Adaptive Request Scheduling for the I/O Forwarding Layer

Francieli Zanon Boito

Inria Grenoble

Jean Bez and Philippe Navaux

Federal University of Rio Grande do Sul (Brazil)

Ramon Nou, Alberto Miranda and Toni Cortes

Barcelona Supercomputing Center

Motivation

- **Parallel I/O** is a challenge for HPC
- Decades of research into optimization techniques
  - MPI collective I/O, reordering aggregation techniques, alignment to stripe locks, I/O scheduling, …
- **Results depend** on the workload
  - They are hard to replicate
  - We may lose performance if we use the techniques for inadequate situations
  - Success also depends on the **right values for parameters**
- **This work:** apply a reinforcement learning technique to adapt

Summary

~~Motivation~~

The TWINS scheduling algorithm

Adaptive request scheduling

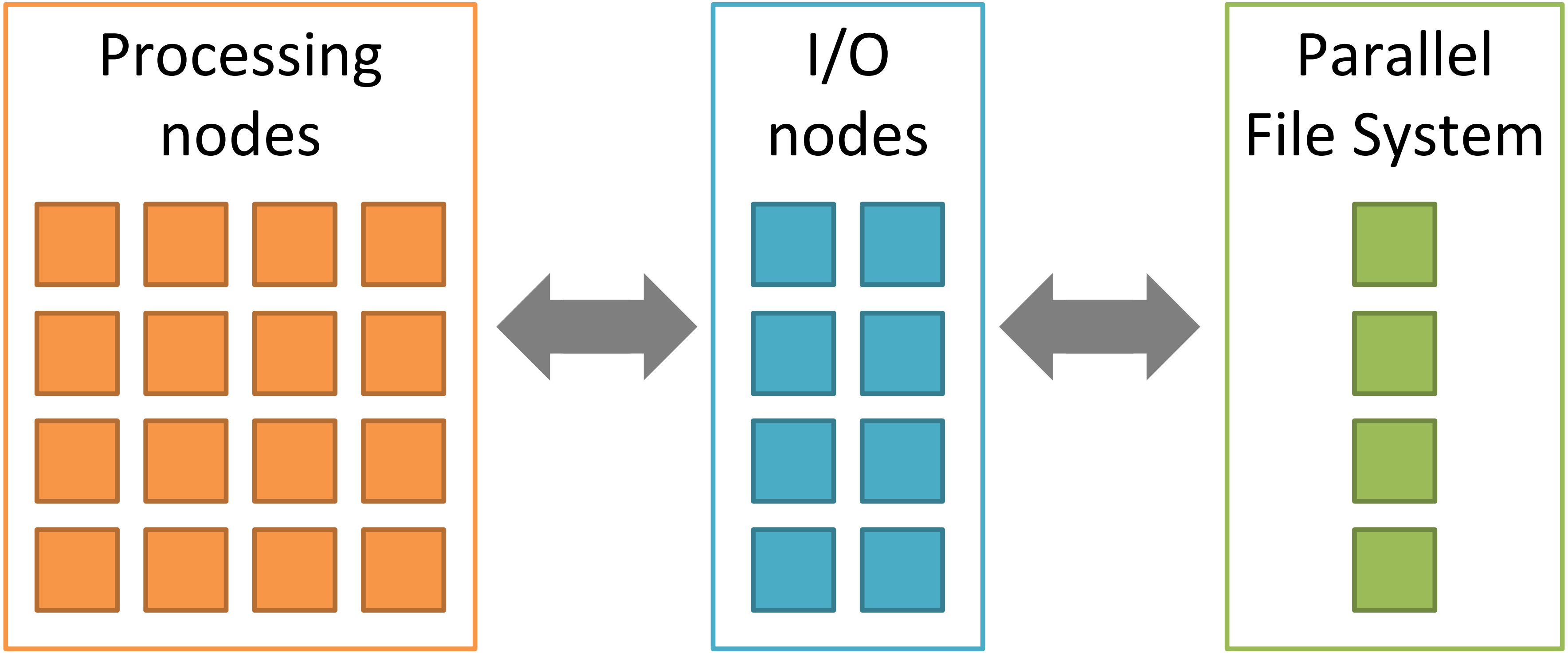
Results

Final remarks

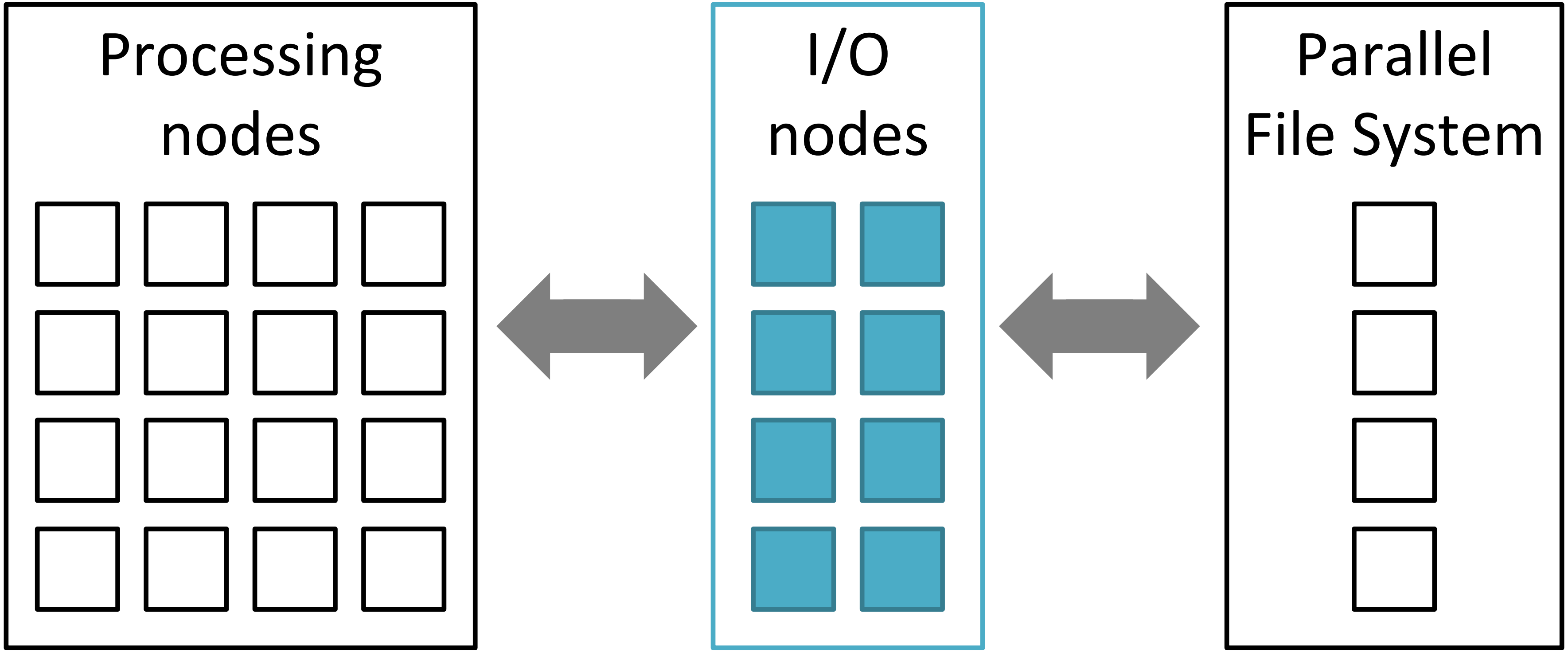
UFRGS
UNIVERSIDADE FEDERAL
DO RIO GRANDE DO SUL

BSC
Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

