CLARISSE: A run-time middleware for coordinating data staging on large scale supercomputers

Florin Isaila

ANL & University Carlos III

Collaborators: Phil Carns (ANL), Jesus Carretero (UC3M), Javier Garcia (UC3M), Kevin Harms (ANL), Rob Latham(ANL), Tom Peterka (ANL), Rob Ross (ANL)

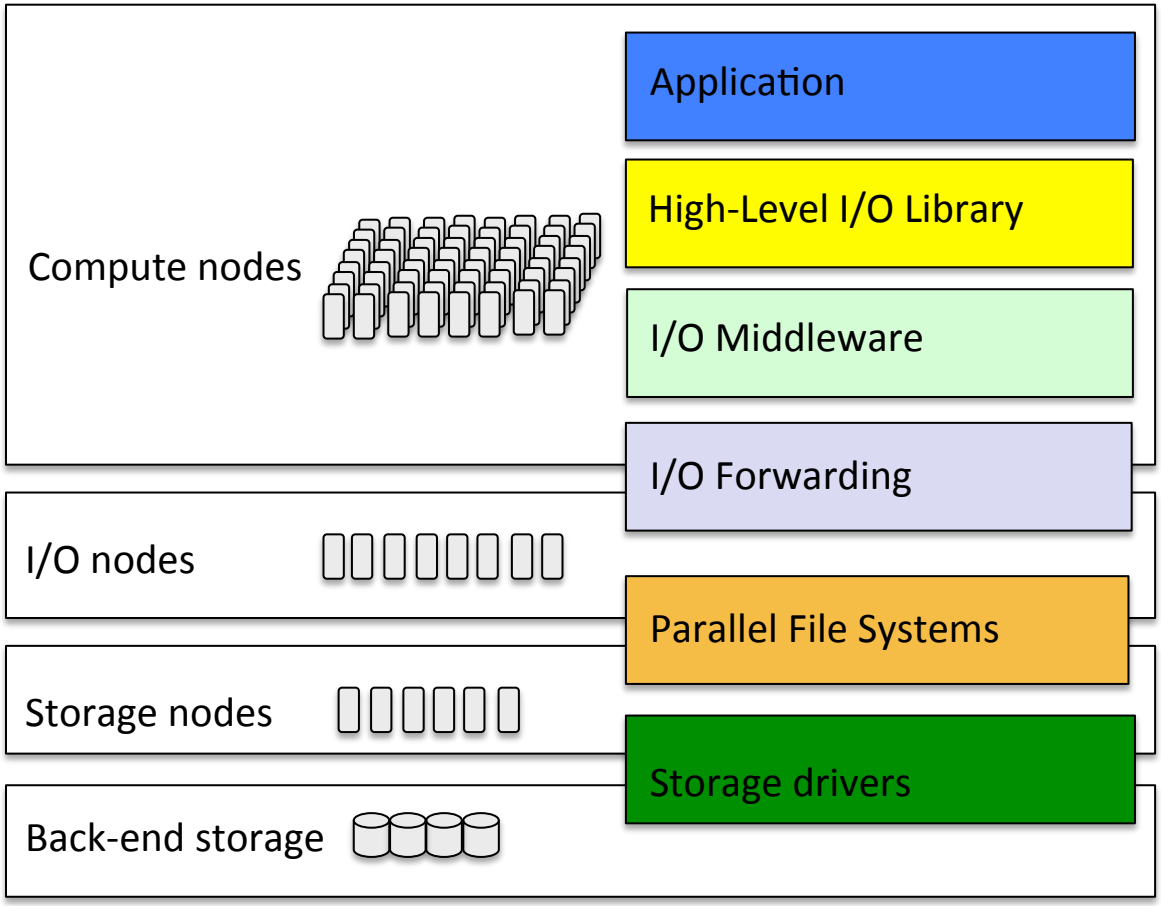
Argonne NATIONAL LABORATORY

ARCOS

Introduction

- Scientific applications (climate, genomics, high energy physics, astronomy etc.) ingest, generate, process increasingly larger data sets
- Future high scale supercomputers need to deal efficiently with huge amounts of data
- Current I/O software stack needs to evolve in order to meet the oncoming scalability challenges

Current problems of storage I/O software stack



Maps application abstractions onto storage abstractions (e.g.: HDF5, ParallelNetCDF)

Reduces the number of file system calls by optimizations like collective I/O (e.g.: MPI-IO)

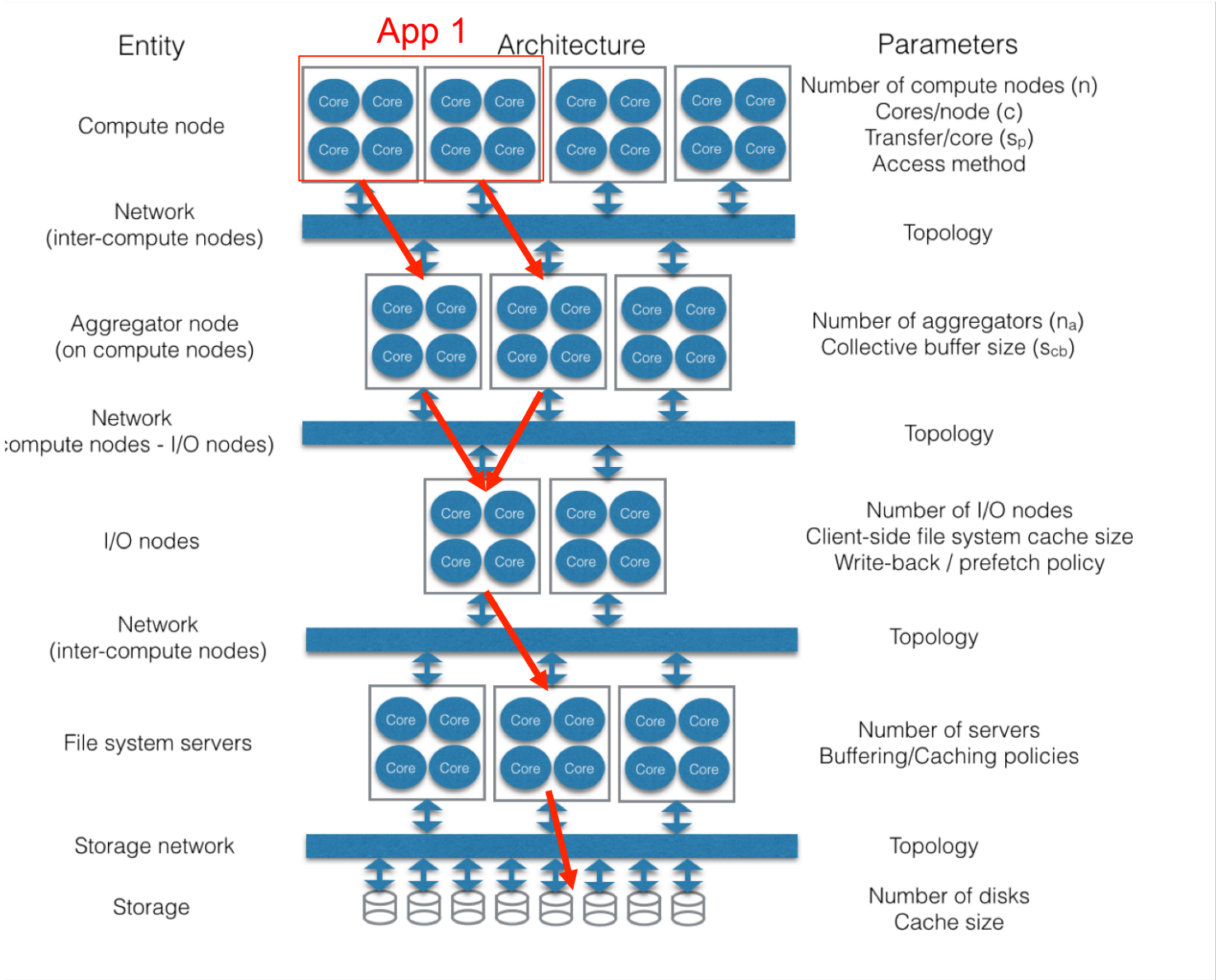
Offloads I/O functionality from compute nodes (e.g.: Mercury, IOFSL)

Offer a global name space and high performance storage access (e.g.: GPFS, Lustre, PVFS)

