JLUG2016

FUJITSU

shaping tomorrow with you

Issues and Directions for the Next Generation Shared File System

Oct. 18 2016

Shinji Sumimoto (Fujitsu Limited)

OpenSFS®

Outline of This Talk

- FEFS Development towards Exascale
  - Exascale File System Design
  - Exascale Storage Design

- 次世代共有ファイルシステムの課題
  - ファイルアクセス性能向上
  - 耐故障性向上
  - 保守性向上
  - 負荷耐性向上
  - アクセス公平性確保
  - 省メモリ性向上

Fujitsu’s FEFS Development towards Exascale

- Fujitsu will continue to develop Lustre based FEFS to realize the next generation exascale systems.
  - Needs continued Lustre enhancements
- FEFS already supports Exa-byte class file system size
  - However, several issues to realize real Exascale file system

- Topics
  - Exascale File System Design
  - Exascale Storage Design

Exascale File System Design

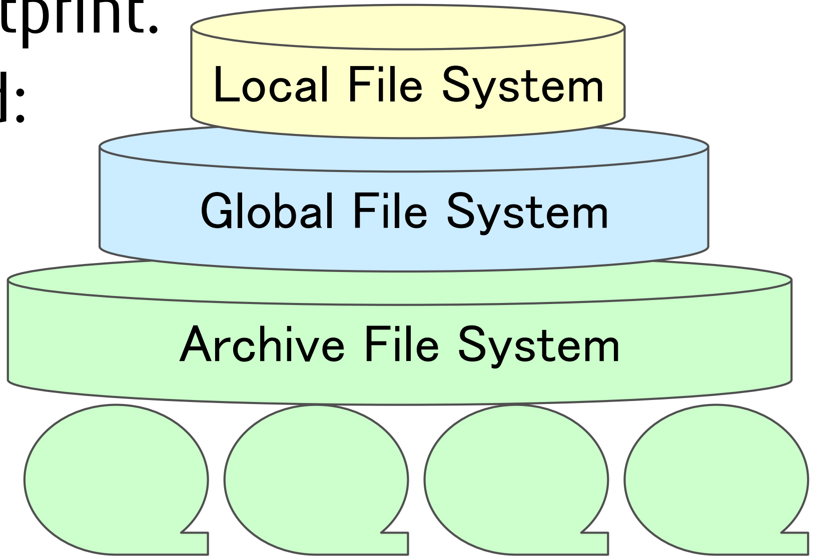
FUJITSU

K computer File System Design

- How should we realize High Speed and Redundancy together?
- How do we avoid I/O conflicts between Jobs?
- These are not realized in single file system.
  - Therefore, we have introduced Integrated Layered File System.

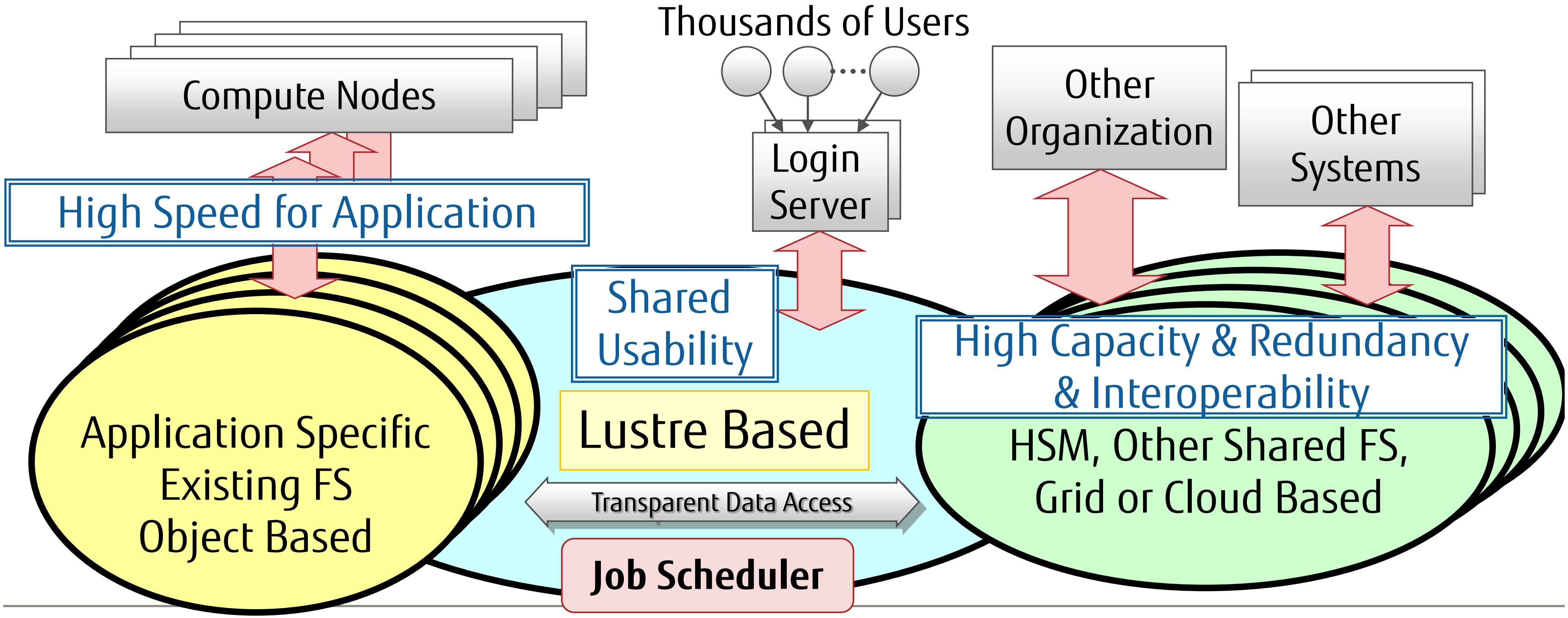
Exascale File System/Storage Design

- Another trade off targets: Power, Capacity, Footprint
  - Difficult to realize single 1EB and 10TB/s class file system in limited power consumption and footprint.
- 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 Post-peta scale System (Feasibility Study 2012-2013)

- Local File System o(10PB): Memory, SSD, HDD Based
  - Application Specific, Existing FS, Object Based, etc..
- **Global File System o(100PB): HDD Based**
  - Lustre Based, Ext[34], Object Based, Application Specific etc..
- Archive System o(1EB): HSM(Disk+Tape), Grid, Cloud Based
  - HSM, Lustre, other file system


