TEXAS TECH UNIVERSITY™

SC11 Seattle, WA Connecting communities through HPC

Towards Scalable I/O Architecture for Exascale Systems

Presented by

Yong Chen

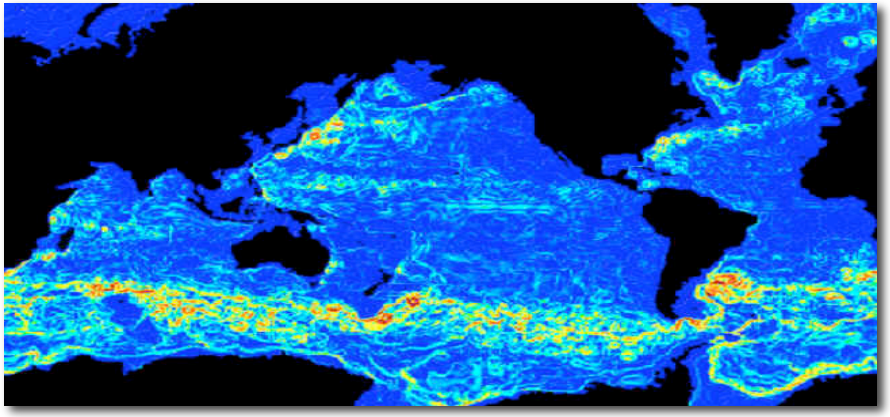
Director, Data-Intensive Scalable Computing Laboratory (DISCL)

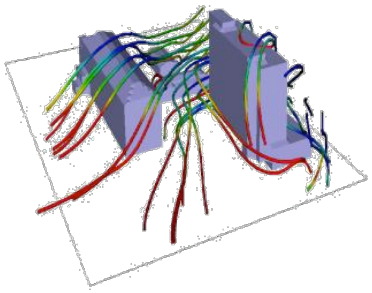
Computer Science Department

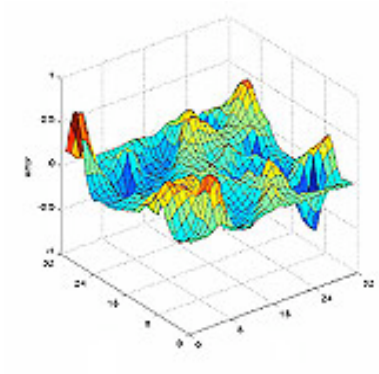
Texas Tech University

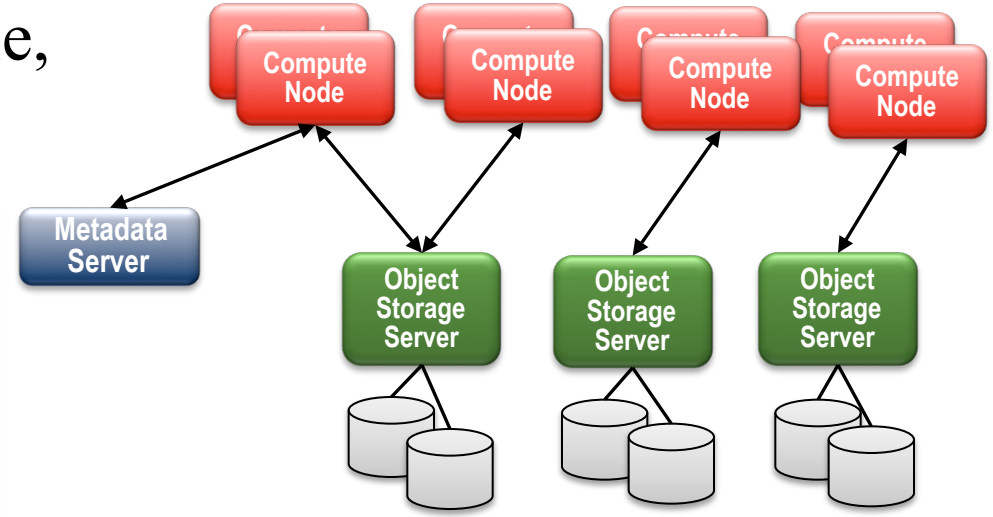
I/O for Large-scale Scientific Computing

- Reading input and restart files
- Reading and processing large amount of data
- Writing checkpoint files
- Writing visualization, movie, history files









Scientific Applications Trend

- Applications tend to be data intensive
  - *Scientific simulation, data mining, large-scale data processing, etc.*
  - *A GTC run on 29K cores on the Jaguar machine at OLCF generated over 54 Terabytes of data in a 24 hour period*
- Pressure on the I/O system capability substantially increases

Data requirements for selected INCITE applications at ALCF

PI	Project	On-Line Data	Off-Line Data
Lamb, Don	FLASH: Buoyancy-Driven Turbulent Nuclear Burning	75TB	300TB
Fischer, Paul	Reactor Core Hydrodynamics	2TB	5TB
Dean, David	Computational Nuclear Structure	4TB	40TB
Baker, David	Computational Protein Structure	1TB	2TB
Worley, Patrick H.	Performance Evaluation and Analysis	1TB	1TB
Wolverton, Christopher	Kinetics and Thermodynamics of Metal and Complex Hydride Nanoparticles	5TB	100TB
Washington, Warren	Climate Science	10TB	345TB
Tsigelny, Igor	Parkinson's Disease	2.5TB	50TB
Tang, William	Plasma Microturbulence	2TB	10TB
Sugar, Robert	Lattice QCD	1TB	44TB
Siegel, Andrew	Thermal Striping in Sodium Cooled Reactors	4TB	8TB
Roux, Benoit	Gating Mechanisms of Membrane Proteins	10TB	10TB

Source: R. Ross et. al., Argonne National Laboratory

High Performance Computing Systems Trend

- The exascale system is projected to appear around 2018
- Anticipated to have millions of nodes, with thousands of cores for each node
- Application teams predicted to process hundreds of terabytes or even petabytes of data in a single run
- Brings significant challenges than ever for the I/O system to meet applications' demand
- The I/O system performance is predicted as one of the most critical challenges

Potential Exascale Computer Design

- Factor changes for peak performance, system size, etc.
- Most significant change is the total concurrency compared to today's system
  - *Will have billions of processes running on millions of nodes to achieve exaflop*
  - *Anticipated as the most challenging issue to achieve an exascale*
- Limited I/O system capability could considerably lower the sustained performance of exascale systems
- Research need in developing an I/O architecture for such concurrency level

Potential Exascale Design for 2018 and Its Relationship to Current HPC designs [VTYR08]

	2010	2018	Factor Change
System Peak	2 Pf/s	1 Ef/s	500
Power	6 MW	20 MW	3
System Memory	0.3 PB	10 PB	33
Node Performance	0.125 Tf/s	10 Tf/s	80
Node Memory BW	25 GB/s	400 GB/s	16
Node Concurrency	12 CPUs	1000 CPUs	83
Interconnect BW	1.5 GB/s	50 GB/s	33
System Size (nodes)	20 K nodes	1 M nodes	50
Total Concurrency	225 K	1 B	4444
Storage	15 PB	300 PB	20
Input/Output Bandwidth	0.2 TB/s	20 TB/s	100