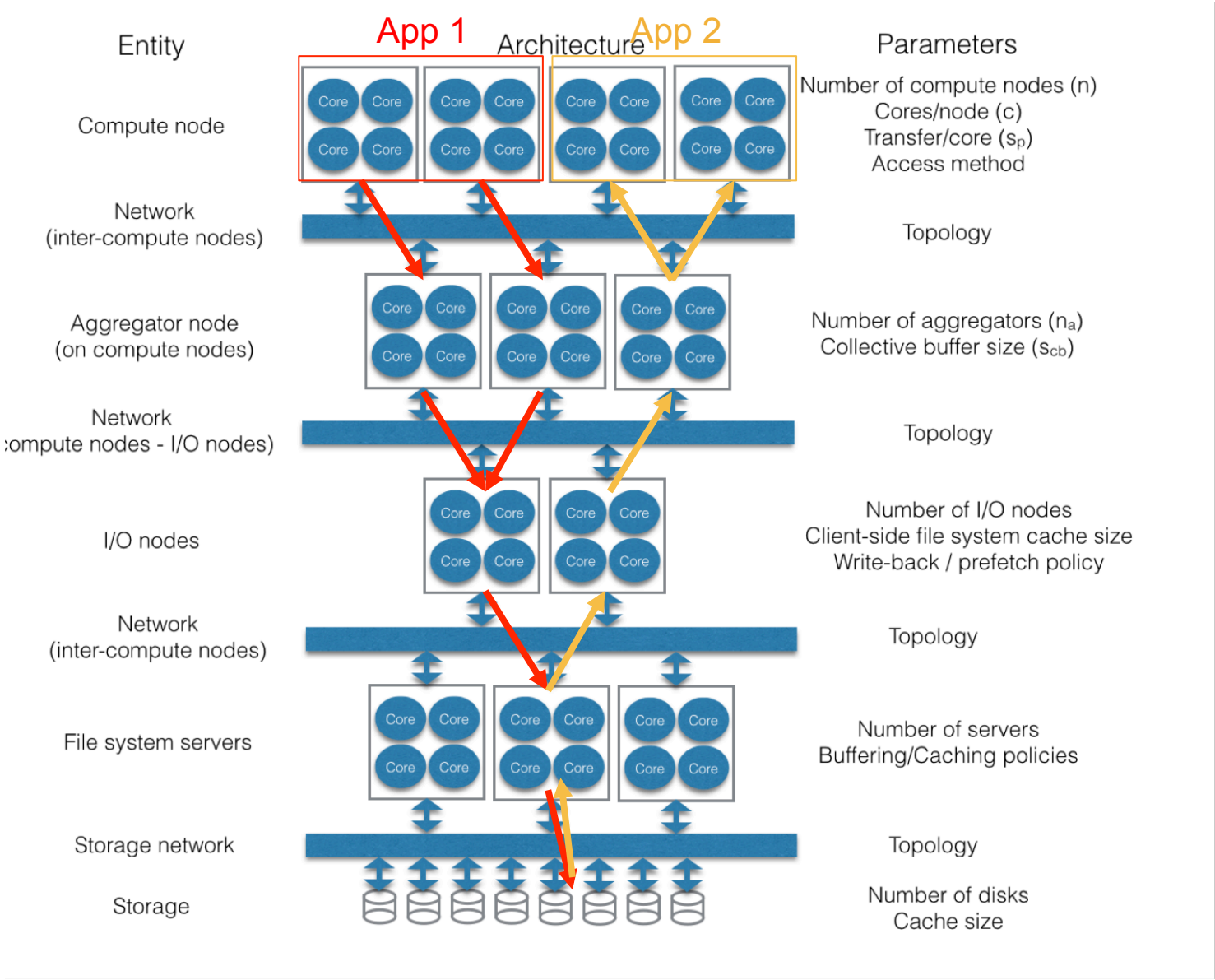
Block and storage object devices

- Optimization: complex stack, deep distributed storage hierarchy
- Coordination: poor state of programmable control mechanisms are not available (e.g., for data staging, dynamic load balancing, resilience)
- Exploit data locality

Data flow in Blue Gene/Q



Data flow in Blue Gene/Q



Data staging challenges

- Concurrent parallel data flows
  - Lack of data staging coordination
    - Among applications
    - Between applications and the system
- Increasing storage hierarchy
- Storage I/O optimizations are local: Difficult to perform global optimizations
- Cross-layer control mechanisms are not available (e.g., for data staging, dynamic load balancing, resilience)
- Lack of standards for dynamic monitoring of large scale infrastructures (e.g. load, faults)
- Coupled control and data mechanisms

Outline

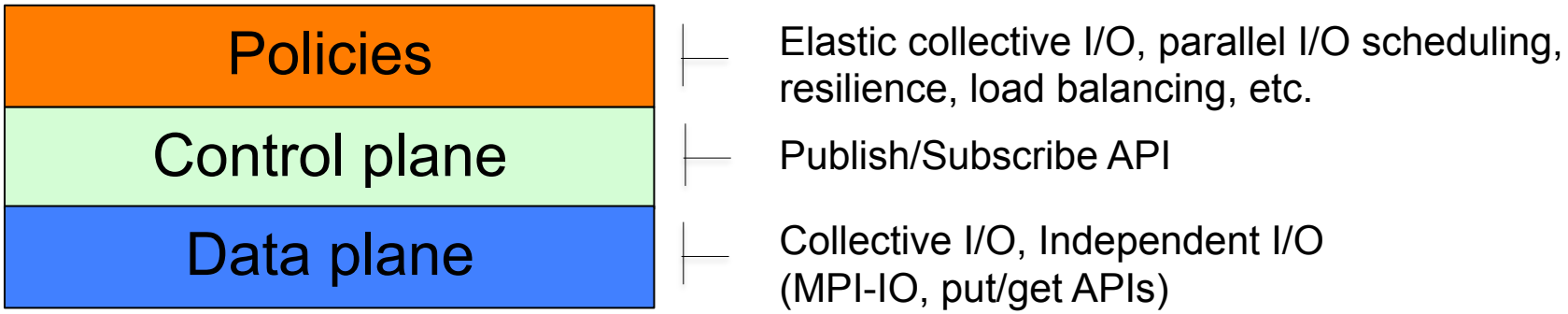
- **CLARISSE overview**
- CLARISSE data plane
- CLARISSE control plane
- CLARISSE policies examples
- Conclusions
- Future work

CLARISSE overview

- Cross-Layer Abstractions and Runtime for I/O Software Stack (CLARISSE)
  - A 3-year project started October 2013
  - European “Marie Curie” International Outgoing Fellowship
  - Collaboration between ANL and UC3M (Spain)
- Goals
  - Novel mechanisms for global data staging coordination to improve
    - Load balance, resilience, parallel I/O scheduling, locality exploitation
  - Improve programmability
  - Facilitate extendability

CLARISSE overview

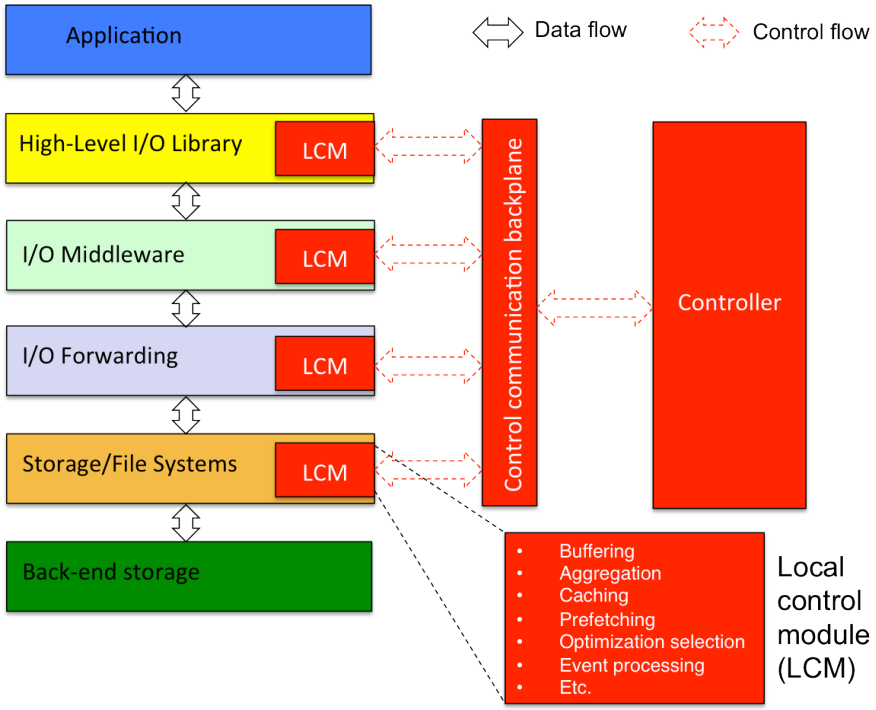
- Decouple the data and control planes
  - Data plane
  - Control plane
  - Policy
- Cross-layer abstractions and run-time
  - Facilitate the flow of control and data across the I/O stack



