SSD Based First Layer File System for the Next Generation Super-computer

Shinji Sumimoto, Ph.D.
Next Generation Technical Computing Unit
FUJITSU LIMITED

Sept. 24^{th}, 2018

Outline of This Talk

- A64FX: High Performance Arm CPU
- SSD Based First Layer File System for the Next Generation Super-computer
- Current Status of Lustre Based File System Development

A64FX: High Performance Arm CPU

- From presentation slides of Hotchips 30th and Cluster 2018
- Inheriting Fujitsu HPC CPU technologies with commodity standard ISA

A64FX Chip Overview

Architecture Features

- Armv8.2-A (AArch64 only)
- SVE 512-bit wide SIMD
- 48 computing cores + 4 assistant cores*

  *All the cores are identical
- HBM2 32GiB
- TofuD 6D Mesh/Torus
  28Gbps x 2 lanes x 10 ports
- PCIe Gen3 16 lanes

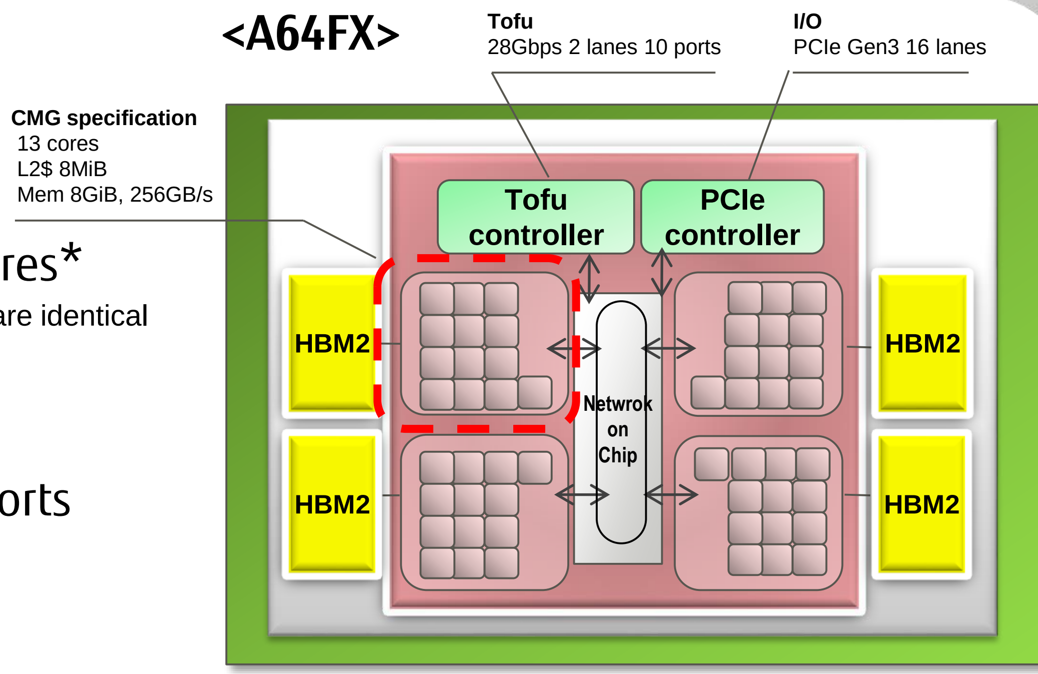
7nm FinFET

- 8,786M transistors
- 594 package signal pins

Peak Performance (Efficiency)

- >2.7TFLOPS (>90%@DGEMM)
- Memory B/W 1024GB/s (>80%@Stream Triad)

<A64FX>



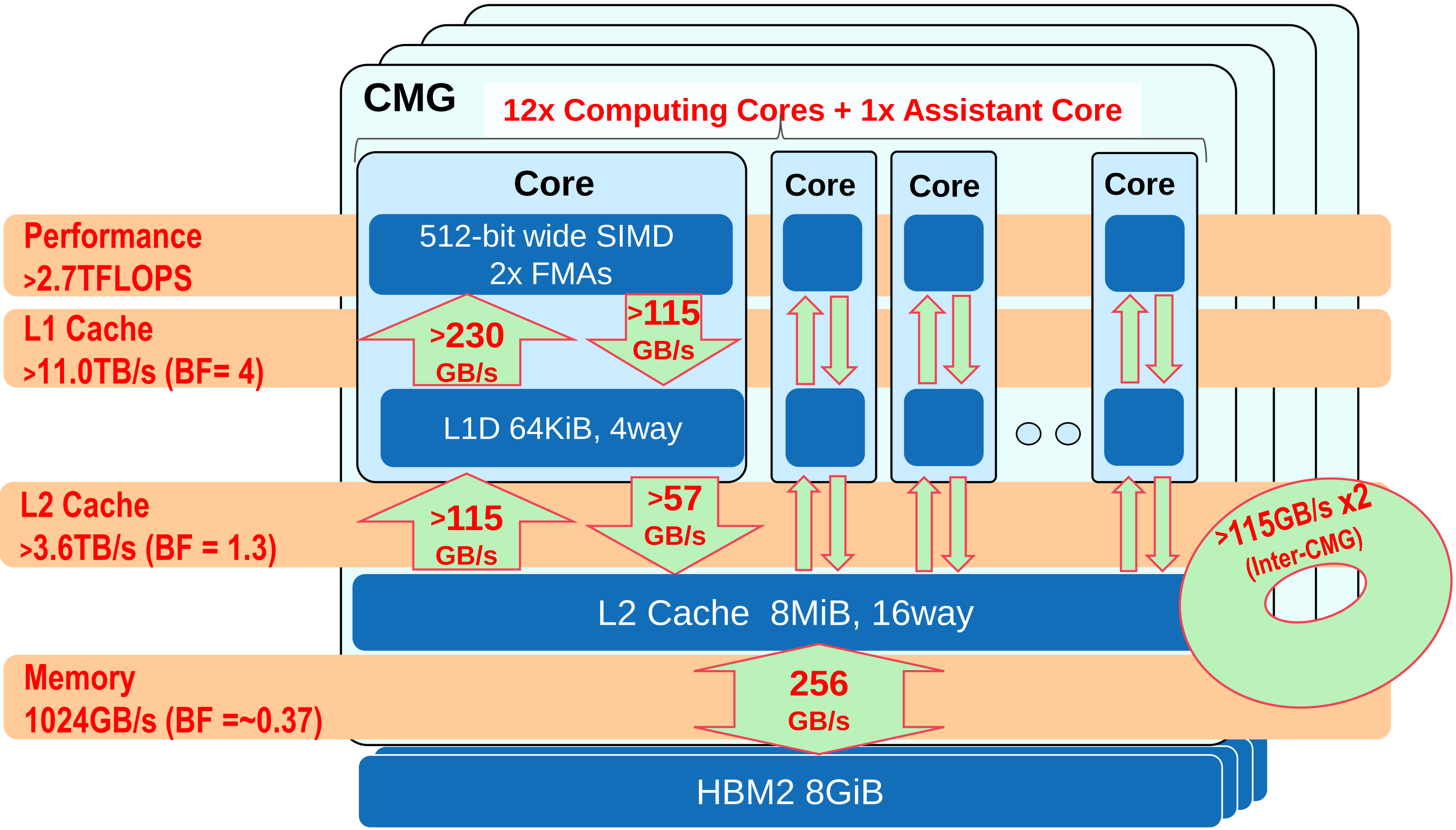
	A64FX (Post-K)	SPARC64 XIfx (PRIMEHPC FX100)
ISA (Base)	Armv8.2-A	SPARC-V9
ISA (Extension)	SVE	HPC-ACE2
Process Node	7nm	20nm
Peak Performance	>2.7TFLOPS	1.1TFLOPS
SIMD	512-bit	256-bit
# of Cores	48+4	32+2
Memory	HBM2	HMC
Memory Peak B/W	1024GB/s	240GB/s x2 (in/out)

