Lustre Developer Day 2016

FUJITSU

shaping tomorrow with you

Upstreaming of Features from FEFS

Shinji Sumimoto

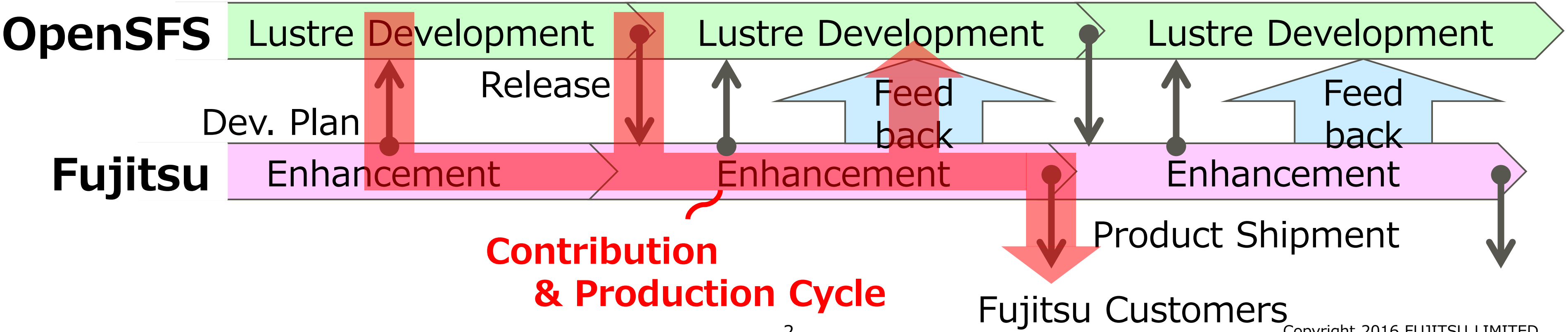
Fujitsu Ltd. A member of OpenSFS

Outline of This Talk

- Fujitsu’s Contribution
  - Fujitsu’ Lustre Contribution Policy
  - FEFS Current Development Status
  - Fujitsu Contributions until 2014
- Some Upstreaming Function Topics from Fujitsu
- Toward Exascale Computing (if we have a time)

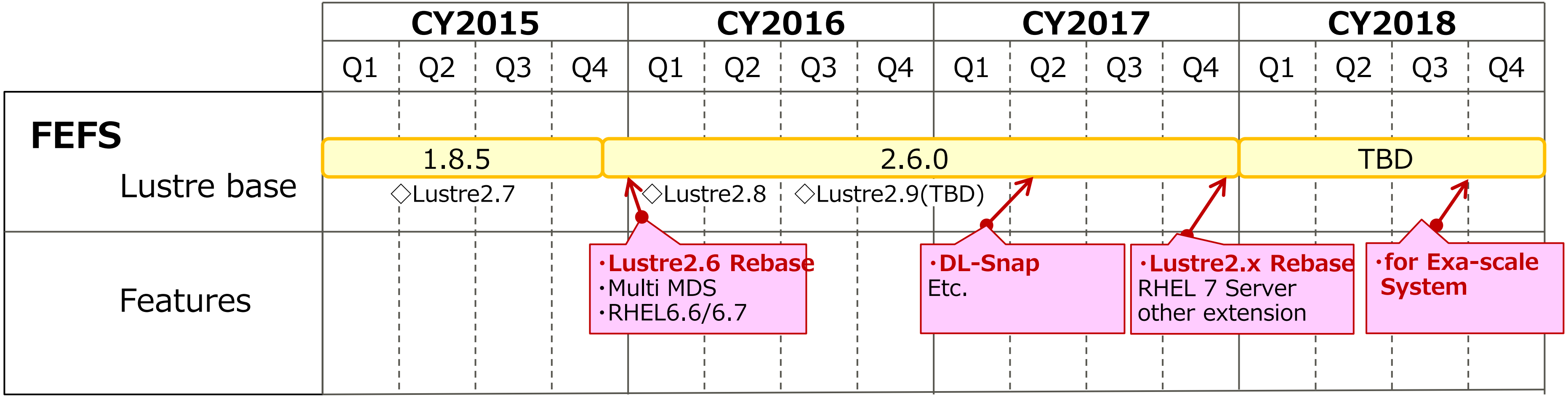
Fujitsu' Lustre Contribution Policy (Presented at LAD 14 in Reims)

- Fujitsu will open its development plan and feed back it's enhancement to Lustre community
- Fujitsu's basic contribution policy:
  - Opening development plan and Contributing Production Level Code
  - Feeding back its enhancement to Lustre community no later than after a certain period when our product is shipped.



FEFS Current Development Status

- Lustre 2.6 based FEFS was shipped as a product in 2015.Q4.
  - It took long time to pass our qualification test for a year.



·Lustre2.6 Rebase
·Multi MDS
·RHEL6.6/6.7

·DL-Snap
Etc.

·Lustre2.x Rebase
RHEL 7 Server
other extension

·for Exa-scale System

Specification Comparison FEFS vs. Lustre

Features		FEFS 1.8 Based	FEFS 2.6 Based	Lustre 2.6
System Limits	Max file system size	8EB	8EB	512PB
	Max file size	8EB	62.5PB	31.24PB
	Max #files	8E	16T	16T
	Max OST size	1PB	2PB	128TB
	Max stripe count	20k	4k	2k
	Max ACL entries	8191	32	32
Node Scalability	Max #OSSs	20k	20k	1020
	Max #OSTs	20k	20k	8150
	Max #Clients	1M	1M	128K
Block Size of *ldiskfs*		~512KB	4KB	4KB

Current Lustre specification limits Lustre 2.6 based FEFS specification

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 Jun 15, rejected
LU-6657	Eviction Notifyer (Automated Eviction Recovery)	Changing method to reconnect
LU-6658	Single stream write performance improvement with worker threads in llite (Single Process IO Performance Improvement)	In Review Jun 15, Fujitsu needs to reconsider the implementation

Fujitsu Contributions in 2015 (2)

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

Jira	Patch	Submission Status
LU-6600	Race lustre_profile_list	Lustre 2.8
LU-6624	LBUG in osc_lru_reclaim	Lustre 2.8
LU-6643	write hang up with small max_cached_mb	In Review May 15, Fujitsu needs to reconsider the implementation
LU-6732	Cannot pick up EDQUOT from ll_write_begin and ll_write_end	In Review Aug 15, One more reviewer needed

Fujitsu Contributions in Future

- Fujitsu will continue submitting new features.