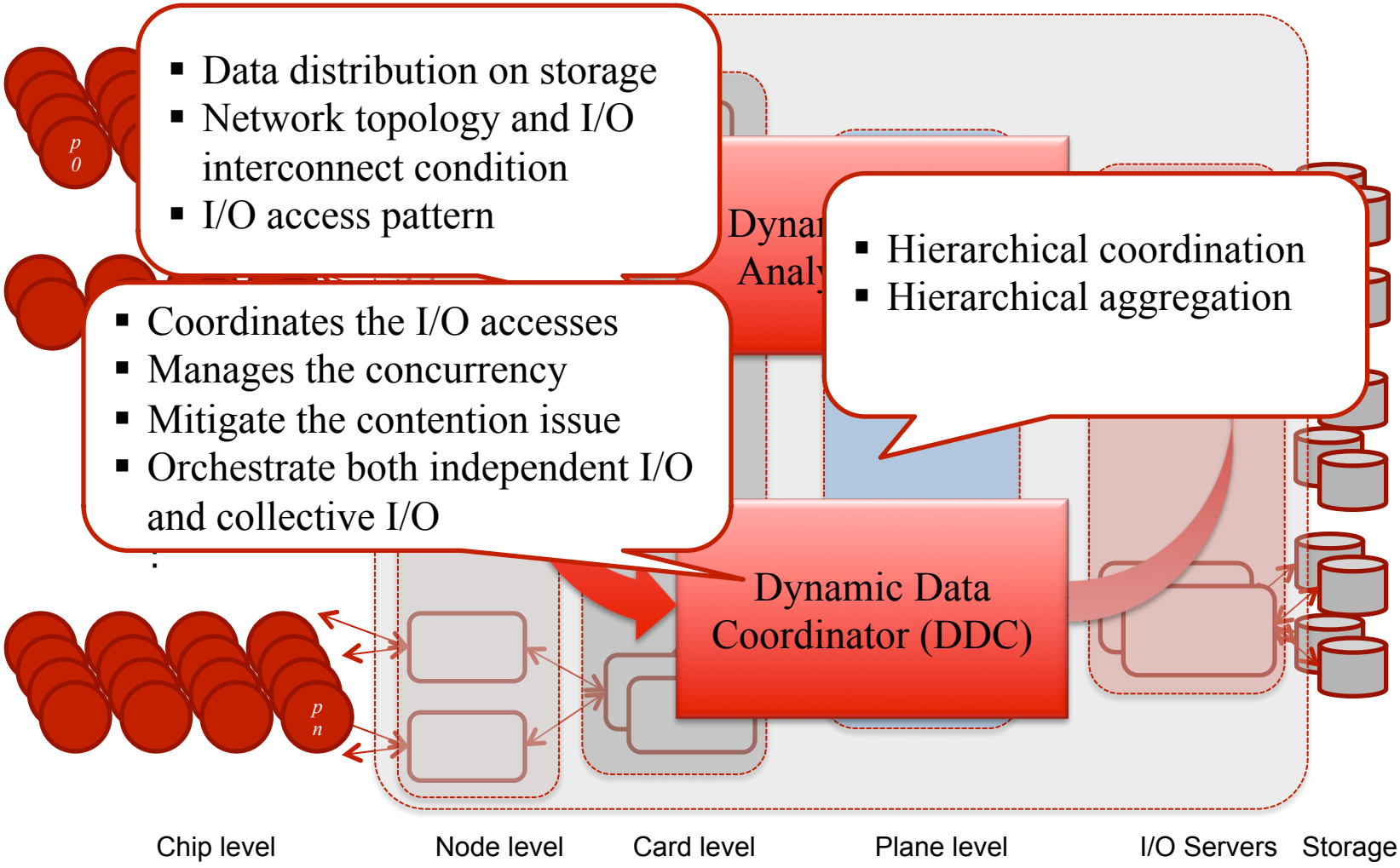
Current HPC I/O Architecture

- HPC I/O system focuses
  - *Exploring parallelism*
  - *Exploring locality*
- Not easily managed/achieved for exascale systems
- Substantial amount of concurrency can cause a critical contention issue at multiple levels
- Destroy the locality and level of parallelism
- Current I/O architecture one-set-for-all and static
- Does not manage the concurrency intelligently and limits the scalability and the potential at an extreme scale

Dynamically Coordinated I/O Architecture

- We propose a dynamically coordinated I/O architecture (or coordinated I/O in short) for exascale systems
- We argue that coordinating I/O accesses is critical and fundamental to exascale systems
  - *According to access pattern, network topology, and data distribution*
- Goals of the coordinated I/O
  - *Manage the substantial amount of concurrency*
  - *Achieve the parallelism and avoid the critical contention issue*
  - *Maintain the locality and avoid the interference*
- An important component for many-task computing paradigm

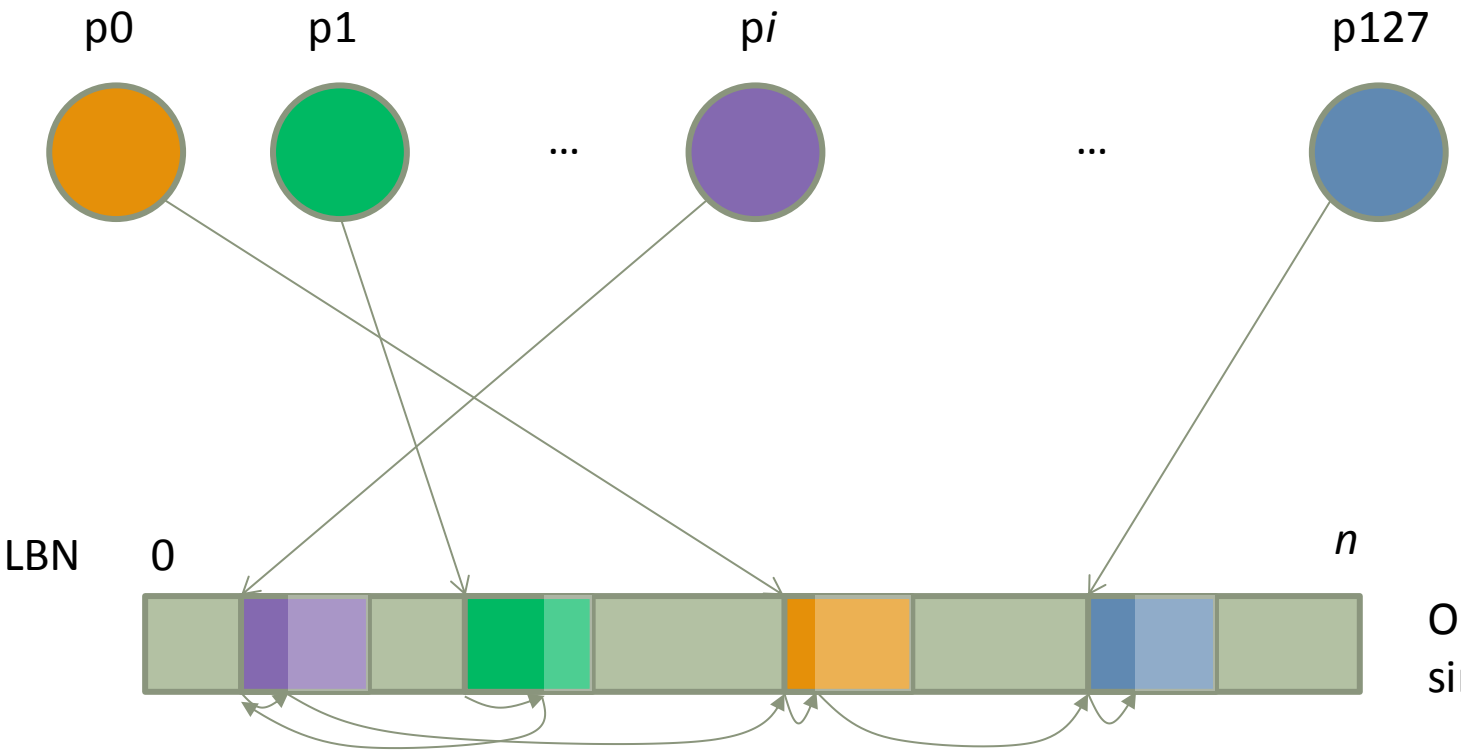
Dynamically Coordinated I/O Architecture



