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)

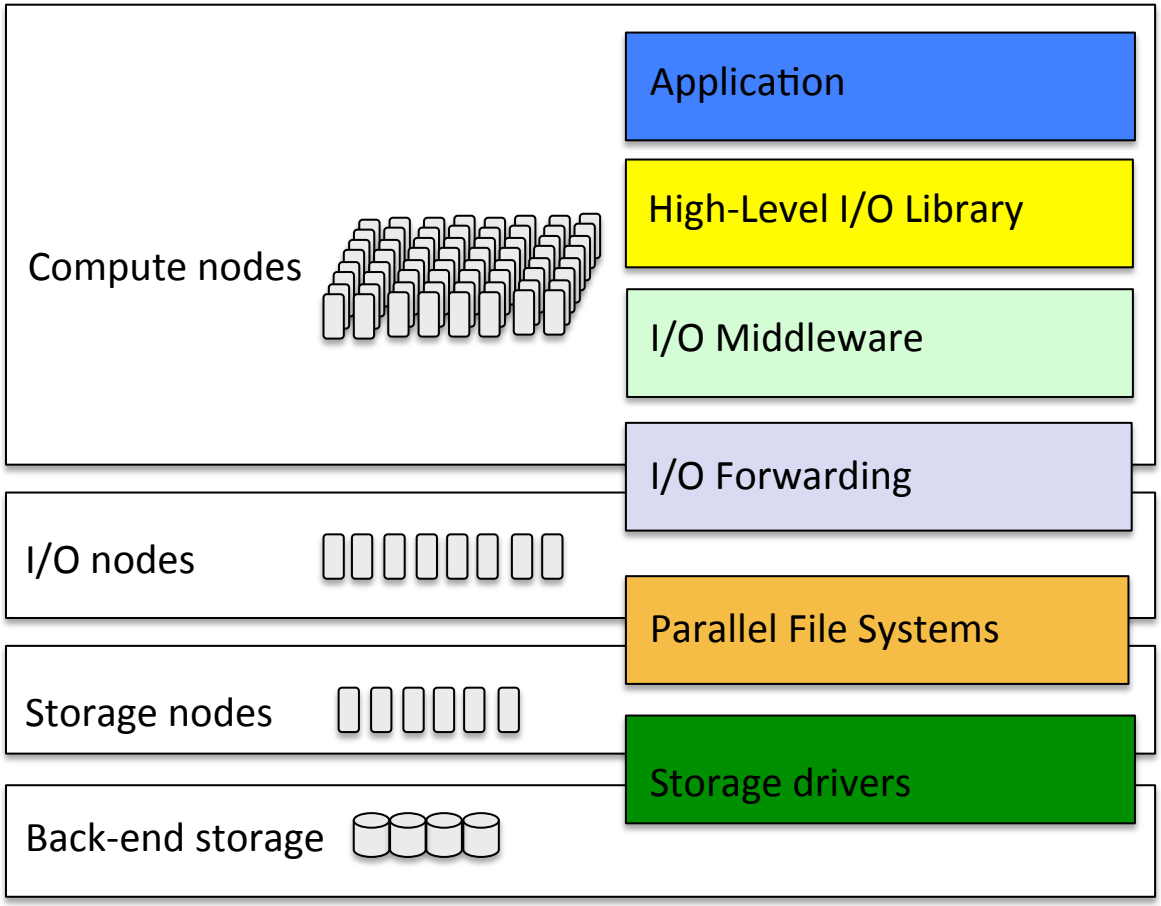
Argonne NATIONAL LABORATORY

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)

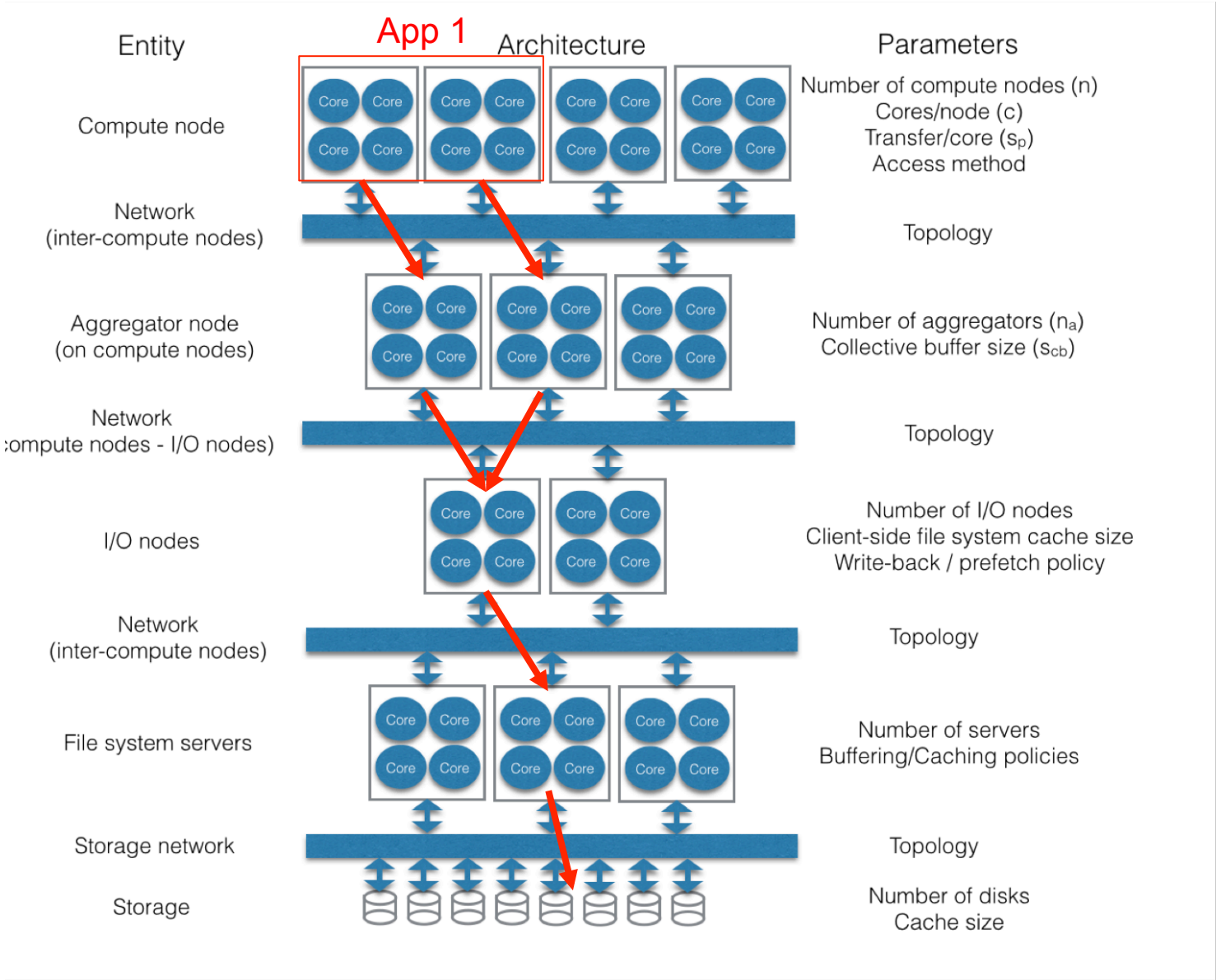
Offloads I/O functionality from compute nodes (e.g.: Mercury, IOFSL)

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

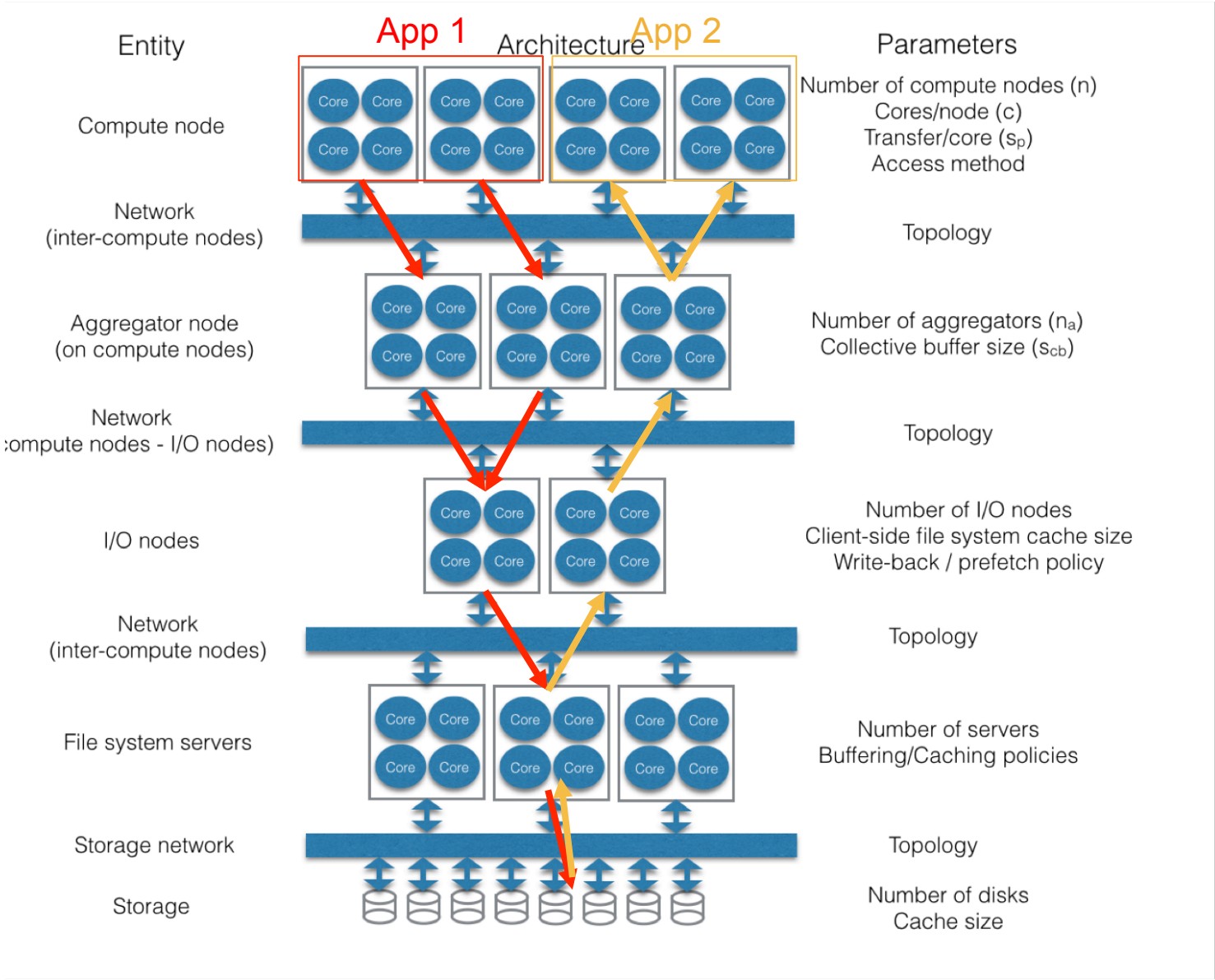
Block and storage object devices

- 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
- Goal: offer novel mechanisms for data staging coordination to improve
  - Load balance
  - Resilience
  - Parallel I/O scheduling