A64FX Memory System

FUJITSU

Extremely high bandwidth

- Out-of-order Processing in cores, caches and memory controllers
- Maximizing the capability of each layer's bandwidth



A64FX Core Features

FUJITSU

- Optimizing SVE architecture for wide range of applications with Arm including AI area by FP16 INT16/INT8 Dot Product
- Developing A64FX core micro-architecture to increase application performance

	A64FX (Post-K)	SPARC64 XIfx (PRIMEHPC FX100)	SPAR64 VIIIfx (K computer)
ISA	**Armv8.2-A + SVE**	**SPARC-V9 + HPC-ACE2**	**SPARC-V9 + HPC-ACE**
SIMD Width	512-bit	256-bit	128-bit
Four-operand FMA	✓ Enhanced	✓	✓
Gather/Scatter	✓ Enhanced	✓	
Predicated Operations	✓ Enhanced	✓	✓
Math. Acceleration	✓ Further enhanced	✓ Enhanced	✓
Compress	✓ Enhanced	✓	
First Fault Load	✓ New		
FP16	✓ New		
INT16/ INT8 Dot Product	✓ New		
HW Barrier* / Sector Cache*	✓ Further enhanced	✓ Enhanced	✓

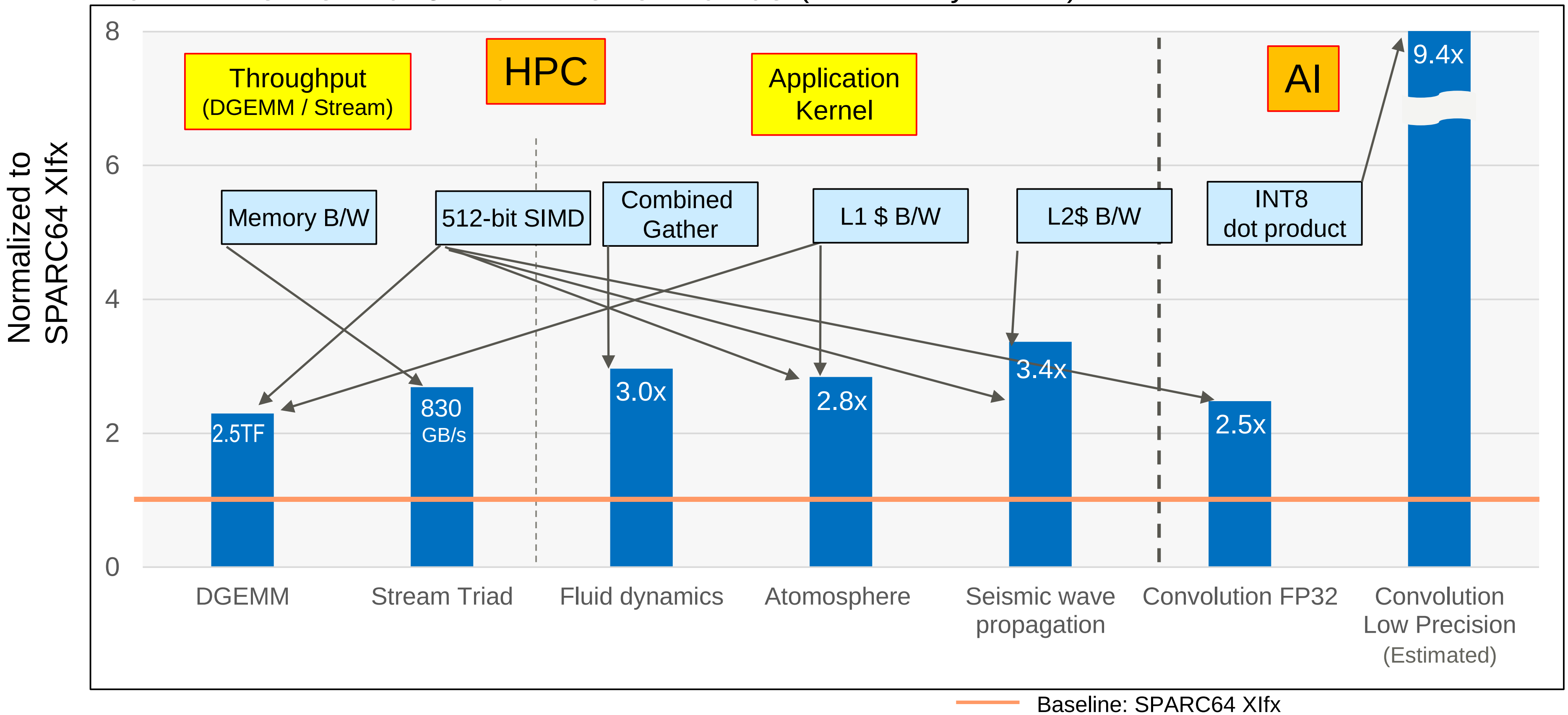
* Utilizing AArch64 implementation-defined system registers

