Oct. 22 2015

Shinji Sumimoto (Fujitsu Limited)

Outline of This Talk

- From LUG-2015
  - Metadata Access Reduction of Large Scale Lustre Based File System
- From Lustre Development Summit 2015
  - Fujitsu Session: Toward Exascale Computing

京

K computer

Apr.14 2015

Shinji Sumimoto*, Shuji Matsui, Kenichiro Sakai, and
Fumichika Sueyasu (Fujitsu Limited)
Fumiyoshi Shoji, Atsuya Uno, Keiji Yamamoto (RIKEN AICS)

OpenSFS

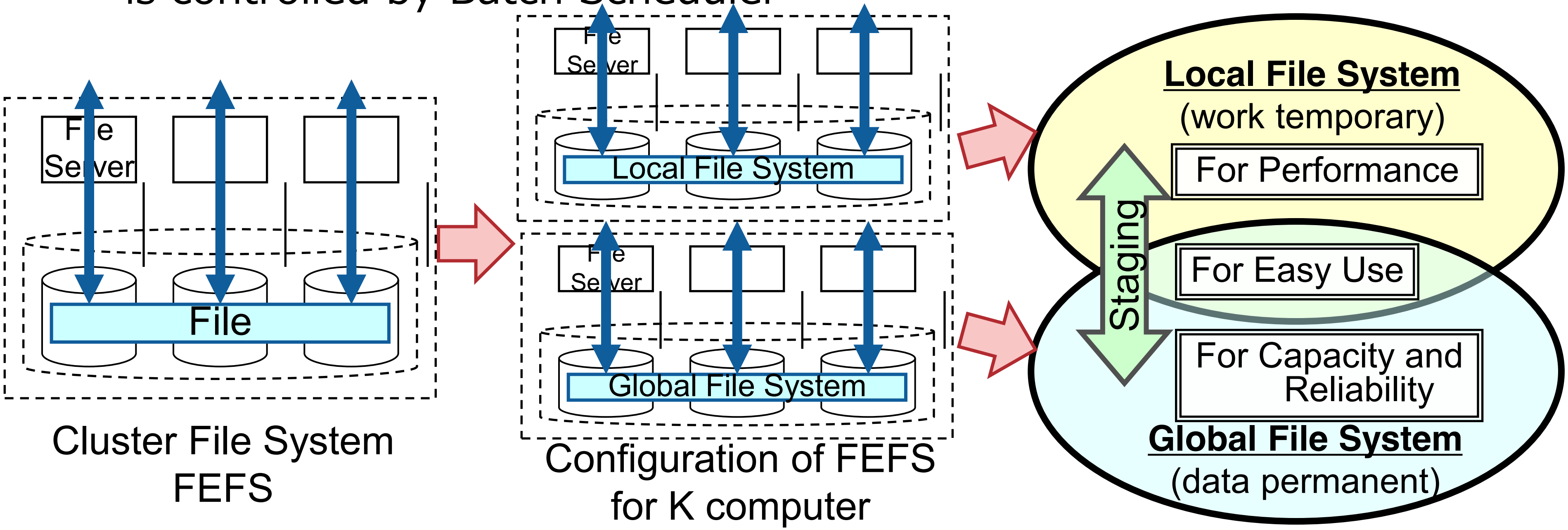
Outline

- File System Usage in User Jobs on K computer
- File Access Issues on Local File System
- Meta Data Access Distribution by Loopback File System
- Evaluation on K computer

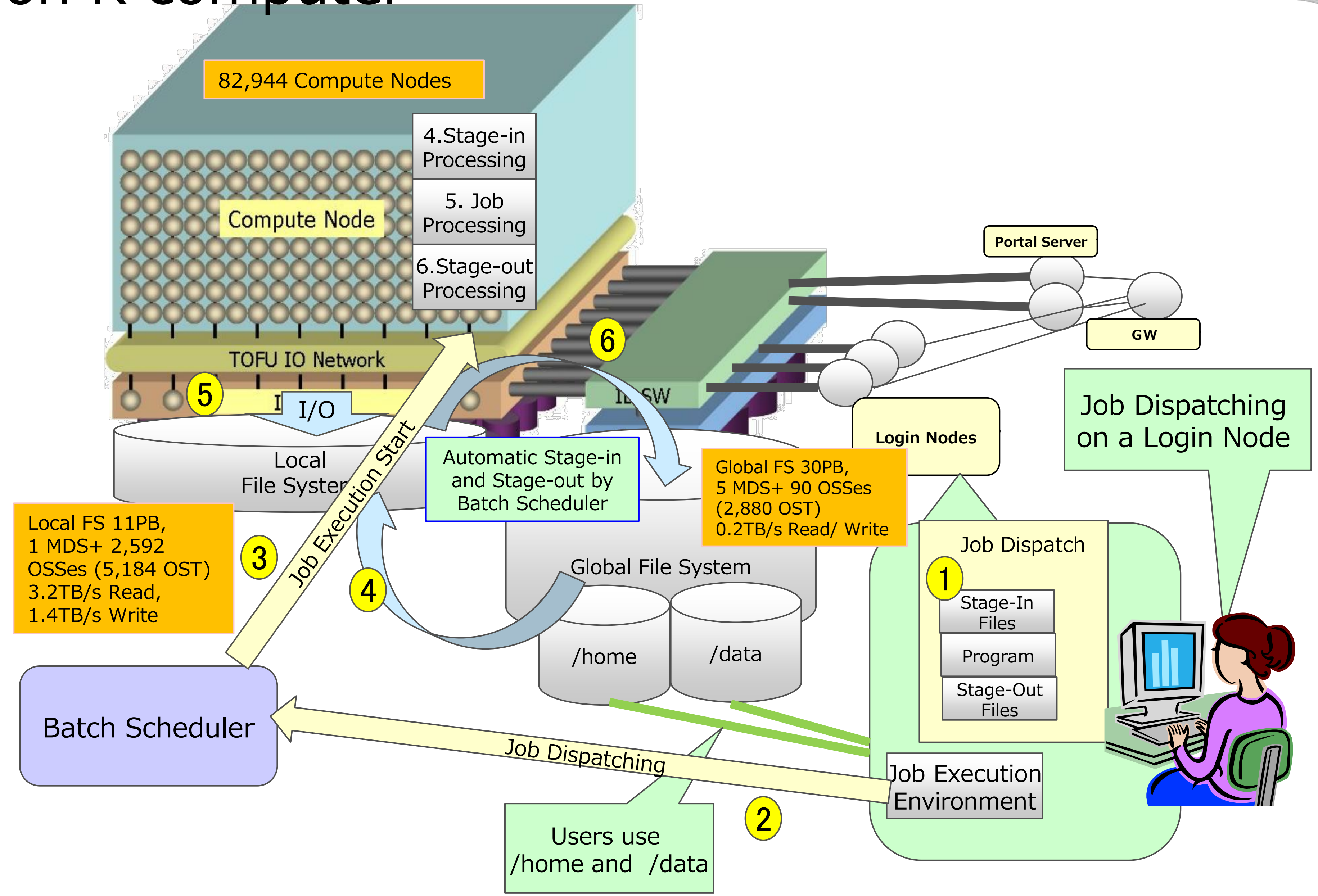
FILE SYSTEM USAGE IN USER JOBS ON K COMPUTER

