Lustre Developer Summit 2014

FUJITSU

shaping tomorrow with you

Fujitsu's Contribution to the Lustre Community

Sep.24 2014

Kenichiro Sakai, Shinji Sumimoto
Fujitsu Limited, a member of OpenSFS

Outline of This Talk

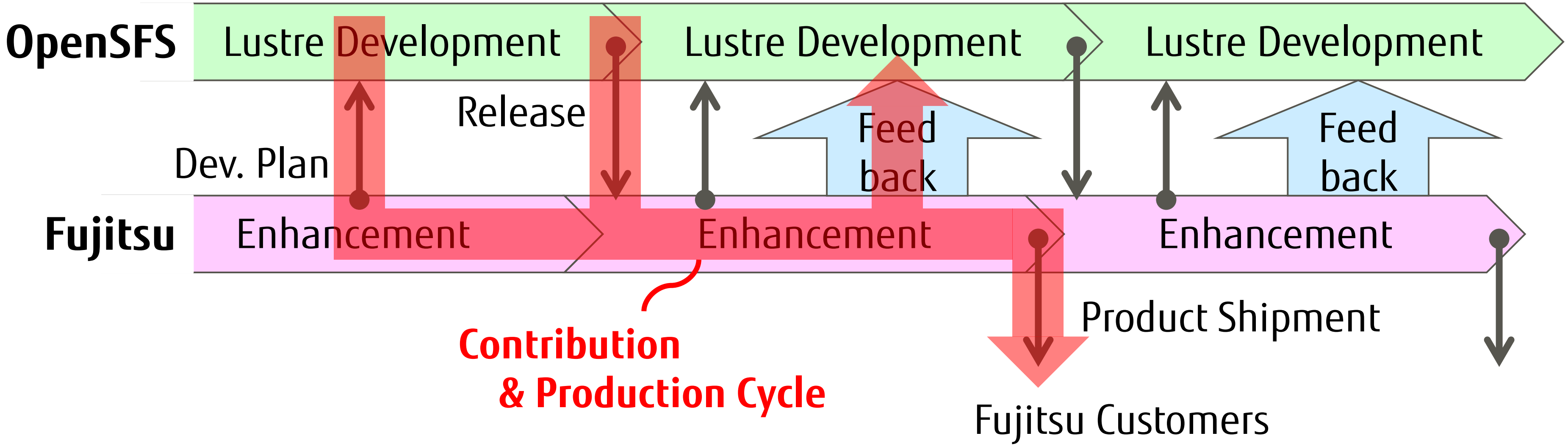
- Fujitsu's Development and Contribution Policies
  - Fujitsu's Lustre Contribution Policy
  - Contribution plan
  - Roadmap
- Introduction of Contribution Features
  - IB Multi-Rail
  - Automated evict recovery
  - Directory Quota
  - Improving single process I/O performance
  - Client QoS
- Challenges Toward Exascale Era
  - Concerns for exascale file system

Fujitsu's Development and Contribution Policies

- Fujitsu's Lustre Contribution Policy
- Contribution plan
- Roadmap

Fujitsu's Lustre Contribution Policy

- Fujitsu will open its development plan and feed back it's enhancement to Lustre community
  - LAD is the most suitable place to present and discuss.
- Fujitsu's basic contribution policy:
  - Opening development plan
  - Feeding back its enhancement to Lustre community no later than after a certain period when our product is shipped.



Contribution Plan

- Fujitsu's now porting our enhancements into Lustre 2.x
  - These features were implemented in FEFS based on Lustre 1.8
  - They've been used in our customer's HPC system, including K computer
- We'll start submitting patches for Lustre in 2015
  - Lustre 2.6 bugs are found during porting → We'll submit their patches too

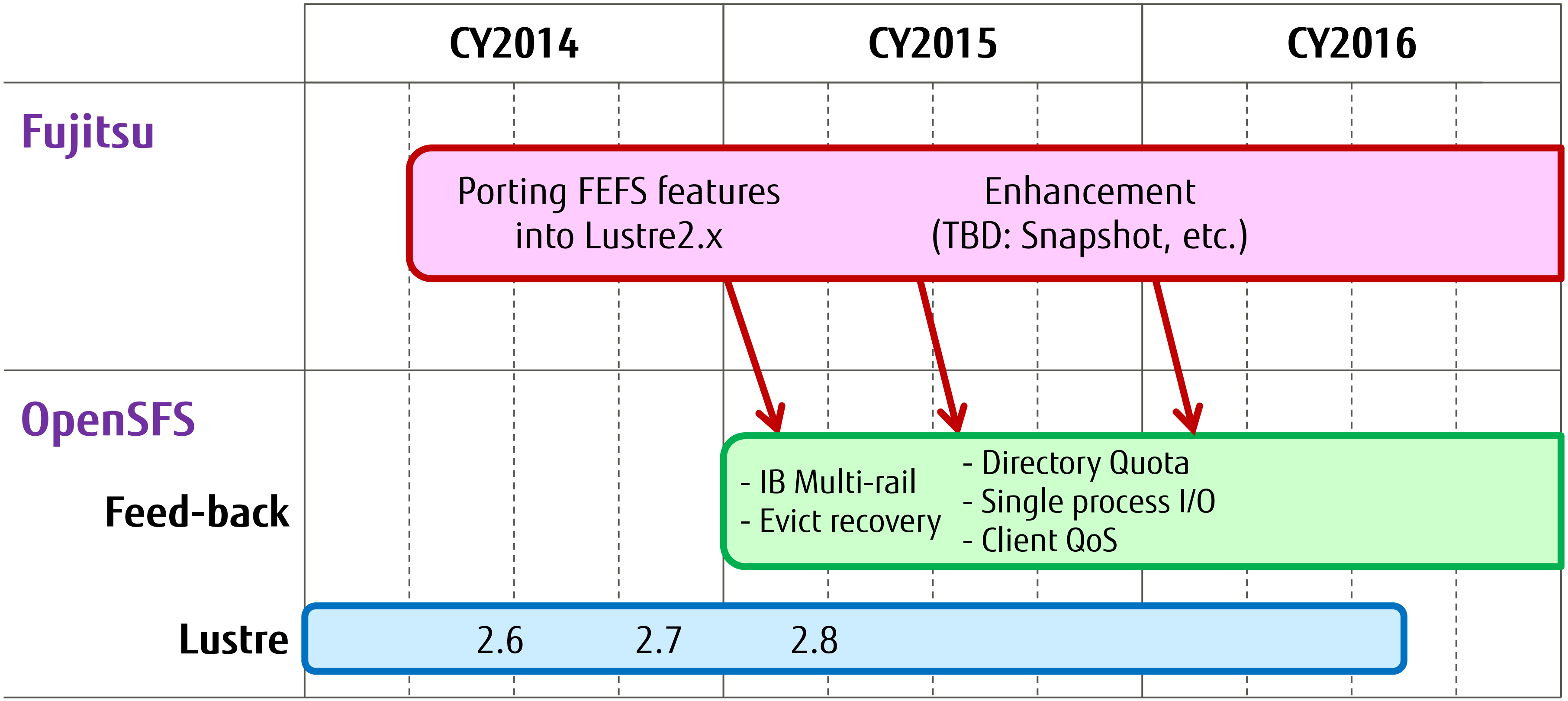
Functions	Submitting Schedule
IB multi-rail	Jan. 2015
Automated Evict Recovery	Apr. 2015
Directory Quota	2nd half of 2015
Improving Single Process I/O Performance	2nd half of 2015
Client QoS	2nd half of 2015
Server QoS	TBD
Memory Usage Management	TBD

→ Details are described in later slides

Roadmap

Fujitsu’s development and community feedback plan

- Schedule may change by Fujitsu’s development/marketing strategy



CY2014

CY2015

CY2016

Fujitsu

Porting FEFS features into Lustre2.x

Enhancement (TBD: Snapshot, etc.)

