




Towards Exascale File I/O

Yutaka Ishikawa
University of Tokyo, Japan

2009/05/21



Background & Overview



- Existing libraries and systems
 - High Level I/O Library
 - Parallel HDF5
 - Collective I/O
 - MPI-IO
 - Global File System
 - Lustre, PVFS, GPFS, ...
- Existing file systems
 - Global file system only
 - Most systems
 - Staging
 - Files are copied to a local disk of each compute node before execution, and then dirty files are copied to the global file system after execution.
e.g., Earth simulator, Riken Super Combined Cluster, PACS-CS@Univ. of Tsukuba
- Environment to develop exascale file systems
- Challenges towards exascale systems
 1. File system configuration
 2. Exascale file access technologies
 3. Exascale data access technologies
 4. Exascale Layered implementation
- Collaboration Scenarios

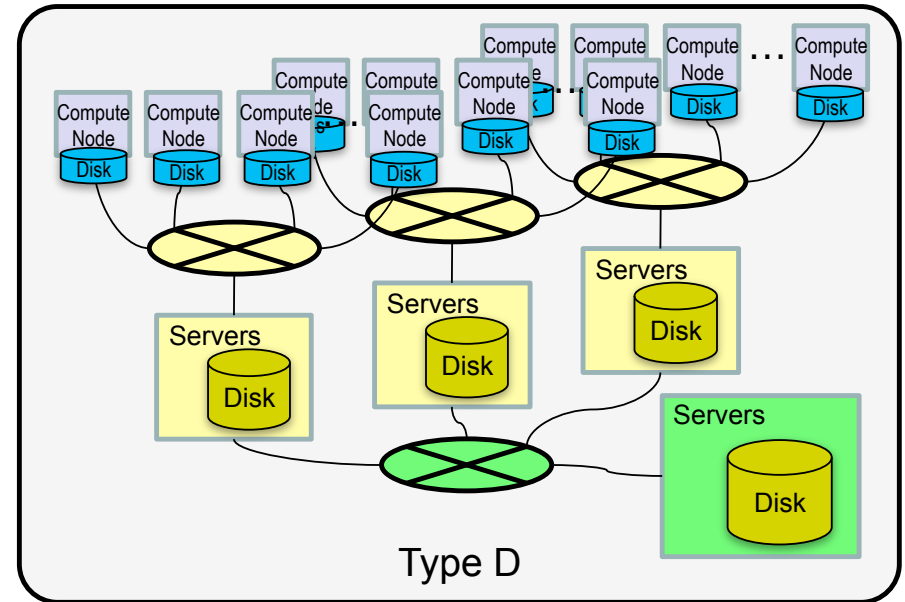
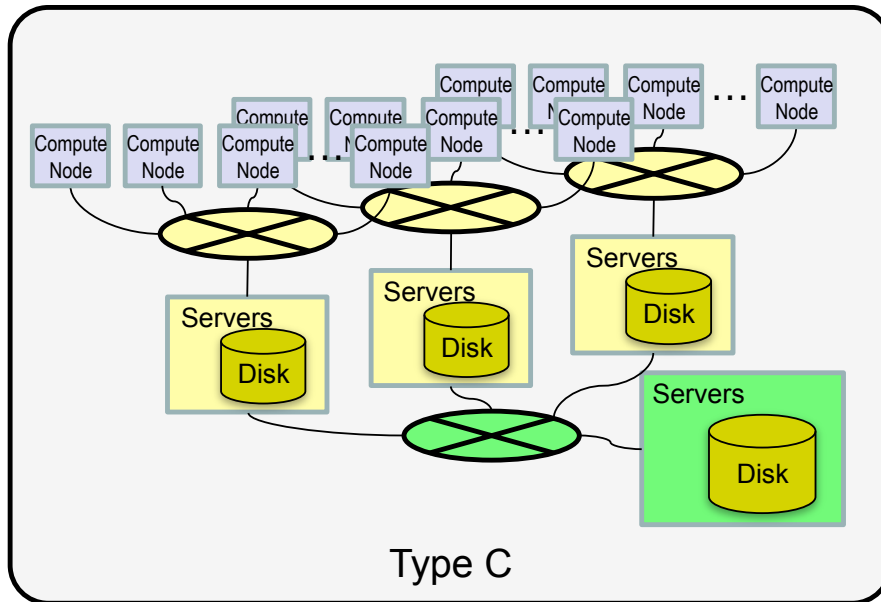
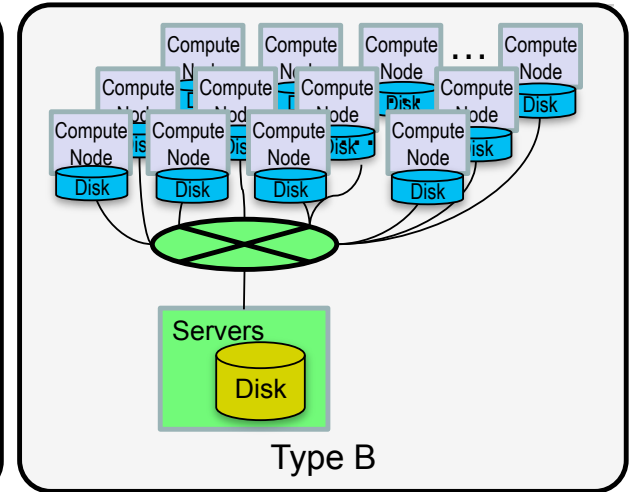
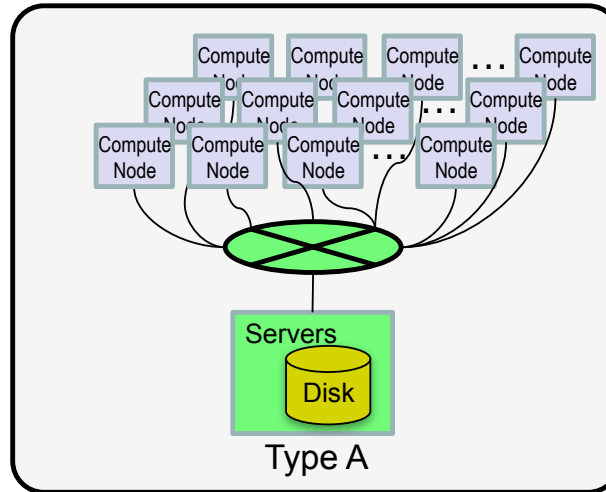
Environment to develop exascale file systems