Inria
inventors for the digital world



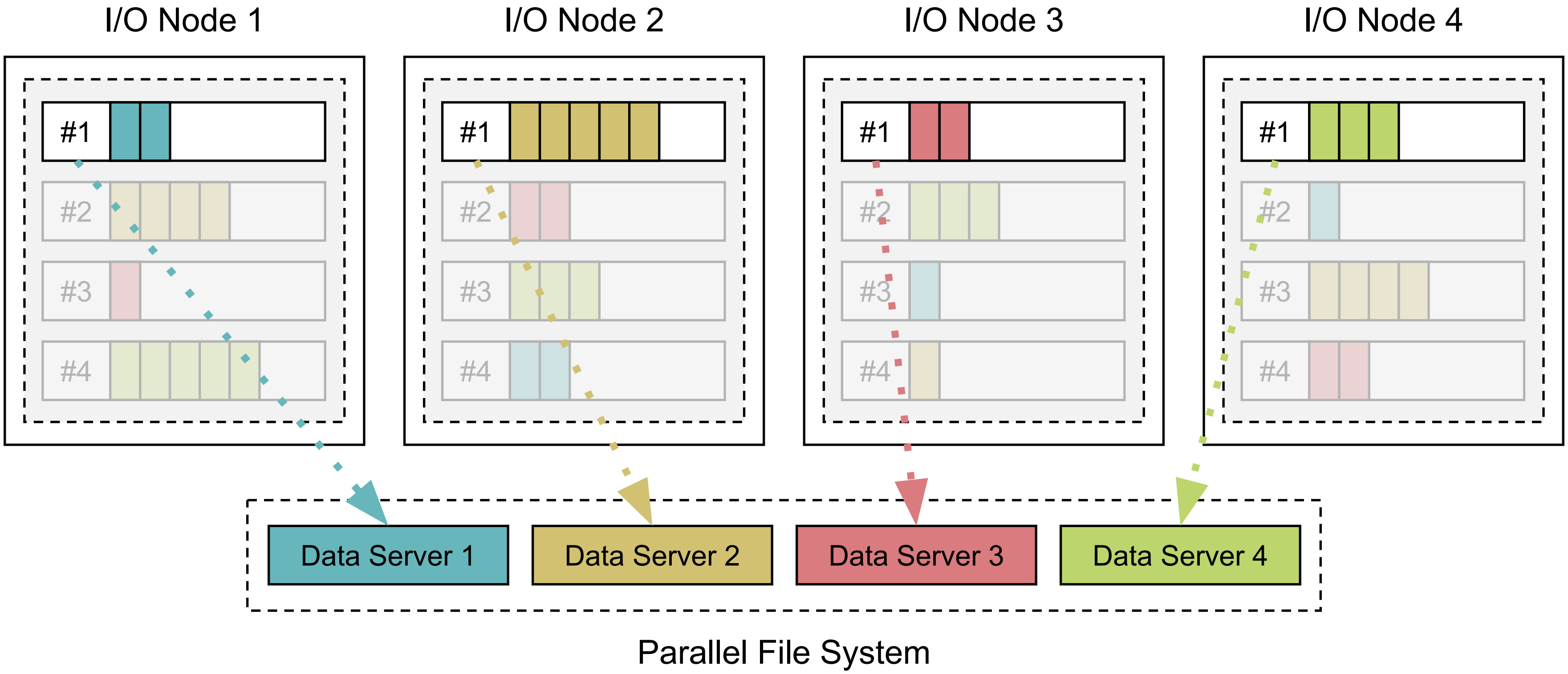
inventors for the digital world


Processing nodes
I/O nodes
Parallel File System

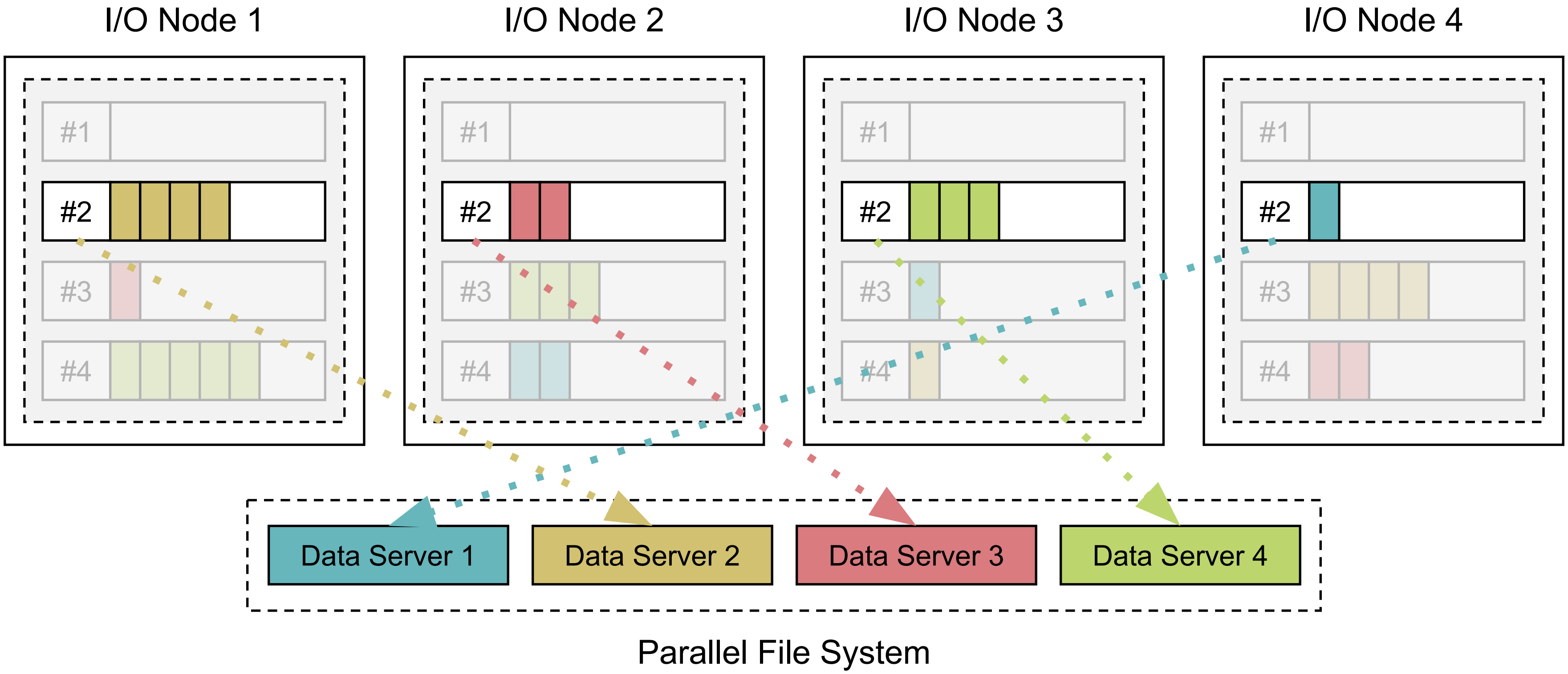
The TWINS scheduler

- One request queue per data server
- The I/O node only accesses one of the servers during each time window
- Different I/O nodes access the servers in different orders

J.Bez, F.Z.Boito et al., 2017, "TWINS: server access coordination in the I/O forwarding layer"