OpenSFS

Feed-back

- IB Multi-rail
- Evict recovery
- Directory Quota
- Single process I/O
- Client QoS

Lustre

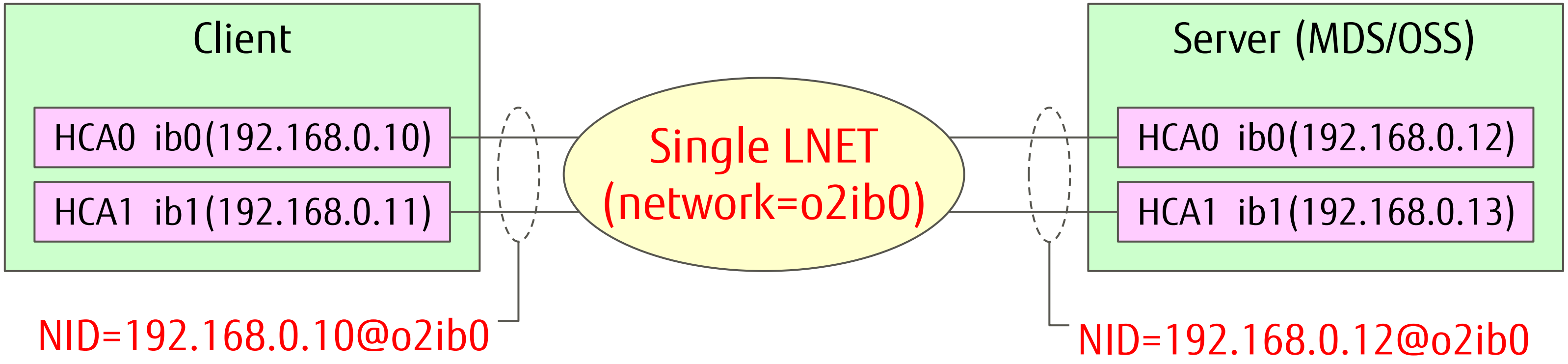
Introduction of Contribution Features

- IB Multi-Rail
- Automated evict recovery
- Directory Quota
- Improving single process I/O performance
- Client QoS

IB Multi-Rail

- Improves LNET throughput and redundancy using multiple InfiniBand(IB) interfaces

- Improving LNET throughput
  - Using multiple IB interfaces as single Lustre NID
  - LNET B/W improves in proportion to the number of IBs on single Lustre node
- Improving Redundancy
  - LNET can continue communicating unless all IBs fail
  - MDS/OSS failover is not necessary when a single point IB failure occurrs



IB Multi-Rail: Related Work (OFED level)

- **IPoIB bonding**
  - OFED has this function already

    → RDMA isn’t supported
- **RDMA bonding**
  - Ongoing work by Mellanox
  - OFED will support RDMA bonding (I’m not sure when…)

    → Our IB multi-rail function might be unnecessary in the future
- **IB partition method**
  - Mr.Ihara (DDN) presented at LUG 2013
  - Multiple bond interfaces are enabled with IPoIB child interfaces

    → Requiring multiple LNET, configurations are complex

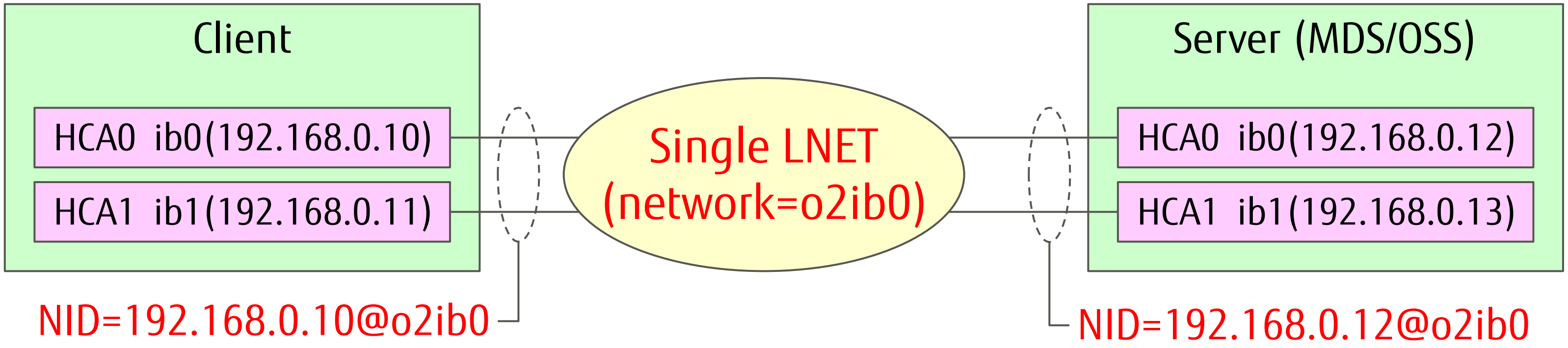
- **At the moment, our approach seems to be better**

IB Multi-Rail: Implementation

- Implemented in LND (ko2iblnd)
  - Other Lustre modules are not changed
  - Keep compatibility with old version of Lustre
- Multiple IB HCAs are handled as single NID
  - Enable constructing single LNET network
- All IBs are active
  - ko2iblnd selects transmission path by round-robin order
  - Multiple LNET requests are transmitted by using all IB paths in parallel

IB Multi-Rail: How to Use

Combining single NID width multiple IB interfaces



LNET setting (modprobe.conf)

```
options lnet networks=o2ib0(ib0,ib1)
```

NID/IPoIB definition

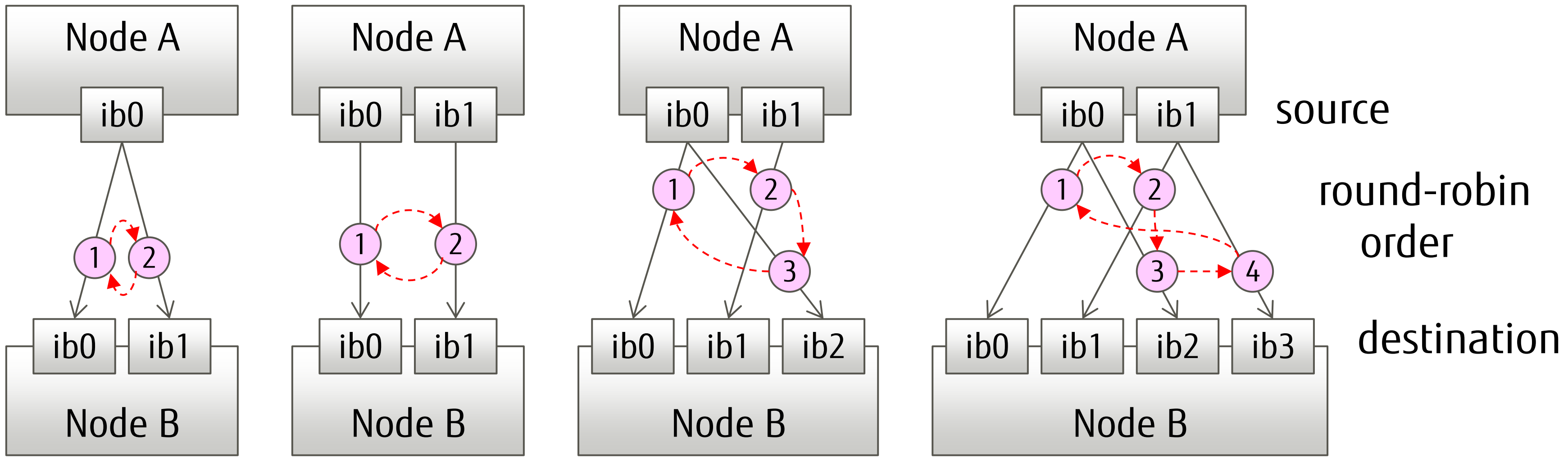
```
# lctl --net  o2ib0  add_o2ibs  192.168.0.10@o2ib0  192.168.0.10 192.168.0.11  → Client
# lctl --net  o2ib0  add_o2ibs  192.168.0.12@o2ib0  192.168.0.12 192.168.0.13  → Server
```