Dynamic Coordination for Collective I/O

- Straightforward form of I/O and is widely used
- Independently by an individual process or any subset of processes of a parallel application
- Independent nature ignores other processes without coordination
- This ignorance of other processes could destroy the locality of requests and hurts overall performance due to contention
- Not only lose the potential benefit of collective I/O optimization, but also deteriorate the I/O performance if without coordination

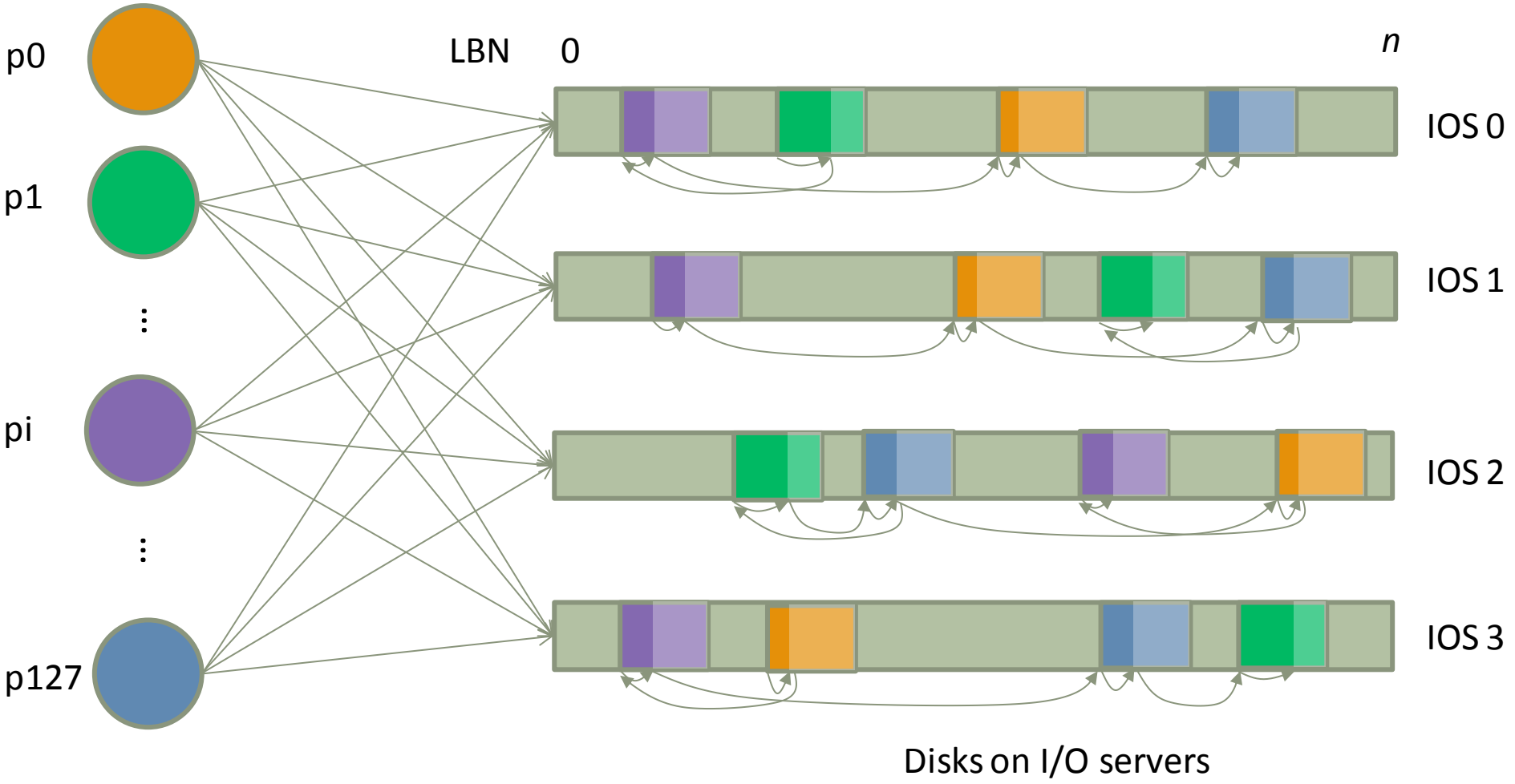
Dynamic Coordination for Independent I/O