A64FX Chip Level Application Performance

FUJITSU

- Boosting application performance up by micro-architectural enhancements, 512-bit wide SIMD, HBM2 and semi-conductor process technologies
  - > 2.5x faster in HPC/AI benchmarks than that of SPARC64 XIfx tuned by Fujitsu compiler for A64FX micro-architecture and SVE

A64FX Kernel Benchmark Performance (Preliminary results)

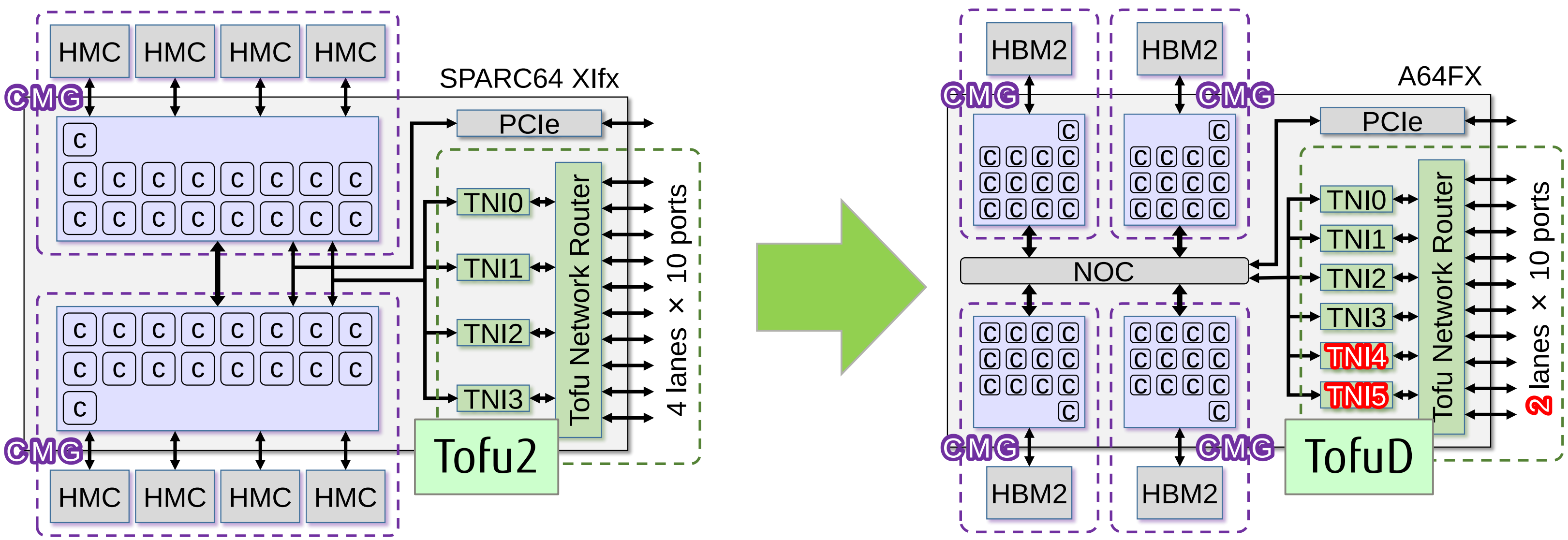


A64FX TofuD Overview

FUJITSU

- **Halved Off-chip Channels**
  - Power and Cost Reduction
- **Increased Communication Resources**
  - TNIs from 2 to 4
  - Tofu Barrier Resources
- **Reduced Communication Latency**
  - Simplified Multi-Lane PCS
- **Increased Communication Reliability**
  - Dynamic Packet Slicing: Split and Duplicate

	Tofu K.comp	Tofu2 FX100	TofuD
Data rate (Gbps)	6.25	25.78	28.05
# of signal lanes per link	8	4	2
Link bandwidth (GB/s)	5.0	12.5	6.8
# of TNIs per node	4	4	6
Injection bandwidth per node (GB/s)	**20**	**50**	**40.8**



TofuD: Put Latencies & Throughput& Injection Rate

FUJITSU

- TofuD: Evaluated by hardware emulators using the production RTL codes
  - Simulation model: System-level included multiple nodes

	Communication settings	Latency
Tofu	Descriptor on main memory	1.15 μs
	Direct Descriptor	0.91 μs
Tofu2	Cache injection OFF	0.87 μs
	Cache injection ON	0.71 μs
TofuD	To/From far CMGs	0.54 μs
	To/From near CMGs	0.49 μs

	Put throughput	Injection rate
Tofu	4.76 GB/s (95%)	15.0 GB/s (77%)
Tofu2	11.46 GB/s (92%)	45.8 GB/s (92%)
TofuD	6.35 GB/s (93%)	38.1 GB/s (93%)