Display multi-rail information

```
# lctl --net o2ib0 show_o2ibs
192.168.0.10@o2ib0 192.168.0.10 192.168.0.11
192.168.0.12@o2ib0 192.168.0.12 192.168.0.13
```

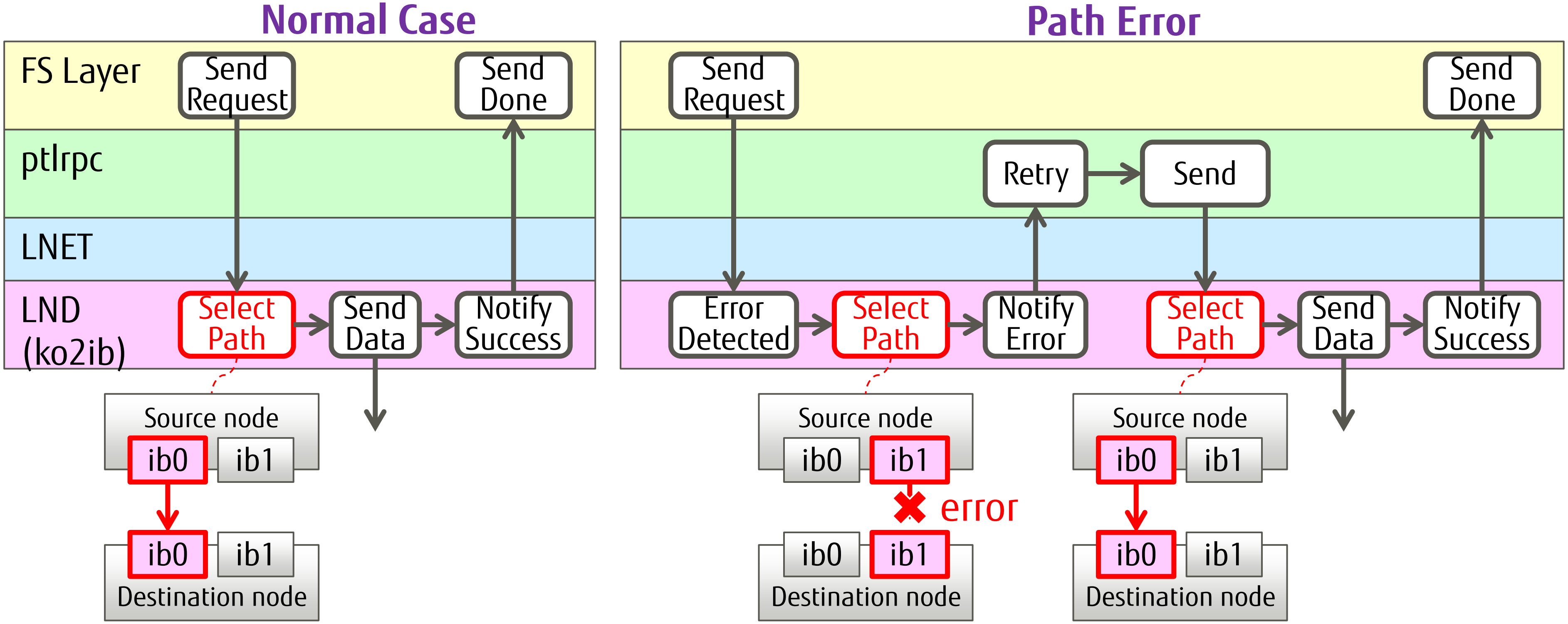
IB Multi-Rail: Path Selection

- **Transmission path is selected in round-robin order**
  - Source and destination interfaces are selected cyclically when each LNET function (LNetPut/LNetGet) is executed



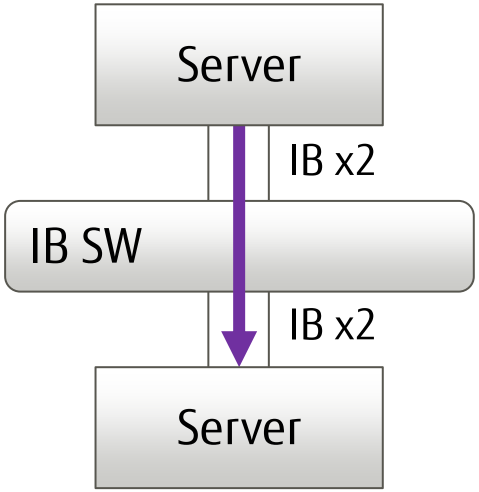
IB Multi-Rail: Error Handling

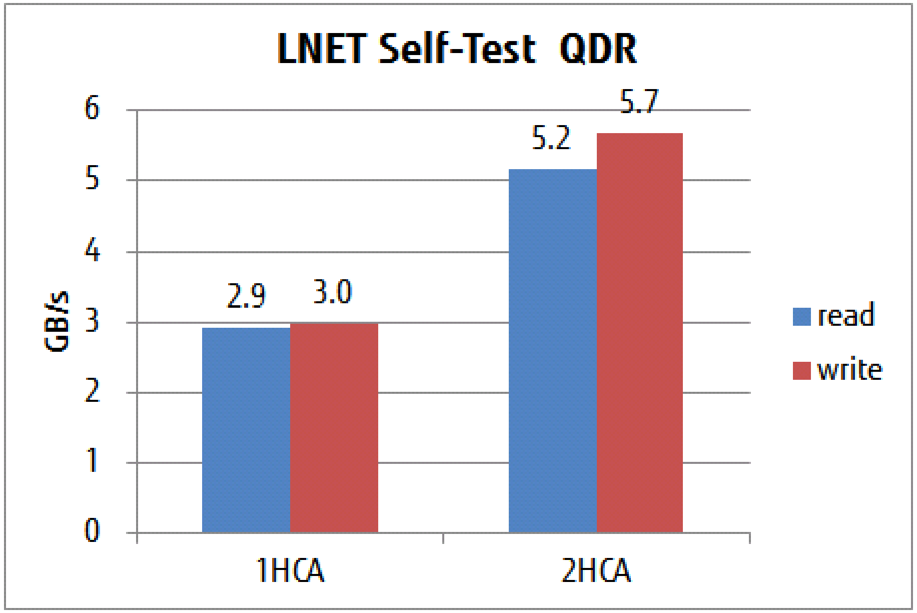
- **Path error**
  - Ptlrpc resends the request that got an error
    → ko2iblnd selects next transmission path in round-robin order and sends it
- **Port down**
  - ko2iblnd removes the transmission path that uses the failed port
    → No error occurs when sending the request

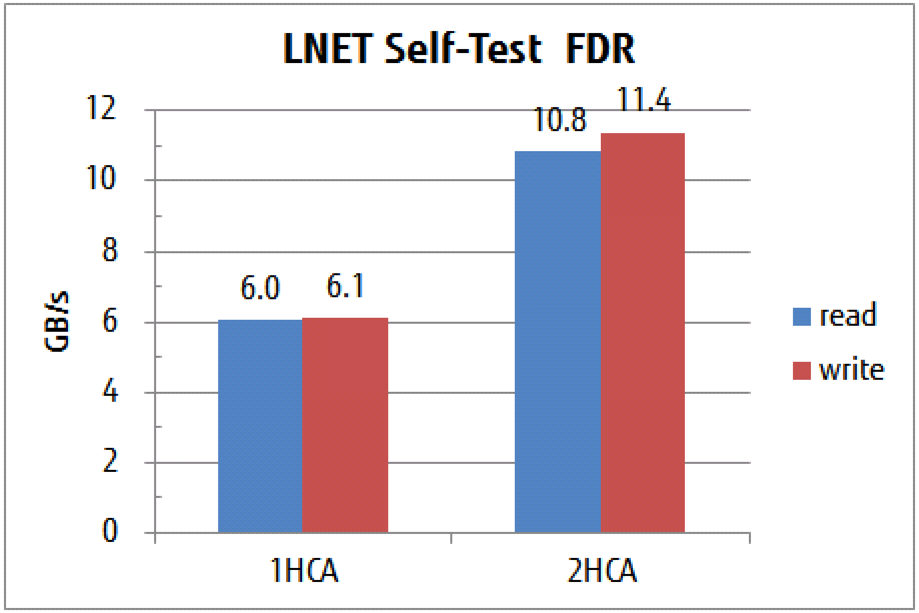


IB Multi-Rail: LNET Throughput

- Server
  - CPU: Xeon E5520 2.27GHz x2
  - IB: QDR x2 or FDR x2
- Result
  - B/W almost scales by #IBs
  - Achieves nearly HW performance