Outline

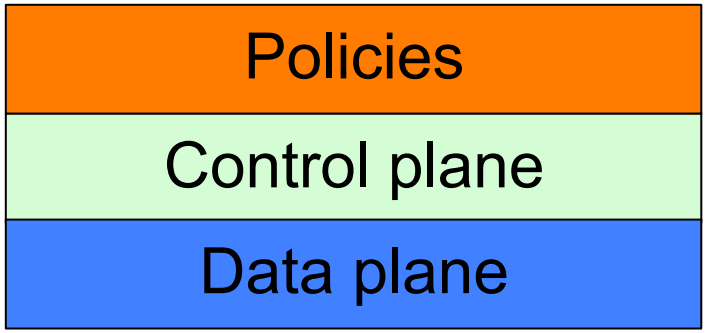
- **CLARISSE overview**
- CLARISSE data plane
- CLARISSE control plane
- Deployment
- Case studies
- Conclusions
- Future work

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

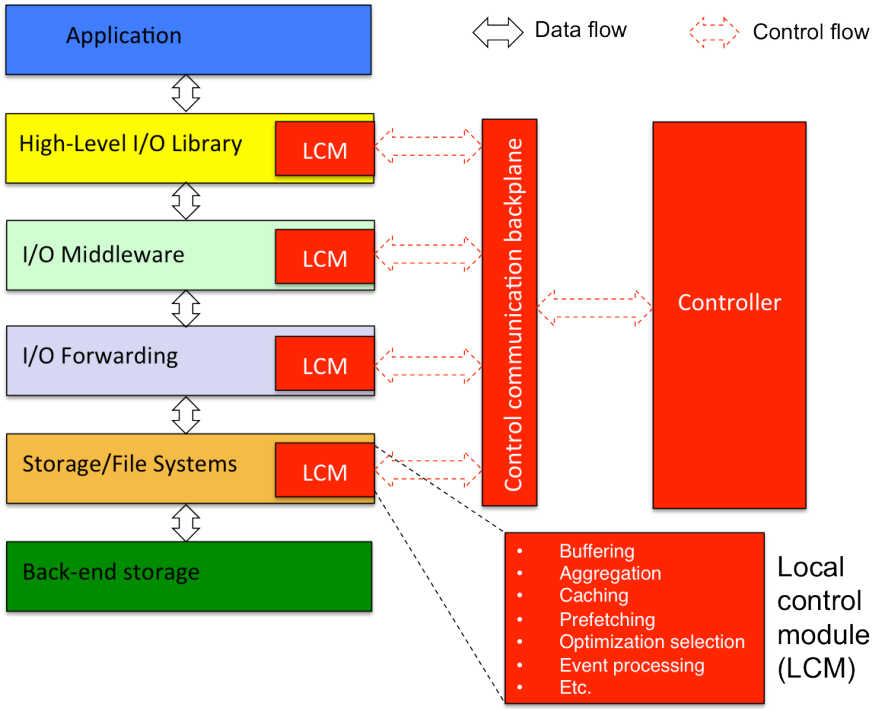
- Decouple the data and control planes
  - Data plane
  - Control plane
  - Policy
- Cross-layer abstractions and run-time
  - Facilitate the flow of control and data across the I/O stack



Outline

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

CLARISSE overview

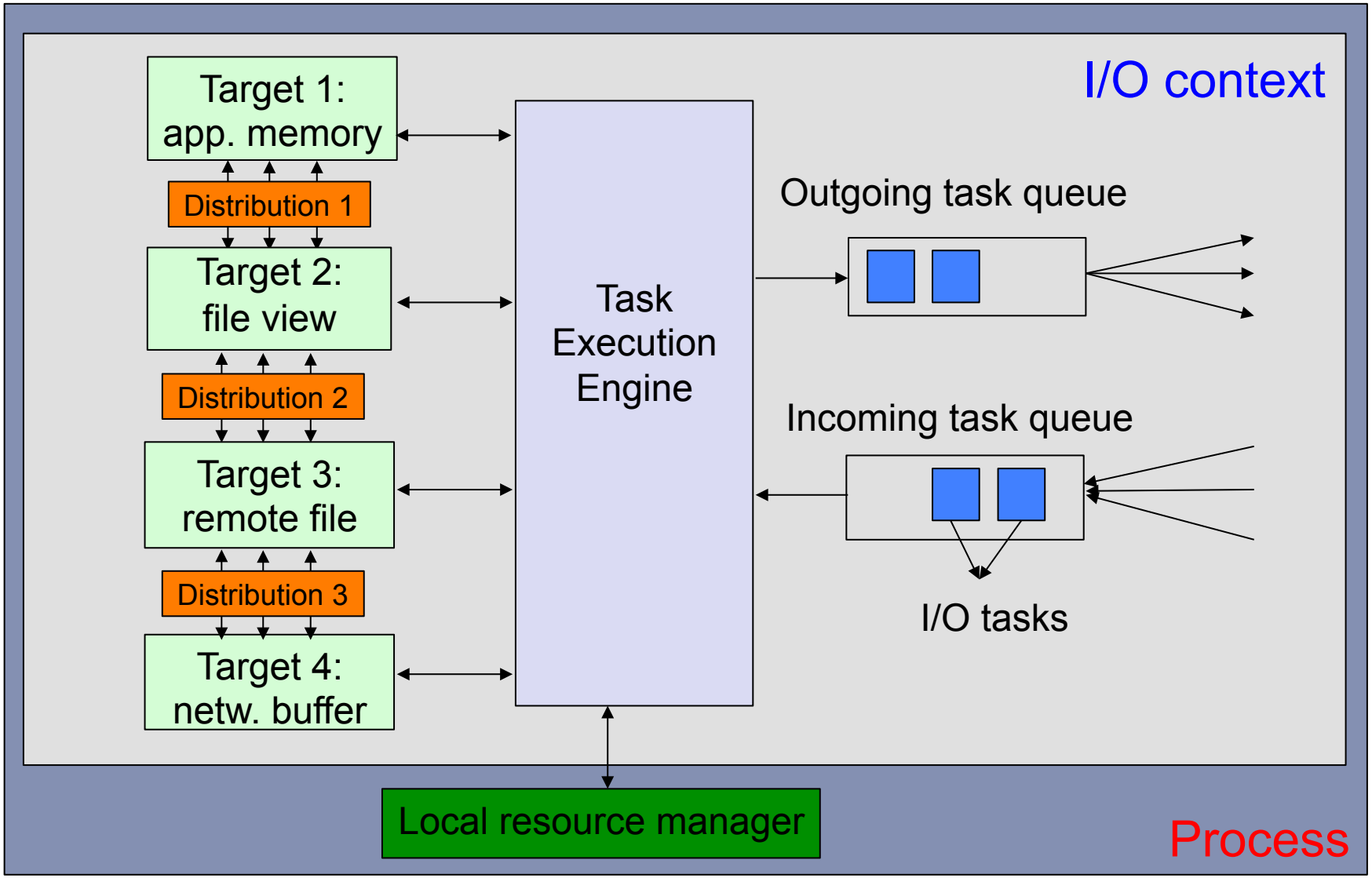


- Data plane
  - Design novel abstractions and mechanisms for supporting data flow optimizations
    - Data aggregation (e.g., collective I/O)
    - buffering / caching, data staging
    - load balance
    - data locality (e.g. in-situ and in-transit data processing)
  - Parallel data-flows based on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

CLARISSE data plane abstractions

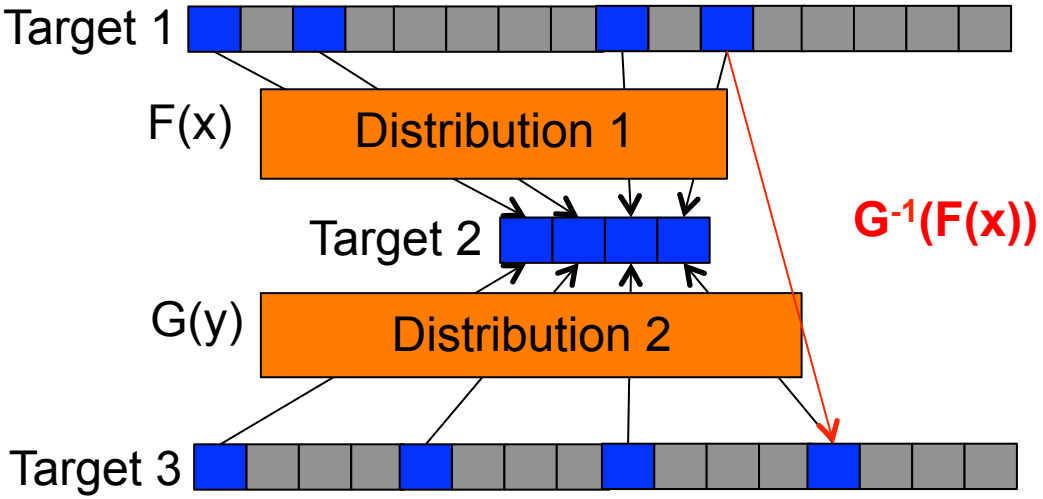
- Five main abstractions
  - Targets
  - Distributions
  - I/O contexts
  - I/O tasks
  - Task queues
- Objectives
  - Represent the storage I/O activity in terms of these abstractions
  - Offer a logically centralized view of these abstractions
- Probably not realistic to expect to have these abstractions present at all stack layers