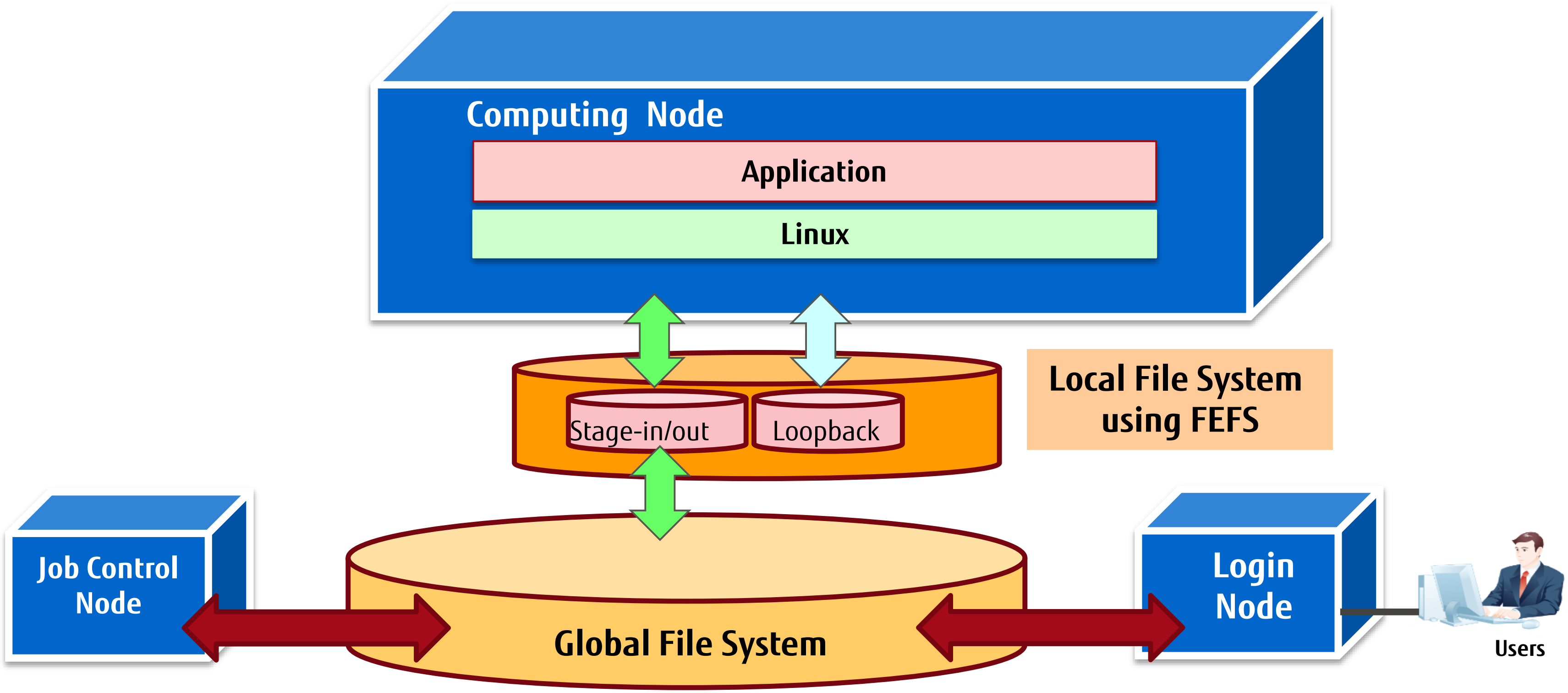
Next Generation File System Design

- Next Generation File System Structure and Design
- Next-Gen 1st Layer File System Overview

K computer: Pre-Staging-In/Post-Staging-Out Method

FUJITSU

- Pros:
  - Stable Application Performance for Jobs
- Cons:
  - Requiring three times amount of storage which a job needs
  - Pre-defining file name of stage-in/out processing lacks of usability
  - Data-intensive application affects system usage to down because of waiting pre-staging-in/out processing



Next-Gen File System Requirement and Issues

FUJITSU

Requirements

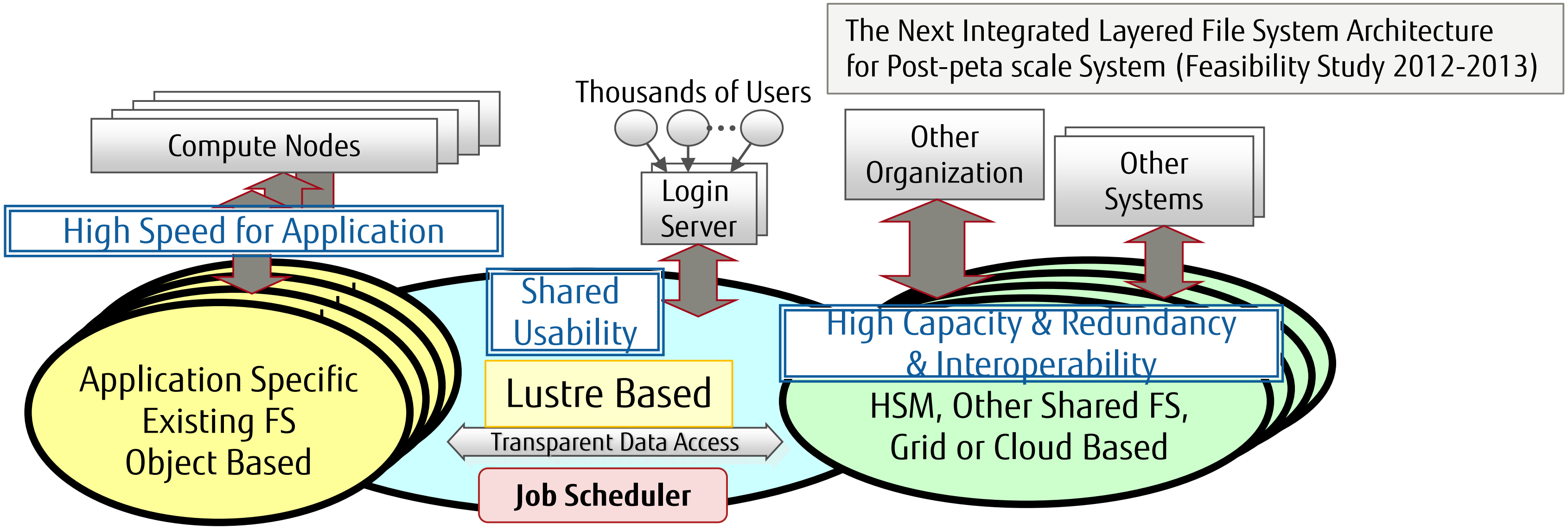
- 10 times higher access performance
- 100 times larger file system capacity
- Lower power and footprint

Issues

- How to realize 10 times faster and 100 times larger file access at a time?