Feature	Submission Schedule
Client QoS	2nd half of 2016
Directory Quota	2017
Snapshot	Mid 2017
Server QoS	TBD
Memory Usage Management	TBD

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

Some Upstreaming Function Topics

- Directory Quota
- IB Channel Bonding
- DL-SNAP: Snapshot
- Client QoS
- Improving Single Process Write Performance

Directory Quota

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

- Implemented on top of the Lustre's Quota framework
  - UID/GID Quota can be used along with DQ
  - Keep compatibility with current Lustre

    Upgrade rpm without mkfs

    Old version of clients can access DQ enabled directory

Directory Quota: How to Use

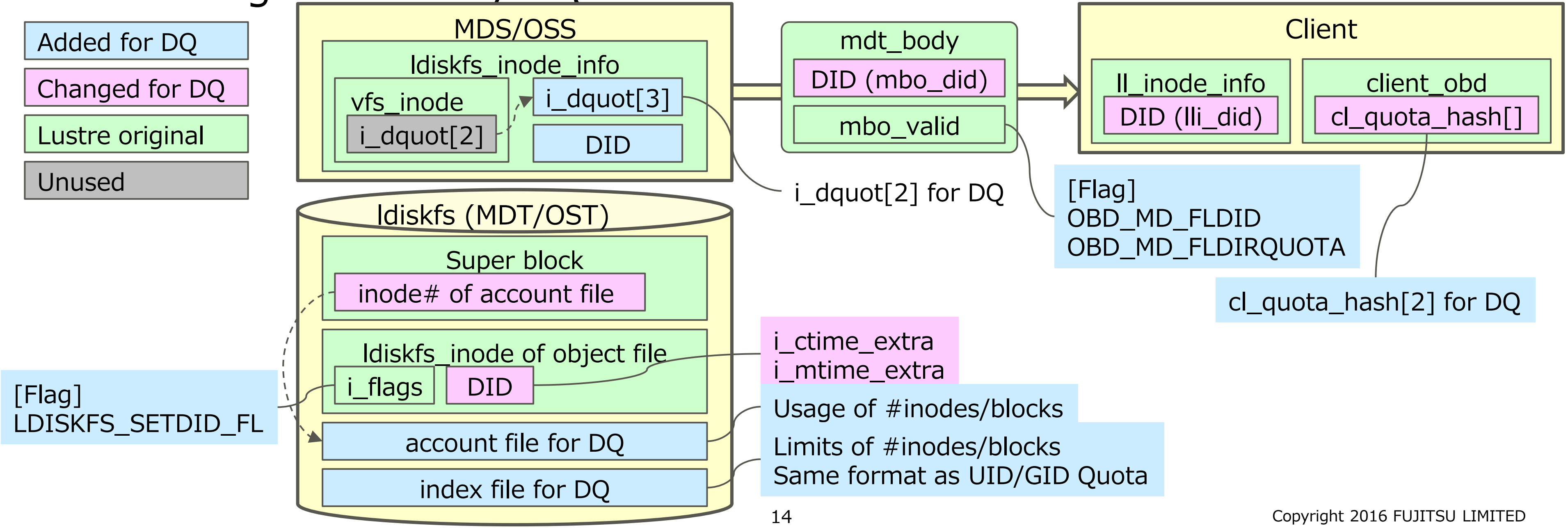
- Operations are same as Lustre’s UID/GID Quota
- Set limits of inodes and blocks
  - # lfs setquota **–d <target dir>** -B <#blk> -I <#inode> <mountpoint>
- Enable limiting by DQ
  - # lctl conf_param <fsname>.quota.<ost|mdt>=<ug**d**>
  - # lctl set_param -P <fsname>.quota.<ost|mdt>= <ug**d**>
- Check status
  - # lctl get_param osd-*.*.quota_slave.info

Directory Quota: Implementation

- Existing processes of UID/GID Quota are used almost as it is
  - Some data structures that stores DQ information are added
  - Disk layout keeps unchanged → mkfs isn’t needed to upgrade PKG
- 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

Directory Quota: Management 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



Added for DQ

Changed for DQ

Lustre original

Unused

MDS/OSS

ldiskfs_inode_info

vfs_inode

i_dquot[2]

i_dquot[3]

DID

i_dquot[2] for DQ

mdt_body

DID (mbo_did)

mbo_valid

[Flag]
OBD_MD_FLDID
OBD_MD_FLDIRQUOTA

Client

ll_inode_info

DID (lli_did)

client_obd

cl_quota_hash[]

cl_quota_hash[2] for DQ

ldiskfs (MDT/OST)

Super block

inode# of account file

ldiskfs_inode of object file

i_flags

DID

[Flag]
LDISKFS_SETDID_FL

i_ctime_extra
i_mtime_extra

account file for DQ

Usage of #inodes/blocks

index file for DQ

Limits of #inodes/blocks
Same format as UID/GID Quota

Current Status of Directory Quota

- Lustre 1.8 based DQ on FEFS has been providing as a product
    - Our customers use DQ function on their system operation