Overview of FEFS for K computer

- Goals: To realize World Top Class Capacity and Performance File system 100PB, 1TB/s
- Based on Lustre File System with several extensions
  - These extensions are now going to be contributed to Lustre community.
- Introducing Layered File system for each file layer characteristics
  - Temporary Fast Scratch FS(Local) and Permanent Shared FS(Global)
  - Staging Function which transfers between Local FS and Global FS is controlled by Batch Scheduler



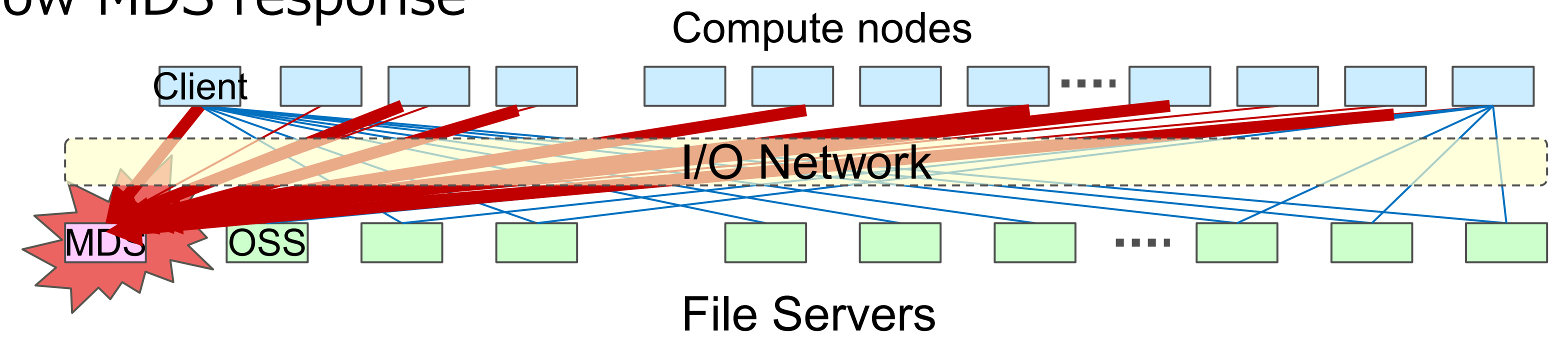
Job Execution and File System Accesses on K computer



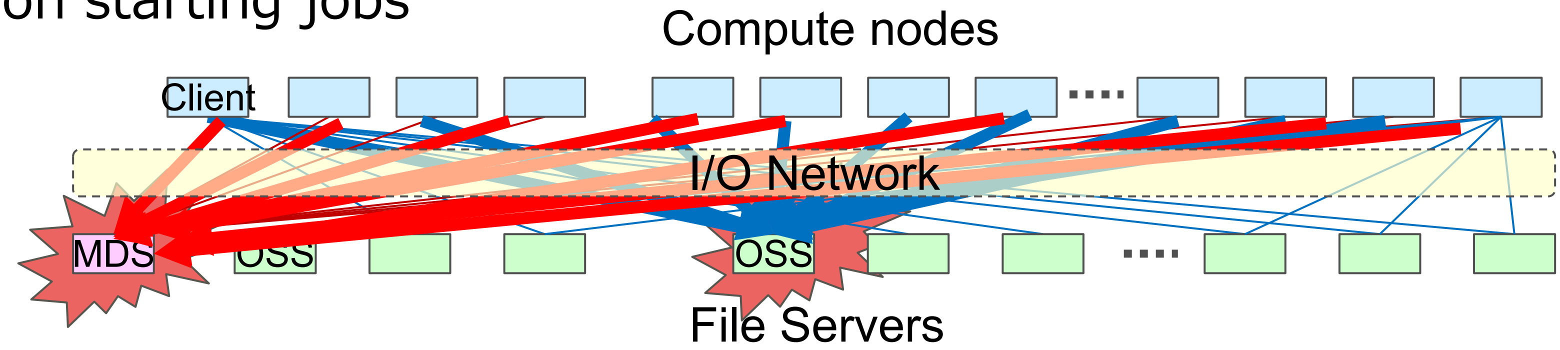
FILE ACCESS ISSUES ON LOCAL FILE SYSTEM

Meta Data Access Issues of Local File System on several 10,000 node job.

- Creating a lot of files per MPI rank at a time.
  - 1,000 file per rank creation becomes 10 M file creation per job.
  - Creating and deleting files take several hours to finish and cause slow MDS response