Outline

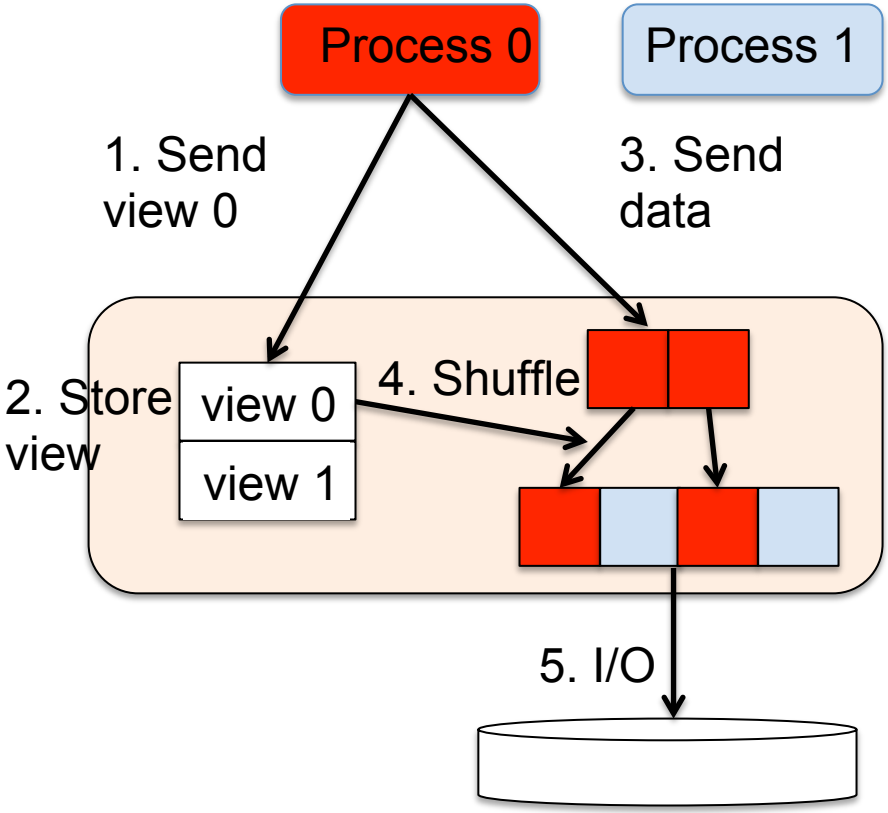
- CLARISSE overview
- **CLARISSE data plane**
- CLARISSE control plane
- CLARISSE policies example
- Case studies
- Conclusions
- Future work

CLARISSE overview

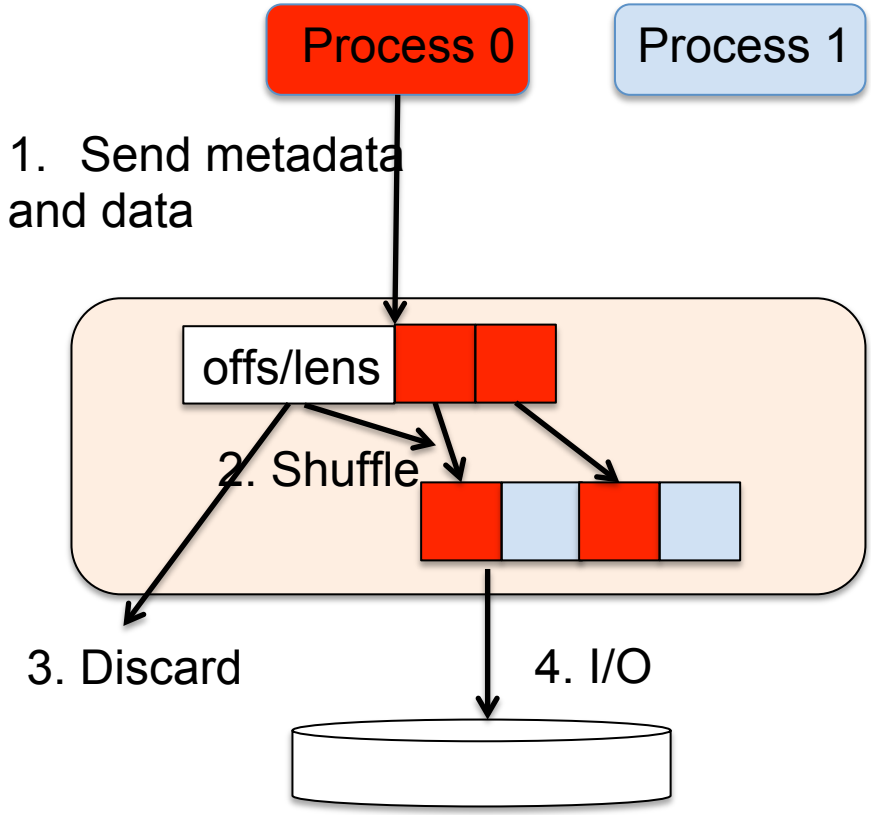


- Data plane
  - Design novel abstractions and mechanisms for supporting data flow optimizations
    - Data aggregation (e.g., collective I/O)
    - buffering / caching, data staging
    - load balance
    - data locality (e.g. in-situ and in-transit data processing)
  - Parallel data-flows based on the these abstractions

Collective I/O



a. View-based collective I/O



b. List-based collective I/O

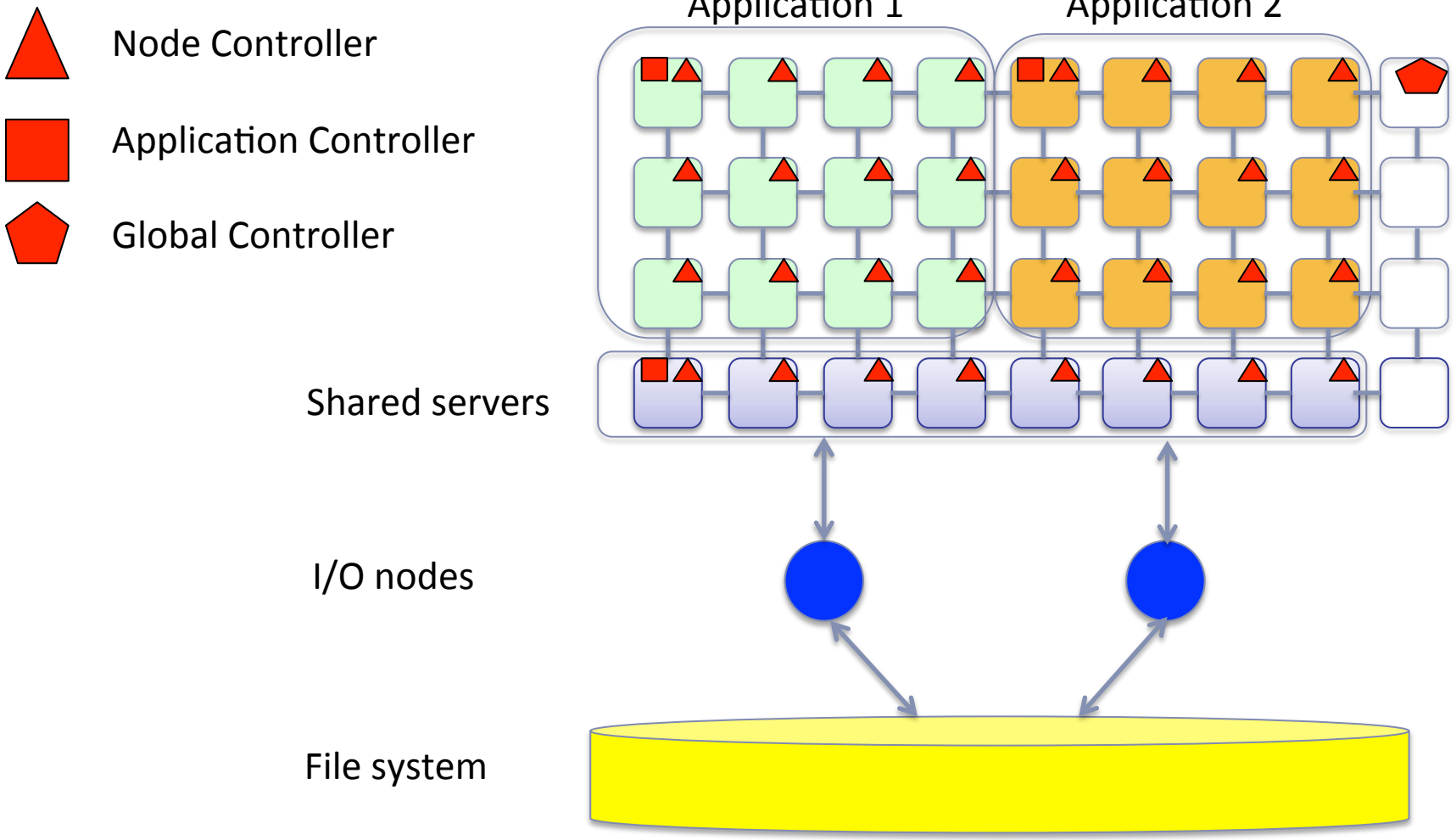
Outline

- CLARISSE overview
- CLARISSE data plane
- **CLARISSE control plane**
- CLARISSE policies examples
- Case studies
- Conclusions
- Future work

CLARISSE control backplane

- Control backplane
  - Based on a publish/subscribe substrate (e.g. Beacon)
  - Processes can subscribe to events having certain properties
    - Associate call-back
    - Wait for an event
    - Check for the arrival of an event
- Allows building any distributed/replicated control architecture
- In this work: Hierarchical control
  - Global controller
  - Application controller
  - Node controller
  - All nodes participate in control

CLARISSE hierarchical control infrastructure



Outline

- CLARISSE overview
- CLARISSE data plane
- CLARISSE control plane
- **CLARISSE policies examples**
- Conclusions
- Future work