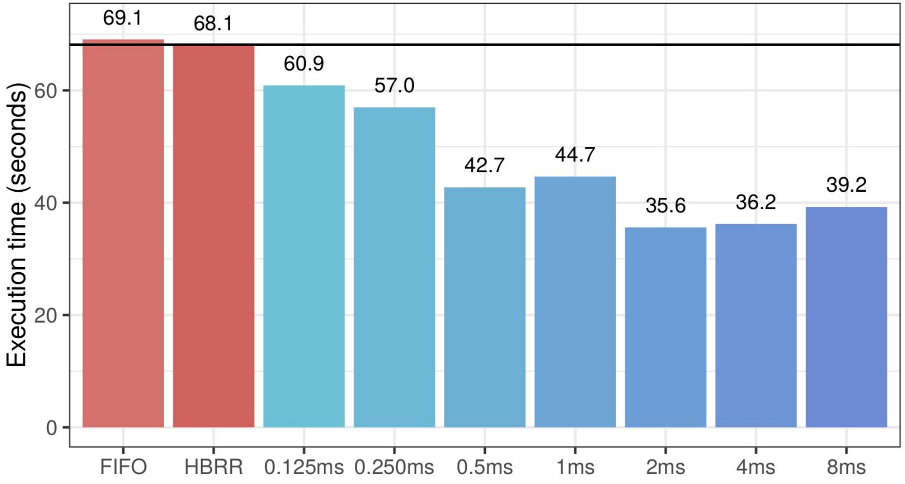


inventors for the digital world

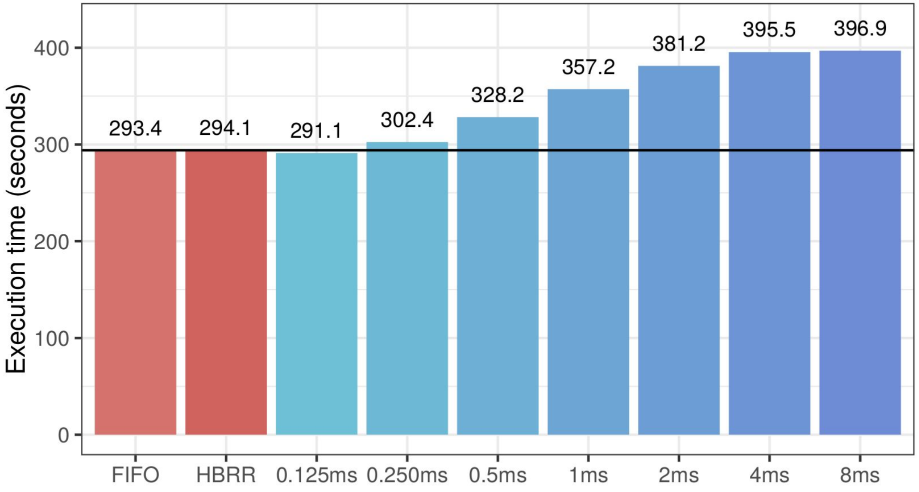


inventors for the digital world

TWINS results



1D-strided read through 8 I/O nodes



Contiguous write through 2 I/O nodes

128 processes, IOFSL forwarding, 4 PVFS servers, 4GB shared file, 32KB requests

Summary

~~Motivation~~

~~The TWINS scheduling algorithm~~

Adaptive request scheduling

Results

Final remarks

Reinforcement Learning approach

- We want to learn, but **without a long training phase**
- Approach: k-armed bandit problem
  - K possible actions, no prior information
  - take one action at each step, observe reward and update value estimates
- We need to learn one policy per access pattern (what is the best value)
  - **Contextual bandit** (associative search task)

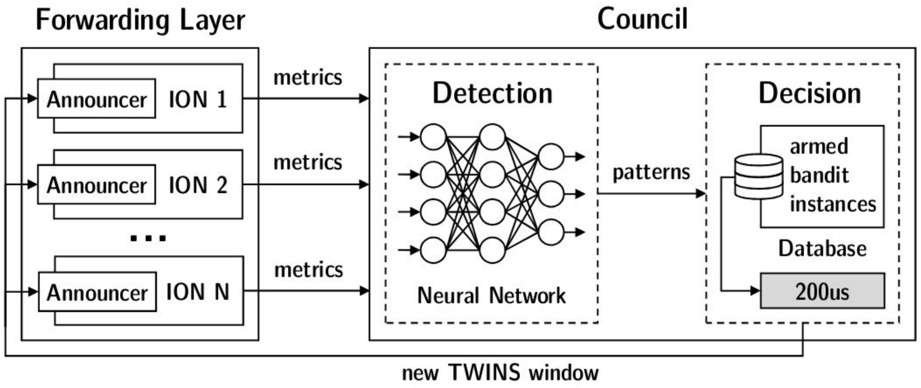
Approach

- Each armed bandit "instance": **ε-greedy algorithm**