Thousands of Users
Compute Nodes
Login Server
Other Organization
Other Systems
High Speed for Application
Shared Usability
High Capacity & Redundancy & Interoperability
Application Specific Existing FS Object Based
Lustre Based
Transparent Data Access
Job Scheduler
HSM, Other Shared FS, Grid or Cloud Based

Exascale Storage System Design

- **Requirements:**
  - Electric Power and Space including Computing System and Storage
    - Total Limitation of Electric Power: 30-40MW
    - Total Limitation of Space: 2000m2
  - Electric power for storage system must be minimized because most of the power should be used for computing.
    - Power Consumption of Storage System: Less than 1MW (as assumption)
- **Basic Estimation for Device Design as First Step**
  - Only Media Device Estimation using Actual Device Parameters

Current Disk Media Parameters: 2016

Seagate	Cap. (TB)	Power W (Idle)	R Perf. (GB/s)	AFR (%)	Price$
SATA 3.5	10.0	8(4.5)	0.254	0.35	850
SAS 2.5	1.8	7.8(4.5)	0.241	0.44	200(0.9T)
PCIe SSD	3.2	5.93(4.38)	1.9 (W0.85)	0.35	9.7K

SATA 3.5inch: Seagate Enterprise Capacity 3.5 10TB
SAS 2.5inch: Seagate Enterprise Performance 10K
SAS SSD: Seagate 1200.2 SAS SSD

- Only disk media factors were calculated
  - Not included: additional parity and spare disks, servers, etc..
- Calculating # of Racks
  - 840 Disks on 1 Rack for 3.5 inch Disks (4U 84drive, 40U 1 Rack)
  - 2520 Disks on 1 Rack for 2.5 and SSD Disks (3 times higher density than 3.5 inch)

10TB/s Storage Estimation

Seagate	Cap. (PB)	Power (W)	# of Racks	Faults /day
SATA 3.5	394	0.3M	46	0.37
SAS 2.5	74	0.3M	17	0.50
PCIe SSD	16.8	0.03M	3	0.05

- Rack space and Power are becoming issues to solve.
  - Recovery of RAID must be done within a day for SATA3.5 case.

1EB Storage Estimation

Seagate	Perf.(TB/s)	Power (W)	# of Racks	Faults /day
SATA 3.5	25.4	0.8M	120	0.96
SAS 2.5	134	4.3M	221	6.7
PCIe SSD	594	1.9M	125	3.0

- Power, Rack Space, and Faults per day are problems.
  - Power consumption becomes impact to realize, rack space also huge.
  - RAID 6 is not enough
- Difficult to realze1EB storage by using disk and SSD drive.
  - Cold Storage and Cloud Wide Storage are options to use.
- Evolutional improvement will be needed, not only computing node, but also storage media.

Issues and Directions for Exascale Storage

- **10TB/s is able to Realize, 1EB is difficult because of Power and Faults/day issues.**
  - Power consumption becomes impact to realize.
    - MAID（Massive Array of Idle Disks）, Tape Storage
  - RAID 6 is not enough:
    - RAID 6+1,6+5,6+6
    - Dynamic Disk Pools（DDP） by NetAPP

- **Another Required Storage: Large Capacity with low power consumption such as HSM (Tape/Cloud Based)**
  - Add third storage as capacity storage based on HSM to realize 1EB storage, and realize 100PB as global storage.

性能 vs. 対故障性 のTrade Off

- PCIe結合のSSDが出てきたことによって高性能なストレージが実現可能になった
- 一方、SSD自体のAFRはSATA HDDと同等レベルと高性能と裏腹に信頼性への課題が残る。
- 性能 vs. 対故障性 のTrade Off
  - 選択肢１ ：　性能優先、対故障性は考えない。
  - 選択肢２ ：　対故障性確保、性能はある程度犠牲
  - 他の選択肢は?
- 使われ方を含めて考える必要がある。

次世代共有ファイルシステムの課題

- ファイルアクセス性能向上
  - メタデータアクセス性能向上
  - 単体I/O処理高速化
- 耐故障性向上
  - フェイルオーバ時のI/O エラー防止
  - ファイルサーバ異常発生からファイルI/O 再開までの時間短縮
  - 両系故障時のアクセスハング抑止
- 保守性向上
  - ファイルシステム復旧時間短縮
- 負荷耐性向上：メタアクセス、データアクセス
  - MDS高負荷防止: 大量ファイルアクセス時のメタアクセスレスポンス低下防止
  - インタコネクト高負荷時のノード異常の誤検出回避
- アクセス公平性確保
  - 特定プロセスのファイルアクセスによりほかのプロセスが大きく影響を受ける
- 省メモリ性向上

ファイルアクセス性能向上

- **Increasing Meta Data Access Performance**
  - Parallelized MDS: DNE
  - Distributed Meta Data Processing: System Level or User Level

- **Increasing User Data Access Performance**
  - Increasing File I/O Performance using system call
  - Increasing Lustre Client File I/O Performance