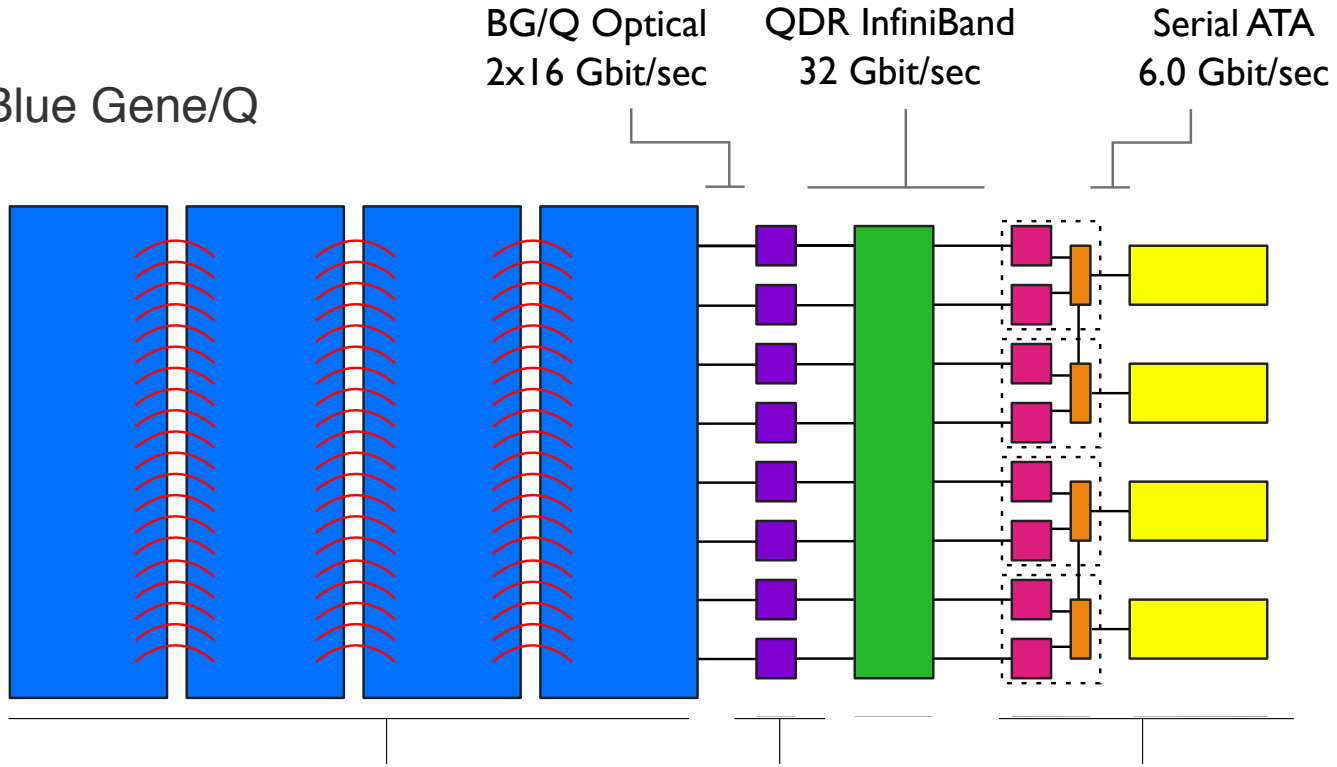
CLARISSE policies examples

- Elastic collective I/O
- Parallel I/O scheduling

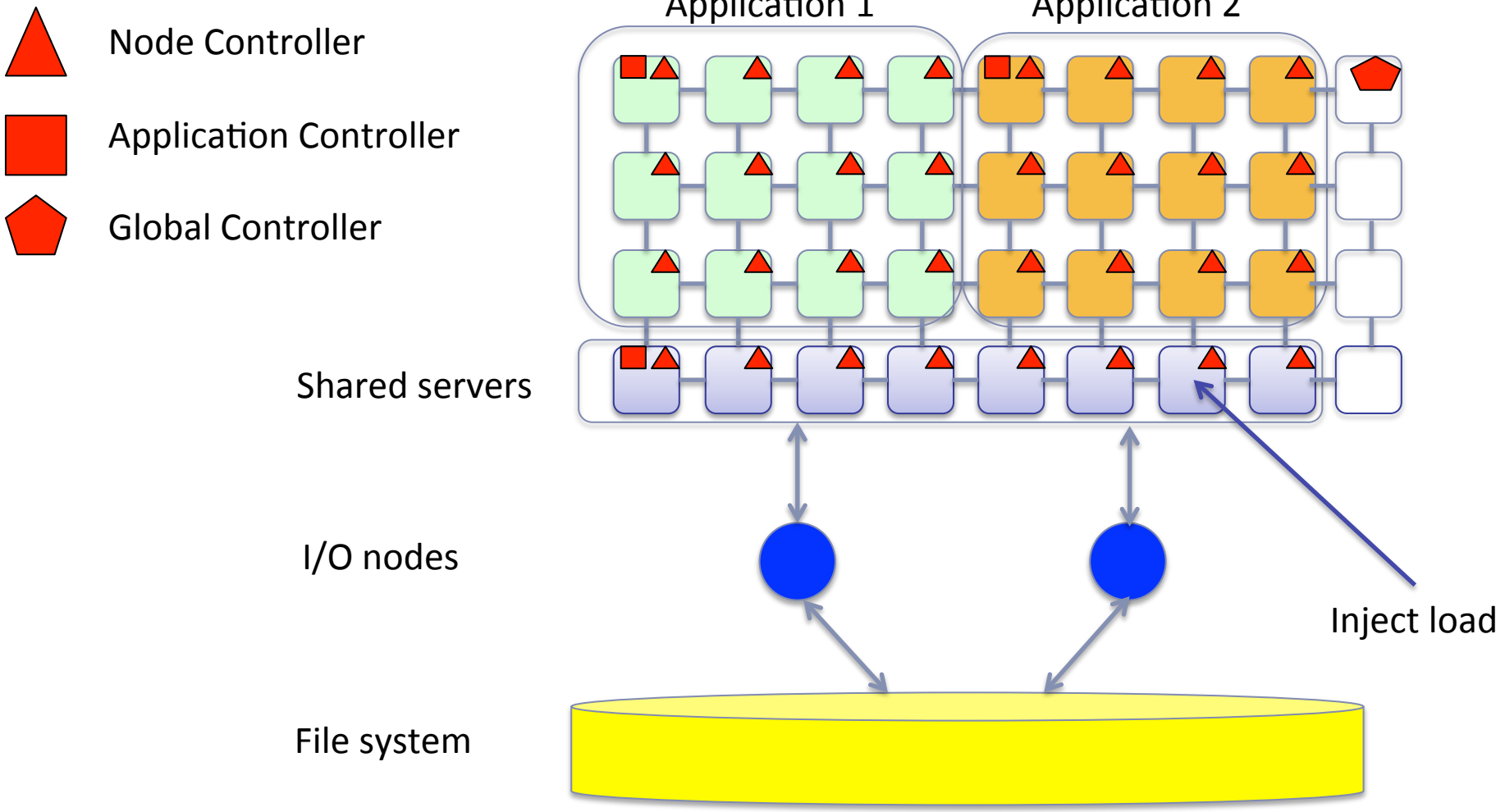
- Evaluation
  - S-IOR
    - IOR pattern with a pseudo-computation phase between two consecutive I/O opearations
  - VPIC-IO
    - a scalable 3D electromagnetic relativistic kinetic plasma
    - wites a 1D array of particles,
  - VORPAL-IO
    - a parallel code simulating the dynamics of electromagnetic systems and plasmas
    - Writes a 3D partition of blocks
  - 10 compute phases interleaved with 10 I/O phases