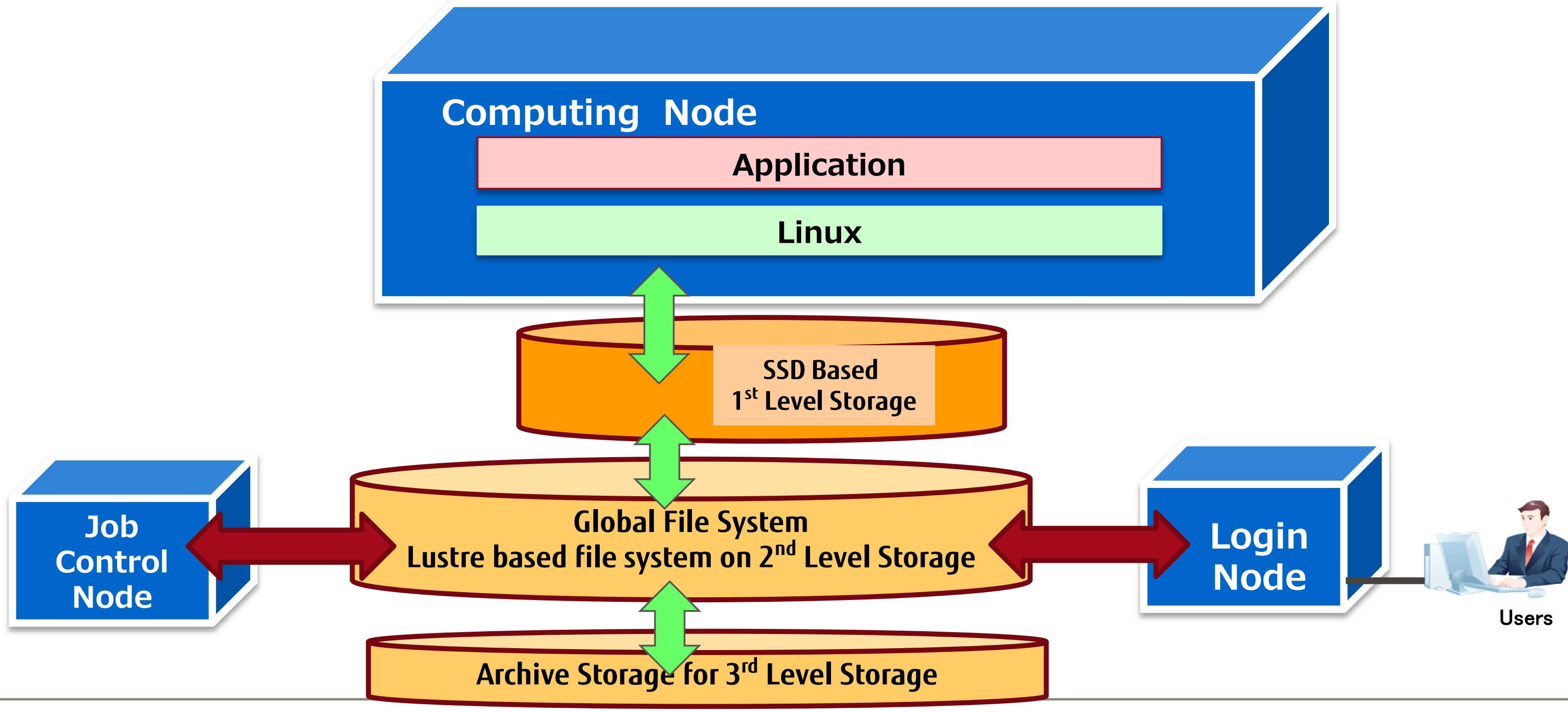
Next-Gen. File System Design

- K computer File System Design
  - How should we realize High Speed and Redundancy together?
  - Introduced Integrated Two Layered File System.
- Next-Gen. File System/Storage Design
  - Another trade off targets: Power, Capacity, Footprint
    - Difficult to realize single Exabyte and 10TB/s class file system in limited power consumption and footprint.
  - Additional Third layer Storage for Capacity is needed:



Next Gen. File System Design

- Introducing three level hierarchical storage.
  - 1st level storage: Accelerating application file I/O performance (Local File System)
  - 2nd level storage: Sharing data using Lustre based file system (Global File System)
  - 3rd level storage: Archive Storage (Archive System)
- Accessing 1st level storage as file cache of global file system and local storage
  - File cache on computing node is also used as well as 1st level storage