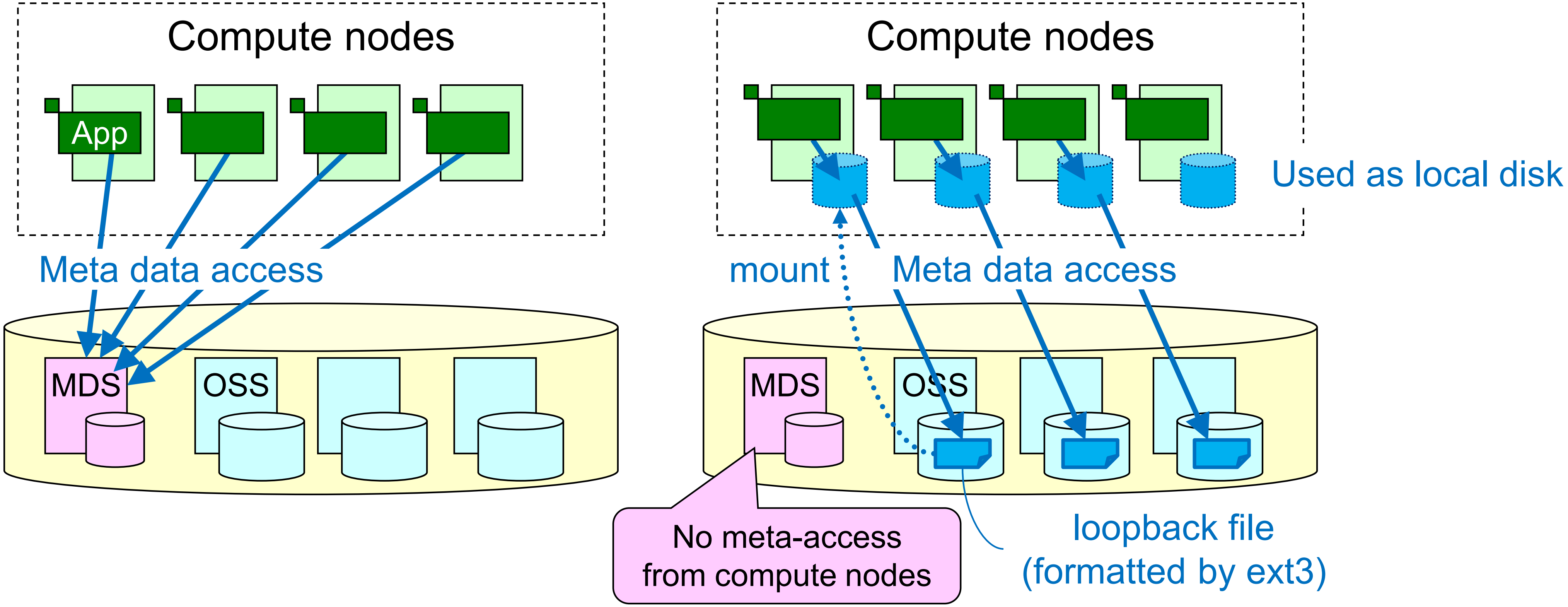
Increasing Meta Data Access Performance

- 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 Ext3 file system and Loopback Layer Overhead

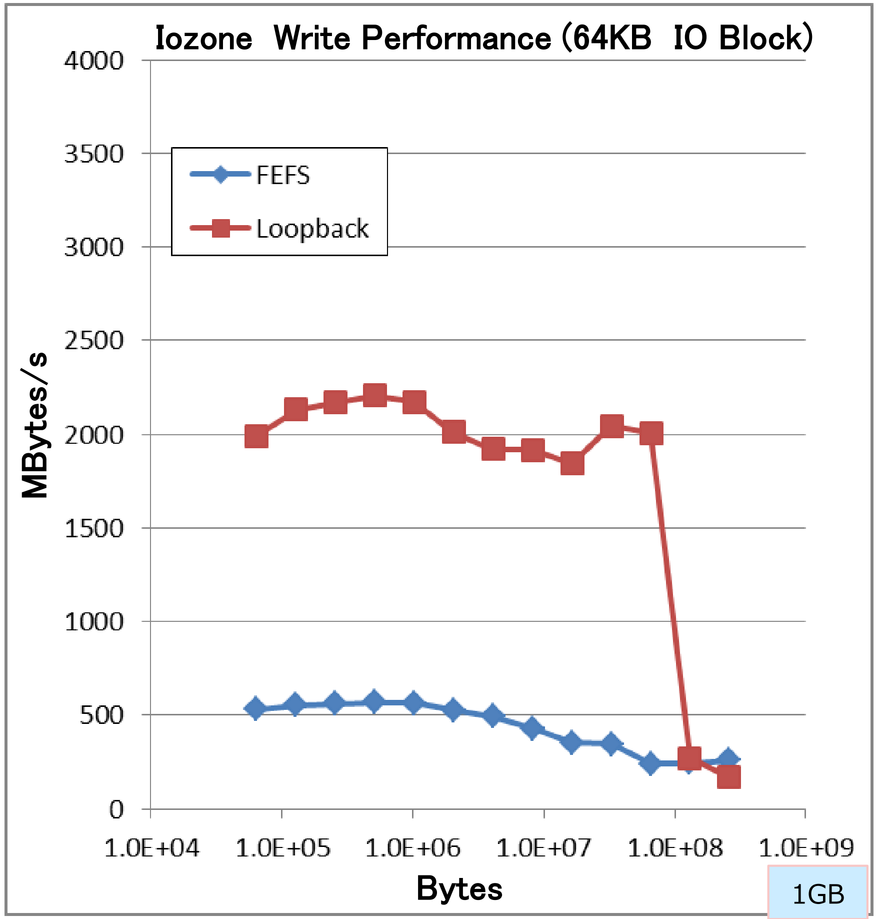
Distributing Meta-access by Loopback Device

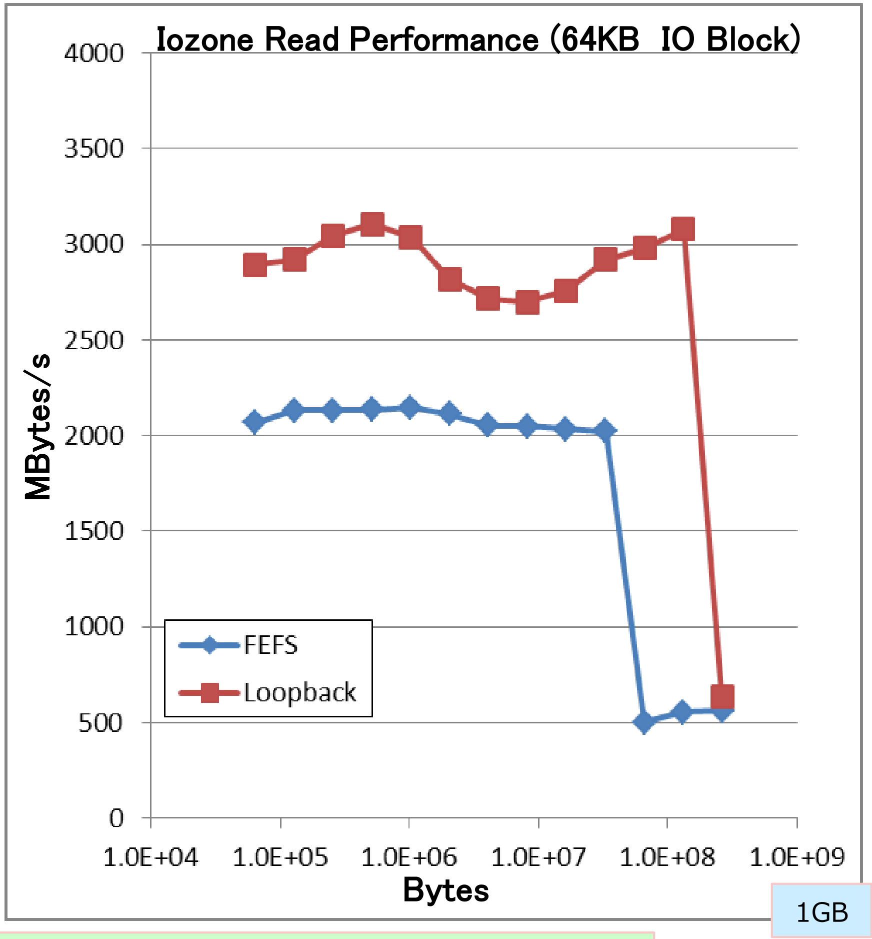
- Move meta-operation from MDS to compute nodes utilizing loopback device.
  - Loopback device can be used as dedicated local disk.
  - meta-access is closed inside of the compute node.
    - MDS accesses doesn't occur in meta-operation such as open/close.