CLARISSE abstractions



Abstractions for the I/O stack

- Targets
  - Virtual linear spaces
  - Memory, files, storage objects, network buffers
  - put/get interface
- Distributions
  - Mappings between targets
  - Mapping functions from non-contiguous regions to a contiguous region
  - Composing the functions for arbitrary non-contiguous to non-contiguous mappings

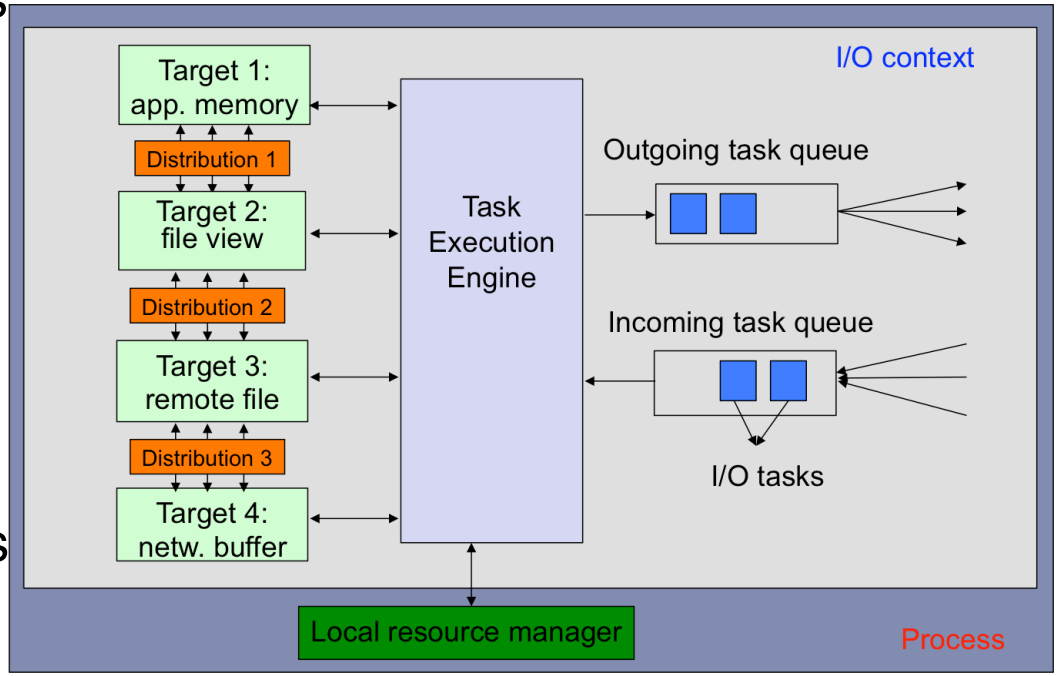


I/O tasks and task queues

- I/O task
  - I/O related set of actions between a local I/O context and a remote I/O context
  - Example of I/O tasks
    - Redistribute data between memory application and network buffer
    - Run custom defined computation (e.g. Code-on-Demand of EVPath)
    - Send/receive data from remote nodes, file/storage systems
- Task queues
  - Queues containing the tasks to be processed by the task engine
  - Control exposed to the control backplane

I/O contexts

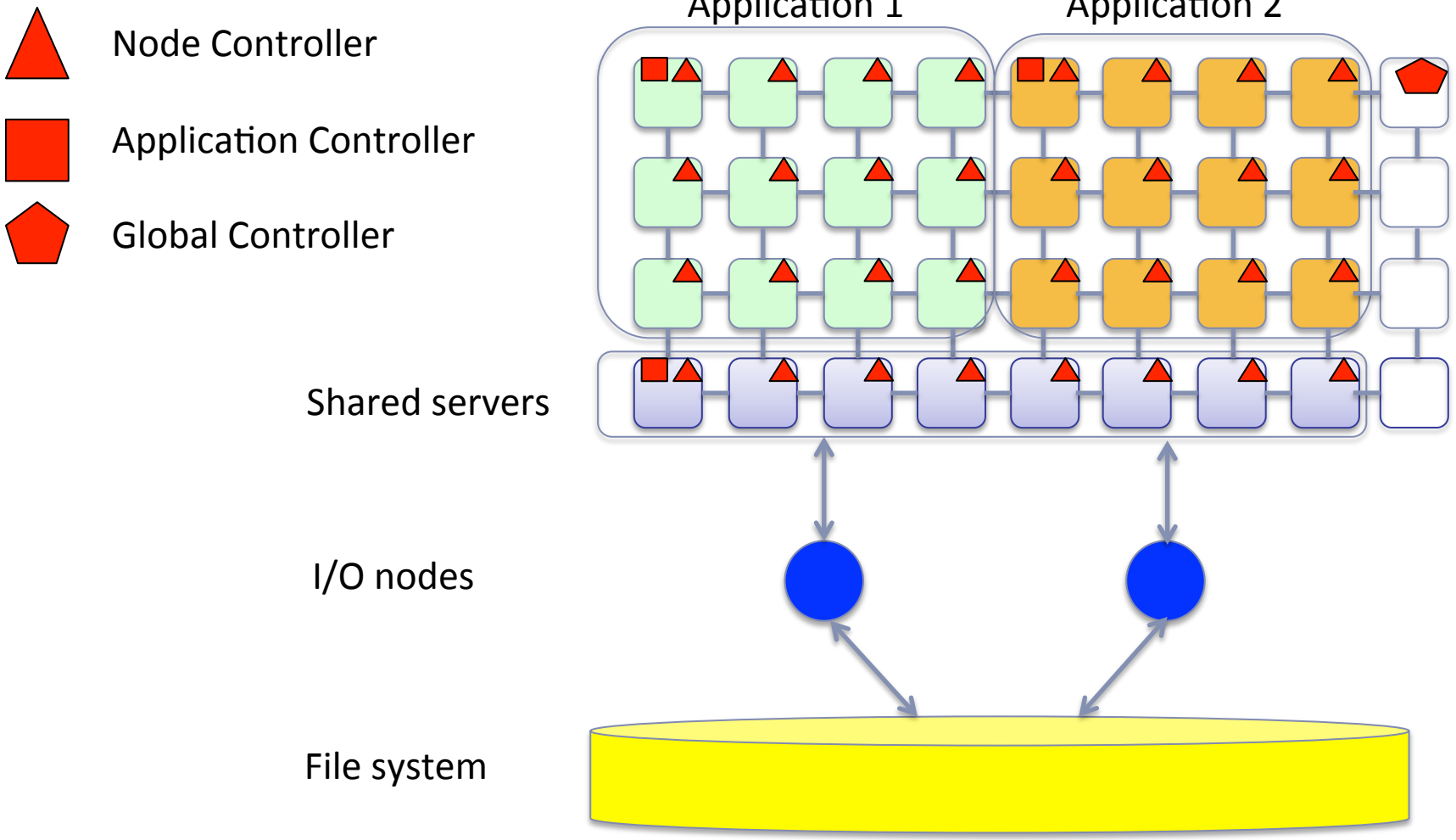
- Local to a node
- Link between one local target and N remote targets
  - Local representation of vertices in the data flow graph
- Two I/O task queues
  - Incoming
  - Outgoing
- Assigned system resources
  - E.g. memory, cores
- Task execution engine
  - Execute task actions
  - Enforce an I/O scheduling policy



Outline

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

CLARISSE hierarchical control infrastructure



CLARISSE control backplane

- Control path: 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
- Hierarchical control
  - Global controller
  - Application controller
  - Node controller
  - All nodes participate in control

Outline

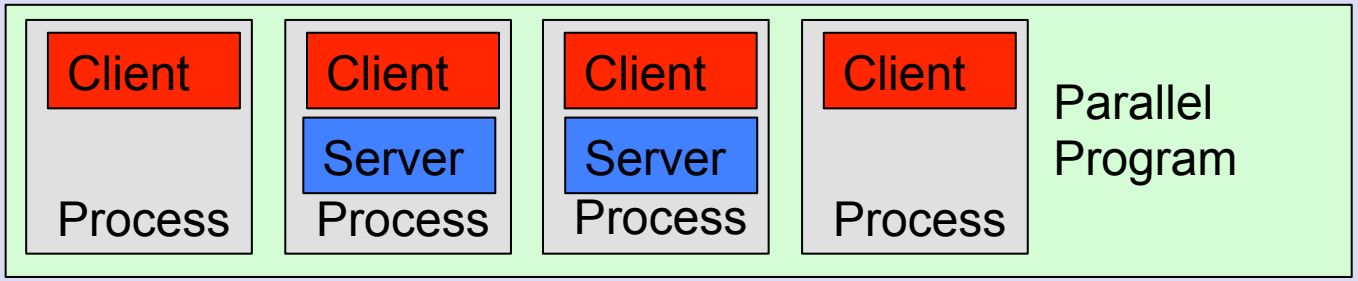
- CLARISSE overview
- CLARISSE data plane
- CLARISSE control plane
- **Deployment**
- Case studies
- Conclusions
- Future work