- Lustre 2.6 based DQ on FEFS has started shipping

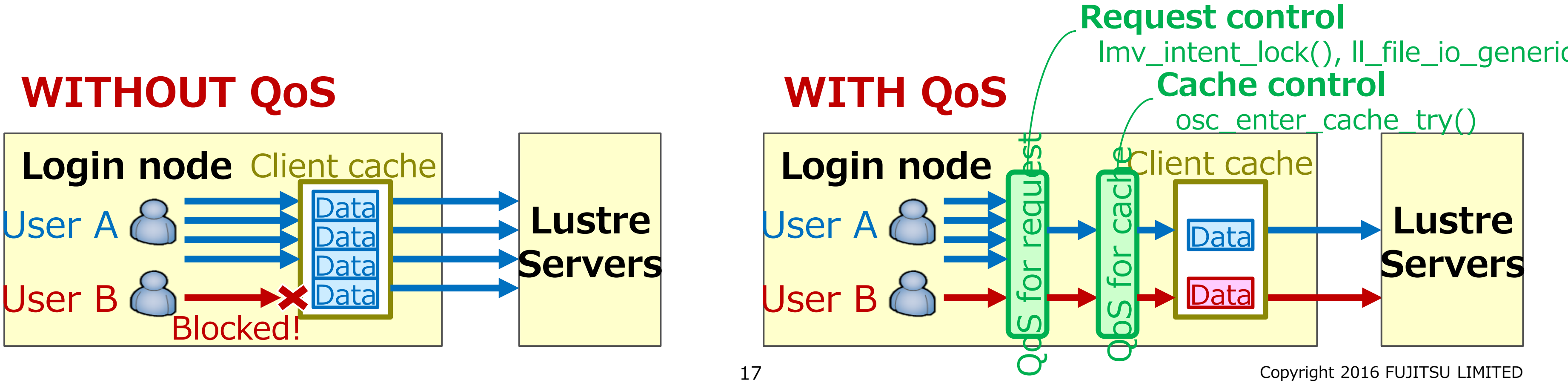
Client QoS

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: 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)

    The number of meta/read/write requests that single user can issue at the same time
- Parameter for cache control
  - qos_cache

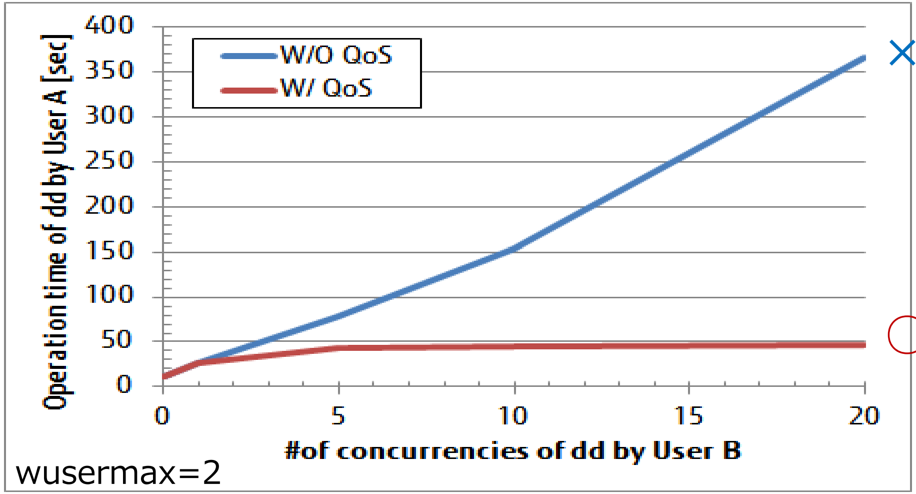
    Enables cache control
  - dpusermax=n (1~100%)

    The amount of client cache(*) that single user can use in the client

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

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



wusermax=2

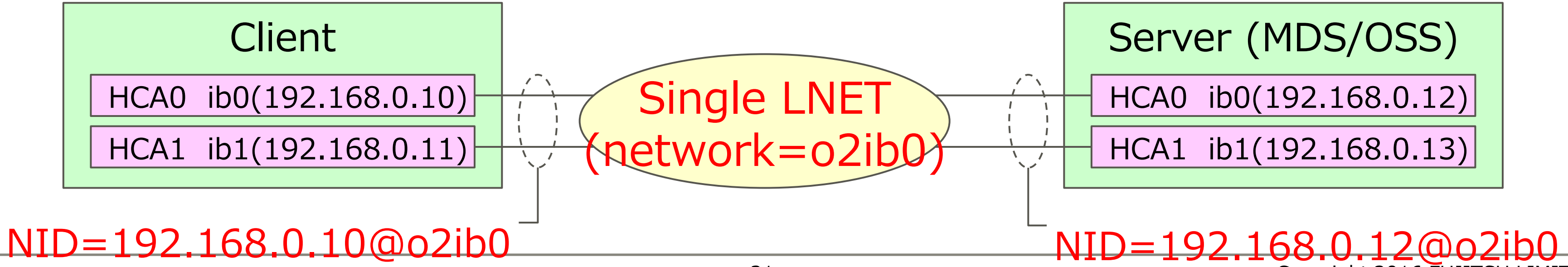
×Execution time becomes very long

○Execution time is almost kept constant

IB Multi-Rail

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, but RDMA isn’t supported
  - RDMA bonding: Ongoing work by Mellanox: OFED will support RDMA bonding (I’m not sure when…).
  - 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
- LNET Level
  - SGI presented LNET level multi-rail at Lustre Developer Summit 2015
    - Only InfiniBand support does not make sense, socket should be supported!
    - RAS feature is not easy to support current LNET level.
- Our approach is better in the point of having a real code to work perfectly.