Single Node File Access Performance on K computer

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



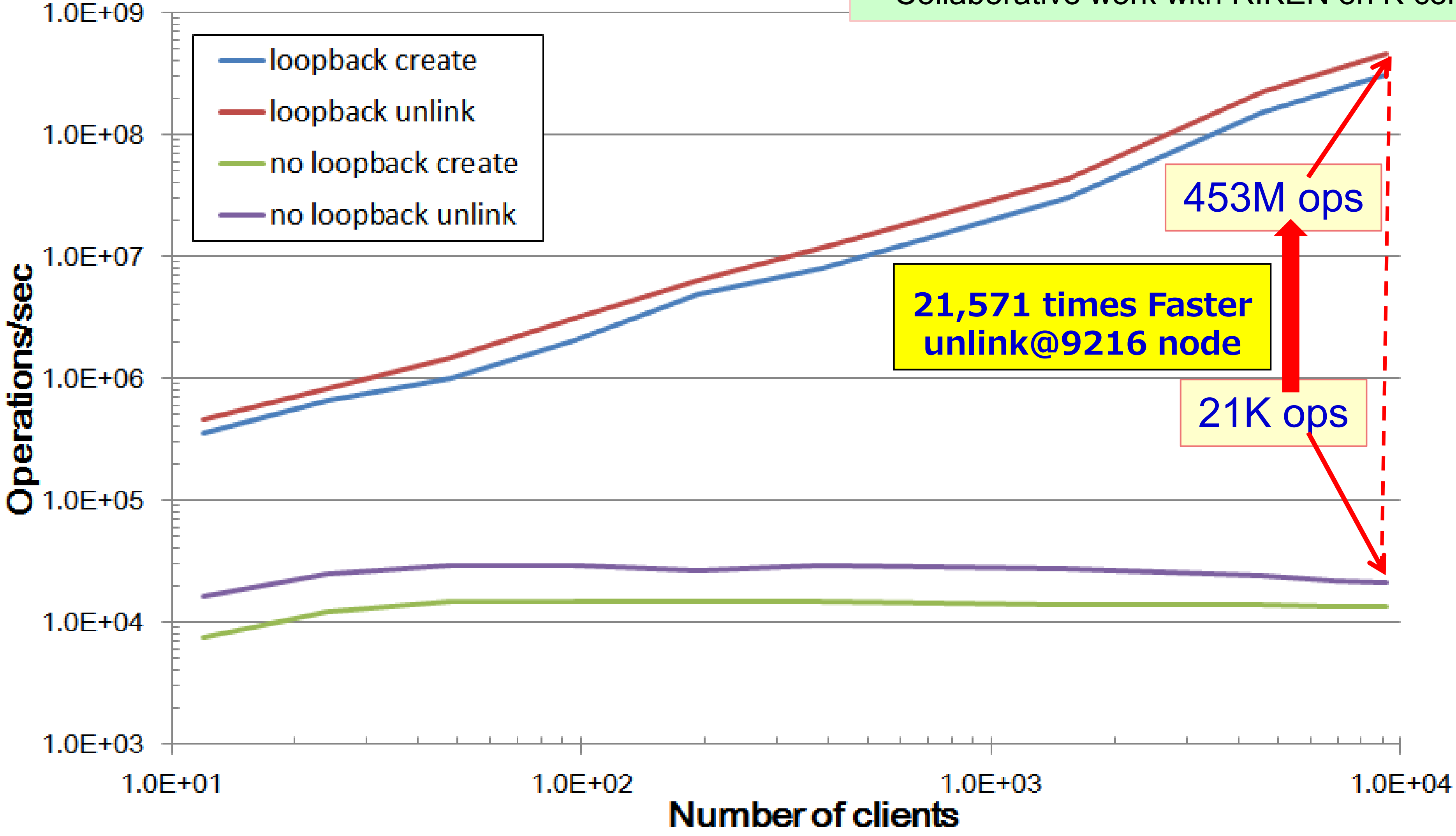


Collaborative work with RIKEN on K computer

Total Meta Data Access Performance

- Loopback Based Local FS Dramatically Scales over 9,000 Nodes!
  - Create 26K ops/node, unlink 37K ops/node by mdtest 100 files/node
  - Providing higher constant meta data access performance for each node

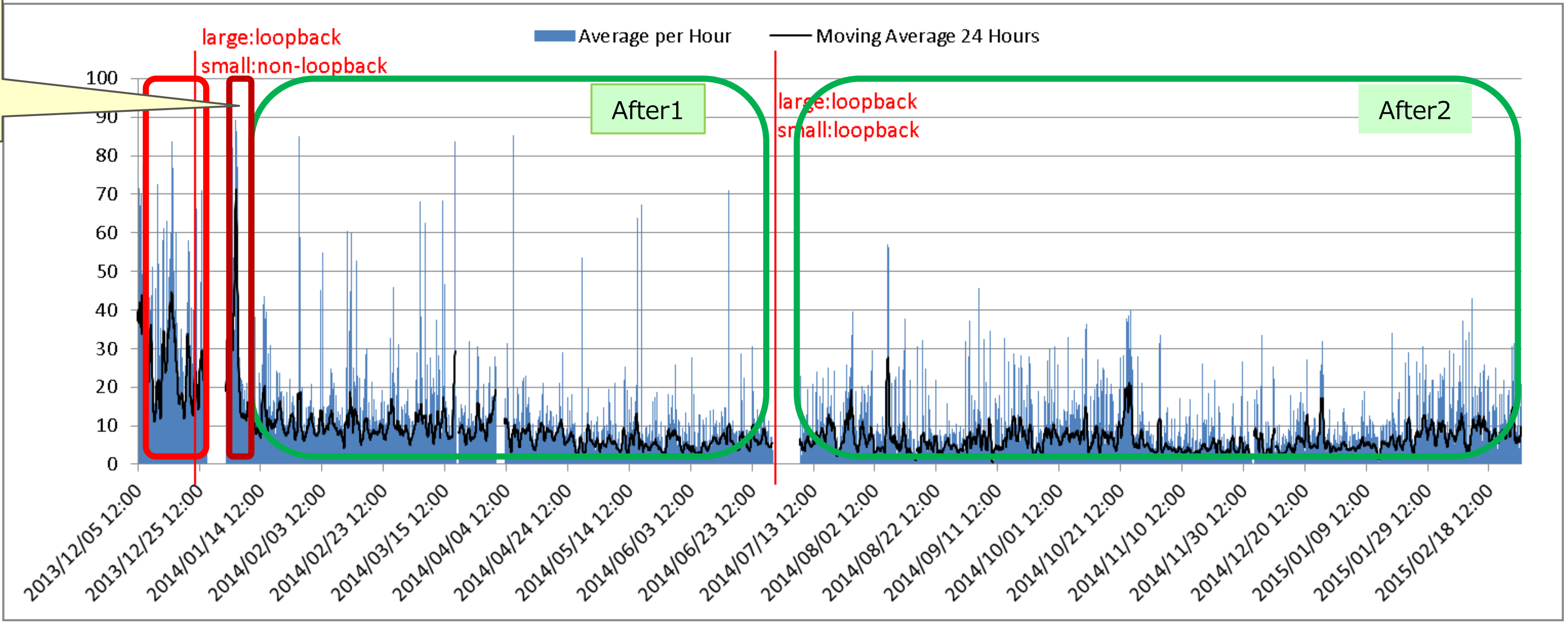
Collaborative work with RIKEN on K computer