Deployment

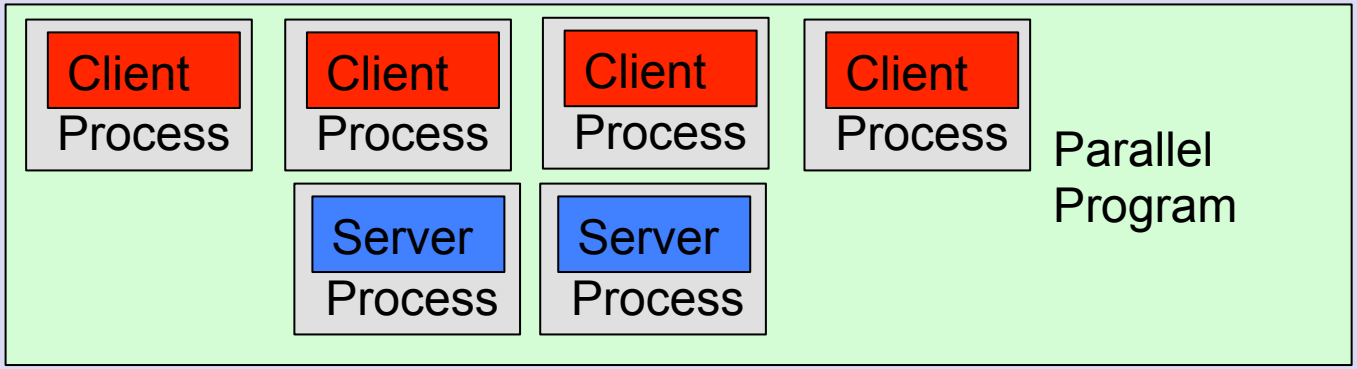
- A process can be:
  - A CLARISSE server (a process that serves remotely data access calls)
  - A CLARISSE client (a process that issues data access calls)
  - Both a server and a client
- Three combinations of parallel servers / parallel clients
  1. Coupled client-server
     - Client and server run in the same process (multi-threaded or not)
     - Locally shared memory
  2. Intra-application decoupled
     - One parallel application
     - Separate client and server processes
  3. Inter-application decoupled
     - Connect different parallel applications or applications to parallel storage systems
     - Construct parallel workflows