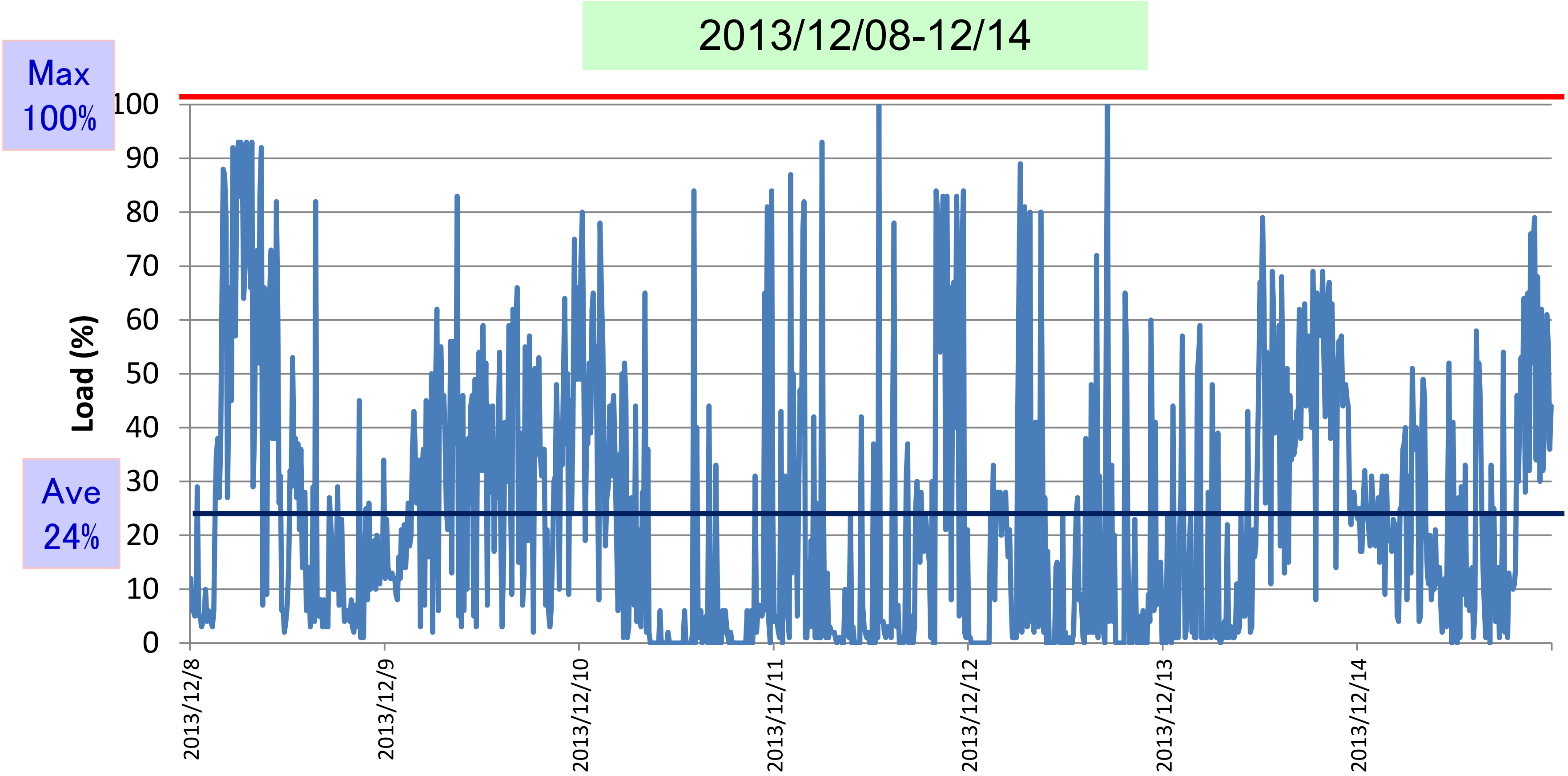


- Execution binaries on shared directory.
  - Concentration access to a single MDS and OST from several 10,000 node takes a long time to finish. Long time delay occurs on starting jobs



MDS CPU Load on Dec. 2013.

- MDS Load was average 24% peak 100% on Dec. 2013.



Issues to Solve and Our Goals

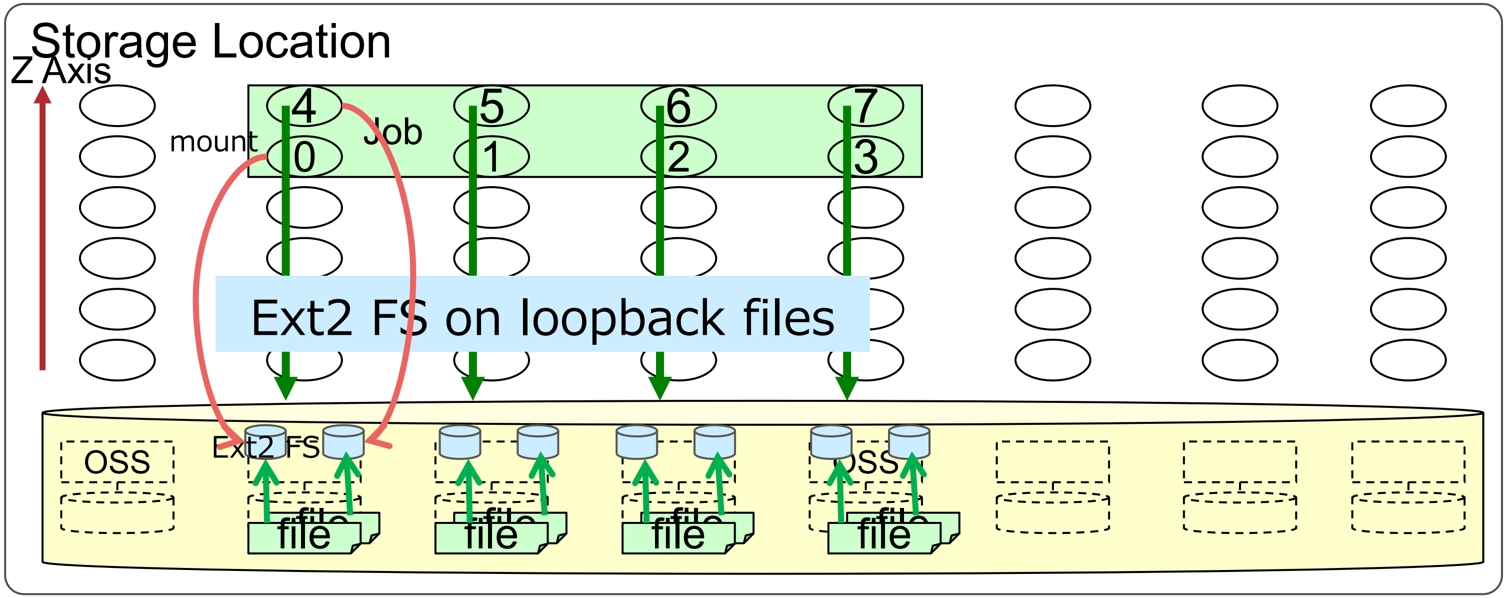
- Issues to Solve:
  - Concentration access to a single MDS or OST on job execution
  - Violent Fluctuations of MDS/OSS load depending on jobs

- Our Goals
  - Distributing and leveraging Meta Data and Data Access
  - Providing faster access performance per MPI rank

META DATA ACCESS DISTRIBUTION BY LOOPBACK FILE SYSTEM

Meta Data Access Distribution by Loopback File System on K computer

- Providing real local file system per rank by using loopback file
  - Creating loopback file and mounting it as Ext2 file system per MPI rank
  - Rank local data and execution binaries are copied to rank local file system
- Job scheduler software automatically manages creating, mounting and deleting the rank local file system.
  - MDS load can be decreased to only one file creation/deletion per rank
  - No fluctuation and no dependence per Job types (Constant Load)



Comparison of Multiple MDS vs. Loopback

- We compare the loopback with multiple MDS which could be the other method to solve high load of MDS.