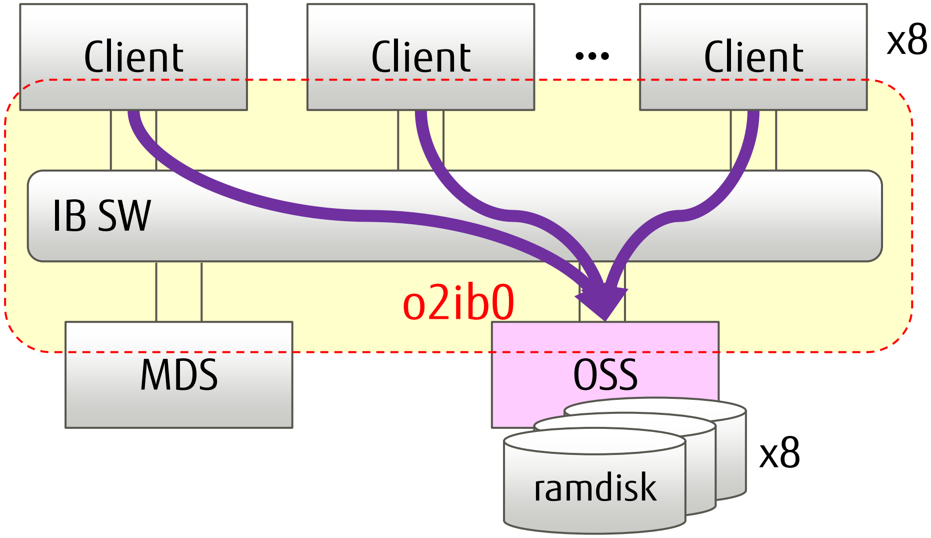


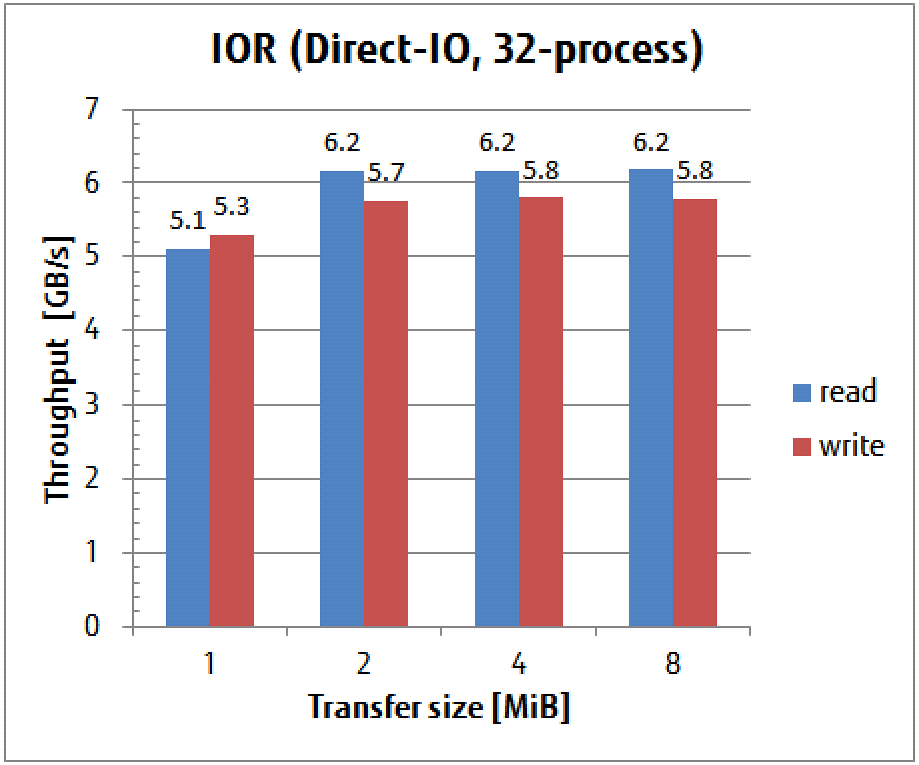
(Concurrency=32)

IB Multi-Rail: I/O Throughput of Single OSS

- OSS/Client
  - CPU: Xeon E5520 2.27GHz x2
  - IB: QDR x2
- OST
  - ramdisk x8 (> 6GB/s)
- IOR
  - 32-process (8client x4)

- Result
  - Throughput almost scales by #IBs
  - Measurement of FDR is planned





Directory Quota (DQ for short)

- Manages maximum files and disk usages for each directory
  - All files/subdirectories under DQ-enabled directory are under control
  - Can not be set to subdirectories under DQ-enabled directory
    - Because of simplicity of implementation and performance

- Implemented on top of the Lustre's Quota framework
  - UID/GID Quota can be used along with DQ
  - Keep compatibility with current Lustre
    - mkfs isn't needed to upgrade PKG
    - Old version of clients can access DQ-enabled directory
      - DQ is not effective to the old version of clients

Directory Quota: How to Use

- Operations are same as Lustre's UID/GID Quota
  - Only "quotacheck" operation differs

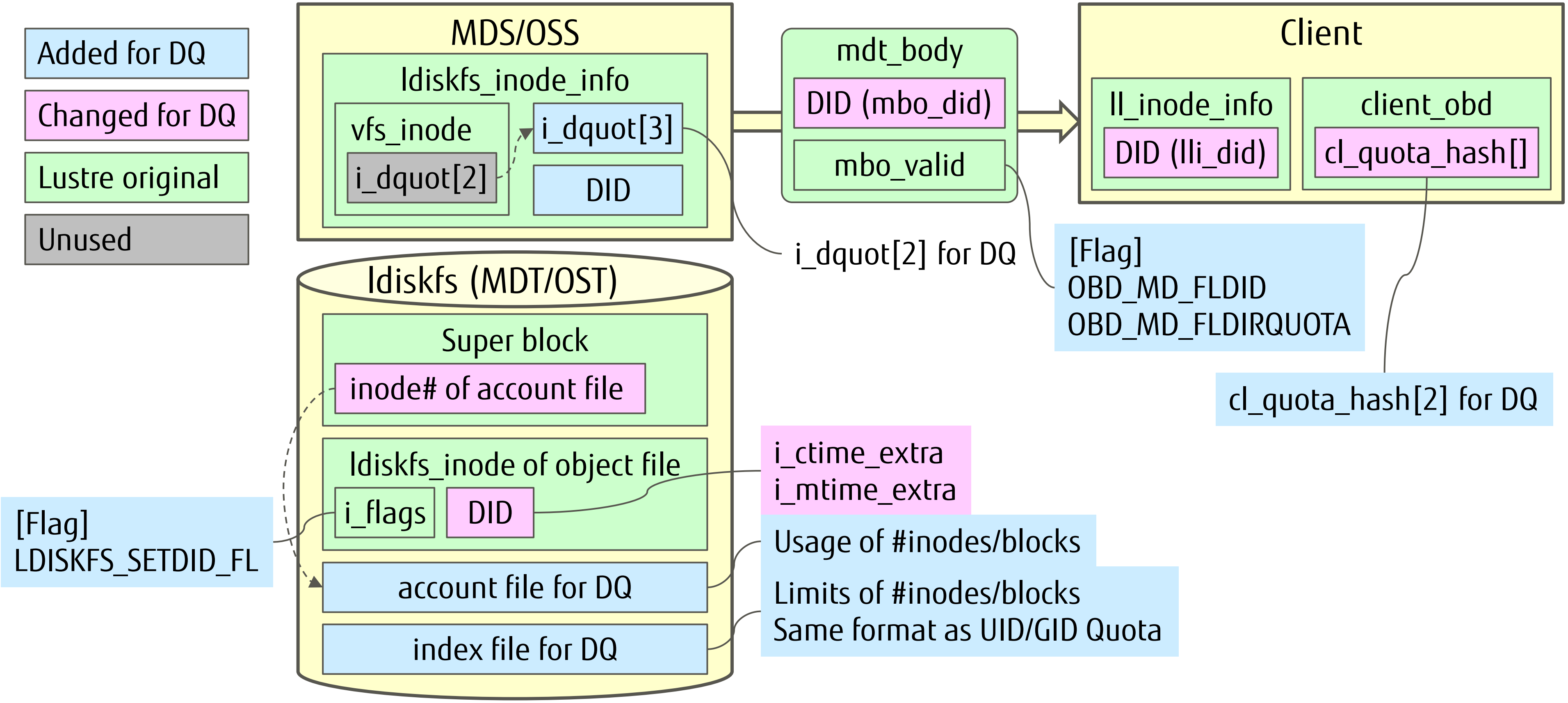
- Set DQ on target directory (=DQ-directory)
  - # lfs **quotacheck –d <target dir>**
    - Counts the number of inodes&blocks of existing files under DQ-directory
- Set limits of inodes and blocks
  - # lfs setquota **–d <target dir>** -B <#blk> -I <#inode> <mountpoint>
- Enable limiting by DQ
  - # lctl conf_param <fsname>.quota.<ost|mdt>=<ug**d**>
  - # lctl set_param -P <fsname>.quota.<ost|mdt>= <ug**d**>
- Check status
  - # lctl get_param osd-*.*.quota_slave.info