Clarisse deployment

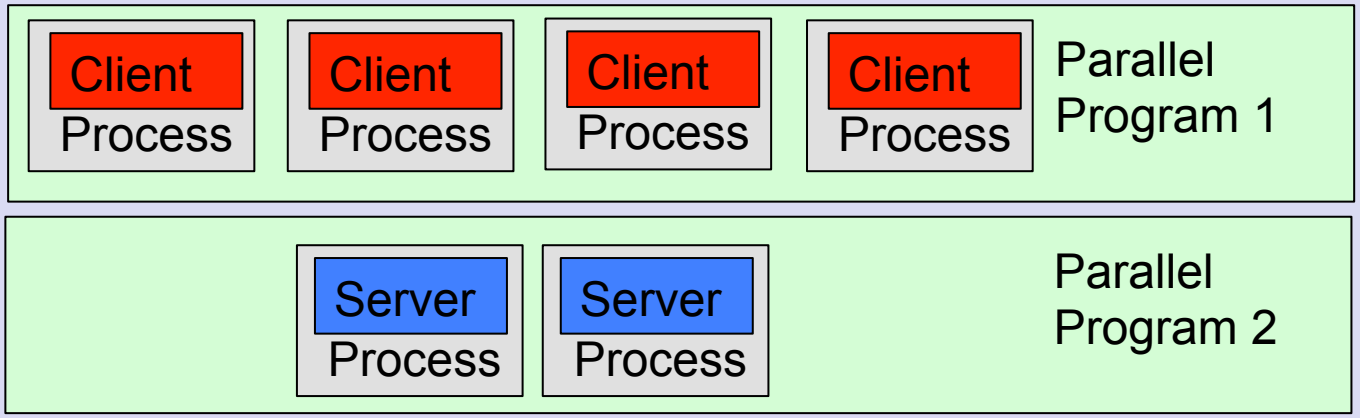
1. Coupled client/ server



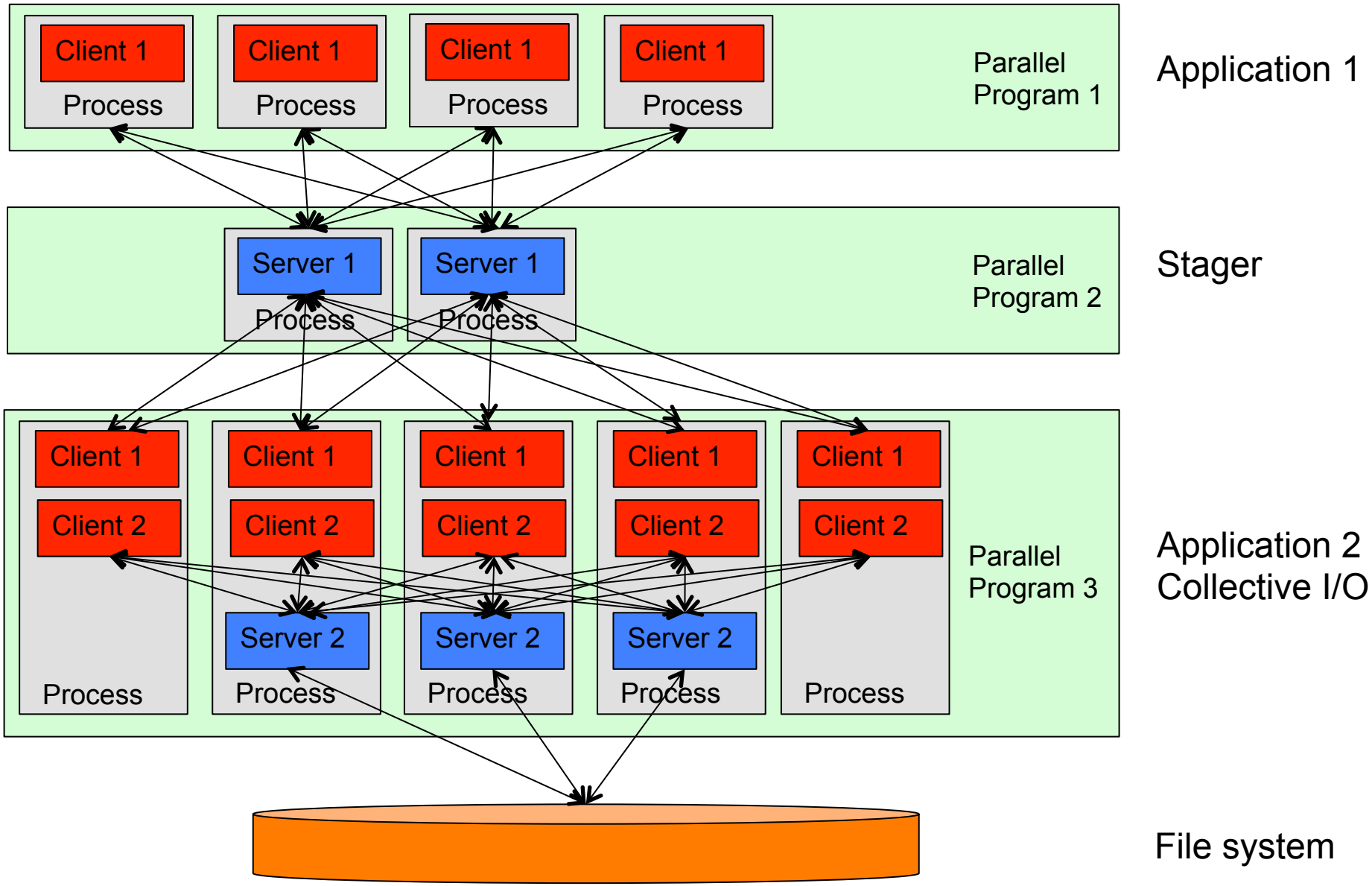
2. Intra-application decoupled



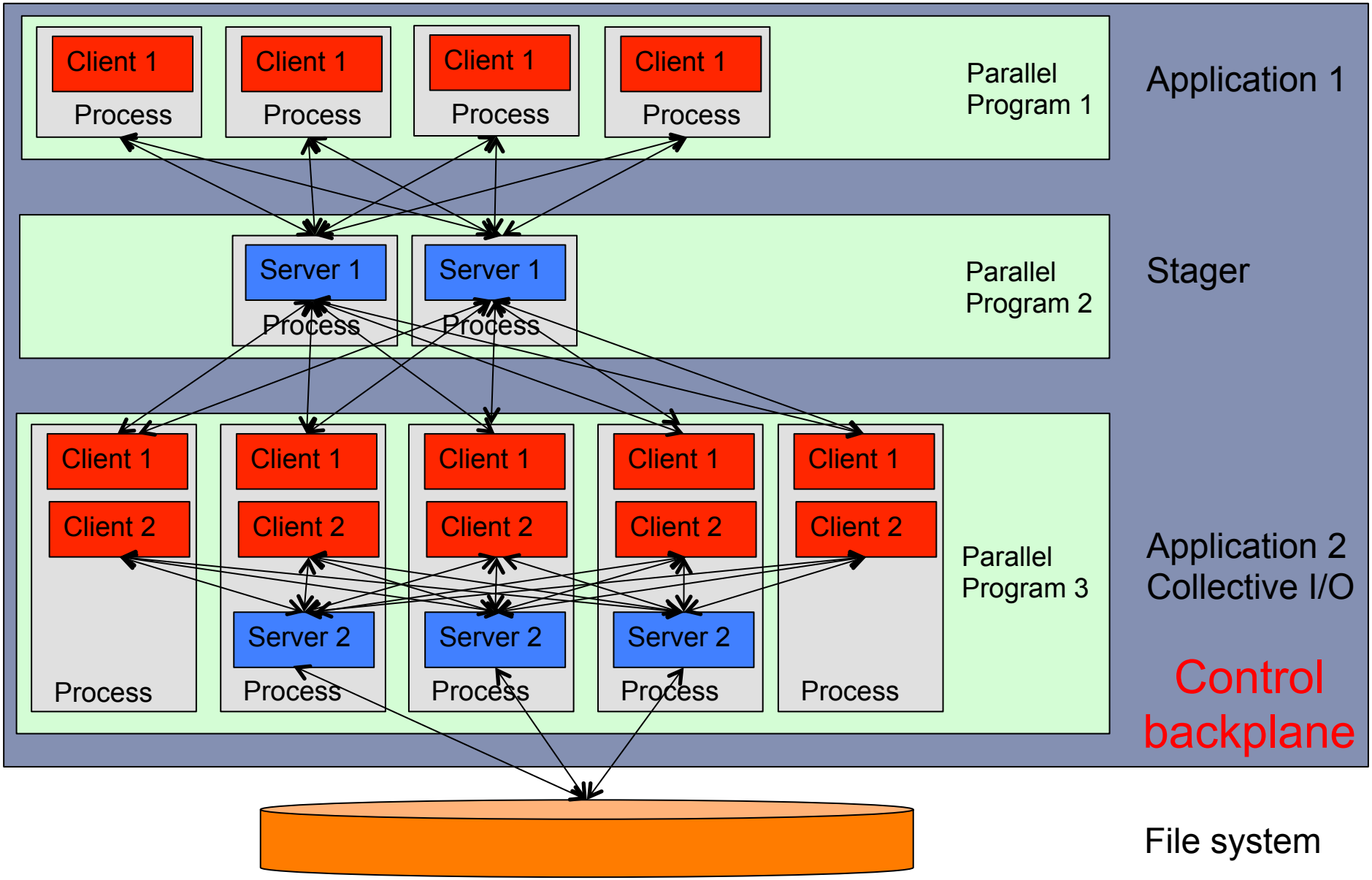
3. Inter-application decoupled



Clarisse parallel data flow example



Clarisse parallel data flow example

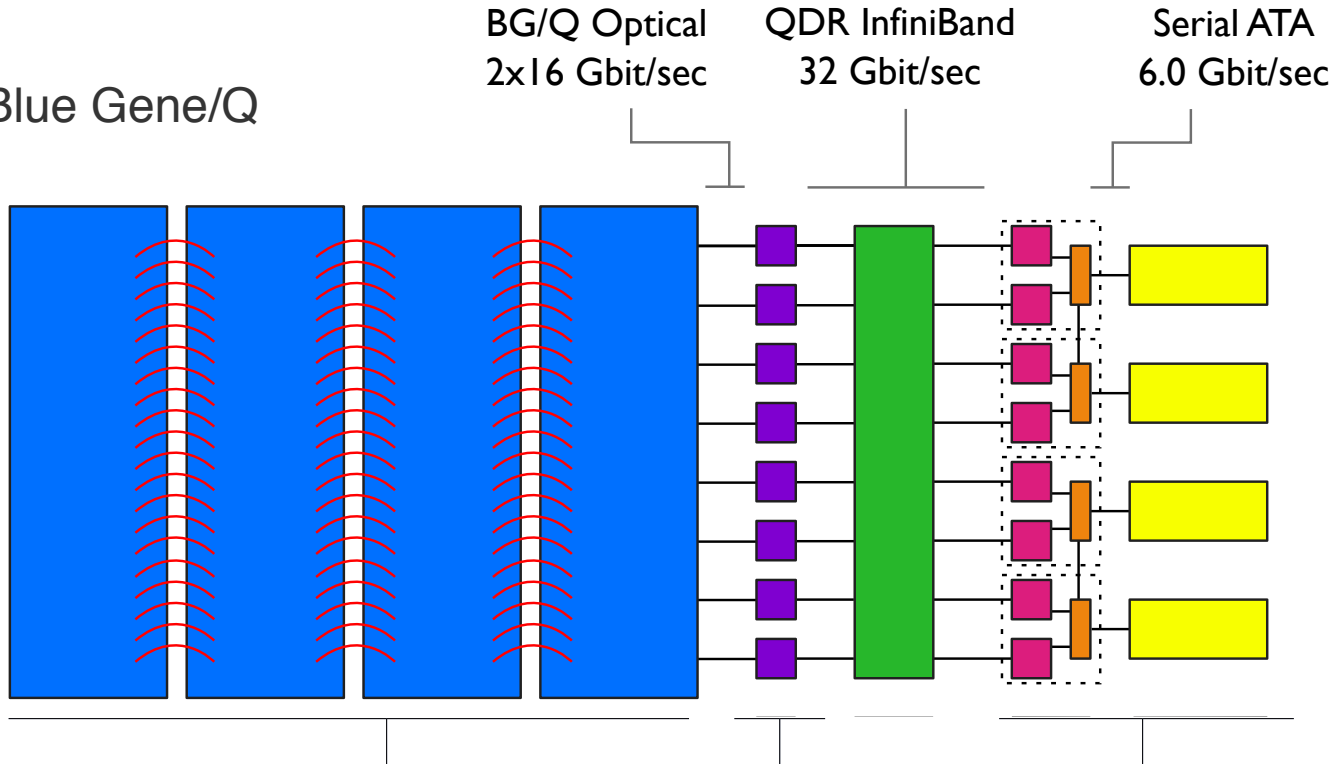


Outline

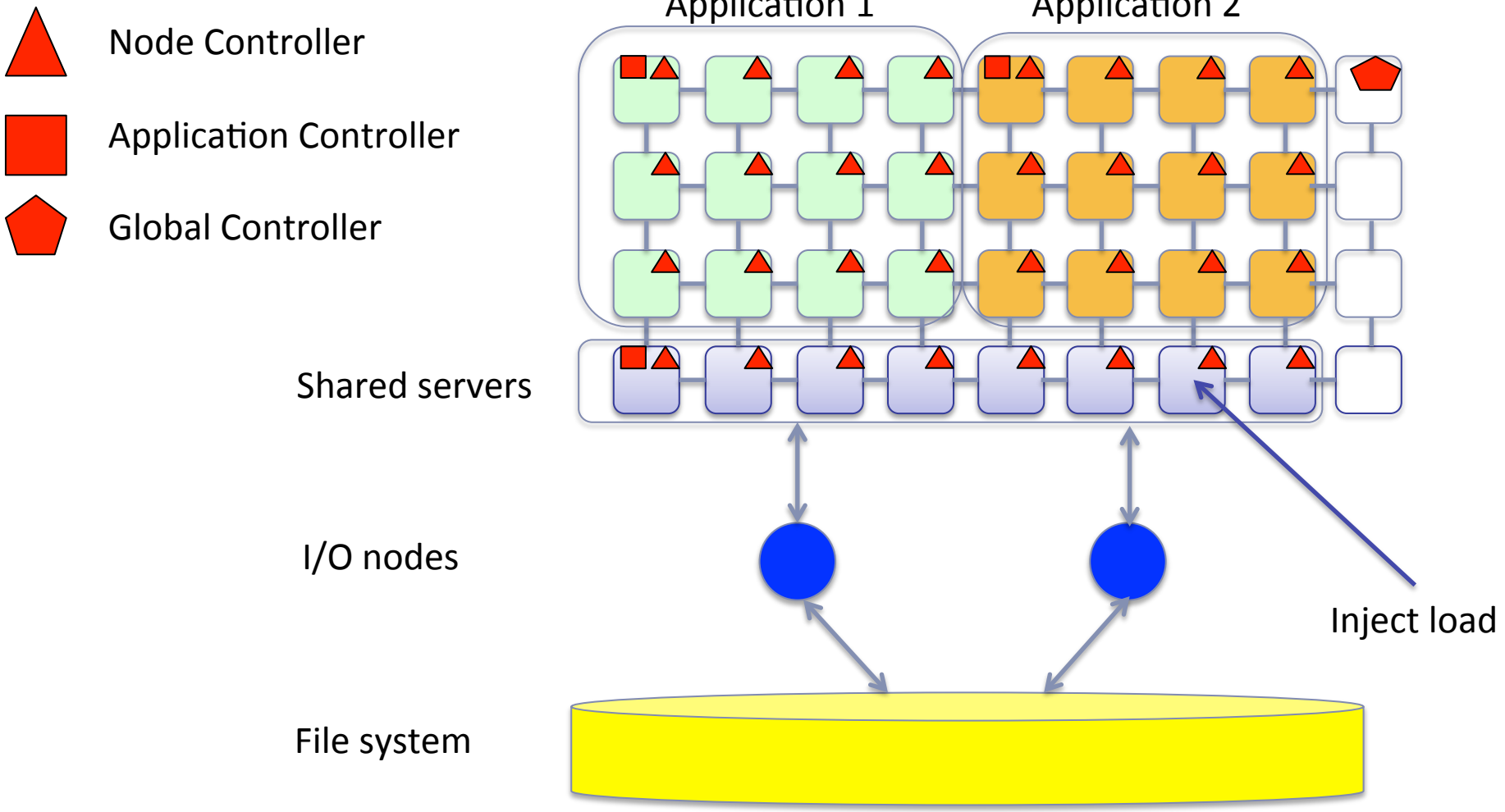
- CLARISSE overview
- CLARISSE data plane
- CLARISSE control plane
- Deployment
- **Case studies**
- Conclusions
- Future work