Next Gen. Layered File System Requirements

FUJITSU

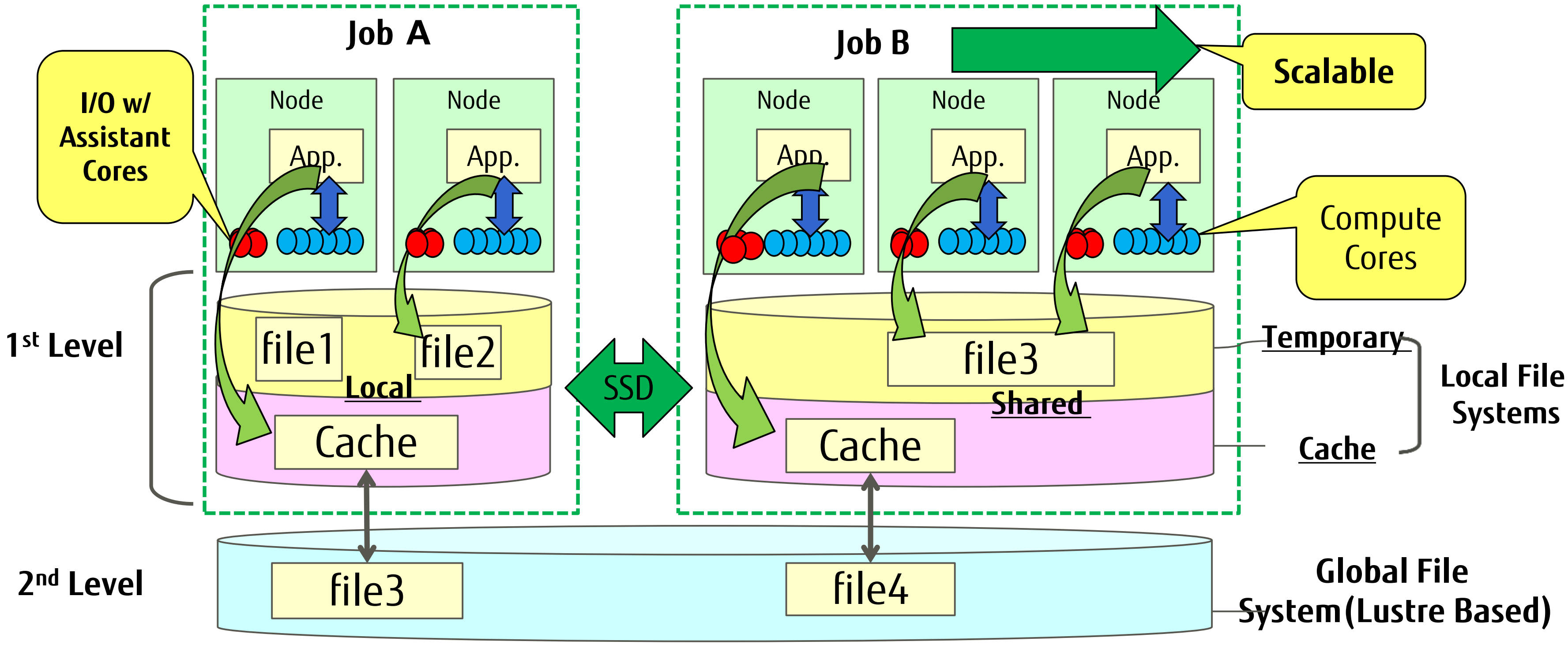
- Application views:
  - Local File System: Application Oriented File Accesses(Higher Meta&Data I/O)
  - Global File System: Transparent File Access
  - Archive System: In-direct Access or Transparent File Access(HSM)
- Transparent File Access to the Global File System
  - Local File System Capacity is not enough as much as locating whole data of Global File System
  - File Cache on node memory and Local File System enables to accelerate application performance

	Meta Perf.	Data BWs	Capacity	Scalability	Data Sharing in a Job	Data Sharing among Jobs
Local File System	◎	◎	×	◎	◎	×
Global File System	○	○	○	○	×	◎
Archive System	×	×	◎	×	×	×

Next-Gen 1st Layer File System Overview

FUJITSU

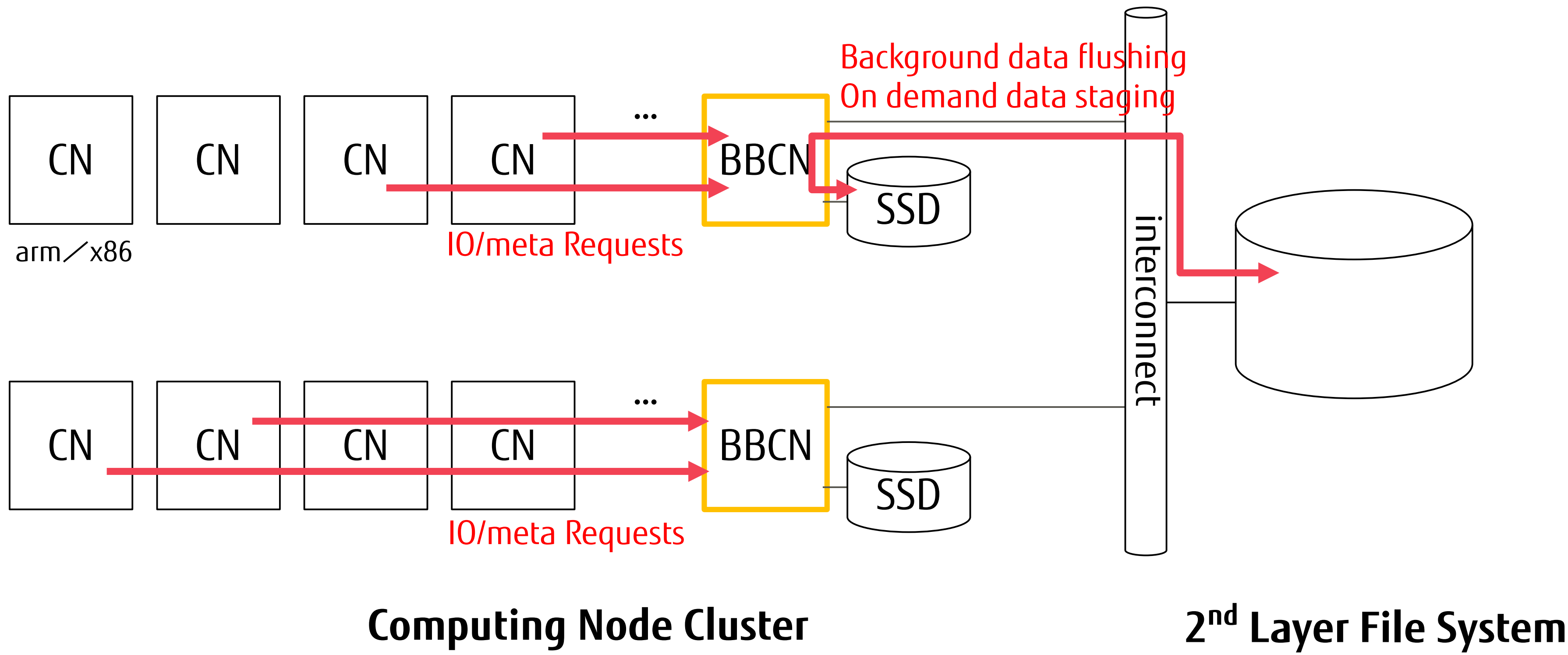
- Goal: Maximizing application file I/O performance
- Features:
  - Easy access to User Data: File Cache of Global File System
  - Higher Data Access Performance: Temporary Local FS (in a process)
  - Higher Data Sharing Performance: Temporary Shared FS (among processes)
- Now developing LLIO(Lightweight Layered IO-Accelerator) Prototype