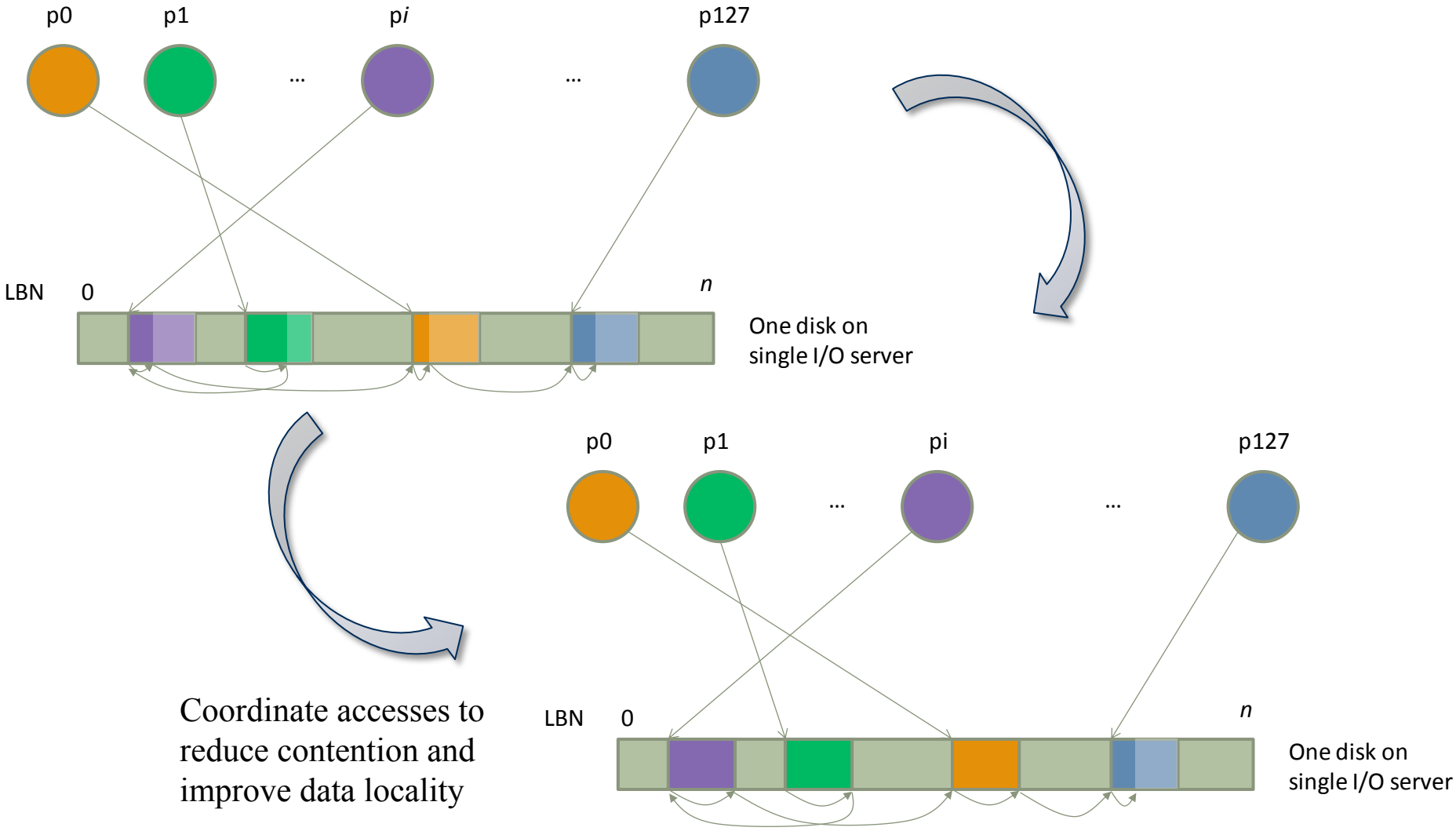
One disk on single I/O server

Lose locality due to contention

Dynamic Coordination for Independent I/O



Dynamic Coordination for Independent I/O

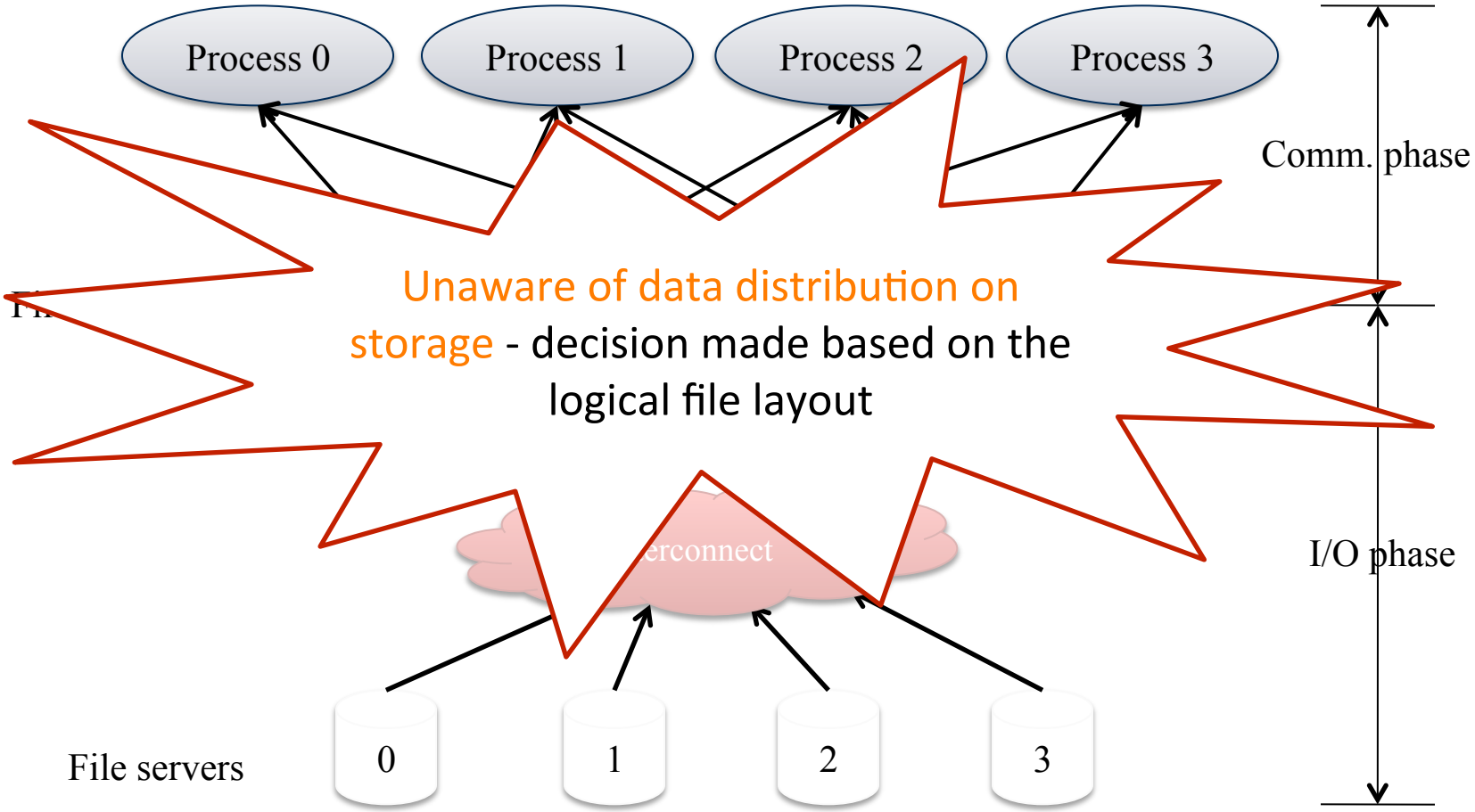


Coordinate accesses to reduce contention and improve data locality

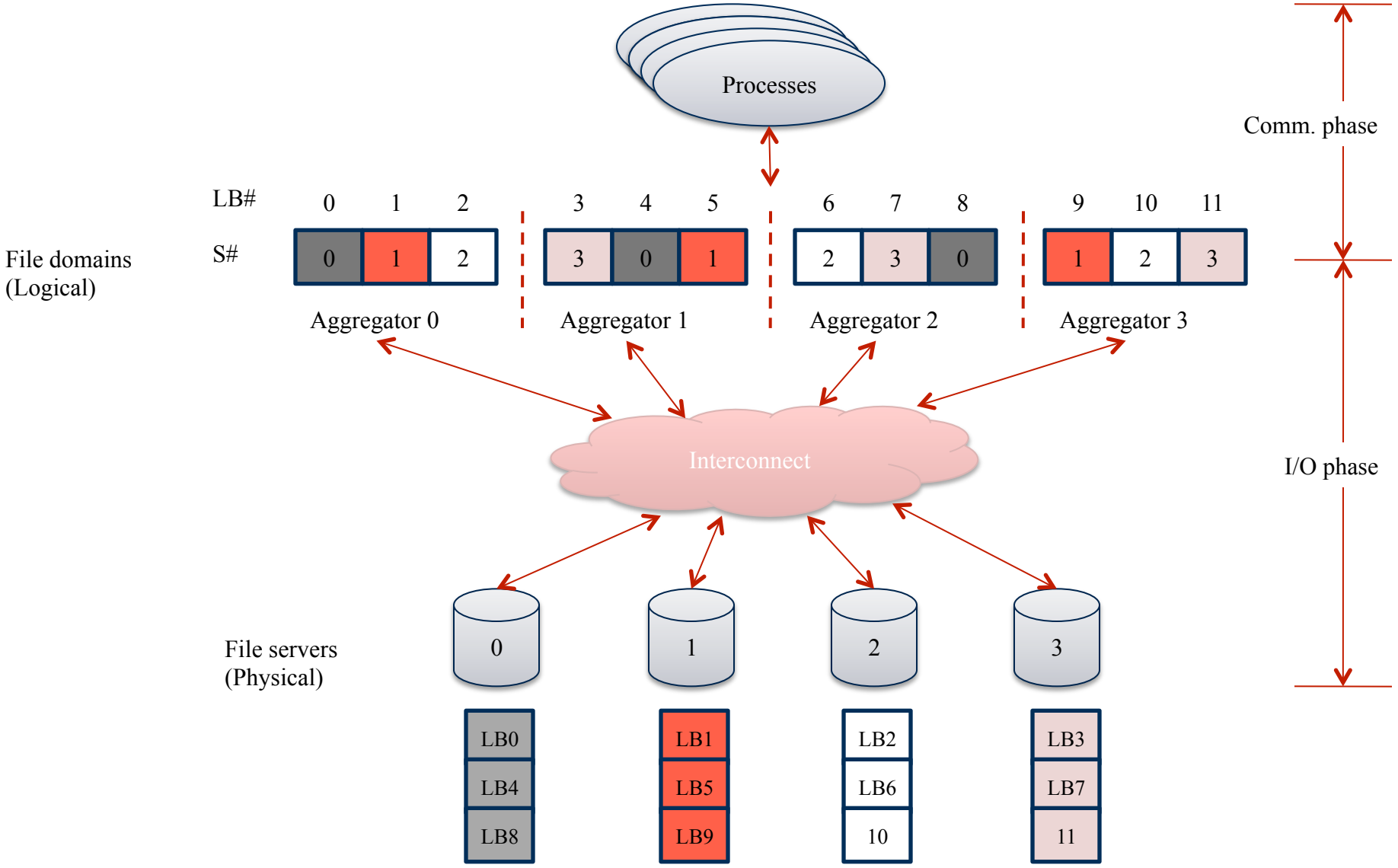
Dynamic Coordination for Collective I/O

- Collective I/O enables a group of processes participate together in reading or writing data