IB Multi-Rail: Implementation

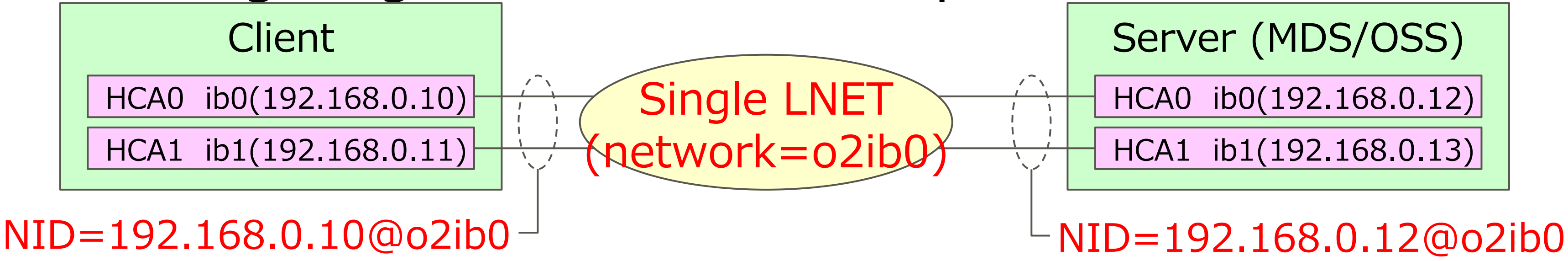
- Implemented in LND (ko2iblnd)
  - Other Lustre modules are not changed
  - Keep compatibility with old version of Lustre (socklnd)

- Multiple IB HCAs are handled as single NID
  - Enable constructing single LNET network

- All IB HCAs 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 (1)

- Combining single NID width multiple IB interfaces



- LNET setting (modprobe.conf)

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

IB Multi-Rail: How to Use (1)

NID/IPoIB definition

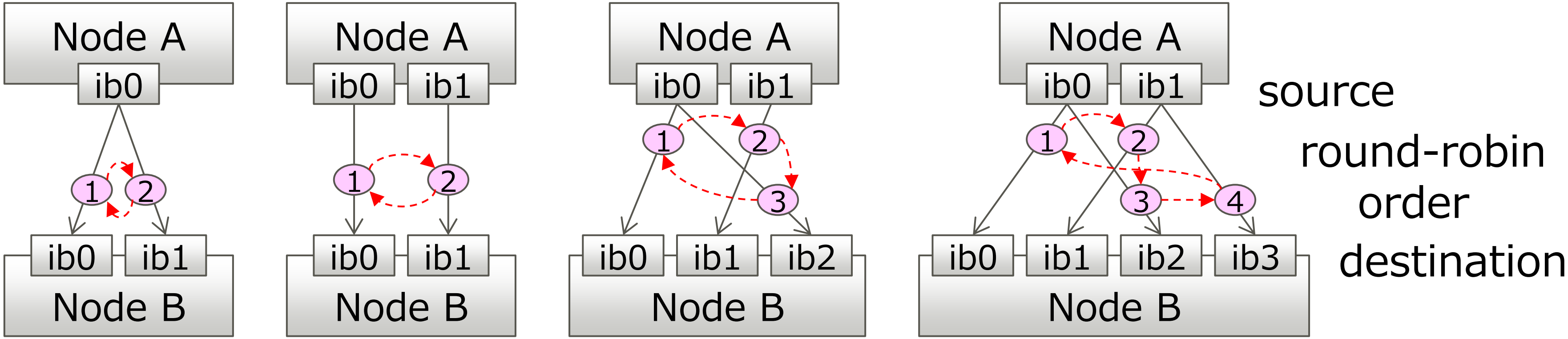
```
# lctl --net  o2ib0  add_o2ibs  192.168.0.10@o2ib0  192.168.0.10 192.168.0.11  → Client
# lctl --net  o2ib0  add_o2ibs  192.168.0.12@o2ib0  192.168.0.12 192.168.0.13  → Server
```