Case studies

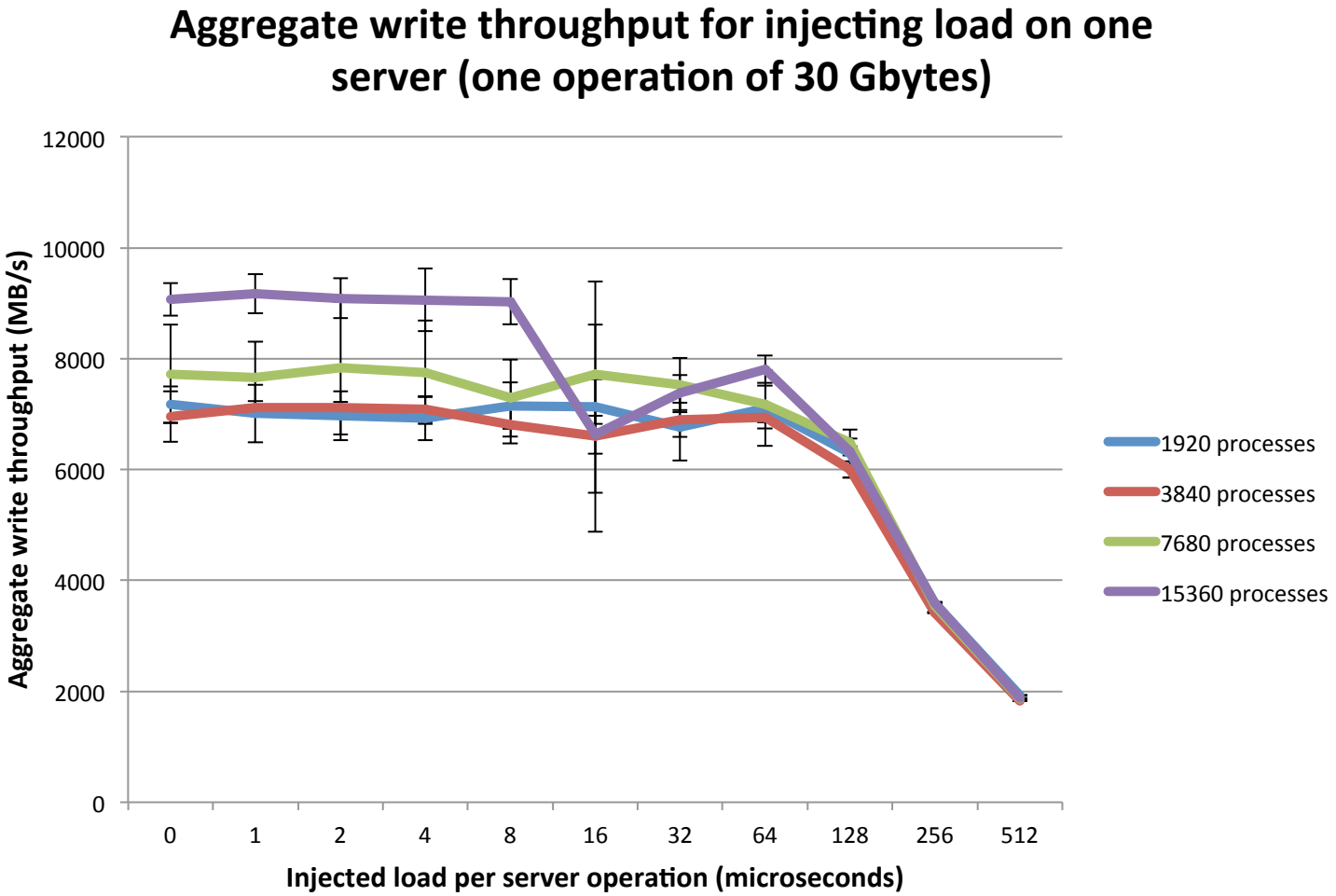
- Dynamic removal of loaded server
- Parallel I/O scheduling
- Vesta Blue Gene/Q



CLARISSE hierarchical control infrastructure

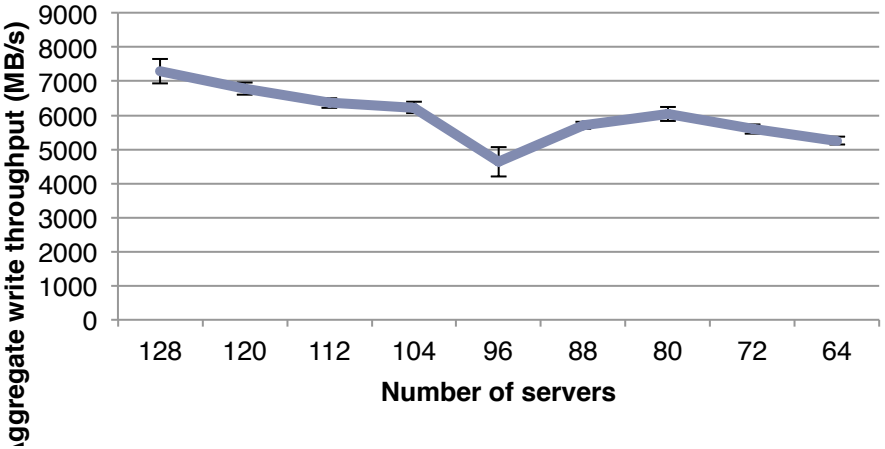


Load injection at 1 server (1 application)

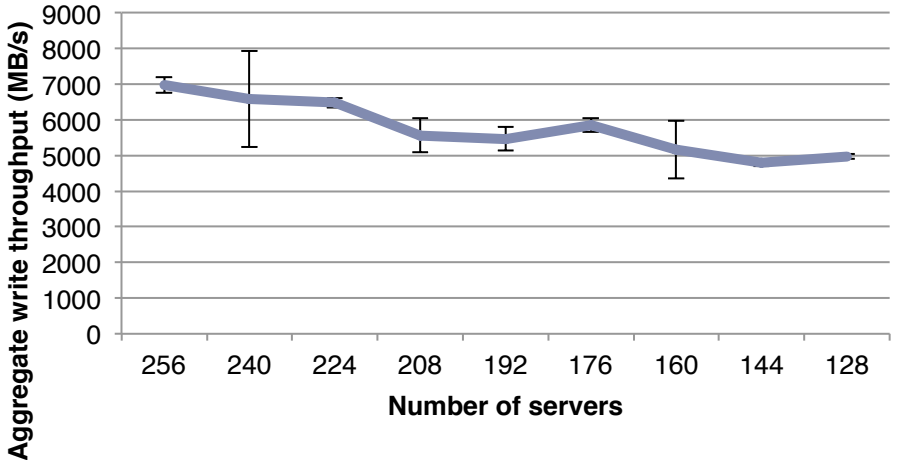


Scale-down number of servers (1 application)

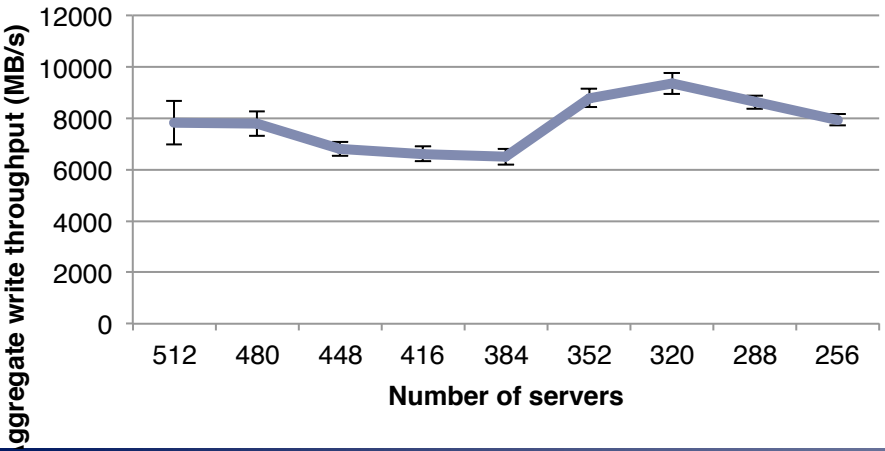
Aggregate write throughput for 1920 processes



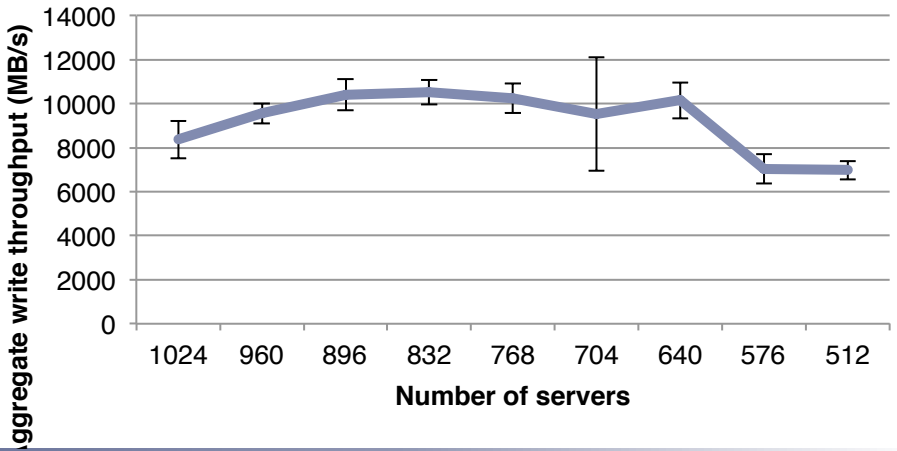
Aggregate write throughput for 3840 processes



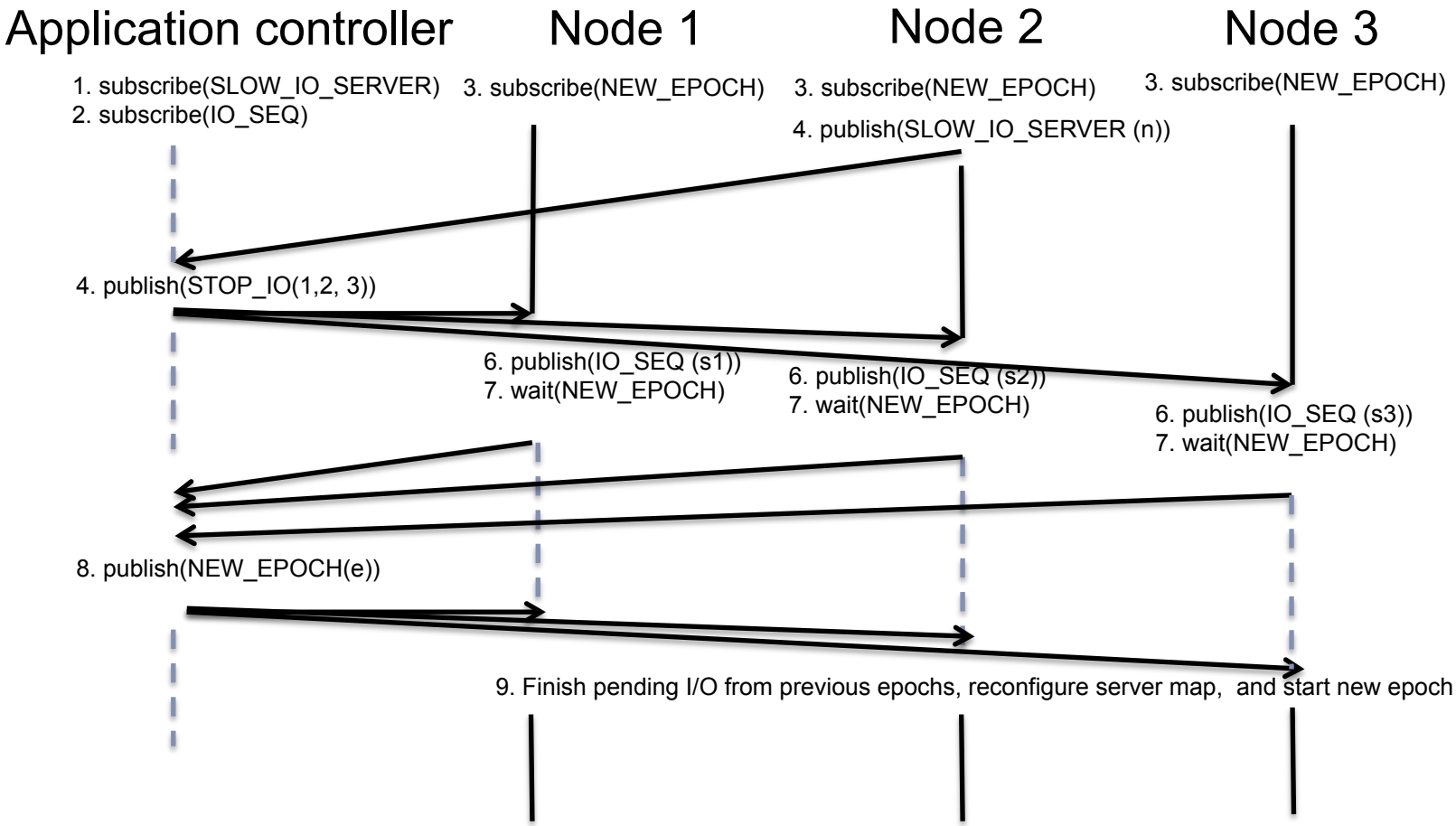
Aggregate write throughput for 7980 processes