Directory Quota: Implementation

- Existing processes of UID/GID Quota are used as far as possible
  - Add some data structures that stores DQ information
  - Keep compatibility with ldiskfs disk layout
- Introduce new ID for DQ (=DID)
  - DID = inode number of DQ-enable directory
  - DID is stored in ldiskfs inode of MDT/OST object files
- Index/account files for DQ are added
  - Usages/limits of the number of inodes/blocks are managed
    - index file: created at first mount
    - account file: created at mkfs
      - Upgrading from no DQ PKG, execute “tunefs.lustre --dirquota”
- ZFS is not supported
  - We don’t have plan to implement DQ in ZFS

Directory Quota: DQ Information

- DID is stored in unused area of ldiskfs inode
  - i_ctime_extra and i_mtime_extra are used
- DQ's index/account files are created on MDTs/OSTs
- Some flags to identify DQ are added

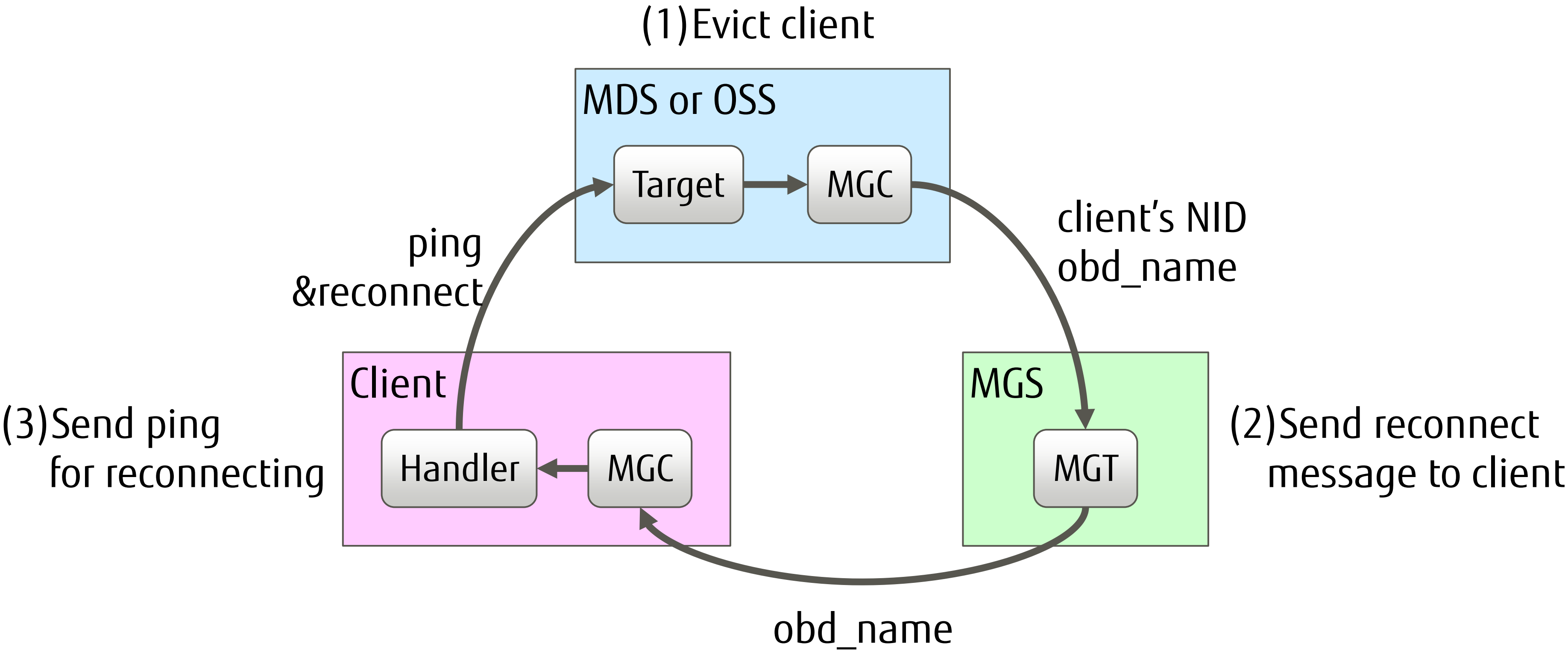


Evict Recovery

- Recovers from evicted-state automatically while disabling periodical pinging (in Lustre 2.4 or later)
- Issue
  - While disabling periodical pinging, clients cannot notice it's eviction
  - First I/O requset from the client to the server gets an error (EIO)
- Approach
  - Reconnect automatically when an eviction occurred
  - Server make evicted client send ping request to the server
- Effect
  - Evicted period is shorten →Frequency of I/O error is minimized