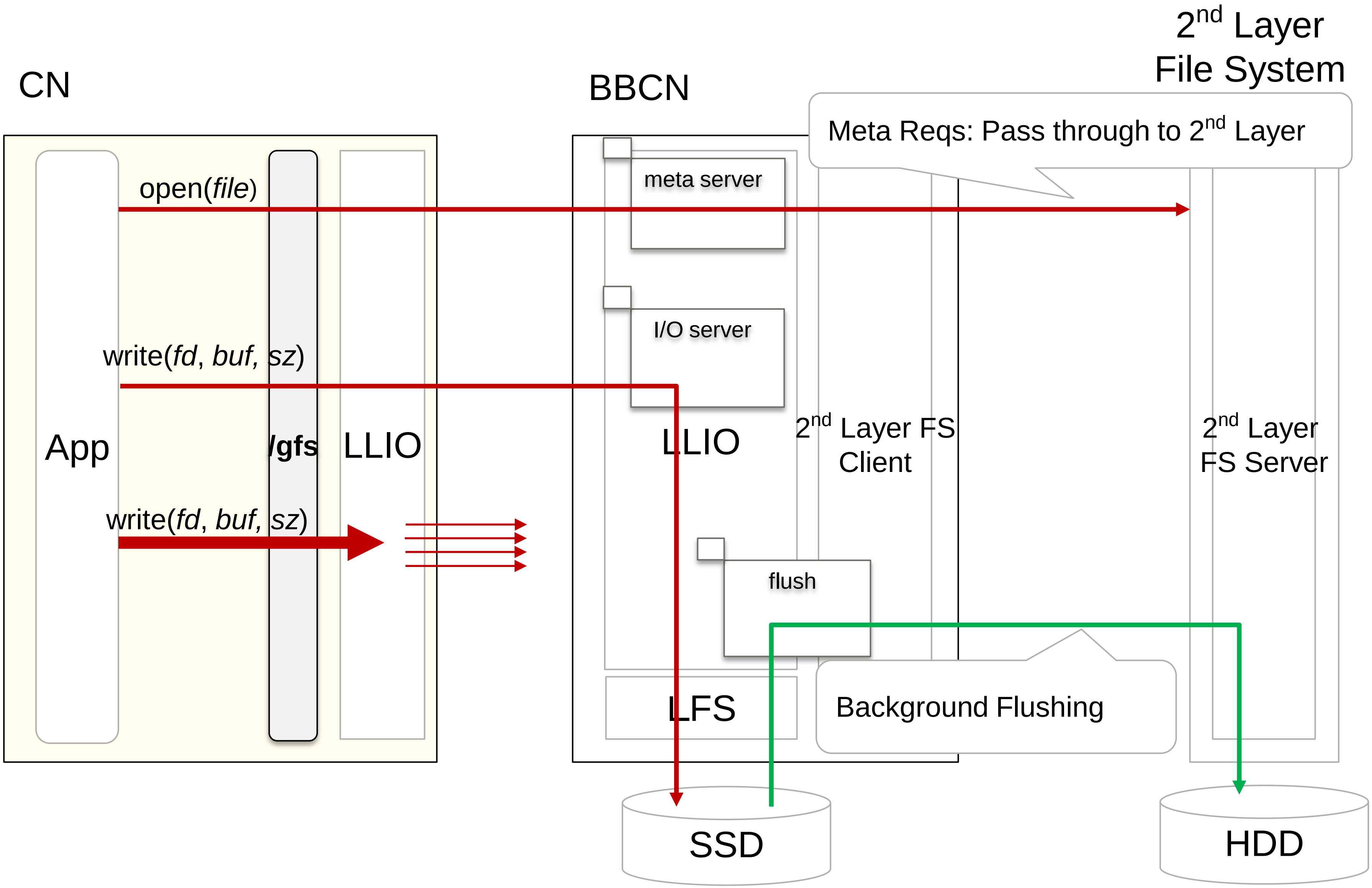
LLIO Prototype Implementation

FUJITSU

- Two types of Computing Nodes
  - Burst Buffer Computing Node(BBCN)
    - Burst Buffer System Function with SSD Device
  - Computing Node(CN)
    - Burst Buffer Clients: File Access Request to BBCN as burst buffer server