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
```

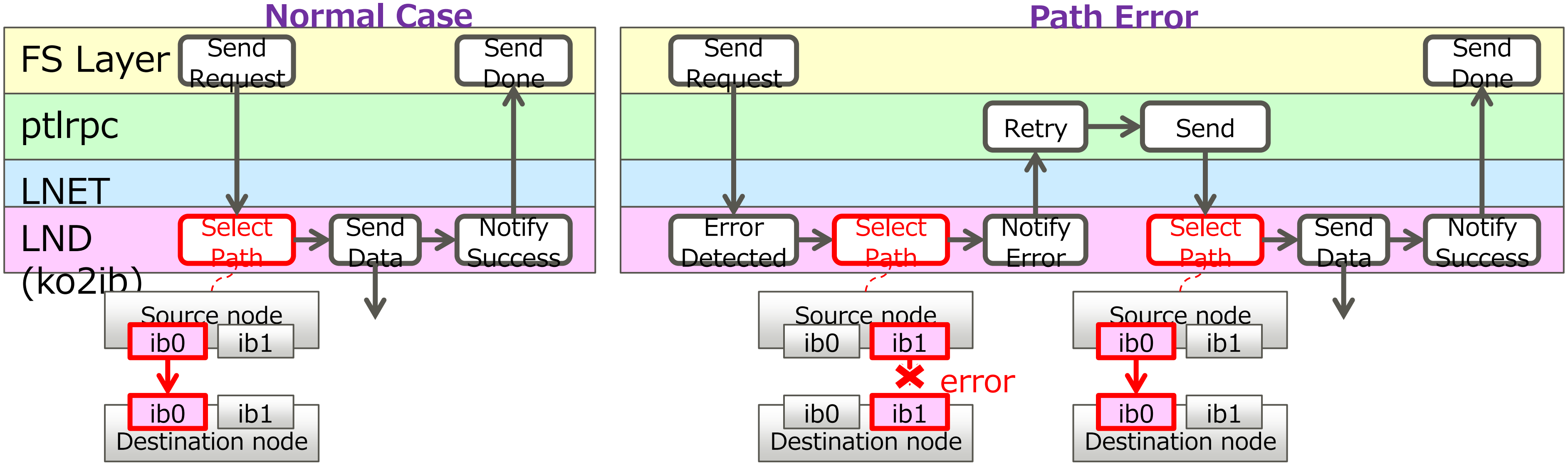
IB Multi-Rail: Path Selection

- 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

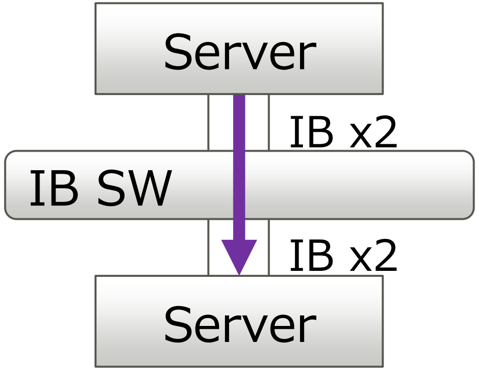
- Path error
  - Ptlrpc resends the request that got an error
    → ko2iblnd selects next transmission path in round-robin order and sends it
- Port down
  - ko2iblnd removes the transmission path that uses the failed port
    → No error occurs when sending the request

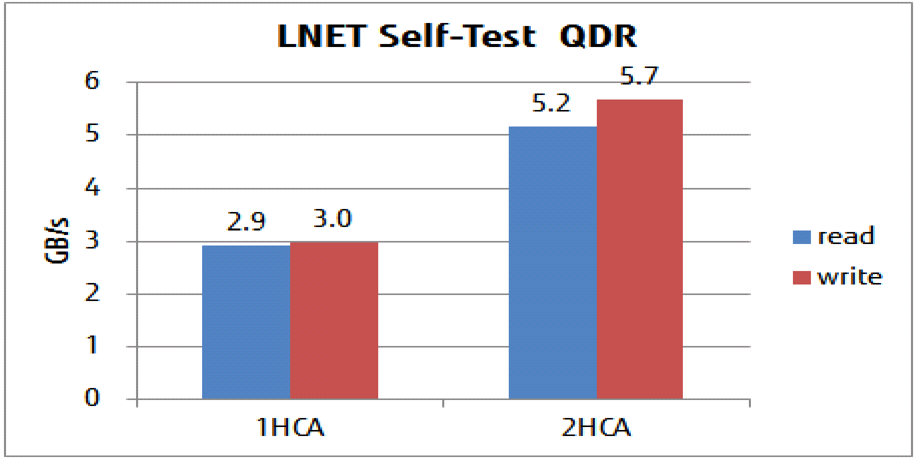


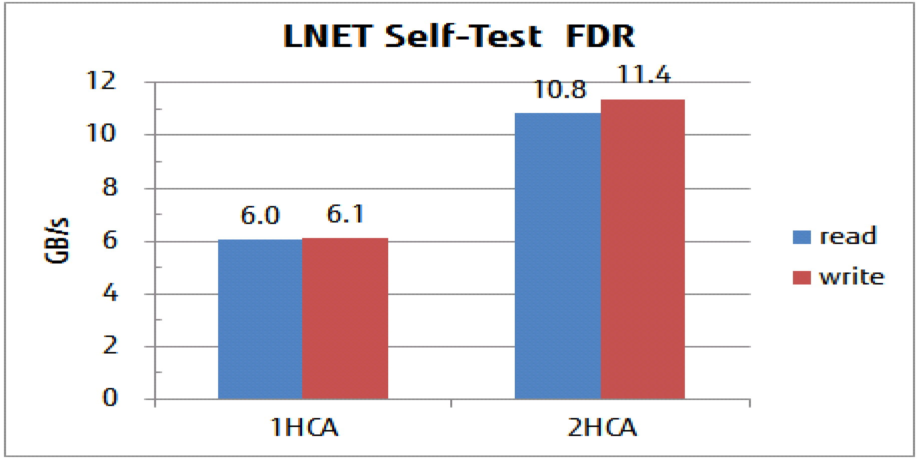
IB Multi-Rail: LNET Throughput

FUJITSU

- Server
  - CPU: Xeon E5520 2.27GHz x2
  - IB: QDR x2 or FDR x2
- Result



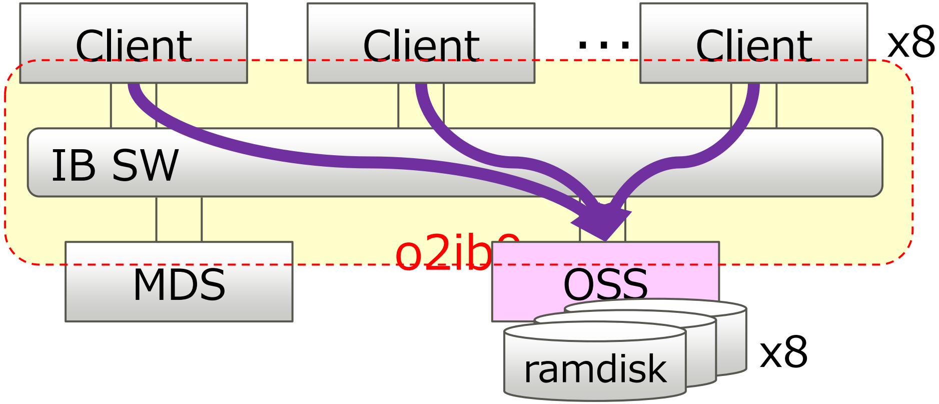




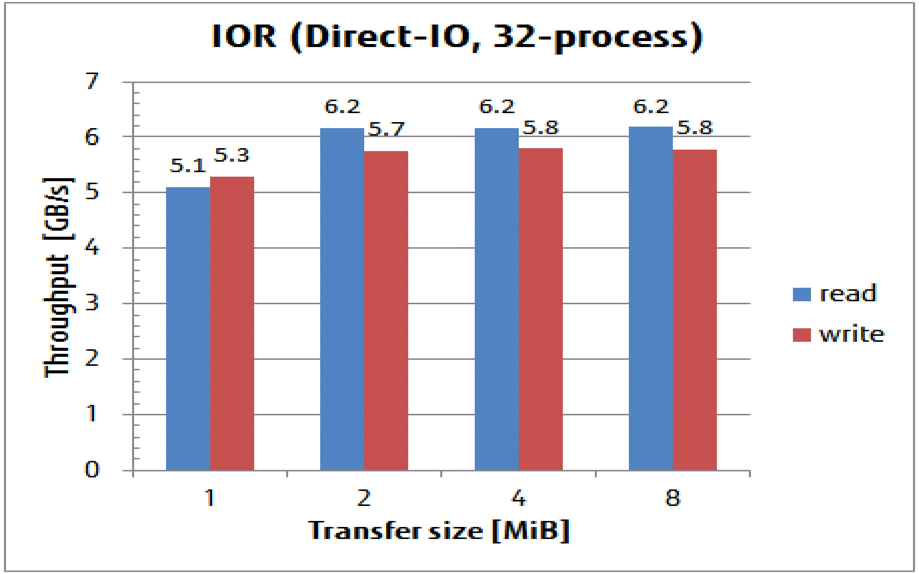
(Concurrency=32)

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



IB Multi-Rail Code Status

- Our IB Multi-Rail is provided as a commercial product for over 5 years
  - K computer: over 90 OSS, 1000 class clients since 2011
  - Realizing Highly available operation for over 5 years
- OFED based implementation can be widely used for other devices
  - RoCE
  - OmniPath
  - Tofu
- We contributed our code to OpenSFS (LU-6531), but rejected with unreasonable reason!
  - We do not mind whether our contribution is accepted or not although our motivation is degraded.

DL-SNAP: Snapshot

Background of DL-SNAP

- It is difficult to make backup on large scale file system.
