https://gitlab.arcos.inf.uc3m.es/admire/hercules
- EpiGraph. epigraph.uc3m.es
- FlexMPI. https://github.com/admire-eurohpc/ic
- Expand. https://www.uc3m.es/ss/Satellite/GruposInvestigacion/en/TextoDosColumnas/1371352039366/
- Sánchez-Checa, P., Sánchez-Gallegos, G., Garcia-Blas, J., Carretero, J., Singh, D.E. Comparative *Analysis of Algorithms for Malleability Decision-Making in Applications and File Systems*. **Euro-Par 2025: Parallel Processing Workshops**. 2025
- Sanchez-Checa, P., Garcia-Blas, J., Carretero, J., and Singh, D.E., *Combining Malleability and Distributed Control Mechanisms to Reduce I/O Contention*. **In High Performance Computing. ISC High Performance 2025 International Workshops**. 2025