- Multiple MDS(Lustre DNE)
  - Pros:
    - Increasing Meta Data performance on shared file system
  - Cons:
    - Requiring additional hardware resource: MDS, MDT Scalability is limited to hardware resource
- Loopback
  - Pros:
    - Completely Scalable Meta Data performance for rank local access
    - No additional hardware
  - Cons:
    - Unable to share among the other nodes
    - Additional Ext2 file system and Loopback Layer Overhead

EVALUATION ON K COMPUTER

Evaluation of Loopback Based Rank Local File System on K computer

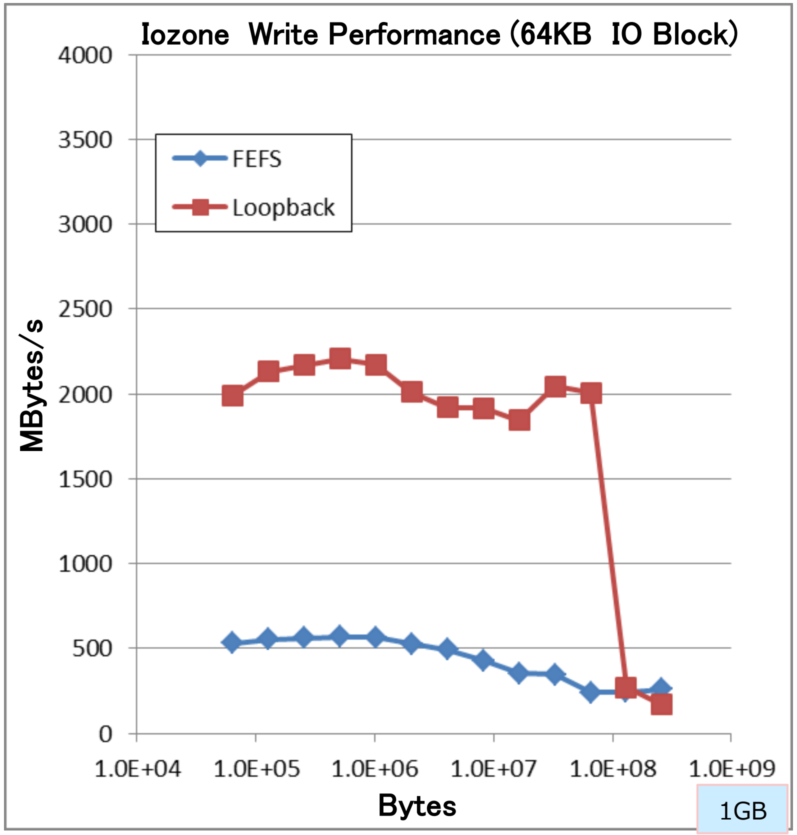
- Single Node File Access Performance

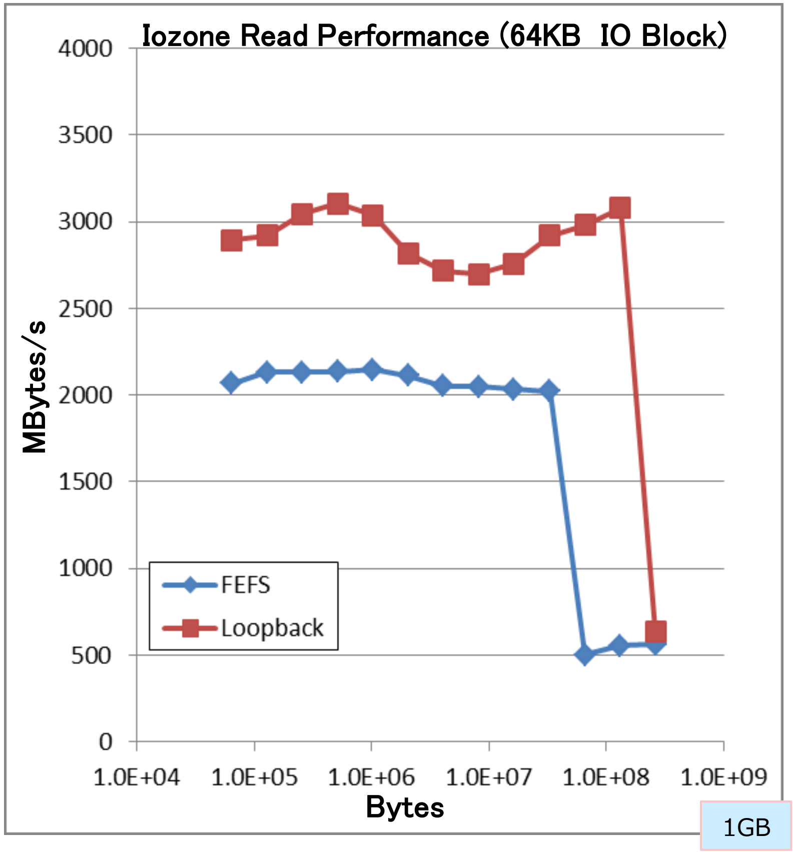
- Total Meta Data Access Performance

- Comparison of MDS Load (Before vs. After)