  - PB class file system backup takes long time and requires its backup space.
- To reduce storage usage and backup time:
  - Using snapshot to reduce duplicate data
- Two level of backup: System level and User level

System Level Backup vs. User Level Backup

- System level backup:
  - System guarantees to backup data and to restore the backup data
  - Therefore, double sized storage space or another backup device is required to guarantee data backup and restore.
  - File Services must be stopped during backup.
- User level backup:
  - User can select backup data
  - File Service does not need to be stopped.

Selected User Level Backup Scheme

- Customer Requirement:
  - Continuing file system service
  - Difficult to guarantees the backup data to restore in system operation
  - Providing some backup service with limited storage space

- Therefore, user level backup scheme is selected.
  - We started to develop DL-SNAP which is user and directory level snapshot

What is DL-SNAP?

- DL-SNAP is designed for user and directory level file backups.
- Users can create a snapshot of a directory using lfs command with snapshot 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.

Quota Support and Utility Commands

- Quota function is also provided to manage storage usage of users
  - a little bit complicate when the owner of the snapshot is different among the original and some snapshot generations.
- Utility Commands: lfs snapshot, lctl snapshot
  - Enabling Snapshot: lctl snapshot on <fsname>
  - Getting Status of Snapshot: lctl snapshot status <fsname>
  - Creating a snapshot: lfs snapshot --create [-s <snapshot>] [-d <directory>]
  - Listing snapshot: lfs snapshot --list [-R] [-d <directory>]
  - Deleting snapshot: lfs snapshot --delete [-f] -s <snapshot> [-d <directory>]

DL-SNAP Implementation

- The implementation of DL-SNAP is copy on write base
  - Implemented on top of current Lustre ldiskfs and limited in OST level
  - Without modification of ext4 disk format
  - Adding special function to create snapshot to MDS.
  - In Lustre point of view, creating snapshot is the same function to create copy.
- OST level modification (more detail on next page):
  - Add a functionality that creates extra-references which points to the existing data blocks on OSTs.
  - Add Copy-on-Write capability to the backend-fs.
- Two Methods to Manage Copy-on-Write Region Blocks
  - Block Bitmap Method
  - Extent Region Method (Selected)

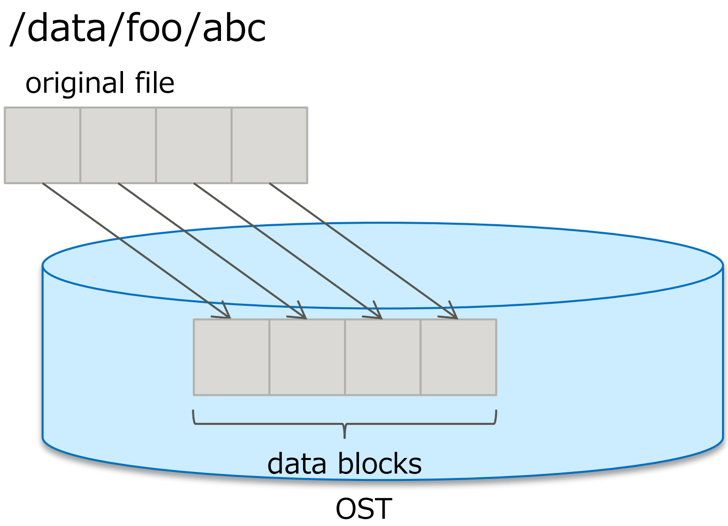
Basic Mechanism of DL-SNAP by Extent Region (1)

Initial state:

- The original file points to the data blocks on OSTs

Taking snapshot:

- Adds another reference and it points the blocks the original file points to.

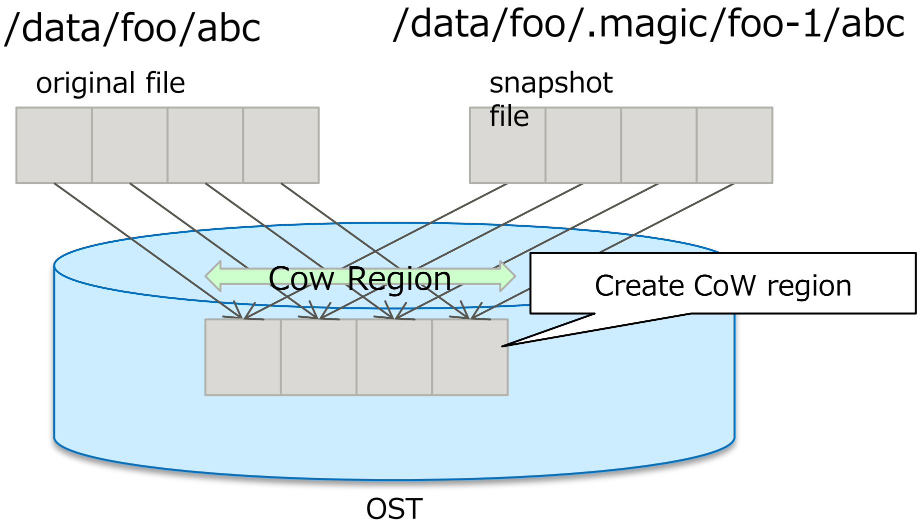


/data/foo/abc

original file

data blocks

OST



/data/foo/abc

original file

/data/foo/.magic/foo-1/abc