Aggregate write throughput for 15360 processes



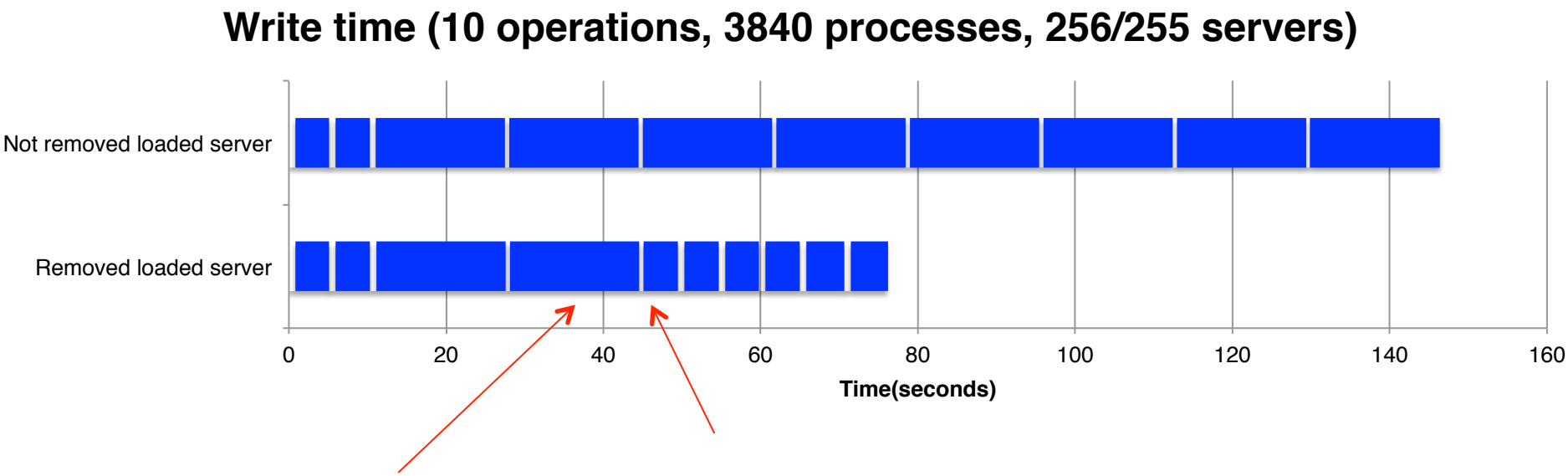
Dynamic removal of loaded server



Dynamic removal of loaded server

- 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

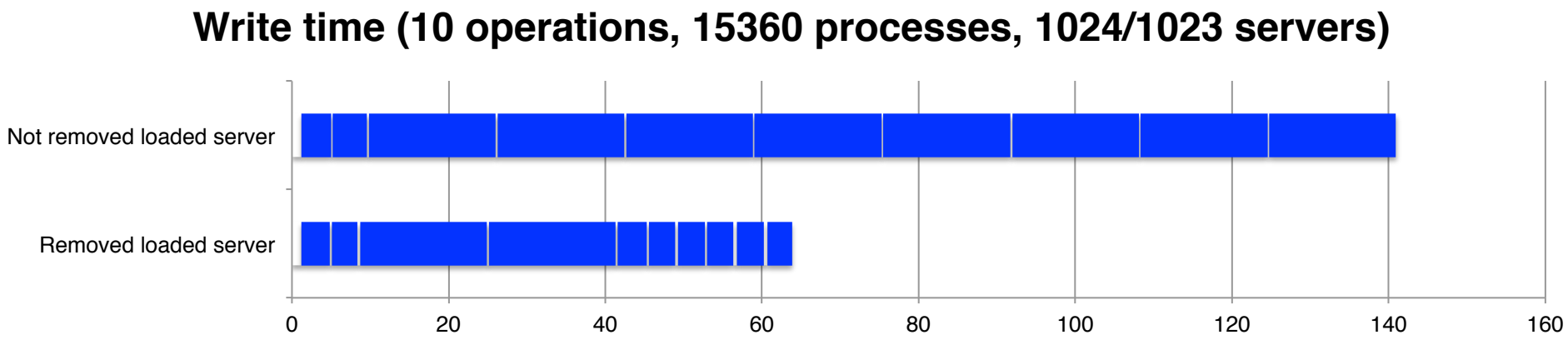
Dynamically scaling-down



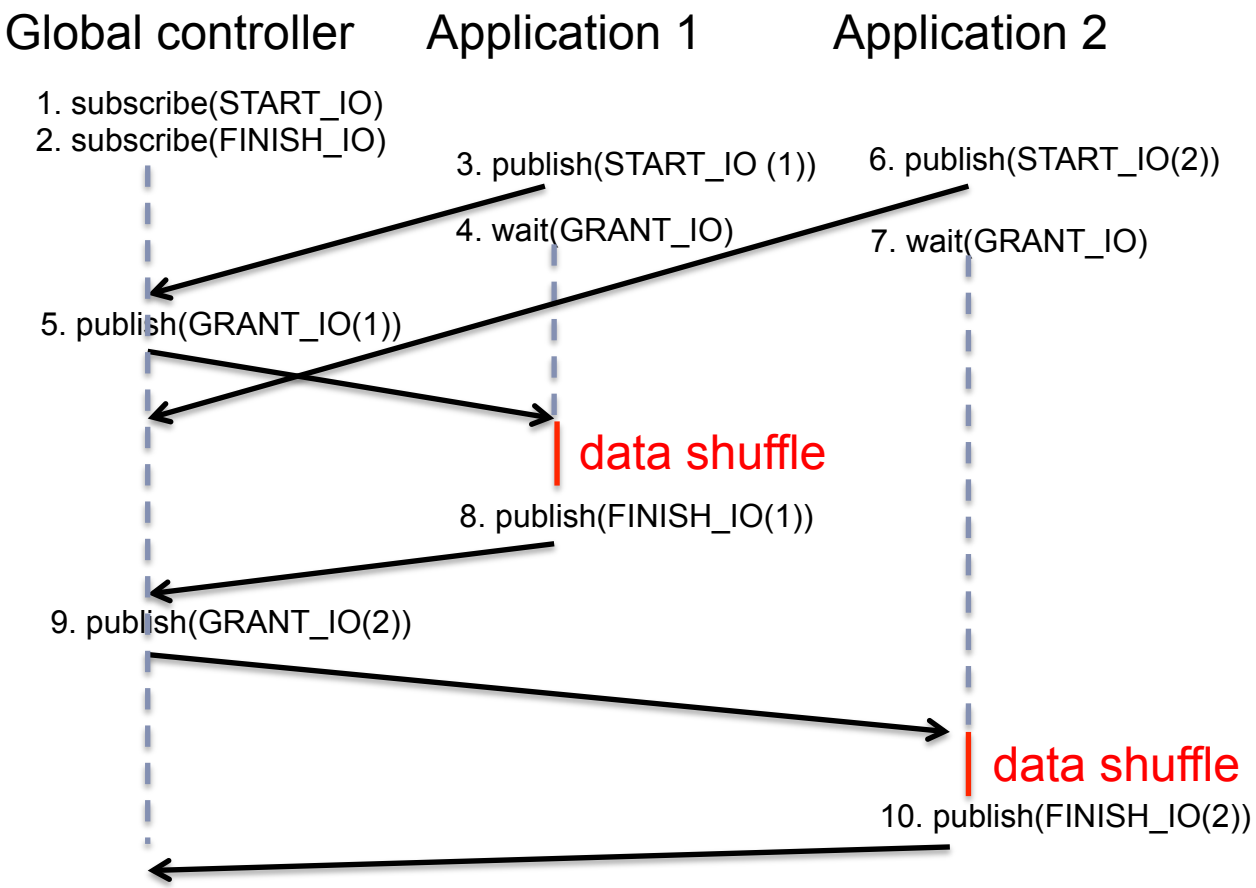
Detect loaded server
Reconstruct server map