Single Node File Access Performance

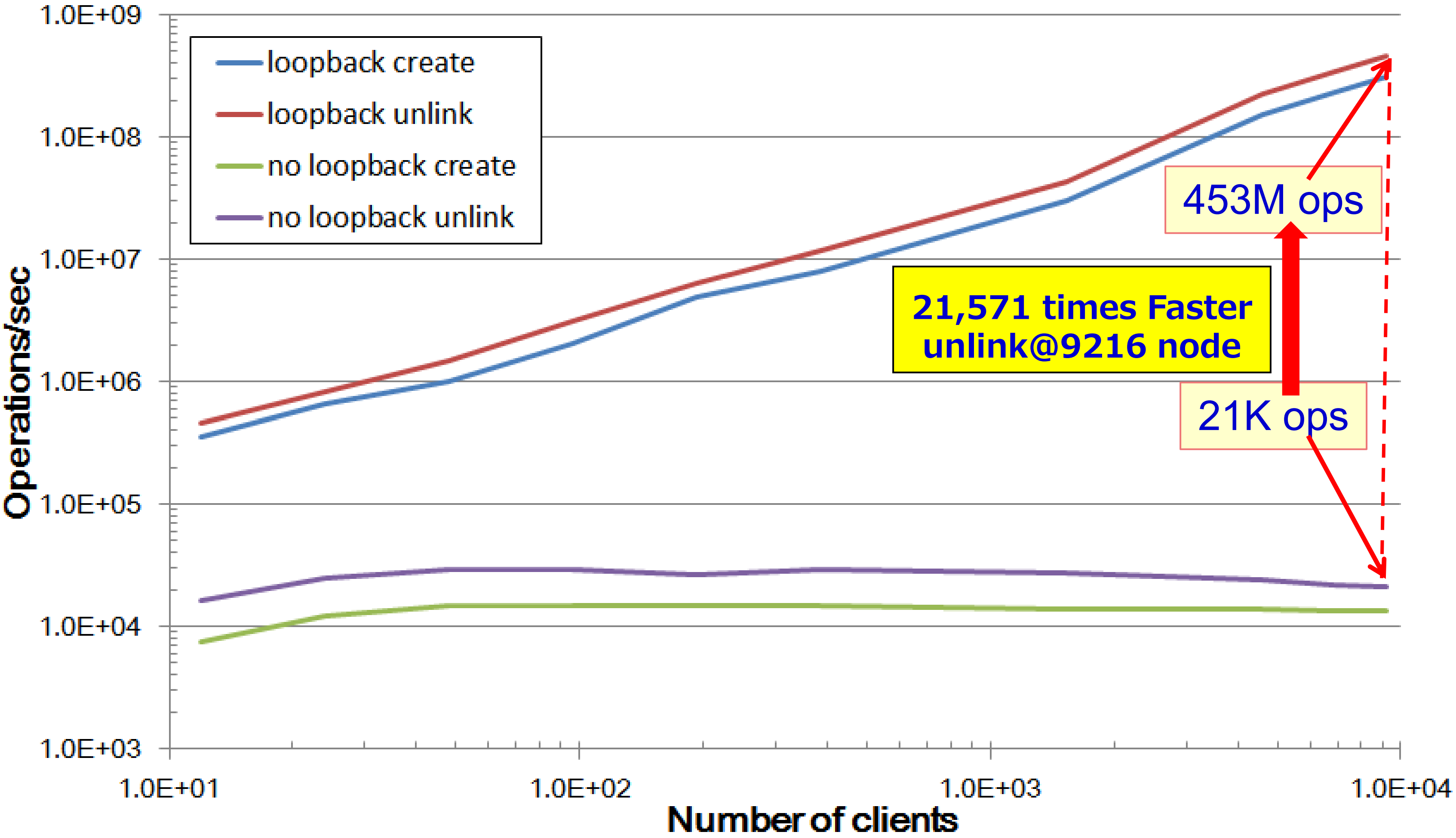
- Single Node File Write/Read Performance by iozone
- Loopback based file system achieved better performance at small file size by file system cache





Total Meta Data Access Performance

- Loopback Based Local FS Dramatically Scales over 10,000 Nodes!
  - Create 26K ops/node, unlink 37K ops/node by mdtest 100 files/node
  - Providing higher constant meta data access performance for each node

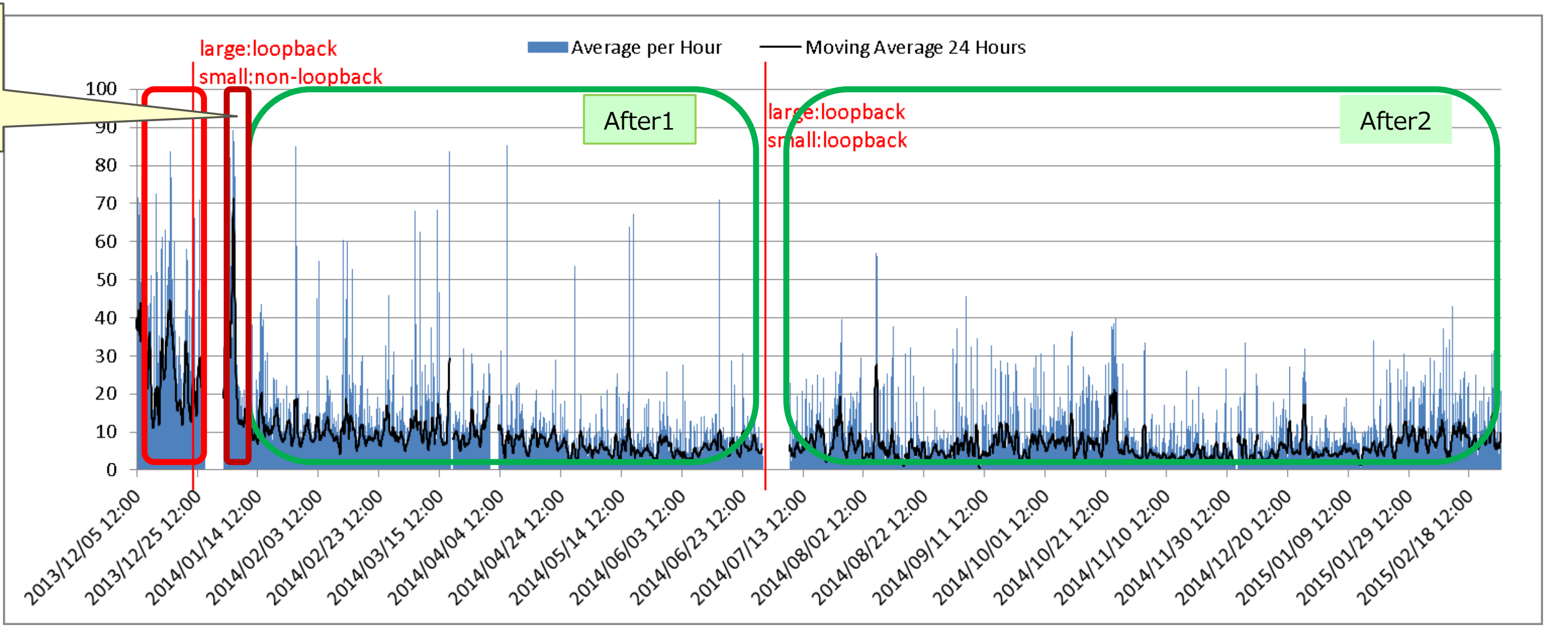


21,571 times Faster
unlink@9216 node

MDS CPU Load Comparison (Before vs. After)
Longtime evaluation except maintenance time(2013/12-2015/2)

- MDS Load average per hour: about 1/3.5
- Peak occurrence times per day(Over 50%,70%): less than 1/30