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



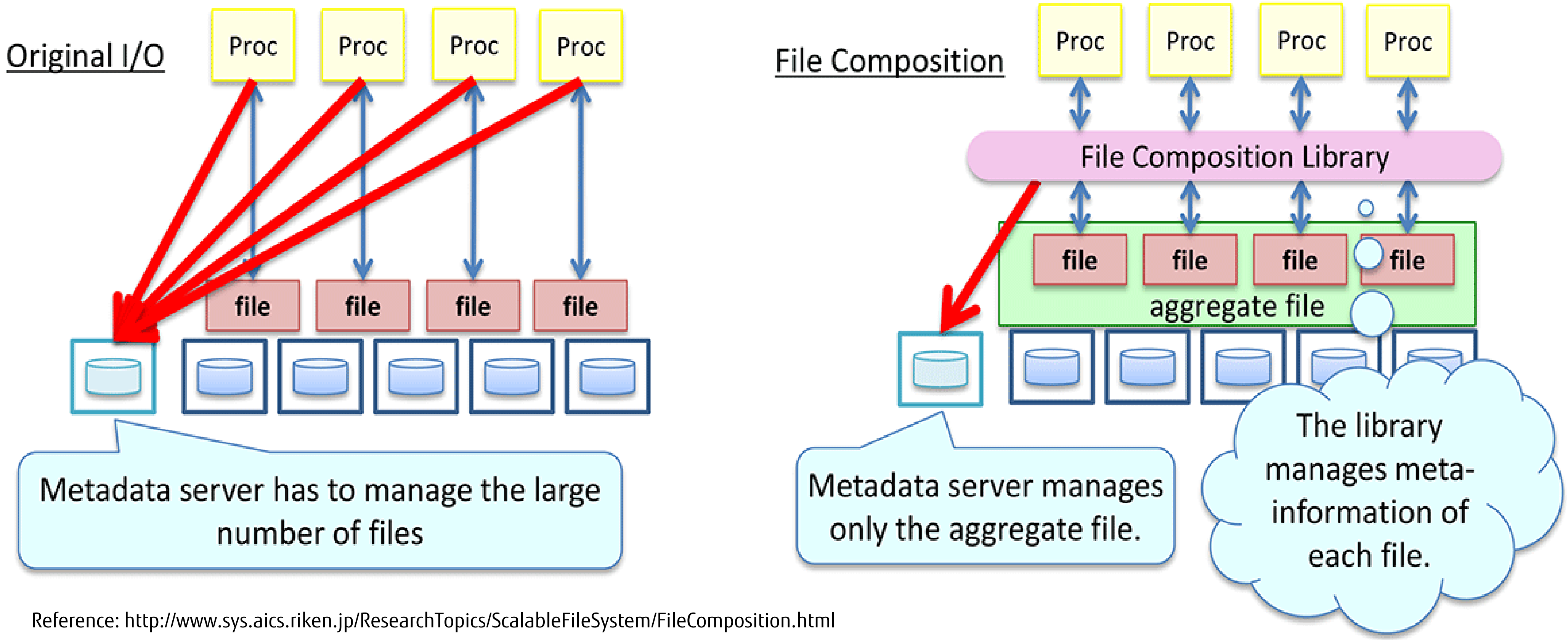
Need to add some option to use loopback based rank directory

	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

Collaborative work with RIKEN on K computer

User Level Approach for Metadata Performance

- Reduce metadata access to MDS by user level library
  - Provide intermediate layer to absorb metadata access between compute node and file system
- File Composition Library by RIKEN AICS



Metadata server has to manage the large number of files

Metadata server manages only the aggregate file.

The library manages meta-information of each file.

Reference: http://www.sys.aics.riken.jp/ResearchTopics/ScalableFileSystem/FileComposition.html

Increasing User Data Access Performance

- **Increasing File I/O Performance using system call**
  - Linux VFS Issues
  - Many Core Processor Issue

- **Increasing Lustre Client File I/O Performance**

Increasing File I/O Performance using system call

- **Linux VFS Issues**
  - Issues:
    - User Space to Kernel Space Copy by Page Size Basis
    - Write() system call needs to fd write lock
  - Solution
    - Needs to Increasing Page Size
    - Uses DIRECT I/O: Current FEFS Zero-Copy transfer.
- **Many Core Processor Issue**
  - Issue:
    - Single Core Copy Performance > Interconnect Bandwidth: Copy performance is bottleneck
  - Solution:
    - Multi-Threaded Copy with elimination of fd locks

Background of High Performance Communication on Many Core Processor for Exa-scale MPI

- Communication Bandwidth of Interconnect continues to increase:
  - Tofu(5+5)GB/s x 4 → Tofu2(12.5+12.5)GB/s x 4
  - Multiple RDMA engines: 4 RDMAs on Tofu and Tofu2
- Single CPU Core Processing Frequency will not increase dramatically because of Power Wall
  - This is because many core CPUs become to use widely
  - Xeon Phi: around 1GHz, FX10: 1.8GHz
- The problem is memory copy performance of single CPU core will not increase dramatically!