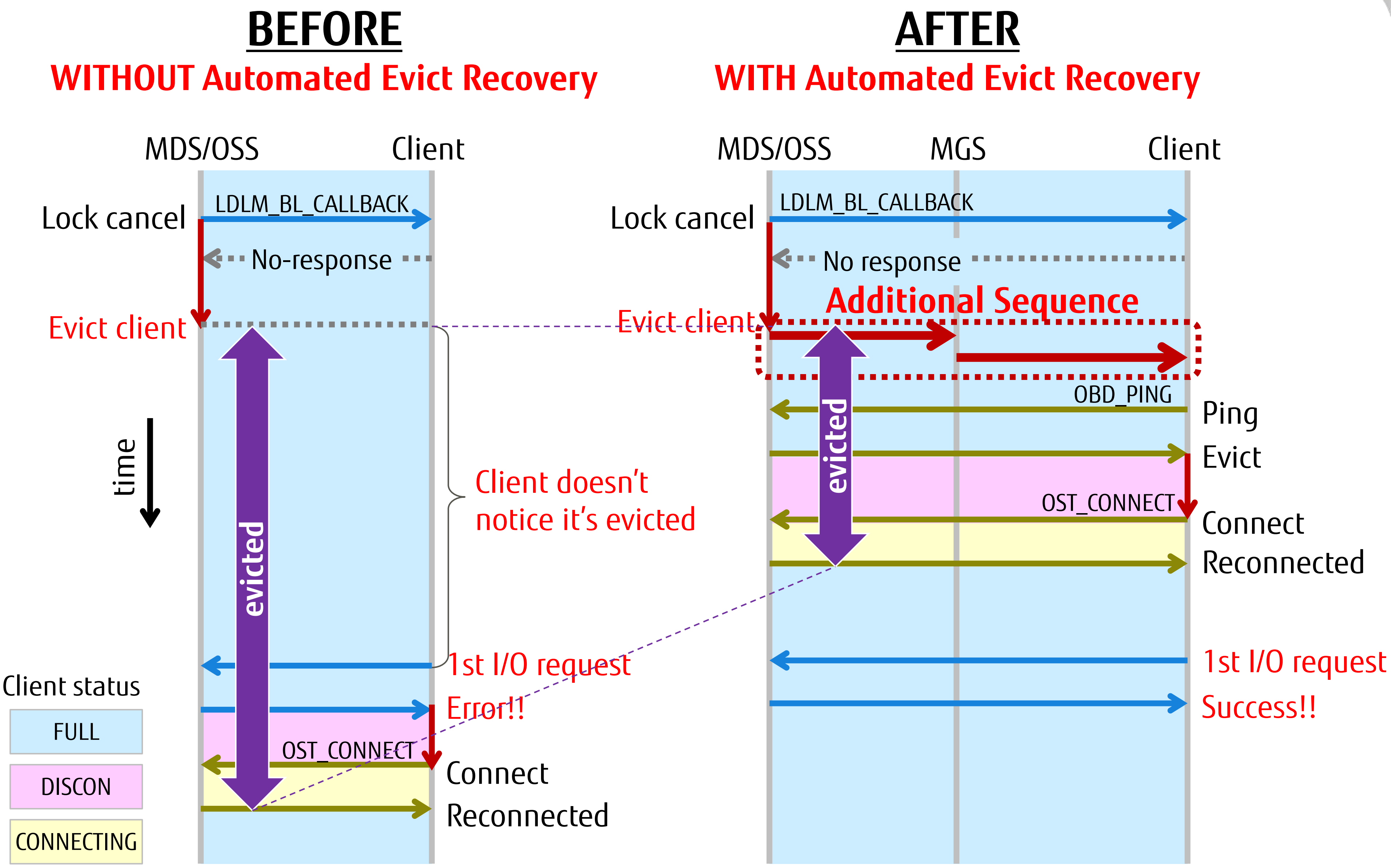
Evict Recovery: Basic Mechanism

- Evict recovery process:
  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



Evict Recovery: Sequences (W/O periodic ping)

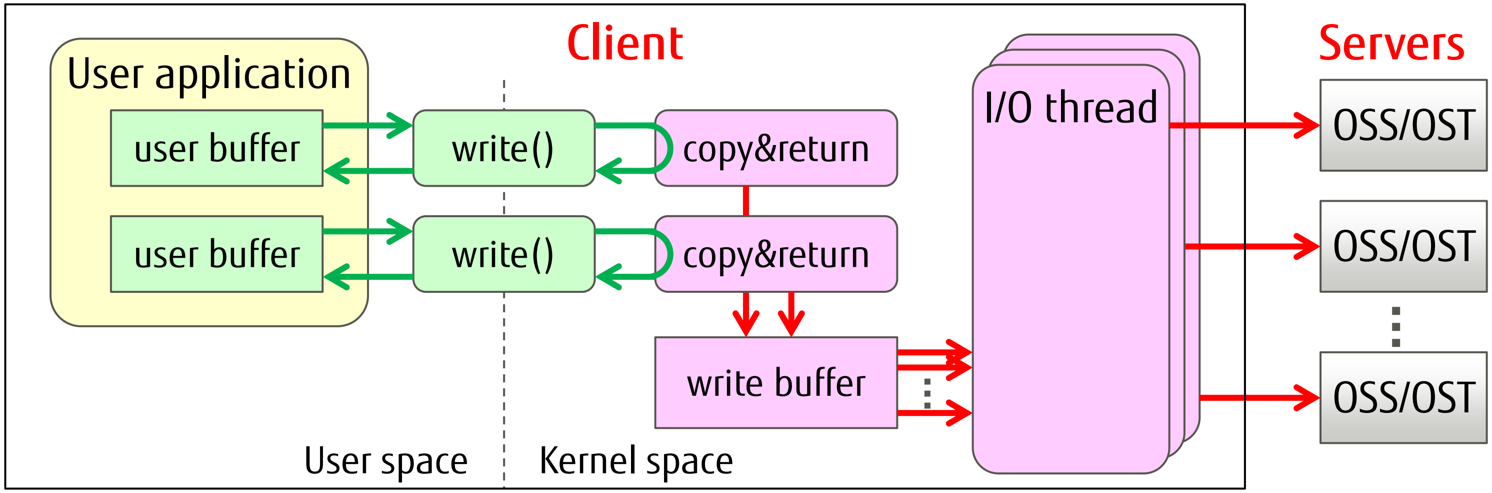
FUJITSU



Improving Single Process I/O Performance

- Important for clients to write a large amount of data such as checkpoint files
- 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
  - Dedicated I/O threads transfer data from the buffer to OSS/OSTs in parallel

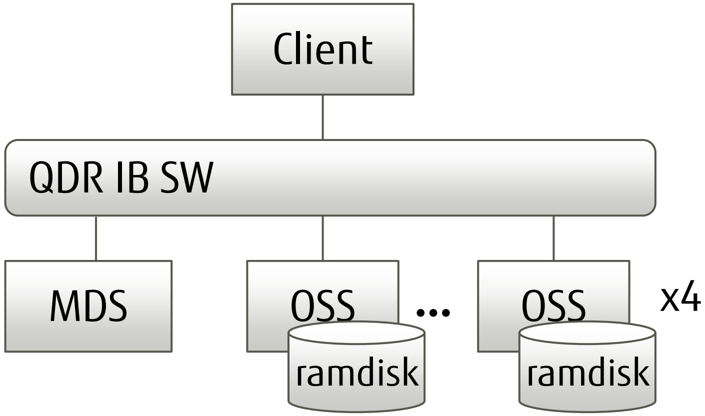
→ write throughput dramatically improves from user perspective



Improving Single Process I/O Performance

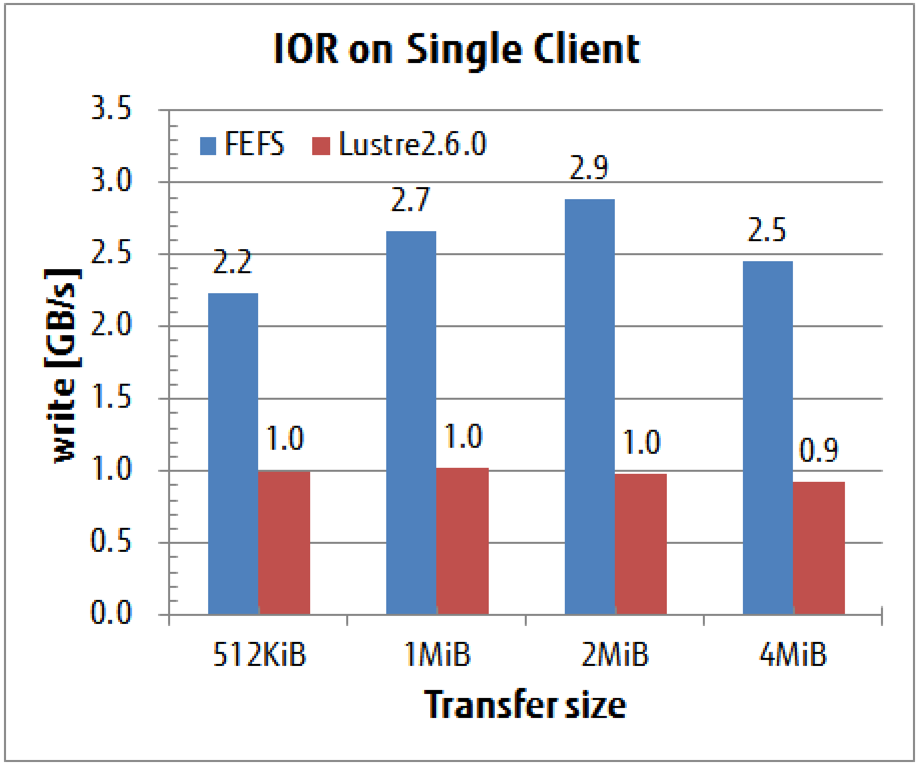
- Lustre 2.6.0 vs. prototype (Lustre 1.8 base)
  - We're re-designing implementation suitable for Lustre 2.x