- Value estimates are incrementally computed sample averages (reward is bandwidth)

$$Q_{t+1}(a) = Q_t(a) + \frac{1}{N(a)}[R - Q_t(a)]$$

- **Global decisions made by a council**
- Chose the value that is the best

  for most I/O nodes



Access pattern classification

- Context is defined by
  - Operation (read or write)
  - Number of files per process (N-to-1 or N-to-N)
  - Average request size
  - Spatiality (contiguous or strided) <- not readily available
- Use a neural network to detect **spatiality** from observed metrics
  - It has to be trained but with less experiments than for the whole thing

Summary

~~Motivation~~

~~The TWINS scheduling algorithm~~

~~Adaptive request scheduling~~

Results

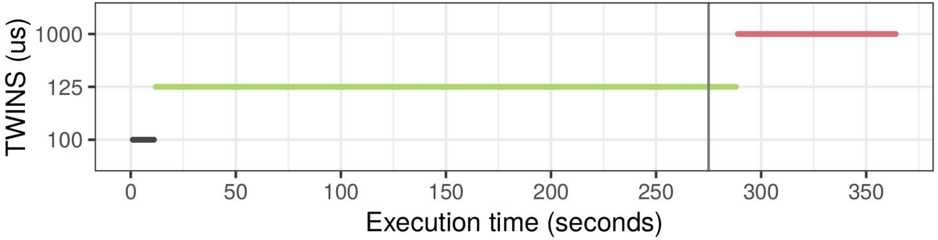
Final remarks

1 - Evaluate the access pattern detection

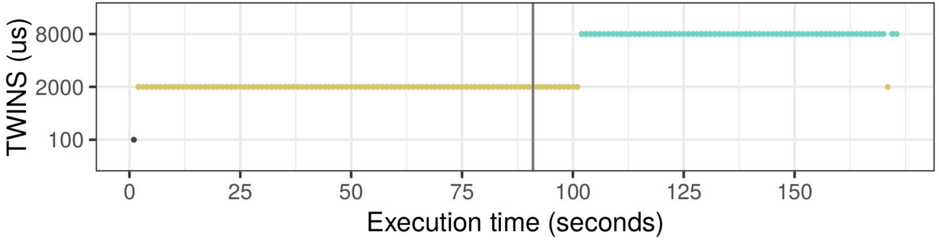
- Assume right context -> right decision
- Offline evaluation (with traces) - precision (%)