Issues of Single Core Memory Copy Performance Limitation on MPI

- Two Major Issues in case of Network Bandwidth >Single CPU Copy Bandwidth:
  - Simple Intra-Node Communication
    - Simple Communication such as MPI_Send, MPI_Recv
  - Collective Communication especially handling multi-rail network
    - Simple Communication such as MPI_Bcast
    - Communication with Arithmetic Computation such as MPI_Reduce
- Solution
  - Hardware Offload: DMA Engine, Loopback Data Transfer
  - Multi Threaded Communication Processing

Xeon-Phi Case: MVAPICH2-MIC

- MVAPICH2-MIC: A High-Performance MPI Library for Xeon Phi Clusters with InfiniBand, by **Sreeram Potluri** at Extreme Scaling Workshop, August 2013.

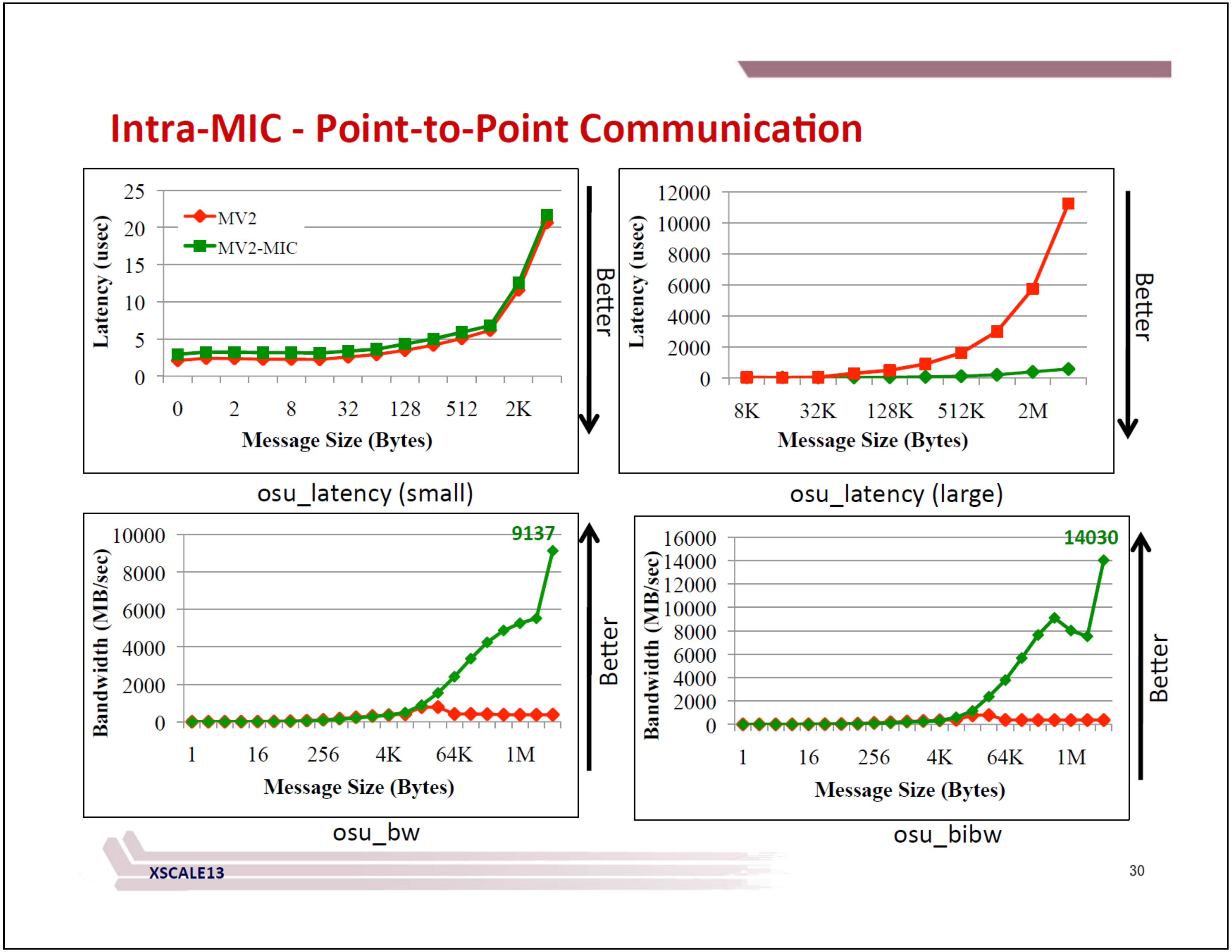
  http://nowlab.cse.ohio-state.edu/publications/conf-presentations/2013/Sreeram-XSCALE13.pdf

- Approach:
  - Short Message Size: Using CPU Copy
  - Long Message Size: Using SCIF(DMA Engine)

- Results:
  - Pros: Simple Intra-Node Performance
  - Cons: DMA engine Resource Bottleneck in case of number of processes Communication with Arithmetic Computation
  - Multi Threaded Communication Processing will be needed

Xeon Phi Case: MVAPICH2-MIC Solution

http://nowlab.cse.ohio-state.edu/publications/conf-presentations/2013/Sreeram-XSCALE13.p



Intra-MIC - Point-to-Point Communication

osu_latency (small)

osu_latency (large)

osu_bw

osu_bibw

XSCALE13

30

Xeon-Phi Case: MT-MPI

- MT-MPI: Multithreaded MPI for Many-core Environments, Min-Si at ICS-14, June 2014

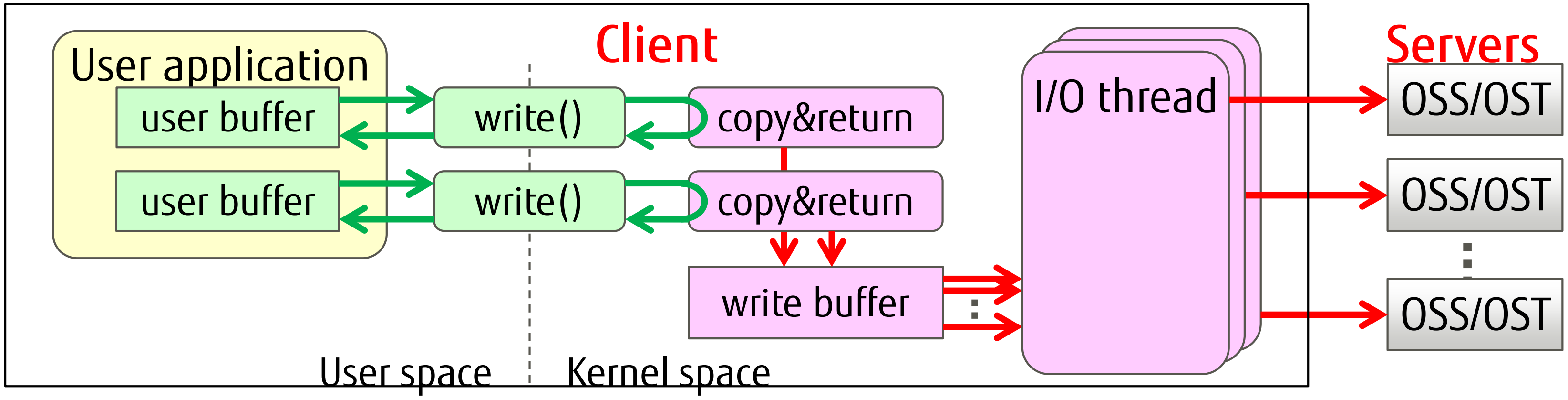
  http://www.il.is.s.u-tokyo.ac.jp/~msi/pdf/ics2014-mtmpi.pdf