CLARISSE policies examples

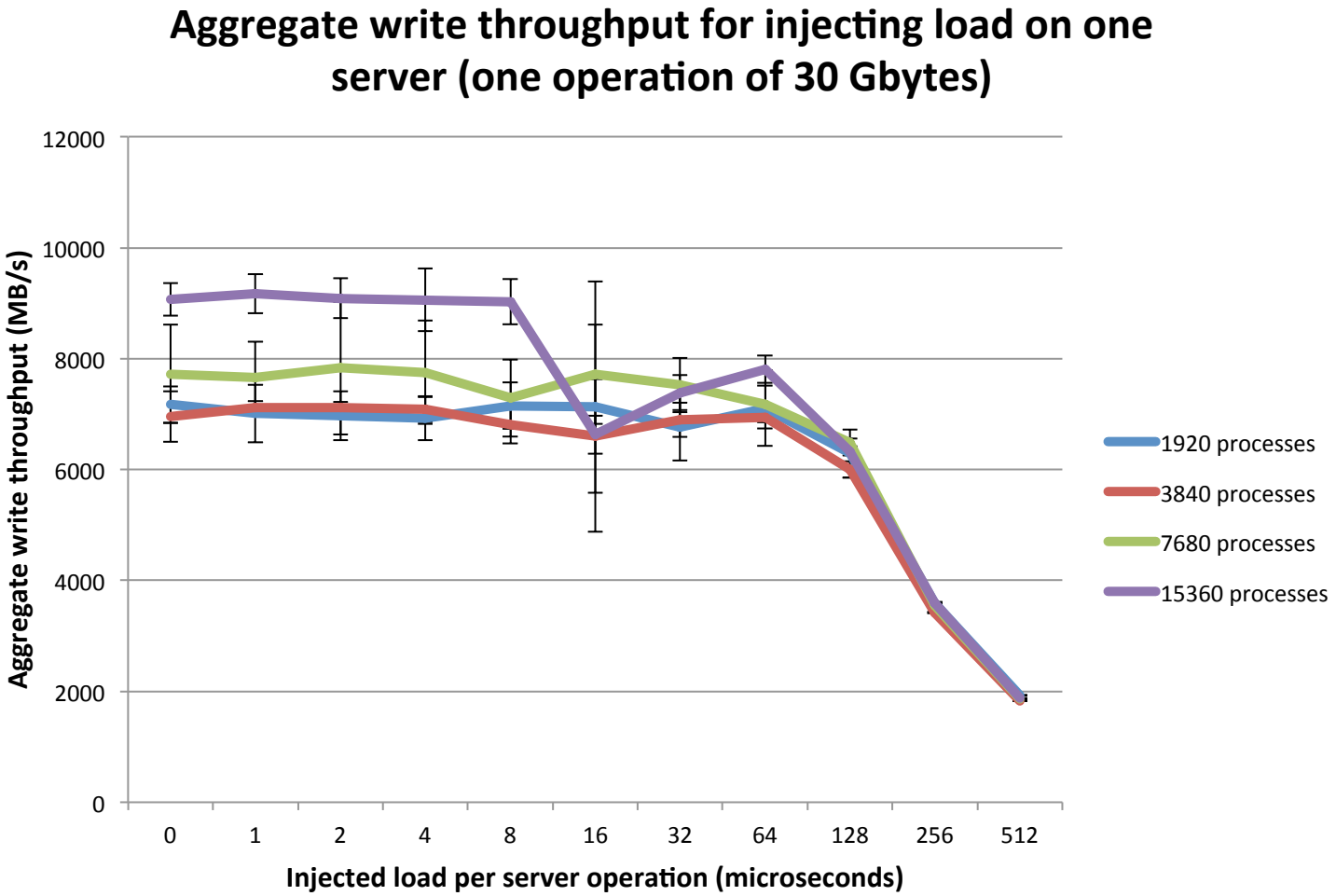
- Elastic collective I/O
- Parallel I/O scheduling
- Vesta Blue Gene/Q



CLARISSE hierarchical control infrastructure

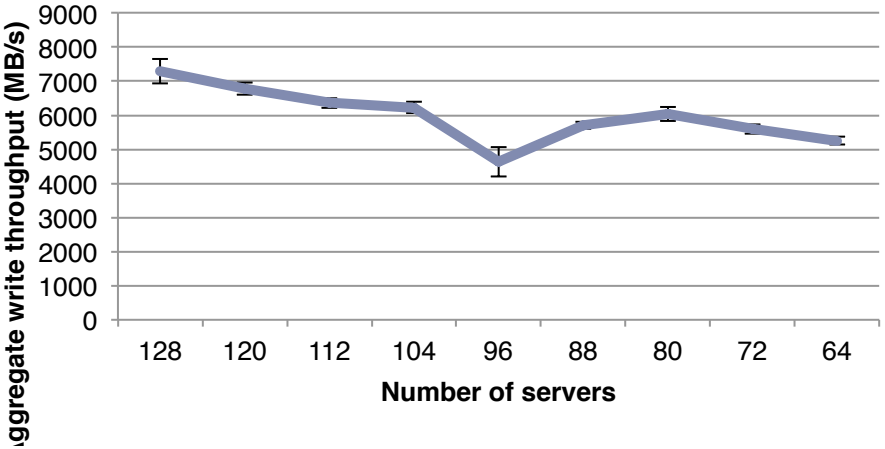


Load injection at 1 server (1 application)

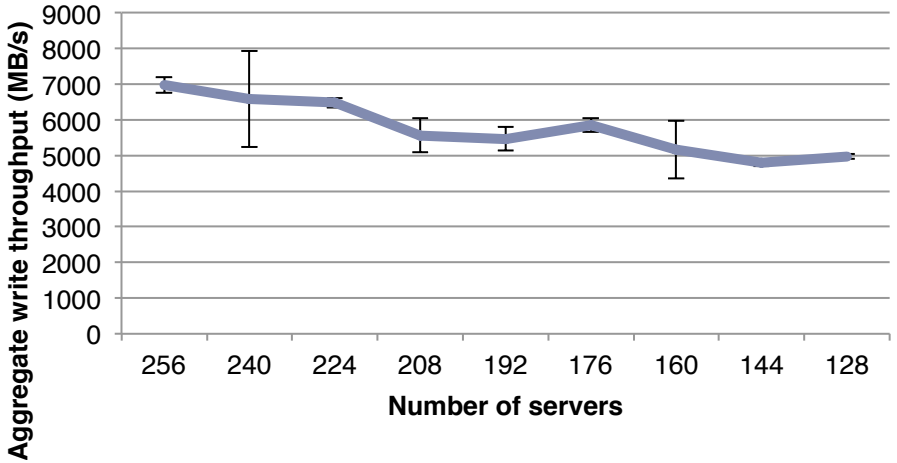


Scale-down number of servers (1 application)

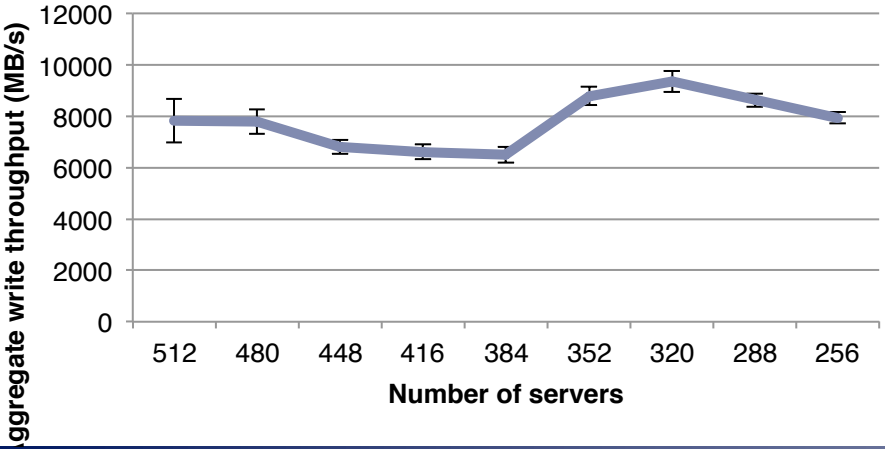
Aggregate write throughput for 1920 processes



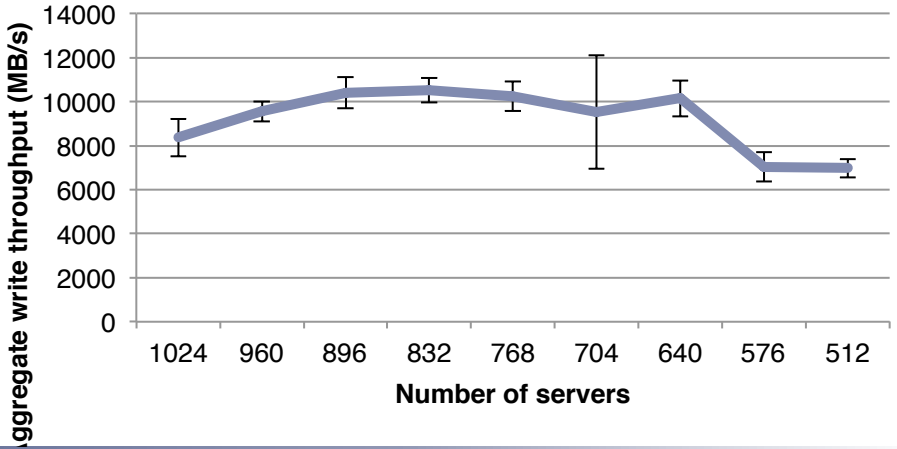
Aggregate write throughput for 3840 processes