	Min	**Mean**	**Median**	**Max**
Read	98	99	100	100
Write	53	97	100	100

- Execution (the council is previously told what is the best window size)



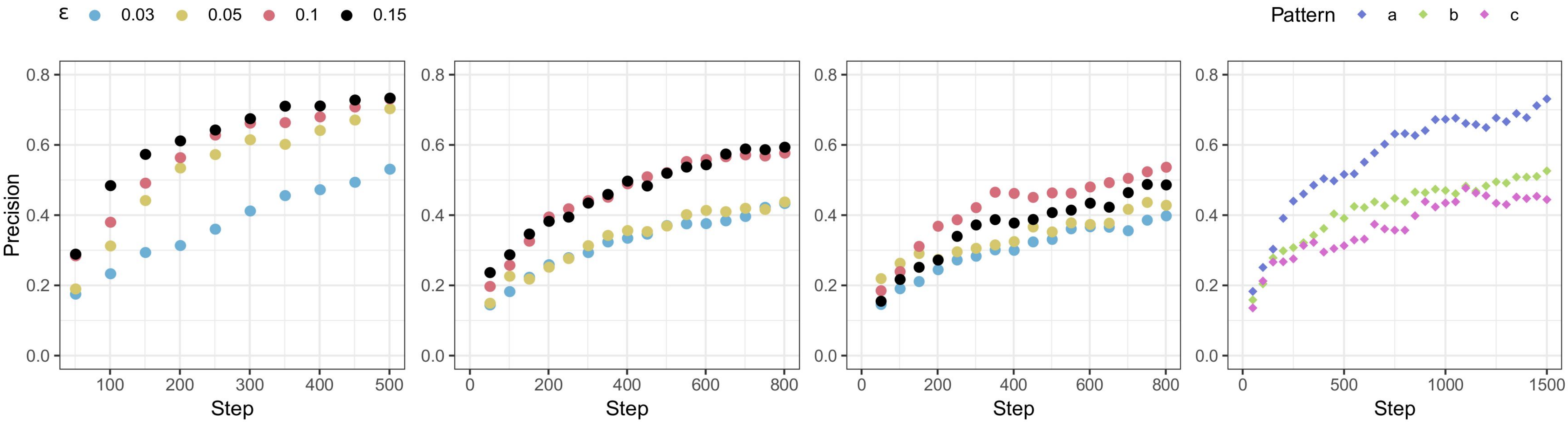
1 I/O node, file per process, contiguous 256KB requests



8 I/O nodes, shared file, contiguous, 256KB requests

2- Evaluate the learning

- Assume perfect access pattern detection
- Offline evaluation (with traces)



Pattern a - 128 procs, read, shared file, 8 I/O nodes, 32KB reqs, 1D-strided

Pattern b - 128 procs, write, shared file, 2 I/O nodes, 32KB reqs, contiguous

Pattern c - 512 procs, read, shared file, 8 I/O nodes, 32KB reqs, contiguous

ε=0.15

Summary

~~Motivation~~

~~The TWINS scheduling algorithm~~

~~Adaptive request scheduling~~

~~Results~~

Final remarks

Final Remarks

- Tuning optimization techniques and parameters is difficult (But important!)
- We used a reinforcement learning technique to **learn the best choices**
  - Tune the window size of the TWINS scheduler
  - Got to 0.98 of the best performance, ~70% of precision
- **This system has a long life!**
- Some caveats
  - Need to select a few good values for the parameter
    - Maybe we don't know the optimal, but we already have some improvement
    - Too many choices = slow learning
  - Need to know what are the parameters that define the context
  - Bandwidth as reward
- Now: investigating the trade-off between centralized decision and scalability

Final Remarks

This paper was submitted and is being reviewed

Available at https://hal.inria.fr/hal-01994677