Some Large Class
Jobs did not apply
Rank Local
File System



Need to add some option to use loopback based rank directory

	Before -13/12/27	After1: -14/6/29	After2: -15/2/28	After (All)
Average MDS Load %	25.1	8.21	6.36	7.13
Over 50% times per day	2.32	0.12	0.02	0.06
Over 70% Times per day	0.68	0.04	0.00	0.02

Summary

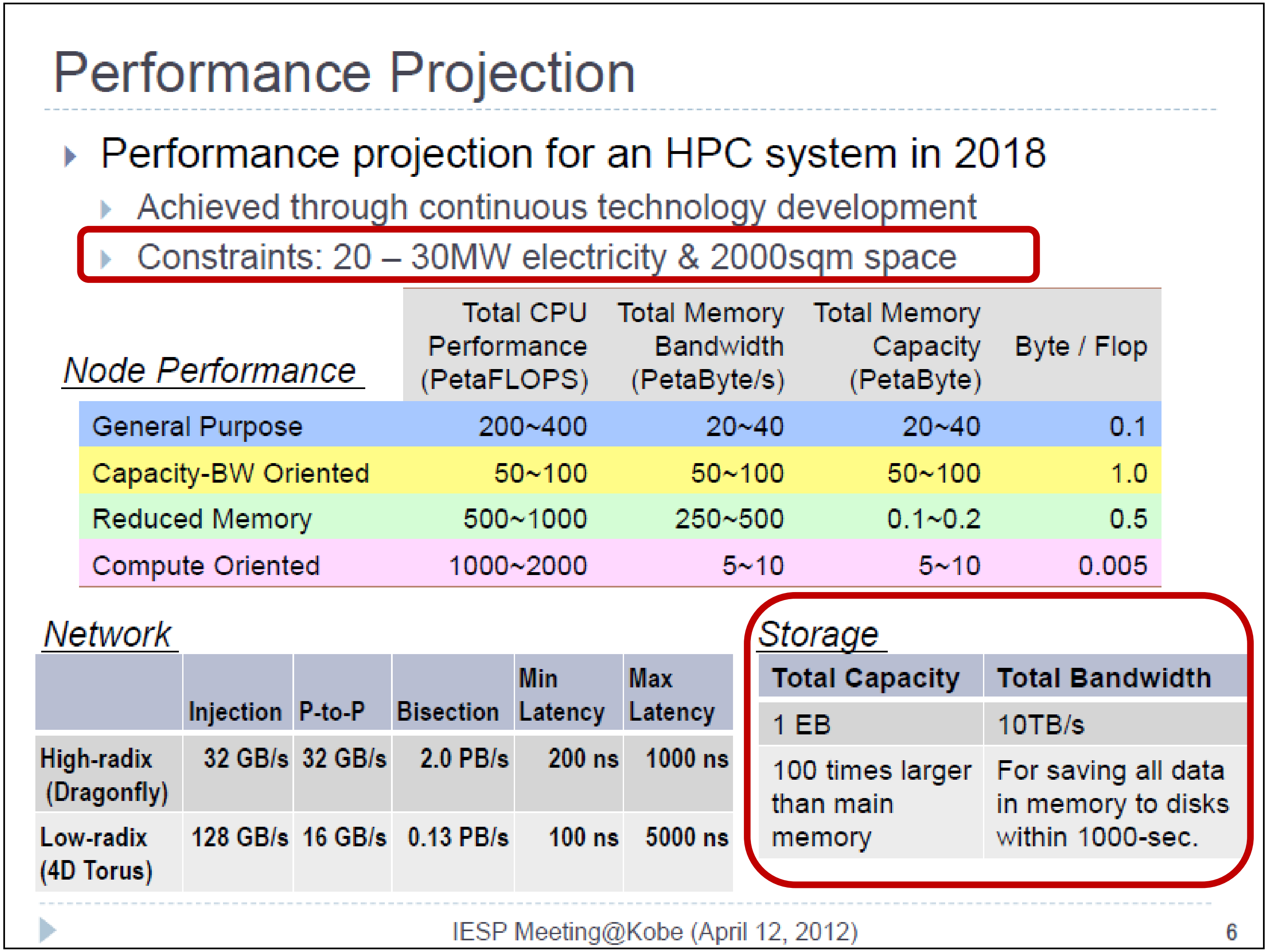
- Meta Data Access Distribution by Loopback File System
  - Distributing and leveraging Meta Data and Data Access
  - Providing higher constant access performance on rank local file
- Evaluation
  - Achieved Better File Access Performance up to 128MB
  - Loopback Based Local FS Dramatically Scales over 10,000 Nodes!
  - MDS Load average: about 1/3.5
  - Peak Occurrence Times per Day: less than 1/30

- Introduction of Loopback based rank local file system is very effective on K computer operation
even if 1 MDS+ 2,592 OSSes (5,184 OSTs) file system.

Yuta Higuchi,

Shinji Sumimoto (Fujitsu Limited)

Storage and System Requirement from the Architecture Roadmap (IESP 2012@Kobe)



Performance Projection

- Performance projection for an HPC system in 2018
  - Achieved through continuous technology development
  - Constraints: 20 – 30MW electricity & 2000sqm space

Node Performance	Total CPU Performance (PetaFLOPS)	Total Memory Bandwidth (PetaByte/s)	Total Memory Capacity (PetaByte)	Byte / Flop
General Purpose	200~400	20~40	20~40	0.1
Capacity-BW Oriented	50~100	50~100	50~100	1.0
Reduced Memory	500~1000	250~500	0.1~0.2	0.5
Compute Oriented	1000~2000	5~10	5~10	0.005

Network

	Injection	P-to-P	Bisection	Min Latency	Max Latency
High-radix (Dragonfly)	32 GB/s	32 GB/s	2.0 PB/s	200 ns	1000 ns
Low-radix (4D Torus)	128 GB/s	16 GB/s	0.13 PB/s	100 ns	5000 ns

Storage

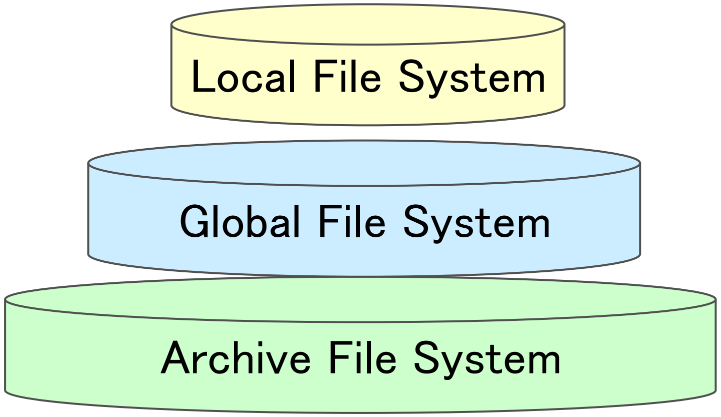
Total Capacity	Total Bandwidth
1 EB	10TB/s
100 times larger than main memory	For saving all data in memory to disks within 1000-sec.