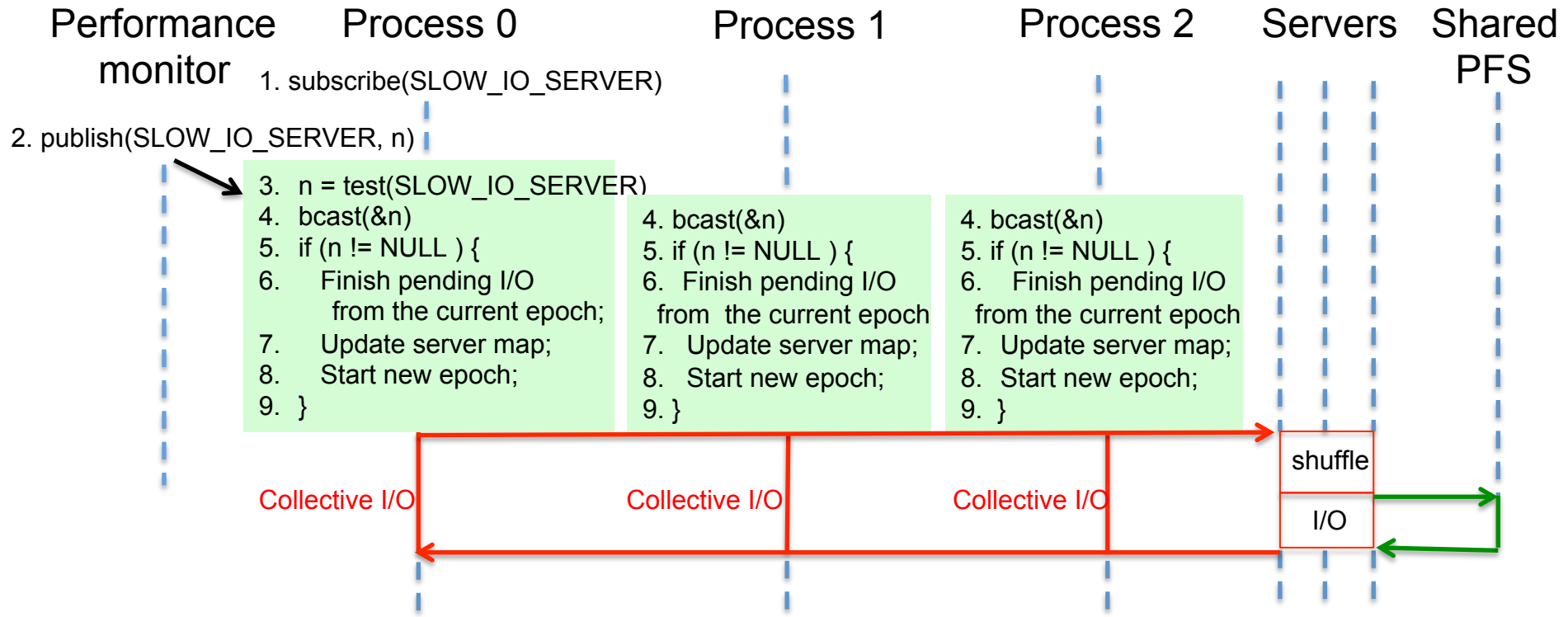
Aggregate write throughput for 7980 processes



Aggregate write throughput for 15360 processes



Elastic collective I/O

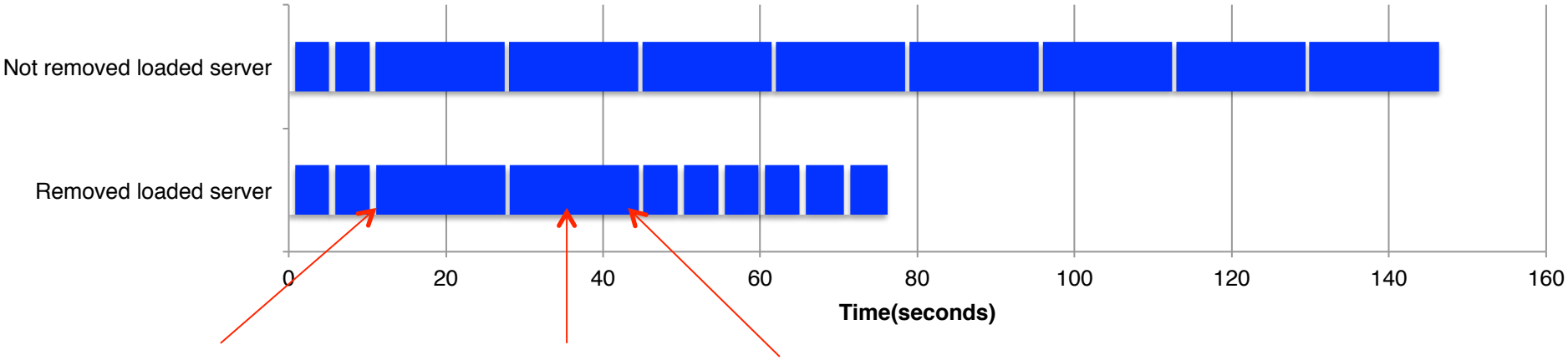


Elastic collective I/O

- Assumes the availability of a load detection mechanism
- One application process detects a loaded server
- Notifies the application controller
- Application controller informs all node controllers and ask them to prepare to start a new epoch with less servers
- Node controller
  - Decides the last operations to be executed from the current epoch
  - Suspends all operation from the future epoch
  - Updates the server map
  - Notifies the application controller
  - Application controller ask all nodes to start a new epoch
- Each node controller resumes the suspended operations if any

Elastic collective I/O performance

Write time (10 operations, 3840 processes, 256/255 servers)

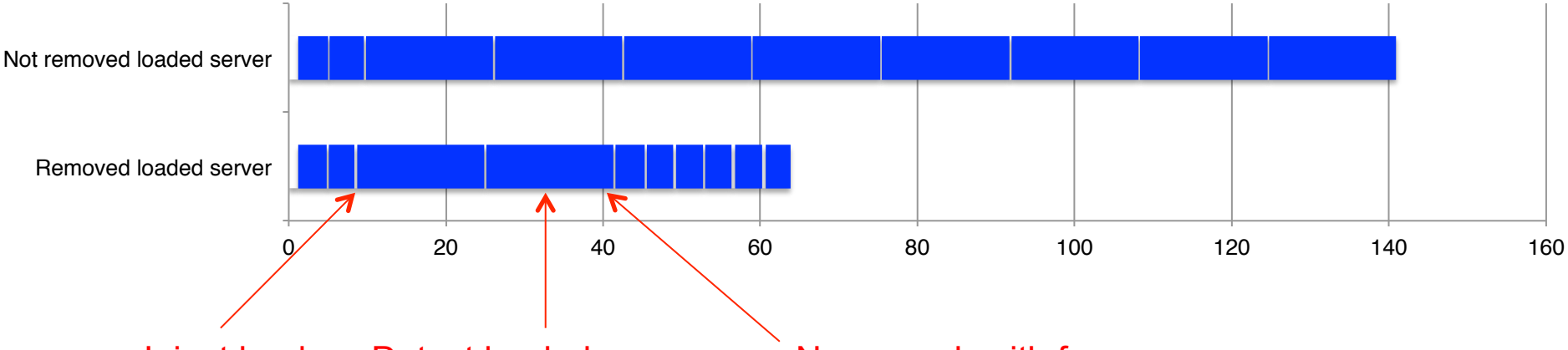


Inject load

Detect loaded server
Reconstruct server map

New epoch with fewer servers

Write time (10 operations, 15360 processes, 1024/1023 servers)

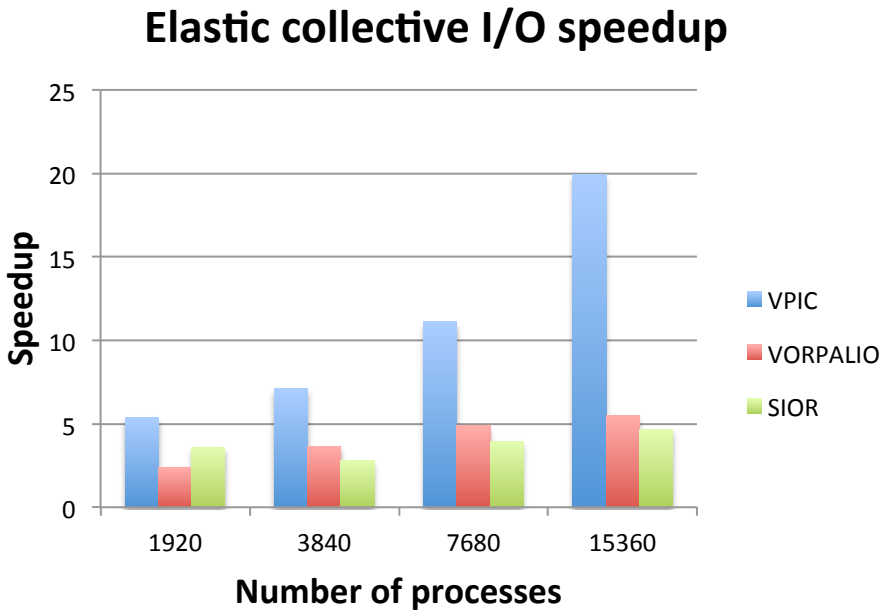


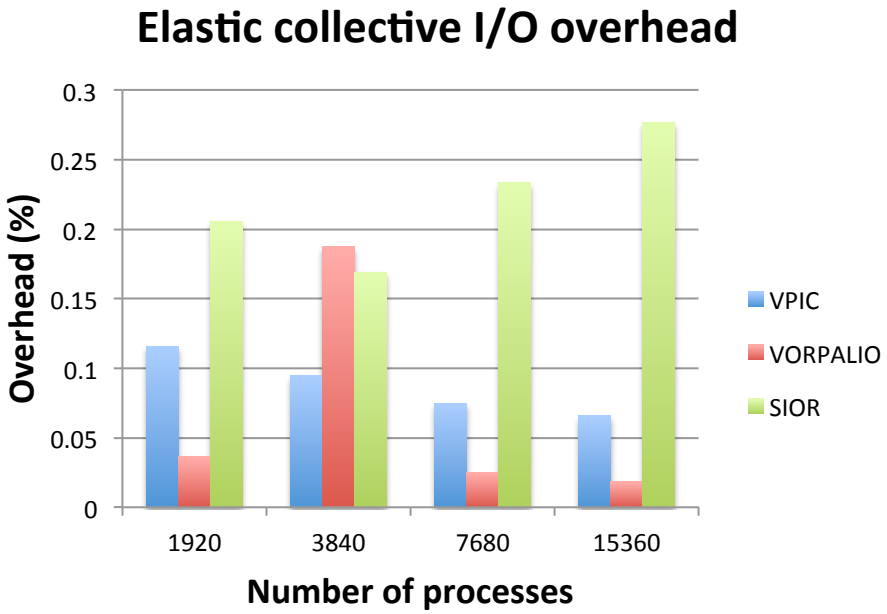
Inject load

Detect loaded server
Reconstruct server map

New epoch with fewer servers

Elastic collective I/O performance





Parallel I/O scheduling

- Several applications share the same servers
- The application controller notifies the global controller
- The global controller schedules the next application to be run
- Several policies possible
  - FCFS evaluation
- Interference factor I