- OSS/Client
  - CPU: Xeon E5520 2.27GHz x2
  - IB: QDR x1
- OST
  - ramdisk x4
- IOR
  - 1-process



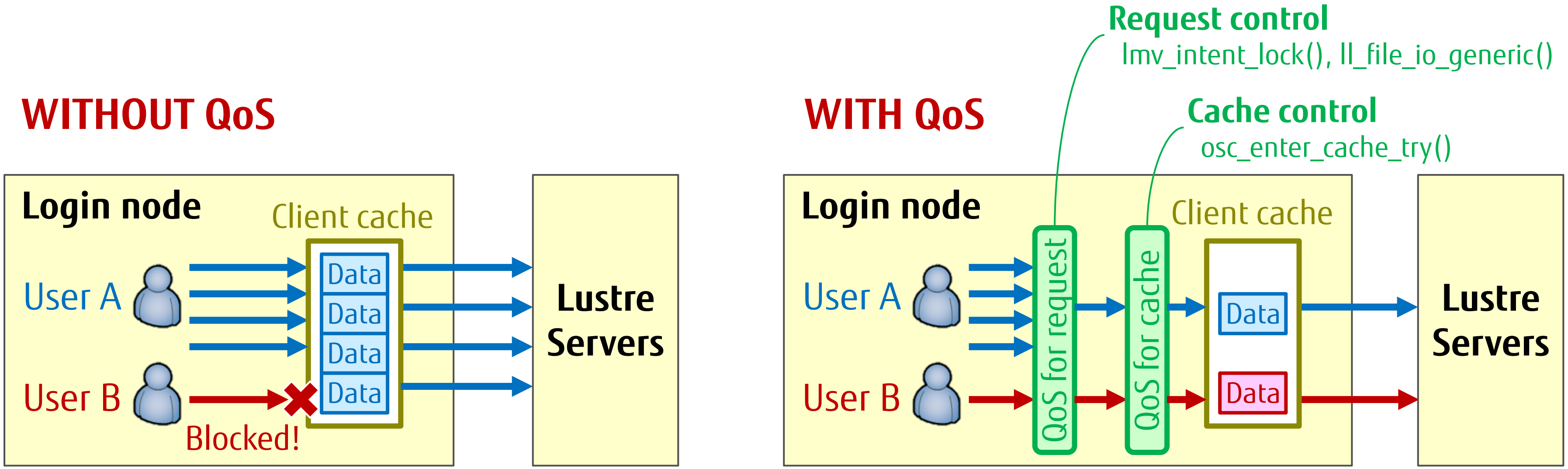
- Result

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



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
  - Request Control: Restricts the max. number of requests issued by each user
    - Prevents a single user occupies requests issued by the client
  - Cache Control: Restricts the max. amount of client cache used by each user
    - Prevents a single user occupies client cache and write from other users are blocked



Client QoS: How to Use

- Parameters for client QoS are specified by mount option

- Parameters for request control
  - qos
    - Enables request control
  - {m|r|w}usermax=n (1~16)
    - Maximum number of meta/read/write requests that each user can issue at the same time

- Parameter for cache control
  - qos_cache
    - Enables cache control
  - dpusermax=n (1~100%)
    - Maximum amount of client cache(*) each user can use in the client

      *per OSC (max_dirty_mb) and per client (obd_max_dirty_pages)

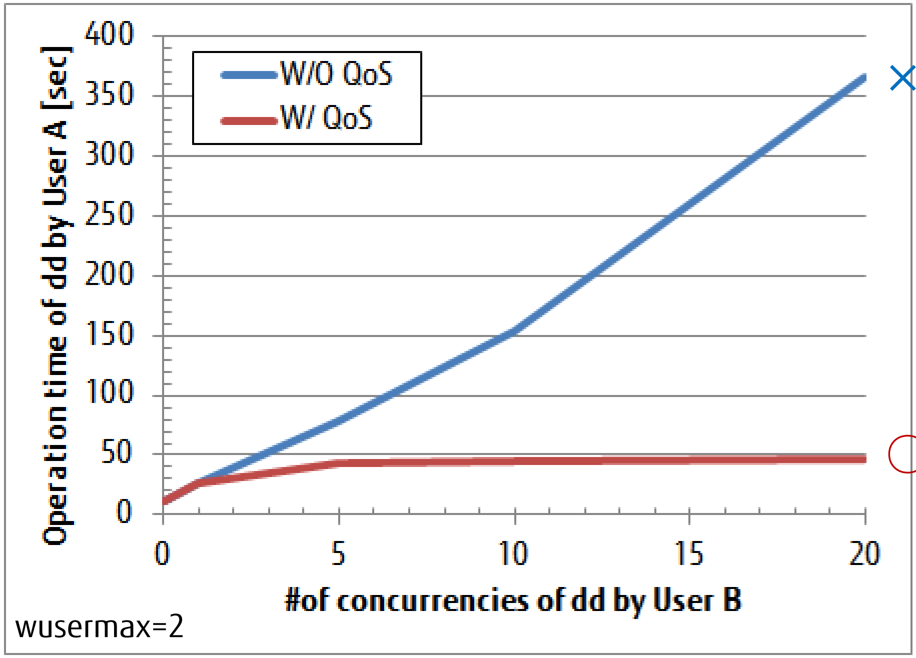
Client QoS: Example of Effectiveness

Test pattern

- dd if=/dev/zero of=/mnt/fefs/out.dat bs=1048576 count=2000 (write 2GB)
- 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

Challenges Toward Exascale File System

- I/O Throughput and Capacity
- Metadata Performance
- System Limits
- Memory Usage
- System Noise

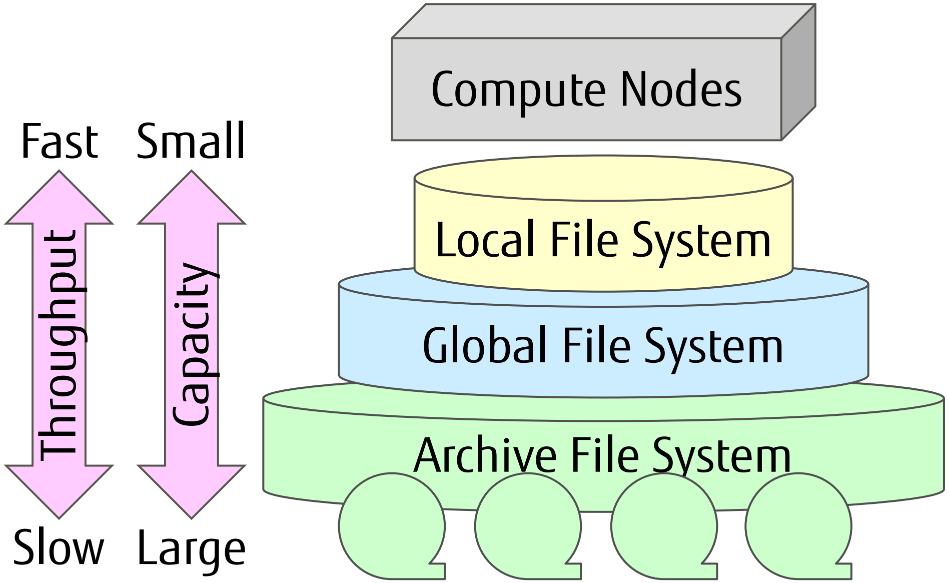
Exascale Concerns: I/O Throughput&Capacity