- Benchmarks and use cases
 - To understand file system performance and reveal the weakness of the file system
 - Involvement of application developers' skill
 - We have to discuss with application developers to understand the application characteristics
 - We have to cooperate with application developers to achieve better file I/O performance
 - Is the following consortium still working ?
 - Parallel I/O Benchmarking Consortium
<http://www-unix.mcs.anl.gov/pio-benchmark>
- Tools
 - File I/O access tracer
 - To understand the application I/O characteristics
- Experimental Equipments
 - 1 K to 10 K nodes
 - The developed code can be deployed to compute nodes and file servers
 - Kernel modification

Research Topics: File system configurations

	Global file system	Local disk on each node	Disk for group of nodes
Type A	✓		
Type B	✓	✓	
Type C	✓		✓
Type D	✓	✓	✓



Research Topics: Exascale file access technologies



- Type A (Global file system only)
 - This configuration may be not applied
- Type B (Global file system + Local disk)
 - Local disk will be SSD.
 - Research topics
 - Both the file and meta-data cache mechanisms in each node
 - File staging
 - If two networks for both computing and file access are installed, some optimization mechanisms utilizing both networks are also research topics
- Type C (Global file system + Group file system)
 - Each group file system provides the file access service to the member nodes of its group
 - Research topics
 - Efficient file staging
 - The group file system as file cache
 - The file service mechanism to some groups if an application runs over those groups
- Type D (Global file system + Group file system + Local disk)
 - The combination of Types B and C

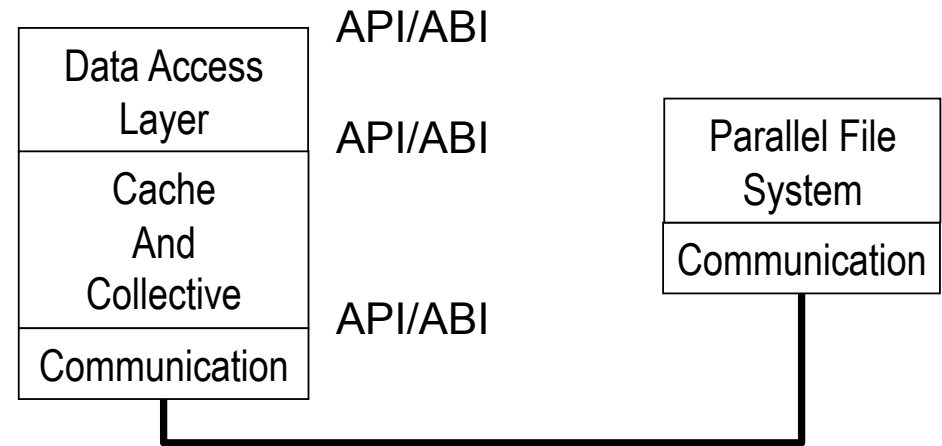
Research Topics: Exascale data access technologies



- Most application developers use the read/write file I/O system calls (, at least in Japan)
- If the parallel HDF-5 is enough capability to describe exascale applications, the following research topics are candidates:
 - Efficient implementation of parallel HDF-5 for exascale parallel file system
 - Optimization over cores in each node
 - Application domain specific libraries on top of parallel HDF-5
- If the parallel HDF-5 is not enough capability,
 - Design of extended API and the implementation data structure is redesigned
- Deployment Issues
 - Portable efficient implementation
 - Tutorials for the application developers

Research Topics: Layered Implementation for collaboration

- Data Access Layer
 - Parallel HDF and others
- Cache and Collective Layer
 - Approaches
 - Memory-mapped parallel file I/O
 - Distributed Shared Memory
 - ...
- Communication Layer
 - Accessing parallel file system
- Parallel File System



Collaboration Scenarios



1. Almost no collaboration
 - Joint workshops are held
2. Loosely coupled collaboration
 - Benchmarks are defined
3. Collaboration with Standardization
 - Network protocol is defined
 - Client-side API/ABI are defined
 - New Parallel File I/O
 - Highly abstracted Parallel File I/O in addition of HDF5 ?
4. Tightly Coupled collaboration
 - Developing the single File I/O software stack