New epoch with fewer servers

Dynamically scaling-down



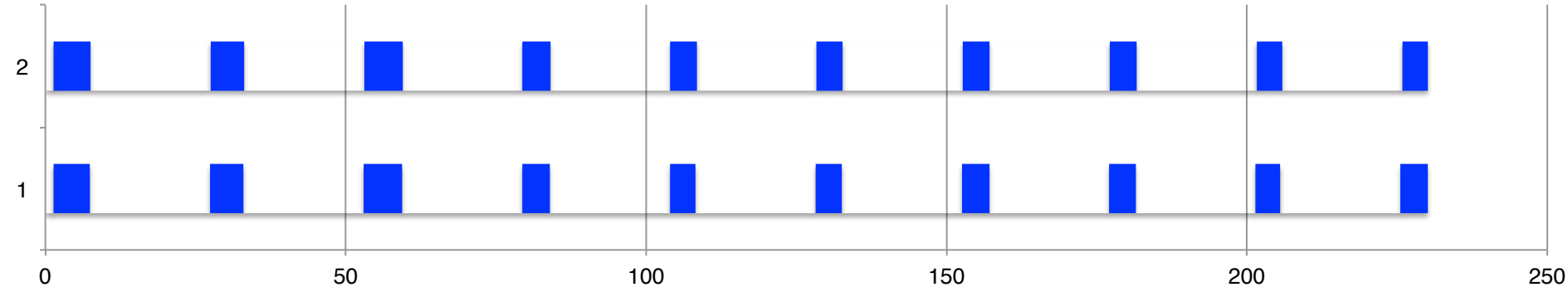
Parallel I/O scheduling: FCFS



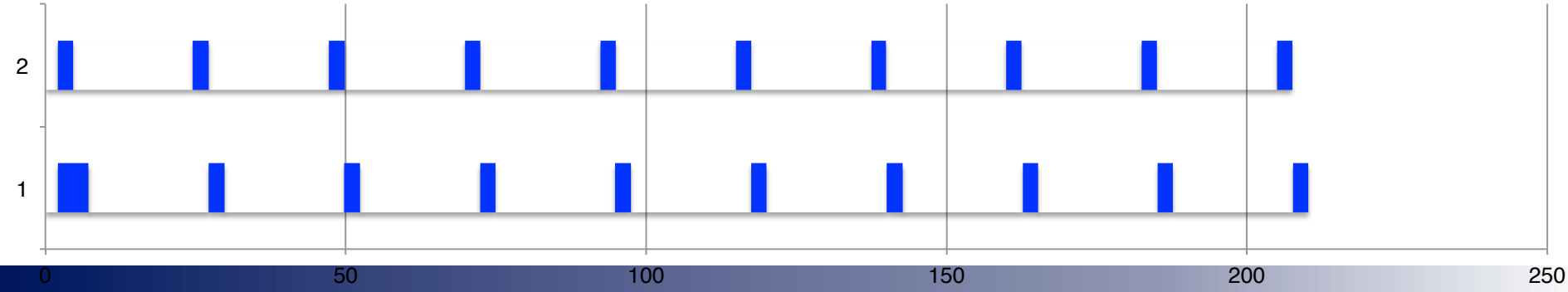
Parallel I/O scheduling

- 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

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

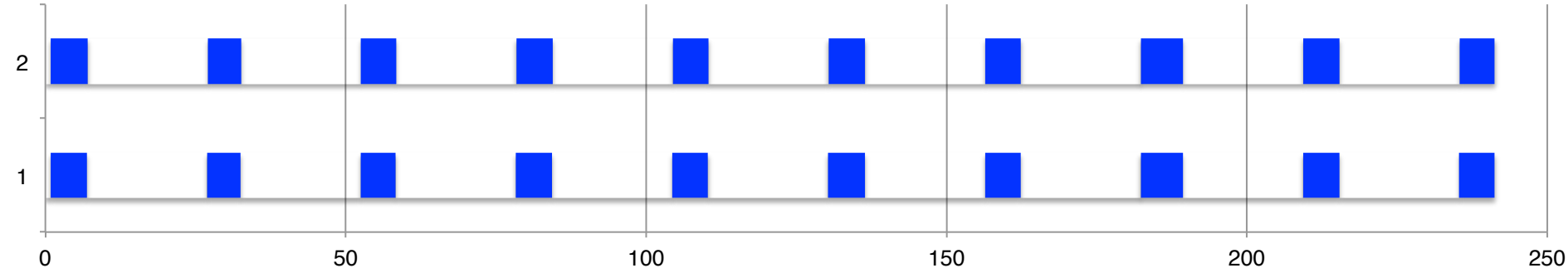


Write timeline for two parallel clients with 960 processes each - FCFS scheduling

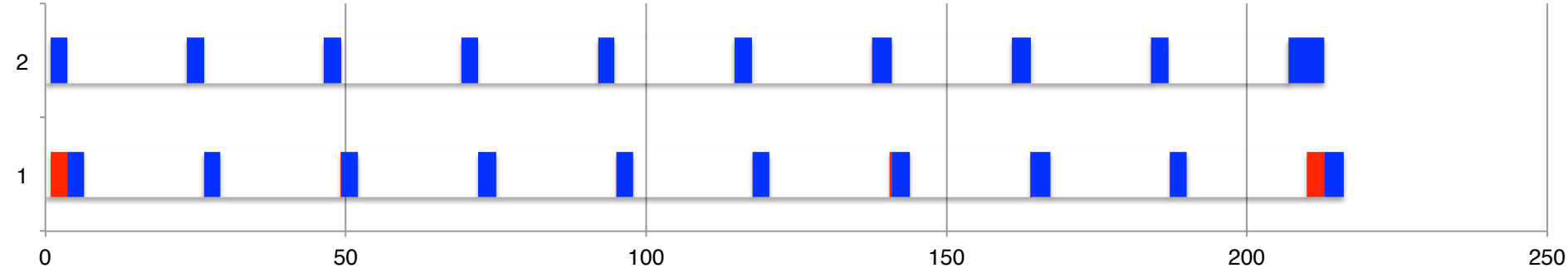


FCFS scheduling versus no scheduling

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



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



Conclusions

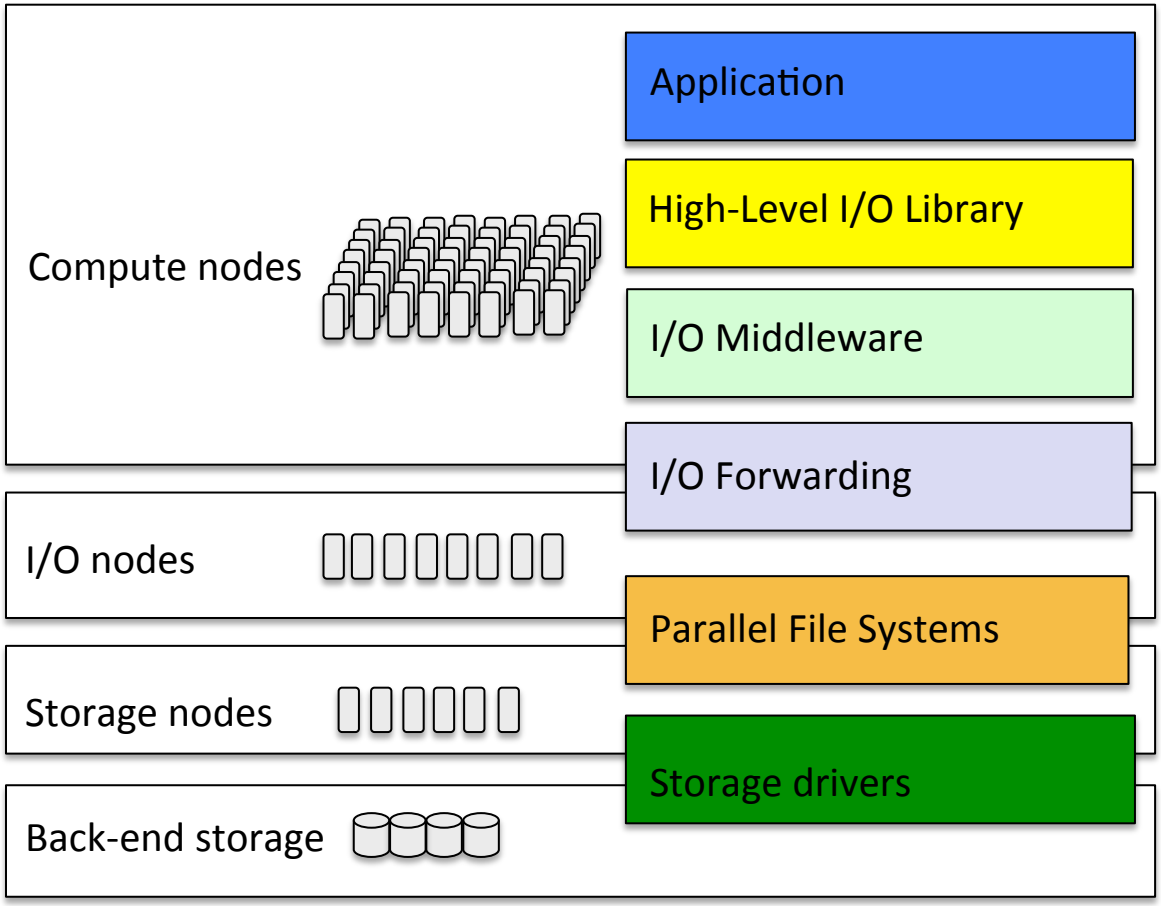
- CLARISSE: Middleware for data staging coordination
- Separation of data and control
- Hierarchical control
- Significant benefits
  - Load/Fault aware sever-scale down
  - Parallel I/O scheduling
- Scalable load and fault monitoring is required

Ongoing and future work

- Topology-aware server/aggregator placement - JL
  Colaboration with E. Jeannot, F. Tessier (INRIA), V. Vishnavath (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

Current problems of storage I/O stack



- Long path from compute nodes to final storage impacts performance (latency, throughput)
- 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)

Related work

- Software Defined Networking (e.g. Open Flow): global control based on separation of control and data flow
- I/O Flow (Microsoft Research): A Software Defined Storage Architecture for virtualized data centers
- Fast Forward (Intel et al.): redesign of the storage I/O stack
- Argo (Argonne et al.), Hobbes (Sandia et al.): system software for exascale based on an OS/Run-time environment
- Decaf (Argonne): decoupling of tightly coupled workflows
- In-situ and in-transit processing: Data Spaces (Rutgers), Flexpath (Georgia Tech), FlowVR (INRIA), Damaris (INRIA), Glean (Argonne)

Blue Gene/Q system at ANL