IESP Meeting@Kobe (April 12, 2012) 6

Exascale File System Design

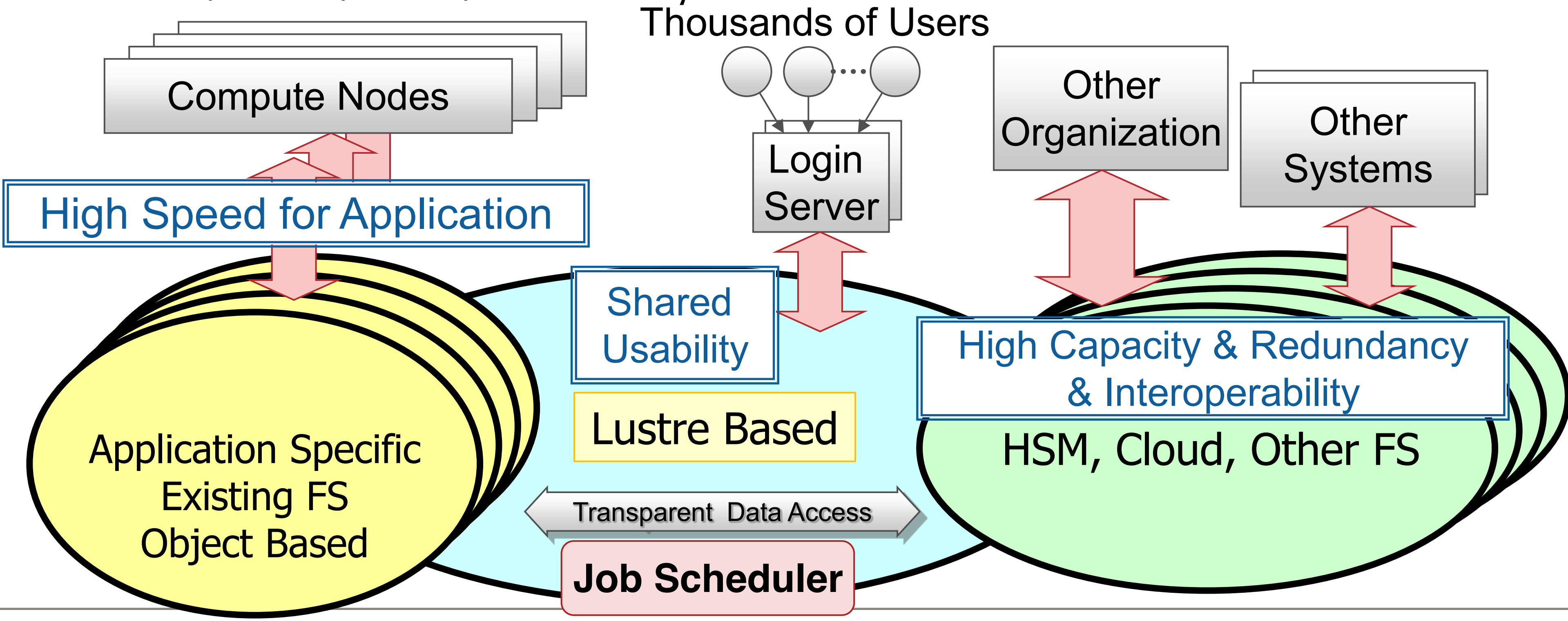
- Trade off: Power, Capacity, Footprint, Costs
  - Difficult to reach1EB and 10TB/s class file system on single file system in limited power consumption.

- Third Storage layer for Capacity is needed: Three Layered File System
  - Local File System for Performance
  - Global File System for Easy to Use
  - Archive File System for Capacity



The Next Integrated Layered File System Architecture for Exascale Systems (Presented at LUG 2013/Panel)

- Local File System(10PB Class): ex: Memory , SSD Based, etc..
  - Application Specific, Existing FS, Object Based, etc..
- **Global File System(100PB Class): ex: Disk Based, etc..**
  - Lustre Based, etc..
- Archive File System(1EB Class): ex: HSM(Disk+Tape) etc..
  - HSM, Lustre, Cloud, other file system



Issues of File System for Exascale Systems

FUJITSU

Discussed at last Summit 2014

- System Limits: Increase the logical upper limits (capacity, # of clients, # of OSTs, etc...)
- Memory Usage: Required memory should not proportional to # of OSTSs
- Meta Data Performance: Reduce metadata access. Lustre DNE improves metadata performance, but requires additional hardware resource, MDS and MDT. So, scalability is limited to hardware resource
- I/O Throughput and Capacity: Achieve higher throughput (10TB/s~) and larger capacity (~1EB) in limited power consumption and footprint
- System Noise: Eliminate OS jitter to maximize performance of massively parallel applications

- Will Be Discussed Today

- Power Consumption: Reduce power consumption of extreme large storage systems
- Dependability: Data must not be lost even if storage(RAID) failure, and operations should be resumed quickly
- Eviction:

Exascale Concerns (Summit 2015)

Power Consumption

- Concern: Reduce power consumption of extreme large storage systems
- Approach: Introduce low power device in hierarchical storage system
    - e.g. SSD for 1st layer (fast job I/O area), Tape device for the bottom layer (archive area)
  - And stopping hardware such HDDs in the storage devices, part of OSSs, etc
    - MAID for HDD (MMP prevents to use this)

Dependability

- Concern: Data must not be lost even if RAID storage gets defective, and operations should be resumed quickly
    - e.g. controller module failures, defective lot of disks, software bug, etc…
    - e.g. Running "lfs find" to find affected files takes a long time ..
    - e.g. Running fsck on the storage cloud take a month.
- Approach?: OST-level RAID(LU-3254 by Jinshan)

← Good idea, but RAID1 requires doubled space. RAID−5 maybe?

- One Approach: File Services should not be stopped even if some storages are offline.