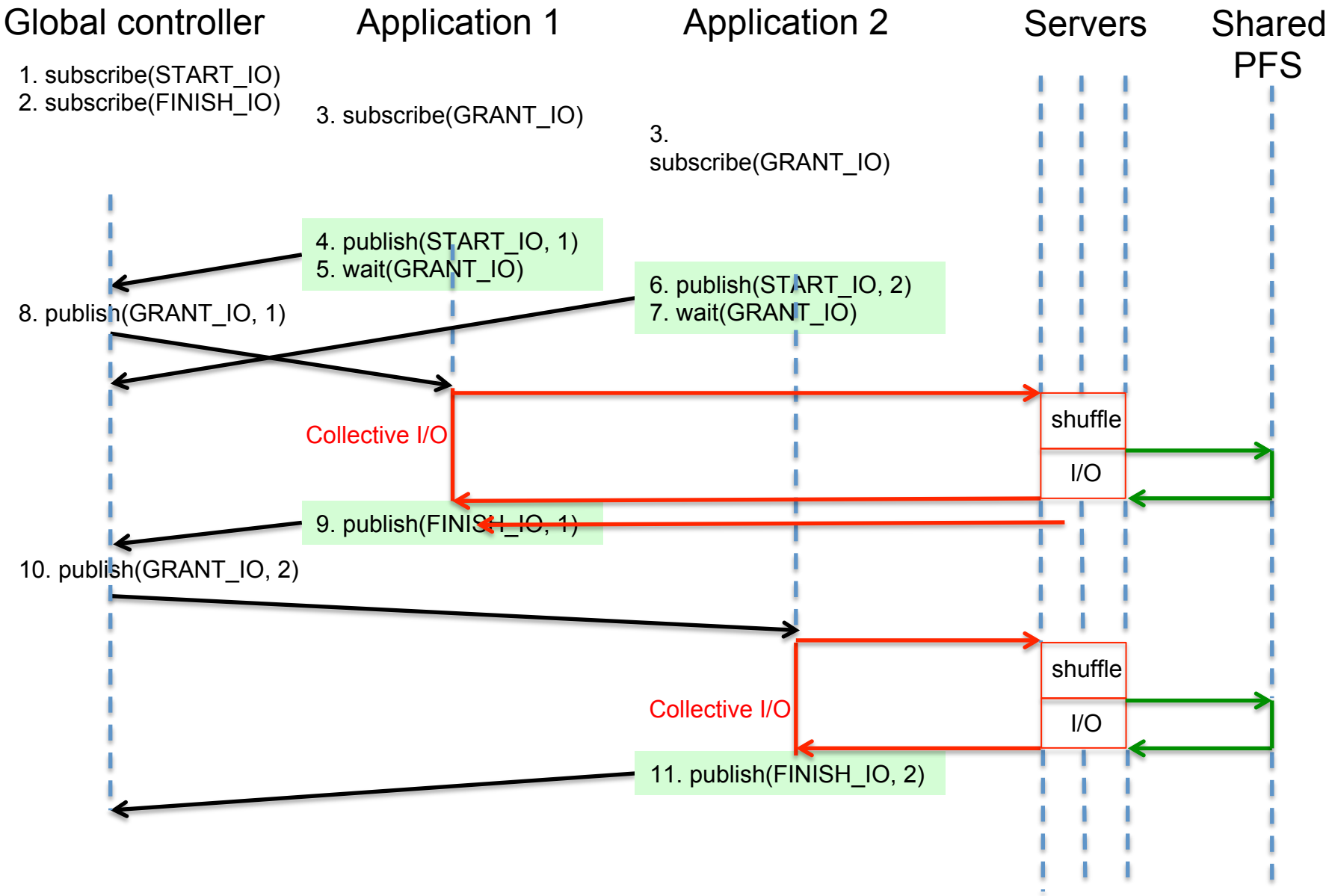
$$I = \frac{T_{nosched}}{T_{alone}} \geq 1$$

- Scheduling cost factor C

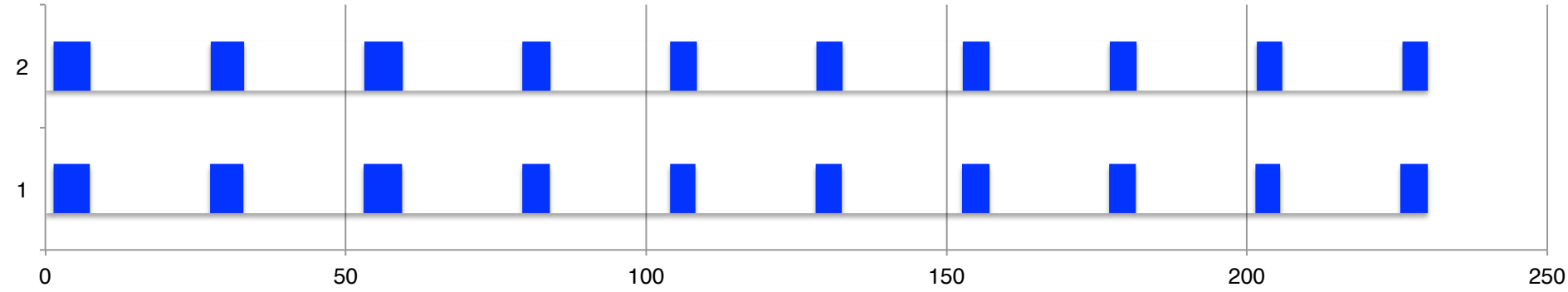
$$C = \frac{T_{sched}}{T_{alone}} \geq 1$$

- Scheduling is efficient if I > C

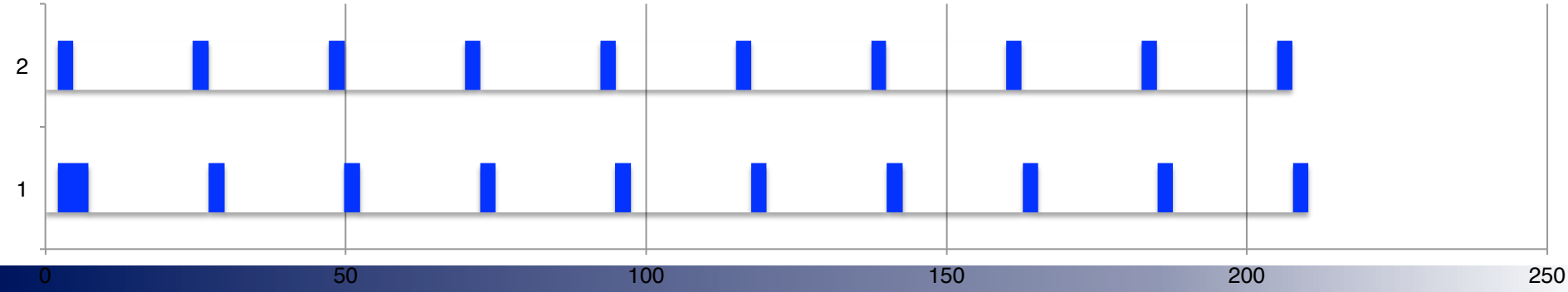
Parallel I/O scheduling



Write timeline for two parallel clients with 960 processes each - No scheduling

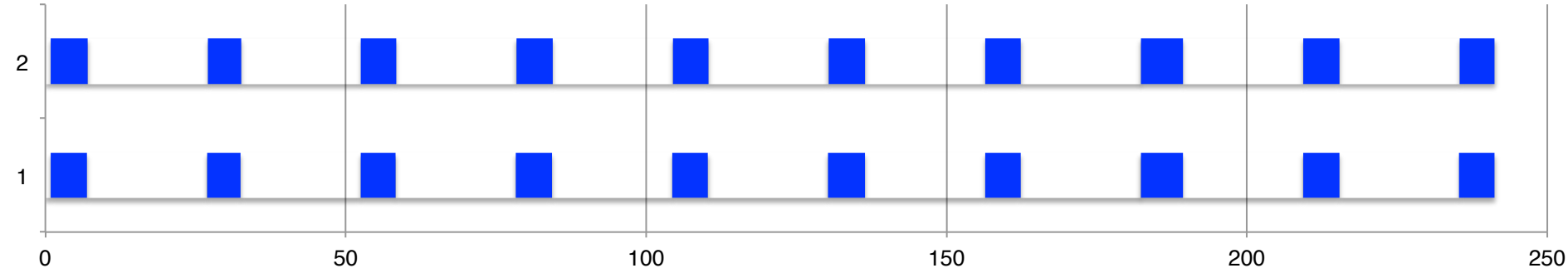


Write timeline for two parallel clients with 960 processes each - FCFS scheduling