- Allows the requests to be serviced together
- Creates opportunity to optimize I/O by exploring correlations
- Benefits
  - *Filter overlapping and redundant I/O requests from multiple processes*
  - *Merge small requests to large and contiguous requests*
  - *Reduce the number of system calls*

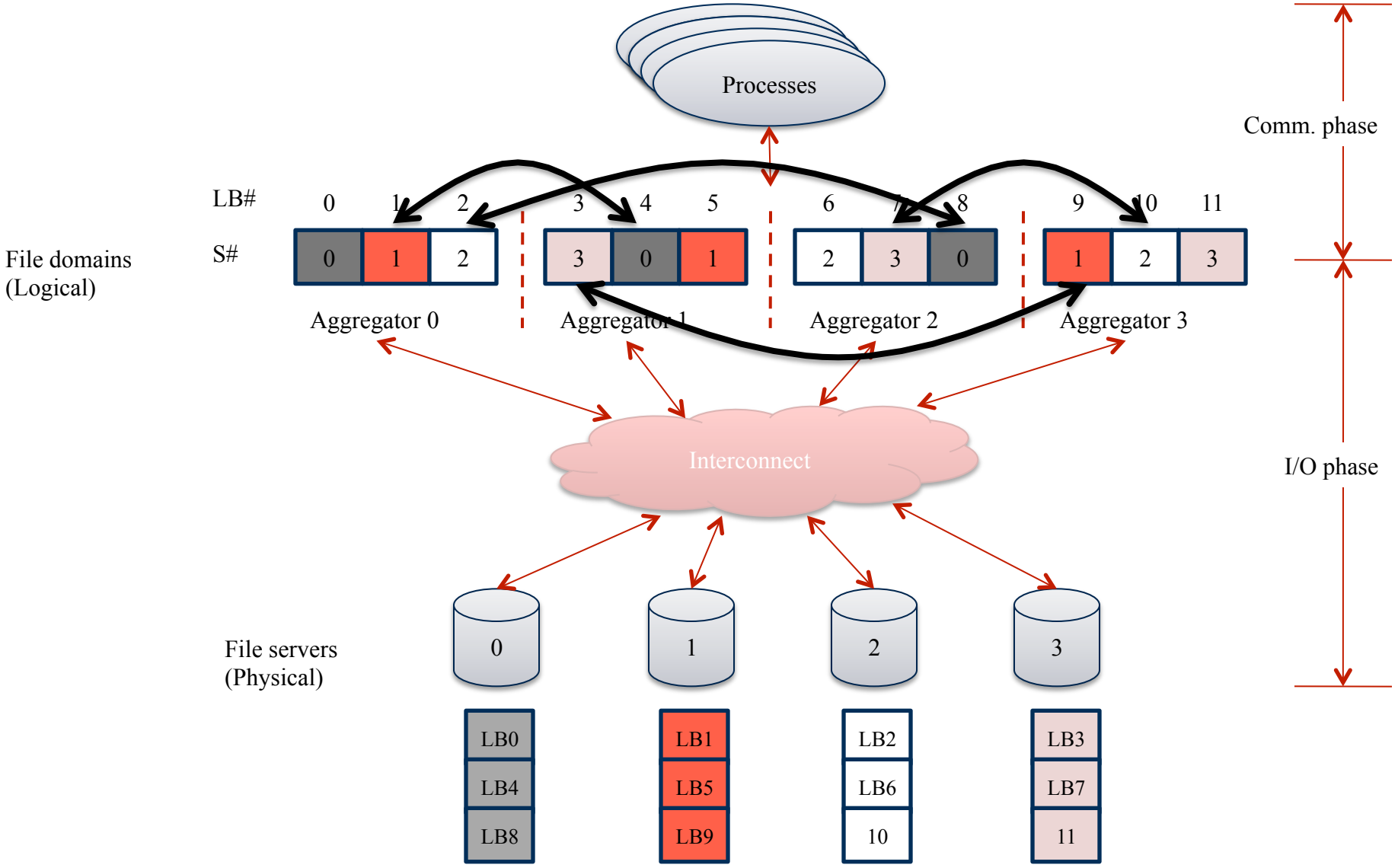
Collective I/O and Two-phase Implementation