Exascale Concerns (Summit 2015) (Cont'd)

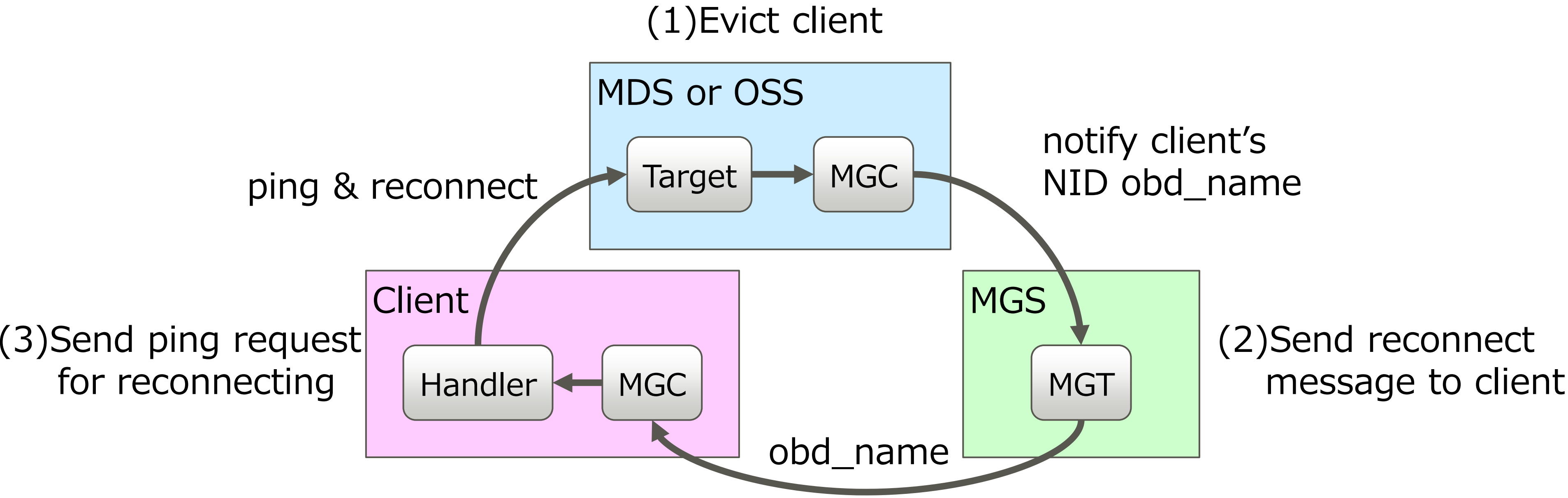
Eviction

- Background:
  - Since OBD ping causes OS jitter, we have suppressed it.
    (LU-2467)
  - Introduced alternative solutions (e.g. cooperation with hardware), but still have problems with notifying the clients their evicted status.
    (reported at LUG2015)
- Concern: The clients do not notice if they are evicted until they do file I/O, which ends up EIO.

Exascale Concerns (Summit 2015) (Cont'd)

- Approach:

1. When a server evicts a client, the server notifies MGS
2. MGS notifies the evicted client to connect the server
3. The client sends ping request to the server
   (LU-6657: eviction notifier)



議論：Dependabilityについて

- Lustre･FEFSはディスク故障は起きない前提で設計されている
  - RAIDやミラーリング、サーバ故障時もFailover時にディスクアクセスが継続
- しかし、稀にストレージ故障によりデータが失われる場合が発生しうる
  - 2台同時故障 on RAID5, Disk Firmware問題など、、
- Lustre･FEFSのデータ復旧は2段階で実行
  - Backend Filesystemのfsck
  - OSTのfsck：lfck
- OSTの規模が大きくなると復旧時間が課題
  - Backend Filesystem復旧中はファイルシステムアクセス不能
  - どのファイルのデータが失われたかを知るには全ファイルのスキャンに時間がかかる
- 復旧時間の短縮に加え、短時間でのサービス再開が求められている
  - Backend Filesystemの復旧ができればサービス復旧は可能
  - しかし、RAID復旧不能の時点でBackend Filesystemは大きなダメージ
  - 一部のOSTがOffline状態でもサービス復旧できるほうがよいケースがあり得る

議論：Dependabilityについて(2)

- RAID故障の対策：RAID故障は本来は起きてはならない
  - システム設計上はストレージで対策すべき問題
  - 原則、更なる多重化しかないのが現状
  - 対策の現状： 機器コスト、性能、運用停止時間を含め総合的に考える必要
- OSTデータの多重化
  - 全データ多重化
    - OST毎多重化： RAID1, RAID1+6, 5+6, 6+6
    - HSM機構を活用した多重化：バックアップとしてHSM機構を活用
  - 部分データ多重化
    - ファイルレベル多重化： ファイルを選択して多重化
    - Backend Filesystemメタデータのみ多重化：復旧高速化、容量は節約可
  - 多重化レベル
    - ストレージを含むBackend Filesystemレベル
    - 多重化に対応したOST： Spare OSTの提案
- 今後も引き続きLustreコミュニティと方策を議論していく

CONTRIBUTION STATUS AND PLAN

Fujitsu Contributions until 2014

- Fujitsu have submitted Lustre enhancements with Intel.

Jira	Function	Landed
LU-2467	Ability to disable pinging	Lustre 2.4
LU-2466	LNET networks hashing	Lustre 2.4
LU-2934	LNET router priorities	Lustre 2.5
LU-2950	LNET read routing list from file	Lustre 2.5
LU-2924	Reduce ldlm_poold execution time	Lustre 2.5
LU-3221	Endianness fixes (SPARC support)	Lustre 2.5
LU-2743	Errno translation tables (SPARC Support)	Lustre 2.5
LU-4665	lfs setstripe to specify OSTs	Lustre 2.7

Fujitsu Contributions in 2015 (1)

- We are submitting new features for Lustre.

Jira	Feature	Submission Status
LU-6531	Fujitsu's o2iblnd Channel Bonding Solution (IB multi-rail)	In Review
LU-6657	Eviction Notifier (Automated Eviction Recovery)	In Review
LU-6658	single stream write performance improvement with worker threads in llite (Single Process IO Performance Improvement)	In Review

Fujitsu Contributions in 2015 (2)

- We are submitting bug-fixes for Lustre as well.

Jira	Patch	Submission Status
LU-6600	Race lustre_profile_list	Landed to Lustre 2.8
LU-6624	LBUG in osc_lru_reclaim	Landed to Lustre 2.8
LU-6643	write hang up with small max_cached_mb	In Review
LU-6732	Cannot pick up EDQUOT from ll_write_begin and ll_write_end	In Review

Fujitsu Contributions in Future

- Fujitsu will continue submitting new features.

Feature	Submission Schedule
Directory Quota	2nd half of 2016
Client QoS	2nd half of 2016
Server QoS	TBD
Memory Usage Management	TBD
Snapshot	TBD

- We will also submit Lustre 2.x bug-fixes in 2016.

FUJITSU

shaping tomorrow with you