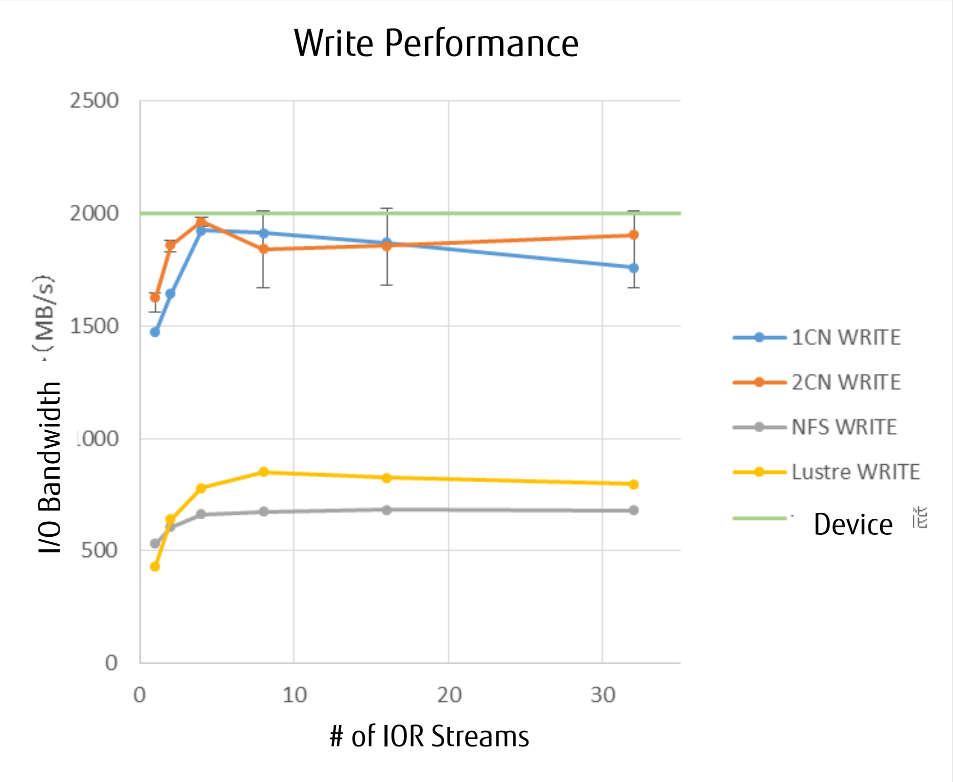


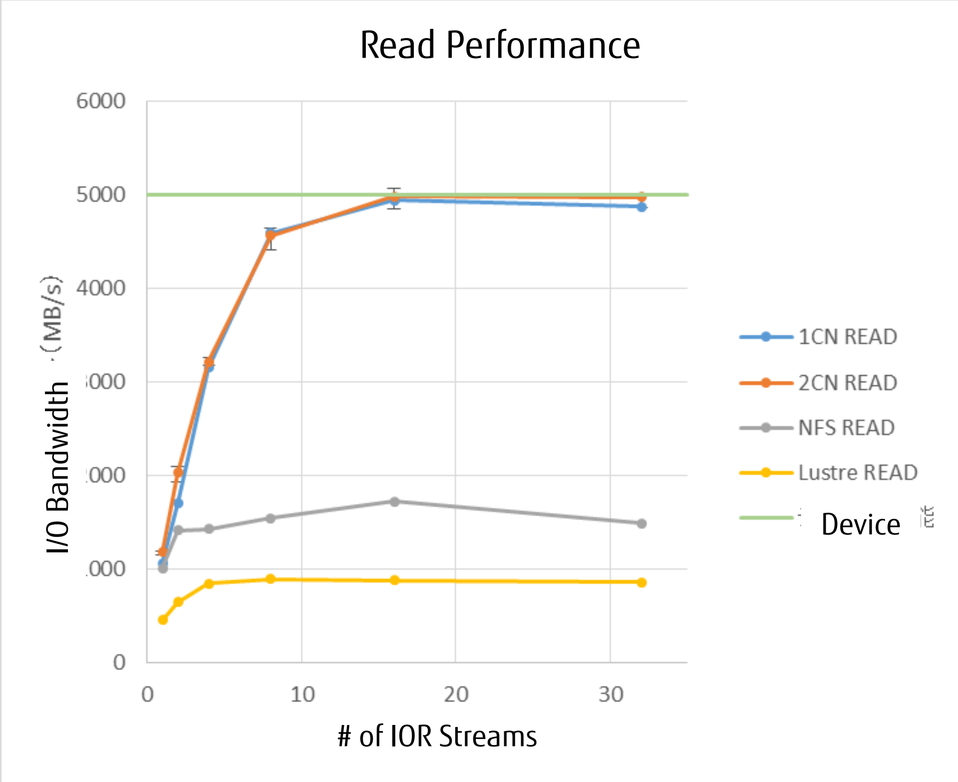
File Access Sequences using LLIO (Cache Mode)



LLIO Prototype I/O Performance

FUJITSU





Evaluated on IA servers using Intel P3608

- Higher I/O performance than those of NFS, Lustre
- Utilizing maximum physical I/O device performance by LLIO

Current Status of Lustre Based File System Development, etc,.

Current Status of Lustre Based File System

- Next-gen. Lustre Based File System: FEFS
  - Planning to develop on Lustre 2.10.x base
- Now testing Lustre 2.10.x based FEFS, and found several problems
  - Planning to fix the bugs and report their fixes.

Contribution of DL-SNAP

FUJITSU

- What is DL-SNAP? Presented@LUG2016, LAD16 (http://cdn.opensfs.org/wp-content/uploads/2016/04/LUG2016D2_DL-SNAP_Sumimoto.pdf)
  - DL-SNAP is designed for user and directory level file backups.
  - Users can create a snapshot of a directory using lfs command with snapshot option and create option like a directory copy.
  - The user creates multiple snapshot of the directory and manage the snapshots including merge of the snapshots.
  - DL-SNAP also supports quota to limit storage usage of users.

- Issue of Contribution:
  - We planed DL-SNAP contribution in 2018
  - We do not have human resources enough to port to latest Lustre version

- Our Strategy for contributing DL-SNAP:
  - We are ready to contribute our current DL-SNAP code for Lustre 2.6
  - We will make a LU-ticket for the DL-SNAP (by the end of Oct. 2018)
  - We need help to port DL-SNAP to the latest Lustre

shaping tomorrow with you