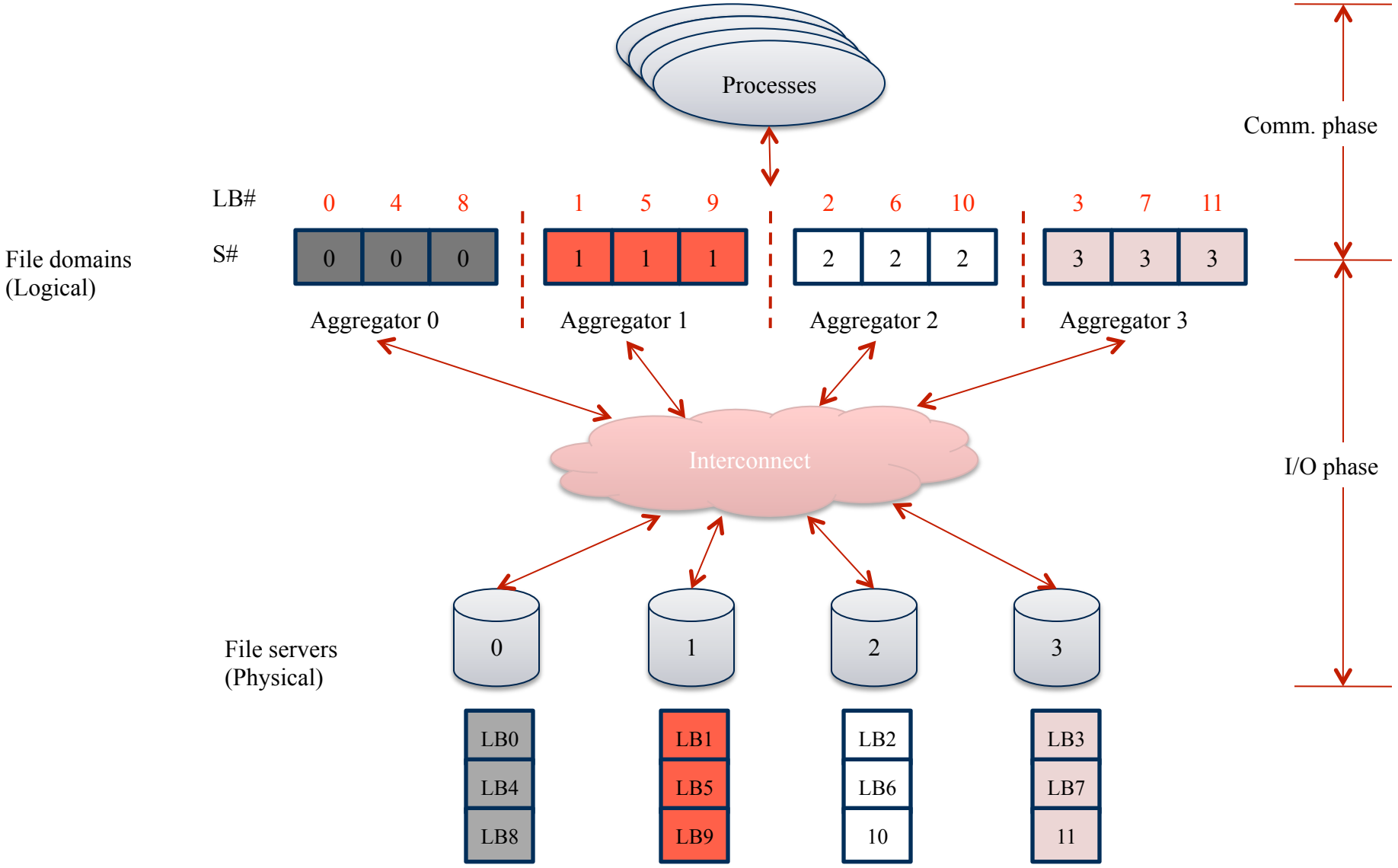
Dynamic Coordination for Collective I/O



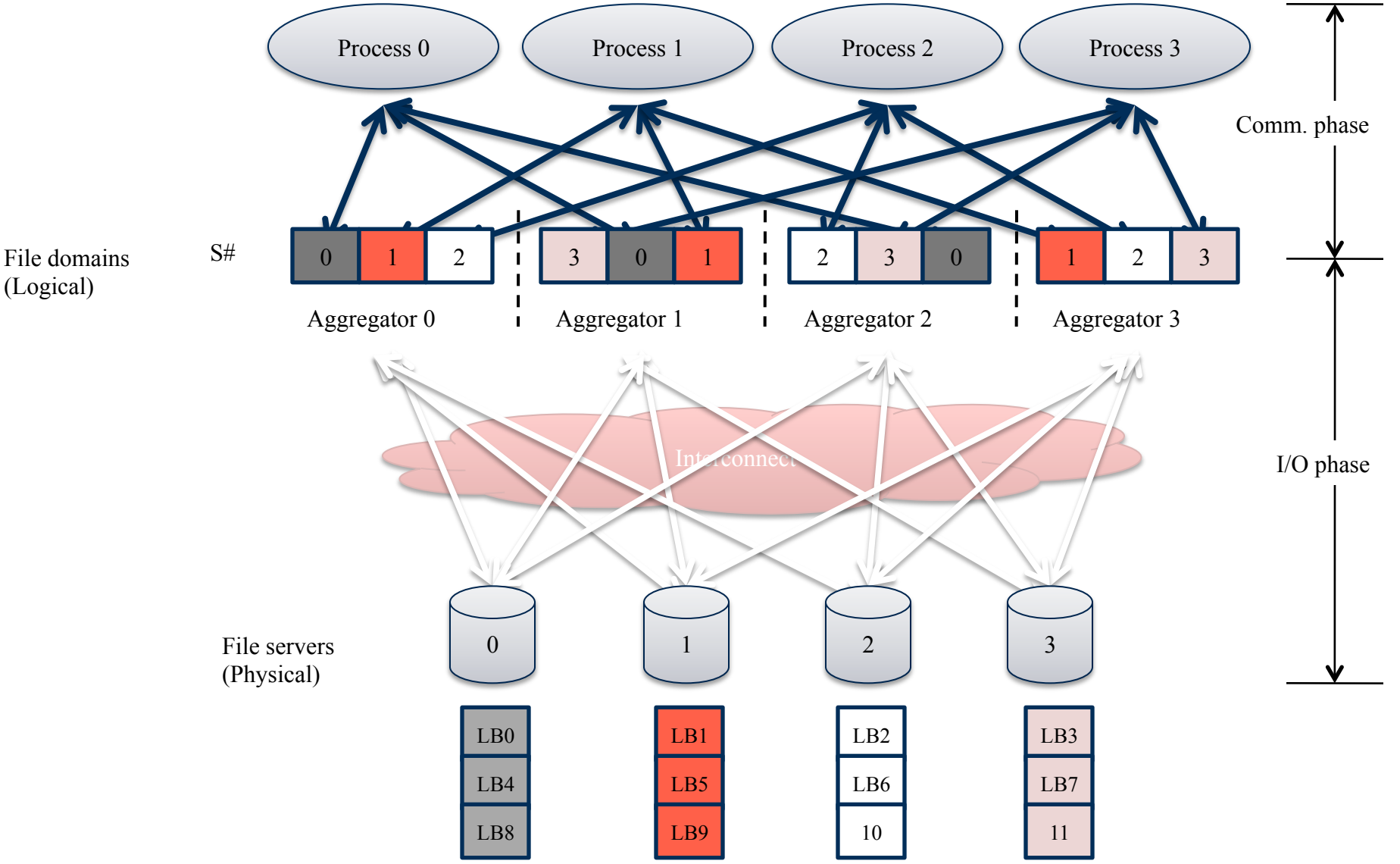
Dynamic Coordination for Collective I/O



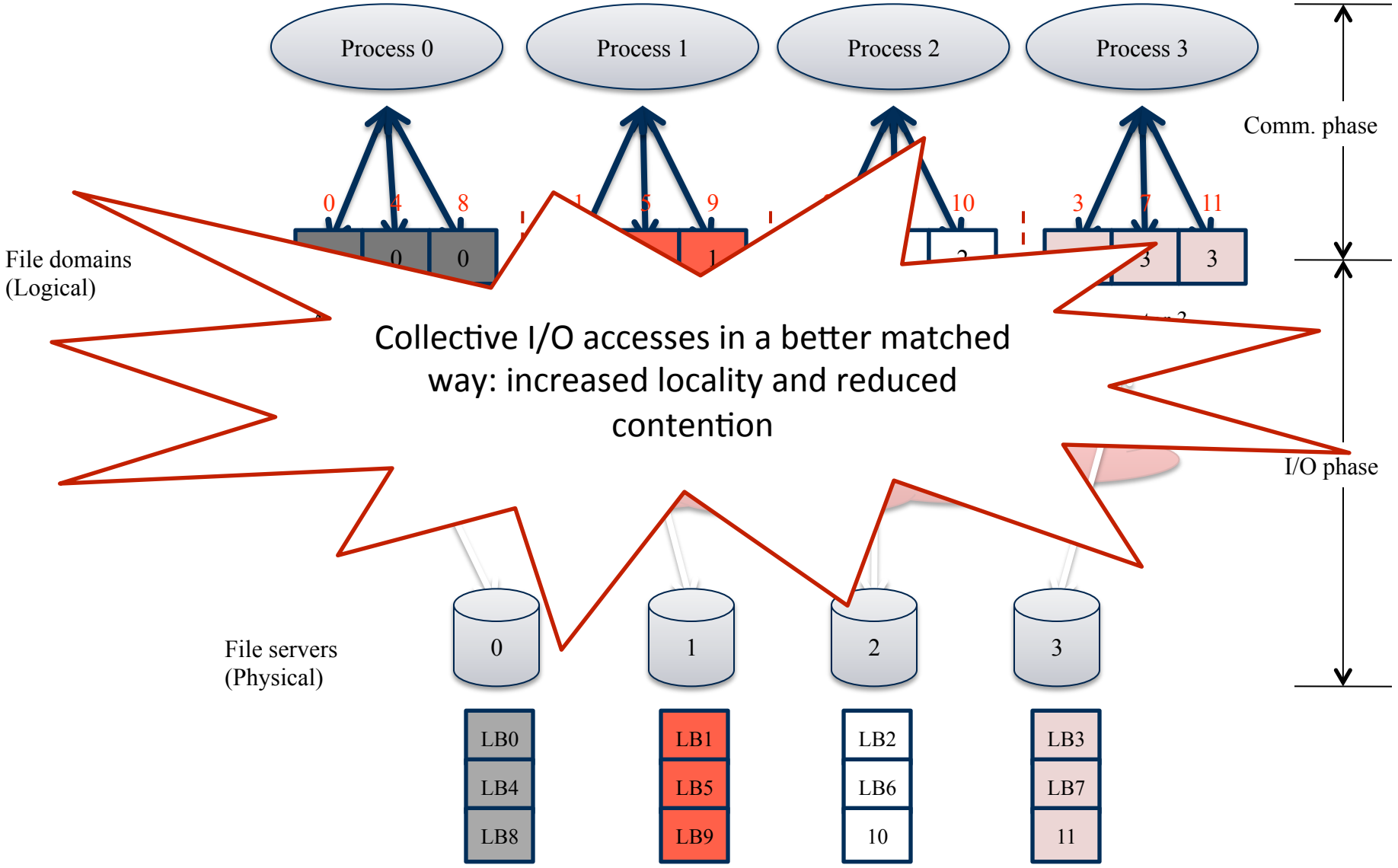
Dynamic Coordination for Collective I/O



Communication and I/O Pattern



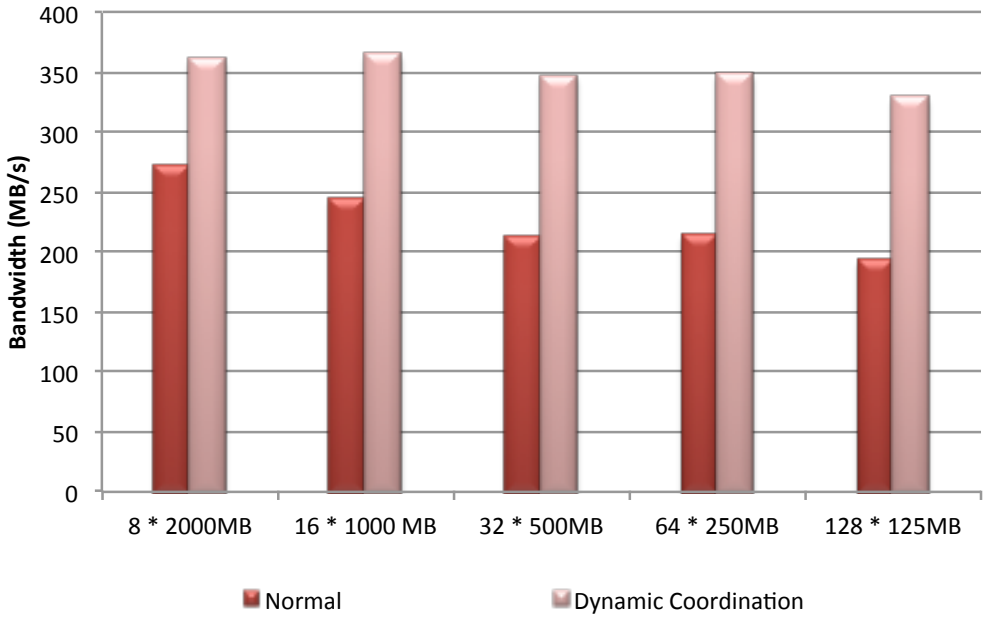
Communication and I/O Pattern



Implementation Strategies

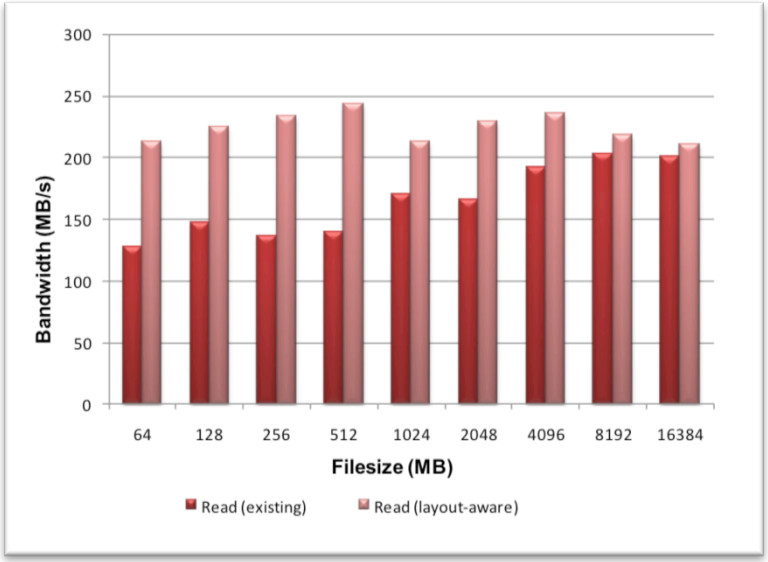
- Data distribution on storage servers can be obtained via parallel file systems API
  - *Happens at the first time an I/O occurs, can then be cached in runtime system*
  - *Such a caching is safe as the data layout of a specific file is determined when it is created and will be static except deleted or explicitly changed.*
- The network topology is static and can be revealed to the DRA component
- I/O interconnection usage information can be periodically sampled and provided
- Application's I/O requests are analyzed for access patterns as well, which is used to direct the coordination