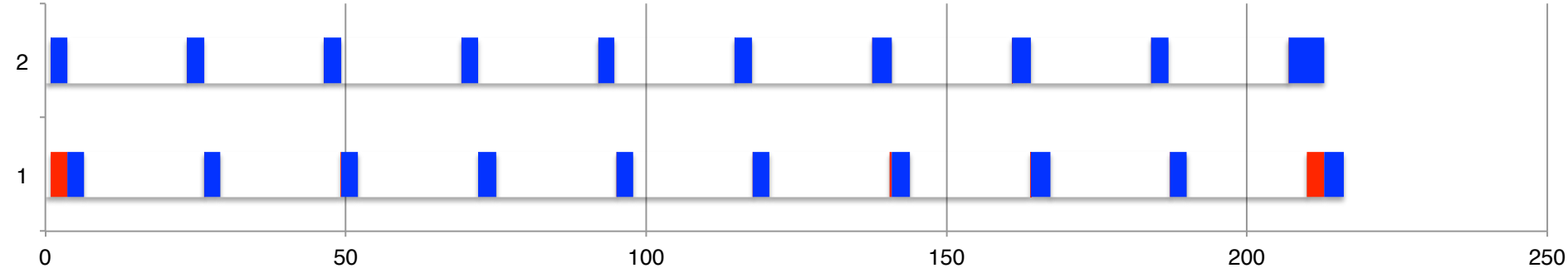


FCFS scheduling versus no scheduling

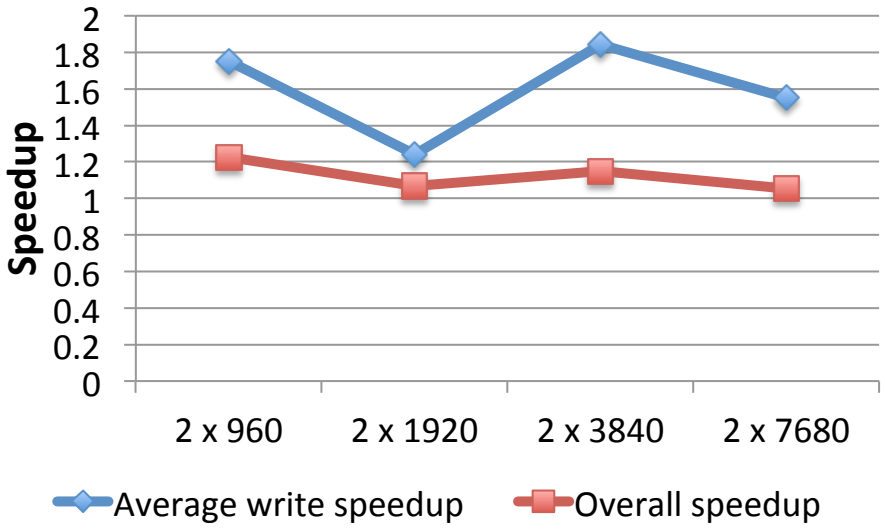
Write timeline for two parallel clients with 3840 processes each - No scheduling

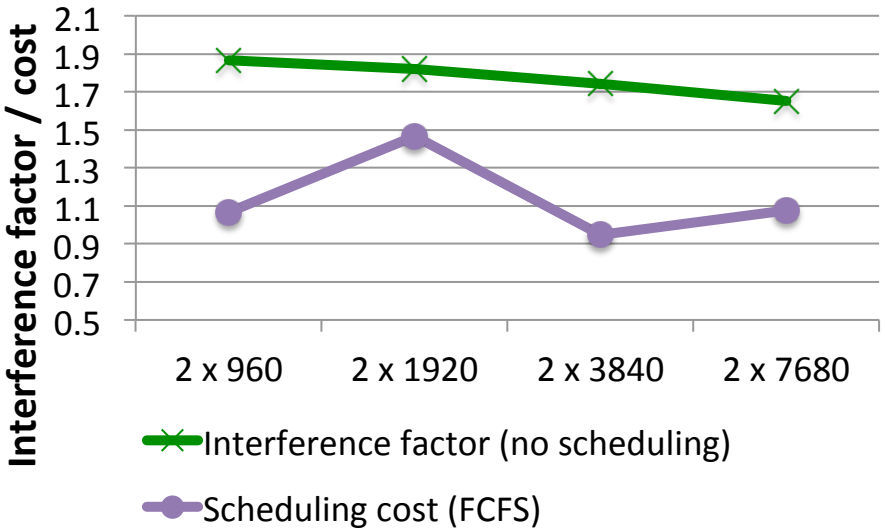


Write timeline for two parallel clients with 3840 processes each - FCFS scheduling

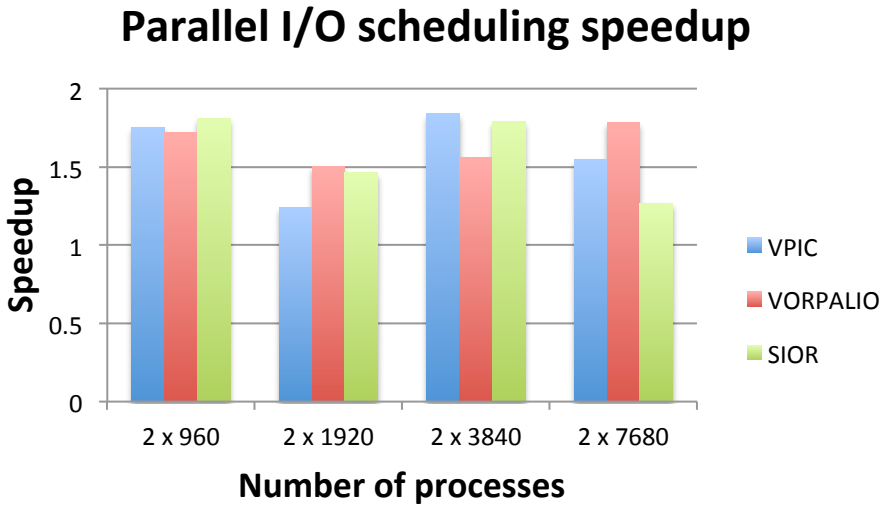


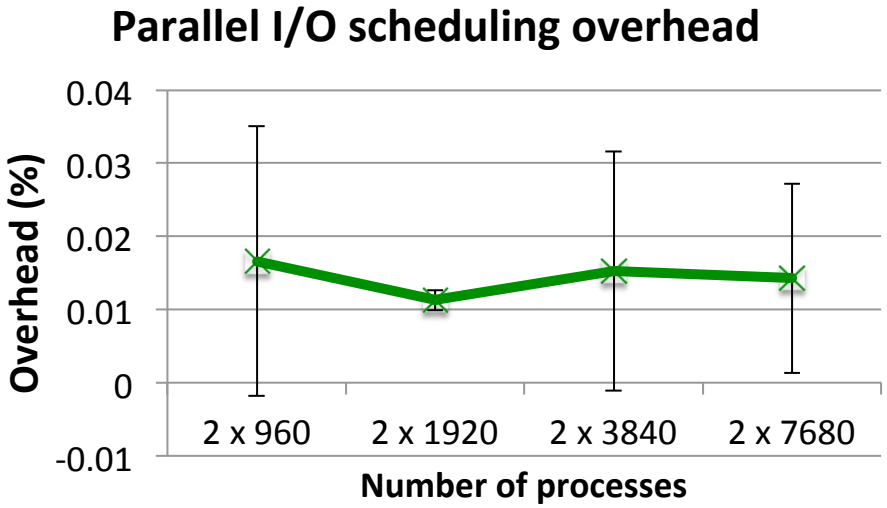
Parallel I/O scheduling performance (VPIC)





Parallel I/O scheduling performance





Conclusions

- CLARISSE: Middleware for data staging coordination
- Separation of data and control
- Hierarchical control
- Significant benefits
  - Load/Fault aware sever-scale down
  - Parallel I/O scheduling
- Scalable load and fault monitoring is required

Ongoing and future work

- Topology-aware server/aggregator placement - JL Colaboration INRIA-ANL
- Multiple stage coordination (aggregation – burst buffer – file system)
- Load prediction based on Omnisc'IO (Mathieu Dorrier – ANL)
- Adaptive buffering in parallel applications workflows (Decaf project)
- Adopt Global Information Bus from Argo and Hobbes (Beacon, Exposé)
- Need for sub-second monitoring and notification

Thank you