- Approach: Multi Threaded Communication Processing by OpenMP
  - Derived Data Type Processing for Pack, Un-pack
  - Shared Memory Communication for Long Messages

- Results:
  - Pros: Higher Performance
  - Cons: Depending on number of Idle Threads

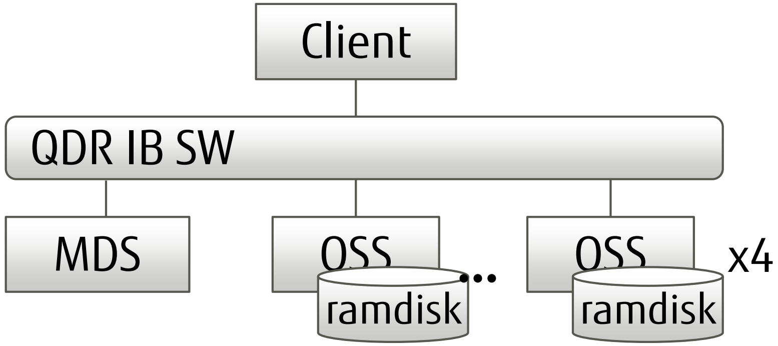
Improving Single Client Process Write Performance

- Important for clients to write a large amount of data such as checkpoint file
- Issue
  - Striping isn't effective to improve single process I/O performance
    - There're some bottlenecks in Lustre's cache method using dirty buffer for each OST
- Our Approach
  - write returns immediately after copying user data to kernel buffer internally
  - Dedicated I/O threads transfer data from the buffer to OSS/OSTs in parallel, therefore write throughput dramatically improves from user perspective

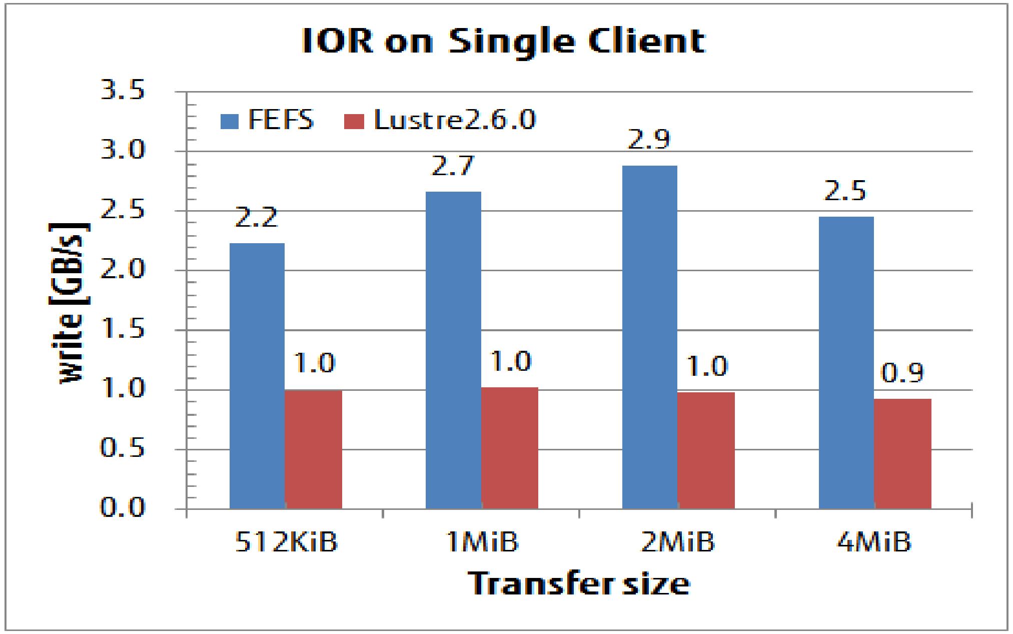


Improving Single Process Write Performance

- Lustre 2.6.0 vs. prototype (Lustre 1.8 base)
- OSS/Client
  - CPU: Xeon E5520 2.27GHz x2
  - IB: QDR x1
- OST
  - ramdisk x4
- IOR
  - 1-process



- Result
  - Lustre 2.6.0 0.9~1.0GB/s
  - Prototype 2.2~2.9GB/s



耐故障性向上

- フェイルオーバ時のI/O エラー防止
  - フェイルオーバ時にクライアントが一定時間内にサーバにアクセスできないとEvictされクライアントが発行したI/OシステムコールがI/Oエラーとなる場合がある
  - 一般的な解決手法としてはタイムアウトを長くすることであるが、フェイルオーバ時間が長くなる
- ファイルサーバ異常発生からファイルI/O 再開までの時間短縮
  - 時間短縮にはタイムアウト時間の短縮が効果があるが、上記のEvict問題が発生
- 両系故障時のアクセスハング抑止
  - これ以降で説明

両系故障時のアクセスハング抑止

- 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

保守性向上:ファイルシステム復旧時間短縮

- 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

負荷耐性向上：メタアクセス、データアクセス

- **MDS高負荷防止: 大量ファイルアクセス時のメタアクセスレスポンス低下防止**
  - MDS処理の負荷分散対応システム設計と大量ファイルアクセスするクライアントからのアクセスをQoSにより抑制する双方面での対策が必要

- **インタコネクト高負荷時のMDS,OSS,ノード異常誤検出回避**
  - MDS, OSS, 計算ノードの故障検出: タイムアウト
    - ネットワークの高負荷時に、故障検出のためのプロトコルがタイムアウト後MDS,OSS,ノード異常と誤検出されてしまう場合がある
  - タイムアウト時間増大による施策：フェイルオーバー時間が長くなり、ファイルI/O中断時間が長くなる問題がある
  - フェイルオーバ中のインタコネクト高負荷状態の副作用：
    - フェイルオーバ時にクライアントがEvictされ、実装中のシステムコールがエラーになることがある