Concern

- Requires high throughput (~10TB/s) and huge capacity (~1EB)
  - Single layered storage system won't be able to satisfy both requirements
  - Device cost, power consumption, footprint

Approach

- Hierarchical storage system architecture
- Use appropriate storage devices in each hierarchy



For example:

1st layer: SSD, fast buffer for job

2nd layer: HDD, shared area (Lustre)

3rd layer: Tape, archive area (Lustre-HSM)

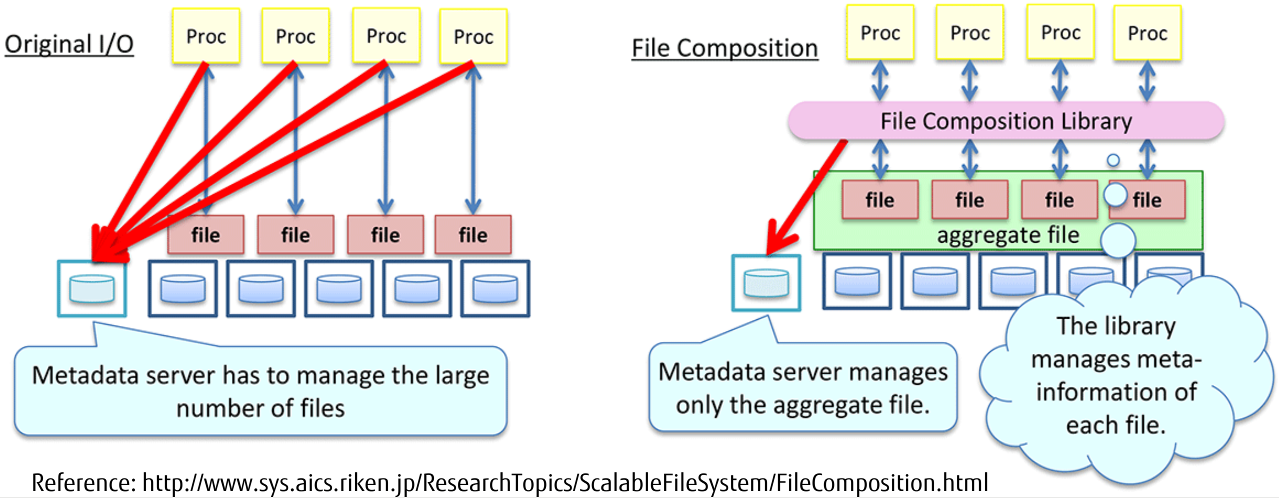
Exascale Concerns: Metadata Performance

Concern

- Metadata performance will hit the limit
  - Exascale applications create several billions of files in a single job
  - E.g. One of exascale application "NICAM" creates 1.8 billion files per job

Approach

- Reduce metadata access to MDS
  - Provide intermediate layer to absorb metadata access between compute node and file system
  - E.g. "File composition library" by RIKEN AICS manages many files as a single file



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

Exascale Concerns: System Limits

Concern

- Capacity of file system must be exabytes class
  - E.g. One of exascale application "COCO" outputs 860PB per job
  - We've extended upper limits of Lustre to satisfy requirements of K computer

Approach

- Eliminating the restriction of logical upper limits
  - E.g. Eliminating 32-bit restriction, etc...

System Limits	FEFS*	Lustre 2.x	Exa
Maximum file system size	8EB	512PB	> 8EB
Maximum file size	8EB	31.25PB	> 8EB
Maximum number of files	8E	4G x#MDTs	
Maximum OST size	1PB	128TB	> 1PB
Maximum stripe counts	20,000	2,000	> 8k
Maximum number of OSTs	20,000	8,150	> 8k
Maximum number of MDTs	1	4,096	

Exascale Concerns: Memory Usage

- Concern
  - Secure sufficient memory to application programs
    - Compute node of K computer ran out of memory only by mounting file system
    - We reduced memory usage drastically for K computer (2.5GB → 490MB in client) (reported at Lustre Developer Summit 2012)
- Approach
  - Controlling memory usage strictly
    - E.e. page cache
  - Break away from scale dependency
    - E.g. number of OSTs

Exascale Concerns: System Noise (OS Jitter)

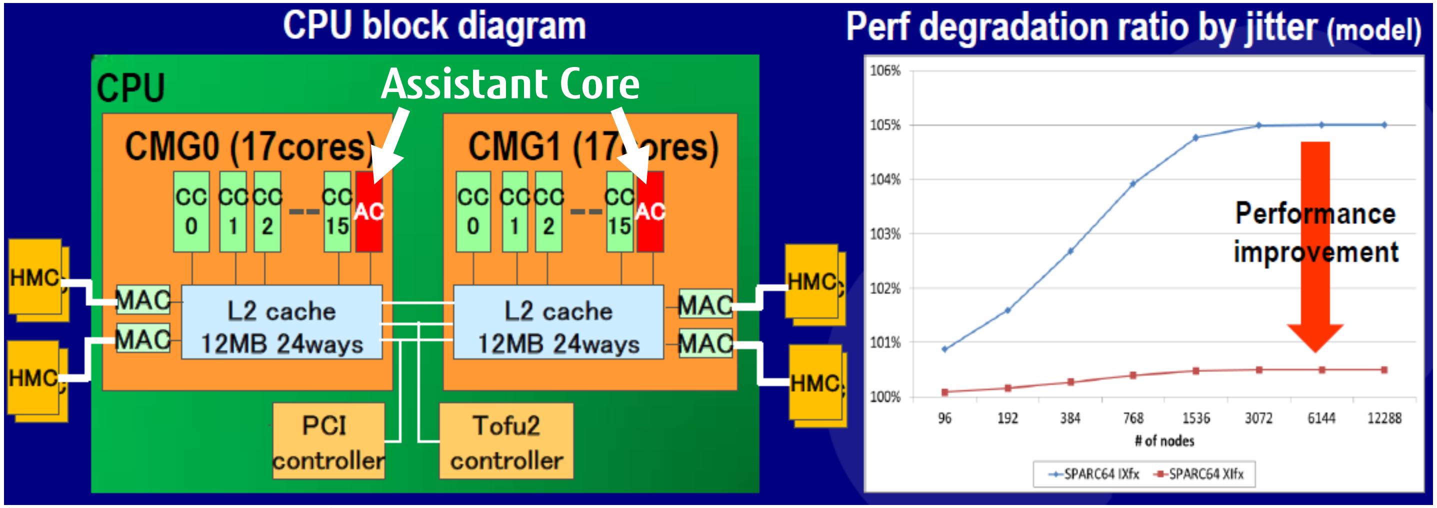
Concern

- Eliminating OS jitter to maximize performance of massively parallel applications
  - We took great effort to reduce system noise in K computer
    - → Shortening execution time of Lustre daemons; ll_ping, ldlm_poold

(Reported at Lustre Summit 2014)

Approach

- Introducing dedicated cores for system daemons (OS timer, file I/O, MPI, etc)
  - E.g. Fujitsu's SPARC64 XIfx CPU for Post-FX10 provides with 2-assistant cores
    - Processing cost of daemons to be reduce?



(Reference: Hot Chips 26)

Summary

- Fujitsu will continue to improve Lustre for exascale systems
  - Take advantage of experience and technology obtained from development of K computer and consumer supercomputers
- Fujitsu will open its development plan and feed back it's enhancements to Lustre community
  - Luster Developer Summit is one of the most suitable place to discuss technical matter
- Several features will be scheduled to be contributed in 2015
  - InfiniBand Multi-rail, Directory Quota, etc.

FUJITSU

shaping tomorrow with you