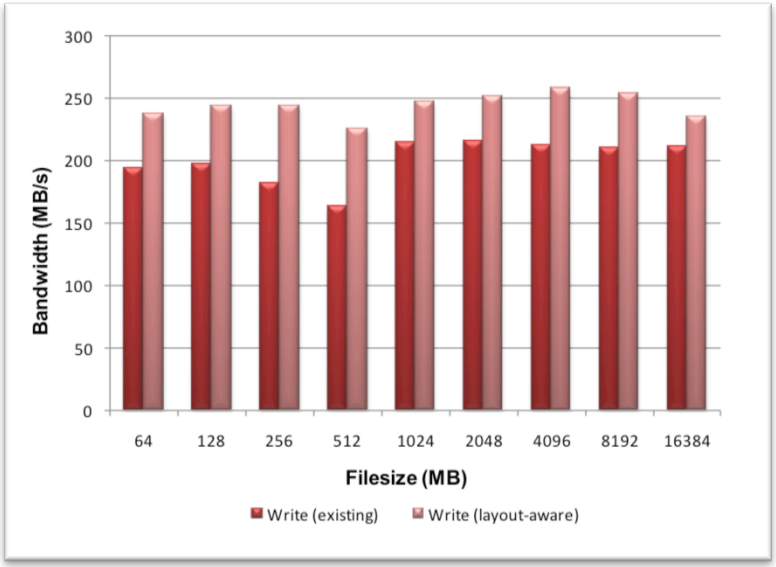
Preliminary Results of Dynamic Coordination for Independent I/O



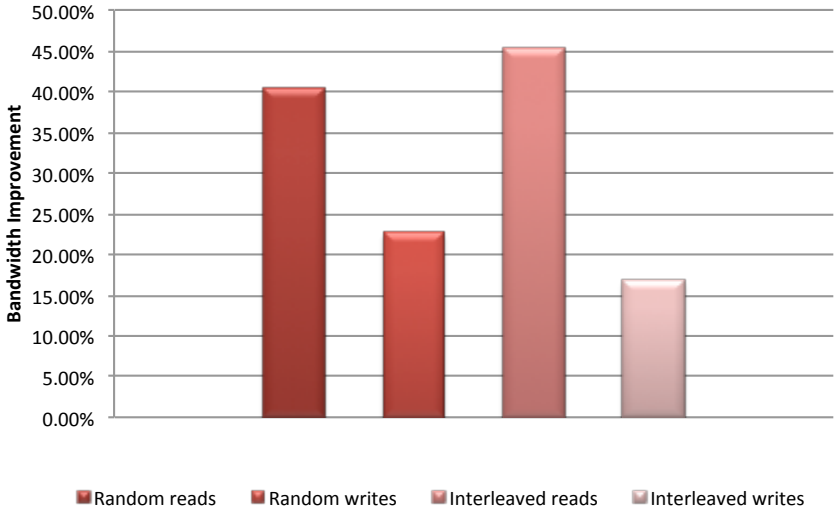
- User-level checkpointing with same total image size
- Performance was nearly doubled with the dynamic coordination in a large scale case with simultaneous I/O
- Bandwidth decreased when the number of processes increased
- Coordinated I/O achieved stable performance and more scalable

Preliminary Results of Dynamic Coordination for Collective I/O





Left: IOR Random Reads Testing
Up to 74% speedup
On average, 40% speedup

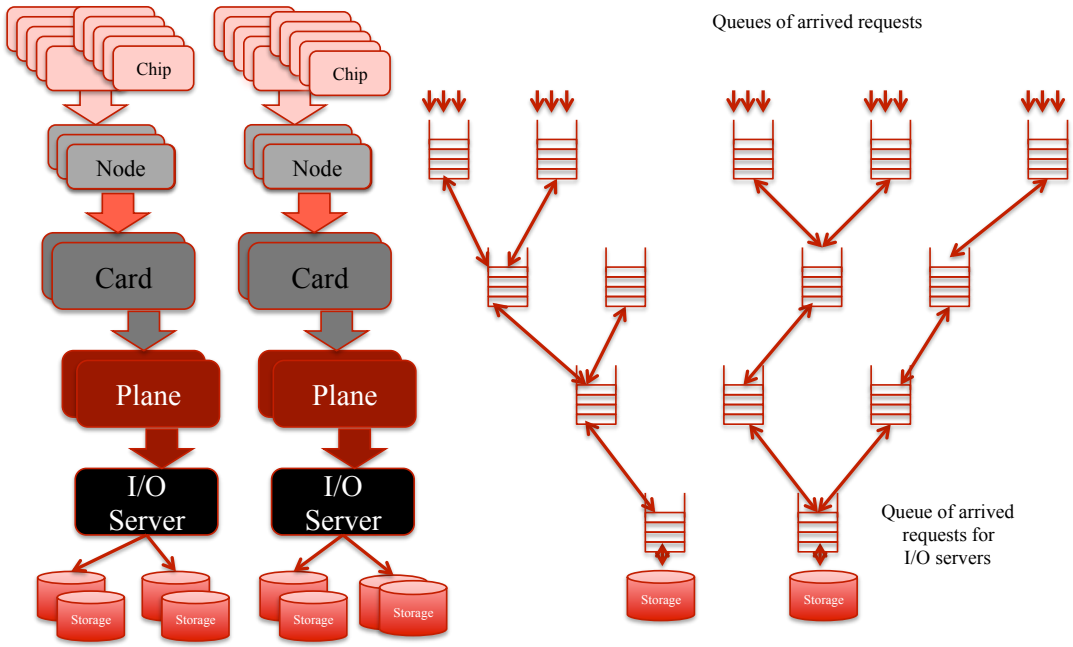


Bandwidth Improvement of Dynamic Coordination

Right: IOR Random Writes Testing
Up to 38% speedup
On average, 23% speedup

Ongoing Work

- Modeling the dynamic coordination and analyze the potential of coordinated I/O in theory
- Hierarchical coordination and aggregation
- Carrying out evaluations at a scale of O(10K-100K) processes



Conclusion

- Exascale systems near the horizon, critical to design and develop a scalable I/O architecture for such ultra large scale systems
- Challenges in terms of substantial amount of concurrency and contention, and reduced locality and increased latency
- The proposed coordinated I/O intends to address these issues
  - *Provides an access-aware, topology-aware, and layout-aware I/O architecture*
  - *Manages the growing amount of I/O concurrency*
  - *Reduces I/O contention and regains diminished locality*
- Long-term goal: provides a scalable I/O architecture to meet the needs of exascale systems and the growing demand of data-intensive science

Any Questions?

Thank You.

For more information please visit

http://discl.cs.ttu.edu/