アクセス公平性確保

- 課題：特定プロセスのファイルアクセスによりほかのプロセスが大きく影響を受ける

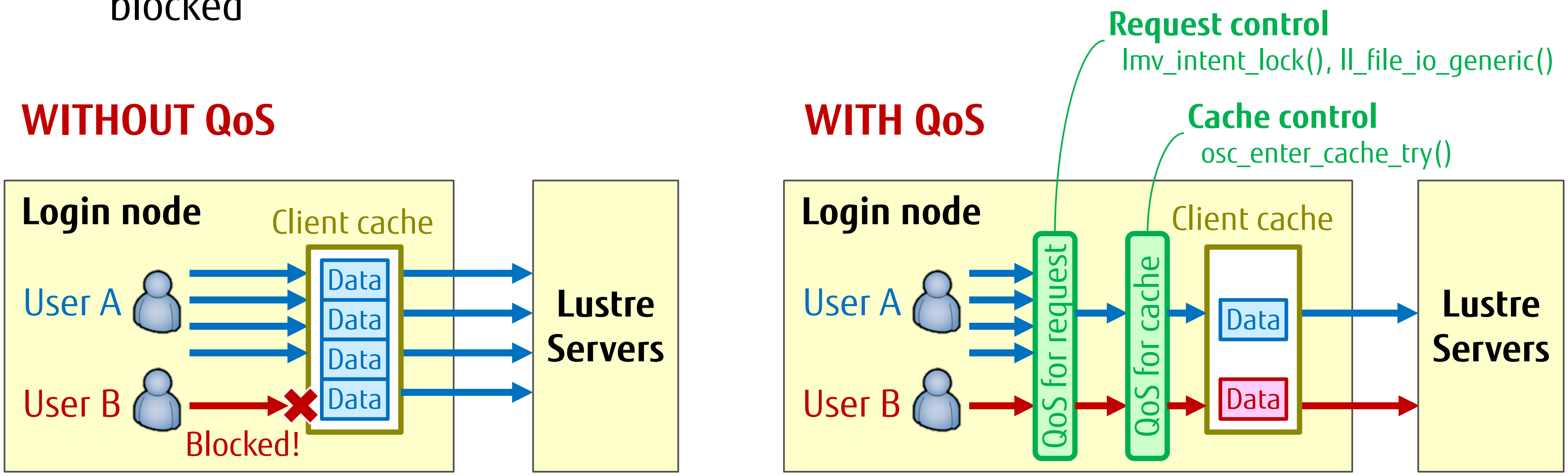
- 解決のアプローチ
  - ユーザ（ジョブ）毎のファイルアクセスを一定数内に制限するQoS機構の導入

- QoS機構の導入視点
  - サーバ・クライアントでのQoS

- FEFS: Client QoS拡張について説明

Client QoS (Quality of Service)

- Provides fair-share access among users on a single Lustre client
- Issue
  - I/O heavy user degrades I/O performance of other users on the same node
- Approach
  - Restricts the maximum number of meta and I/O requests issued by each user
    - Prevents a single user occupies requests issued by the client
  - Restricts the maximum amount of dirty pages used by each user
    - Prevents a single user occupies client cache and write requests of other users are blocked



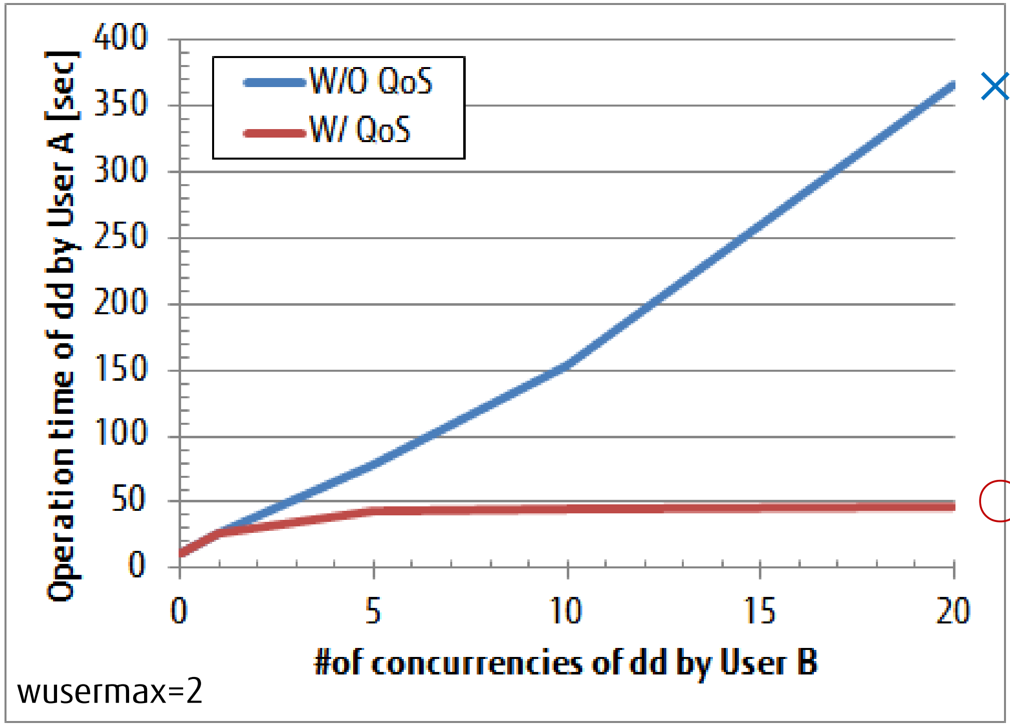
Client QoS: Efficiency

Test pattern

- dd if=/dev/zero of=/mnt/fefs/out.dat bs=1048576 count=2000
- User A: dd x1
- User B: dd x1~20

Result

- Processing time of User A is kept almost constant



× Execution time becomes very long

○ Execution time is almost kept constant

エクサスケール規模実現と省メモリ性向上

- System Limits
  - Concern: File system capacity must be exabyte class
    - e.g. One of exascale application “COCO” could output 860PB per job
  - Approach: Increase the logical upper limits
    - At least, eliminate restriction caused by 32-bit data length
- Memory Usage
  - Concern: secure the certain amount of memory space on the clients for computations
    - Compute node of K computer ran out of memory only by mounting file system
    - We reduced memory usage drastically for K computer (reported at LAD12)
  - Approach: Controlling memory usage strictly (e.g. page cache)
    - Break away from scale dependency (e.g. number of OSTs)

Power Consumption Reduction

- 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)

FUJITSU

shaping tomorrow with you