snapshot file

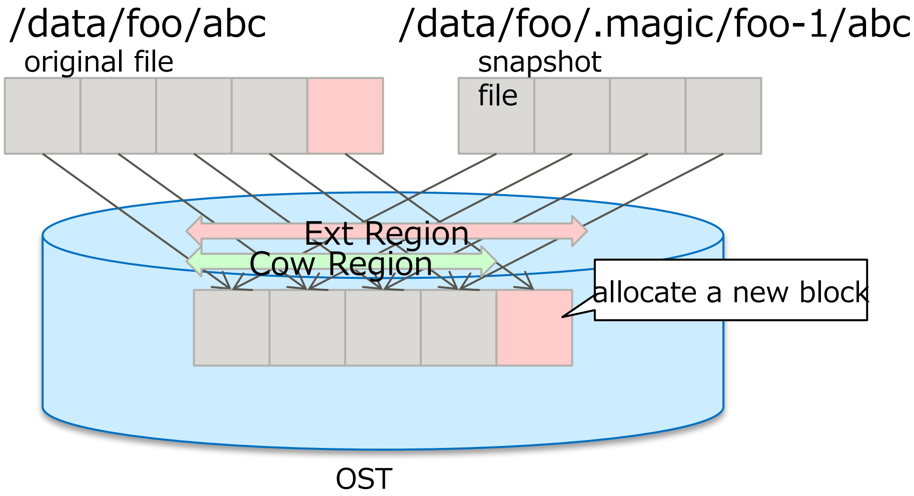
Cow Region

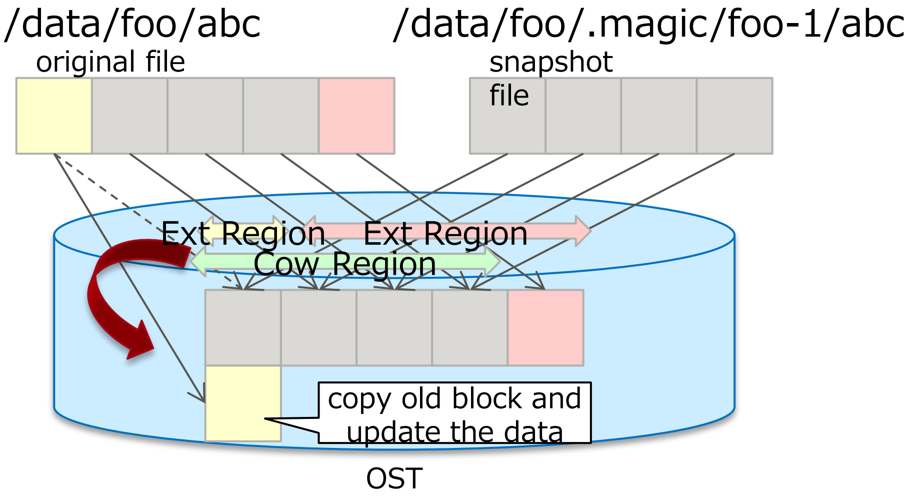
Create CoW region

OST

Basic Mechanism of DL-SNAP by Extent Region(2)

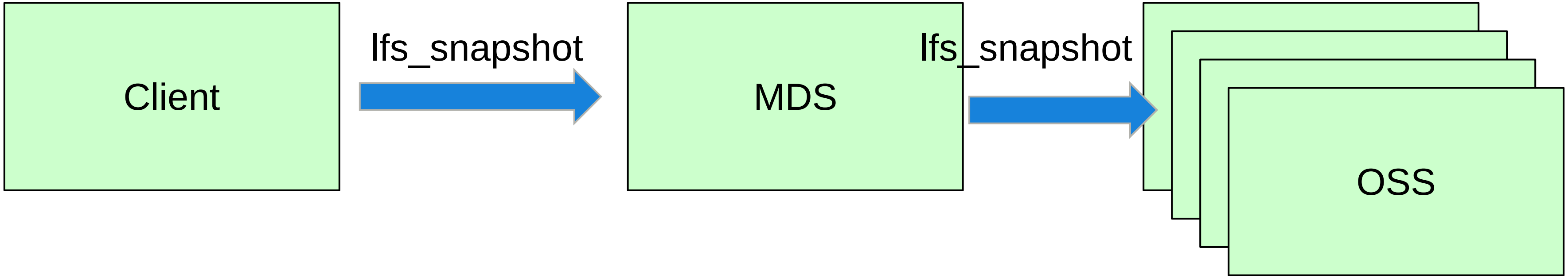
- Append-writing the original file:
  - Allocates a new data block on the OST and writes the data to the data block. Also, creating the original file modification extent of the data block.
- Over-writing the original file:
  - Allocates a new data block on the OST and copy the original data block. Then, the file point the data block.





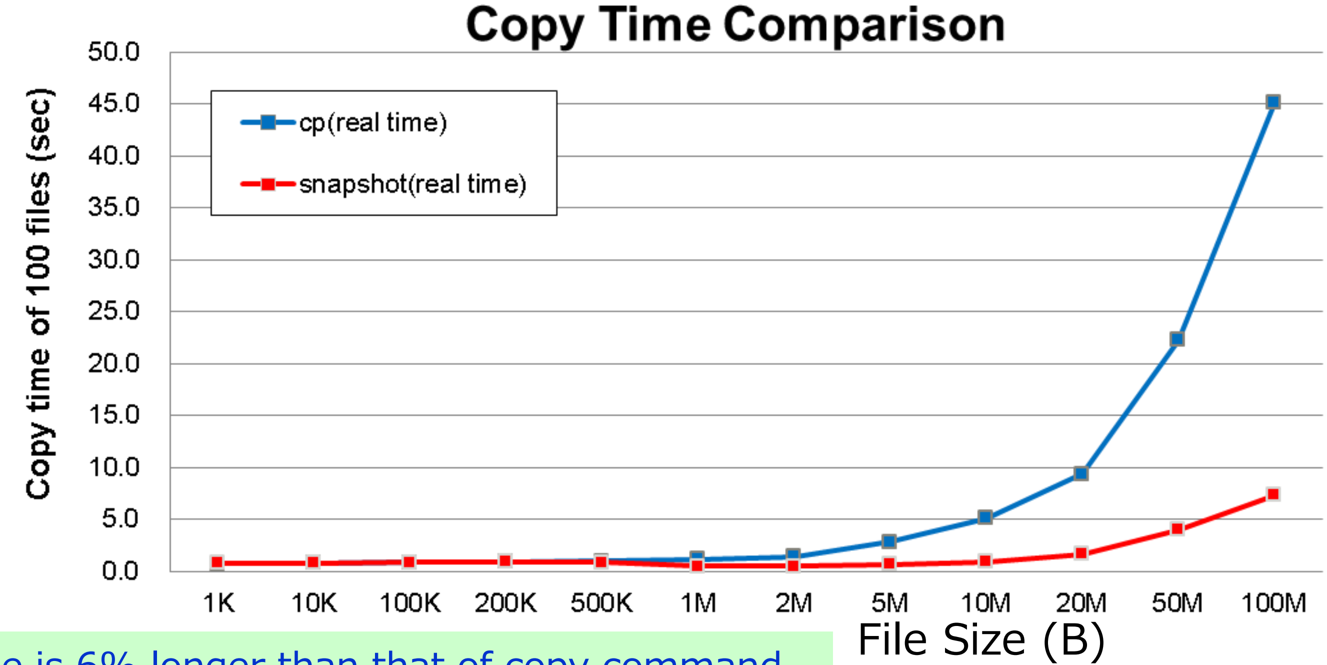
DL-SNAP: Ptlrpc extenstion

- Some Ptlrpc function extension for DL-SNAP



Evaluation of DL-SNAP

- DL-SNAP is faster than normal copy



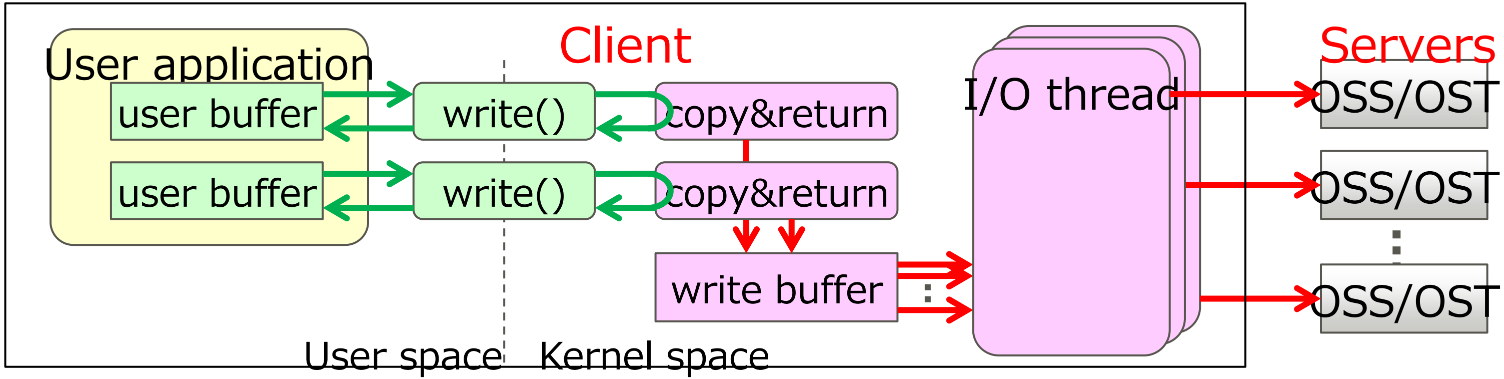
1K byte file is 6% longer than that of copy command, but the time on 100 MB file is over 5 times faster

Improving Single Process Write Performance

Improving Single 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

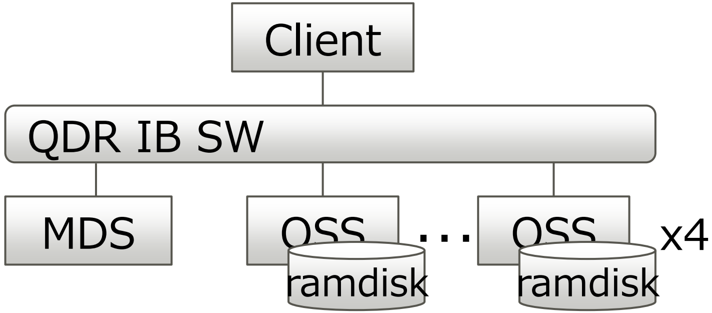
    There're some bottlenecks in Lustre's cache method using dirty buffer for each OST
- Our Approach
  - write returns immediately after copying user data to kernel buffer internally
  - Dedicated I/O threads transfer data from the buffer to OSS/OSTs in parallel, therefore write throughput dramatically improves from user perspective



Improving Single Process Write Performance

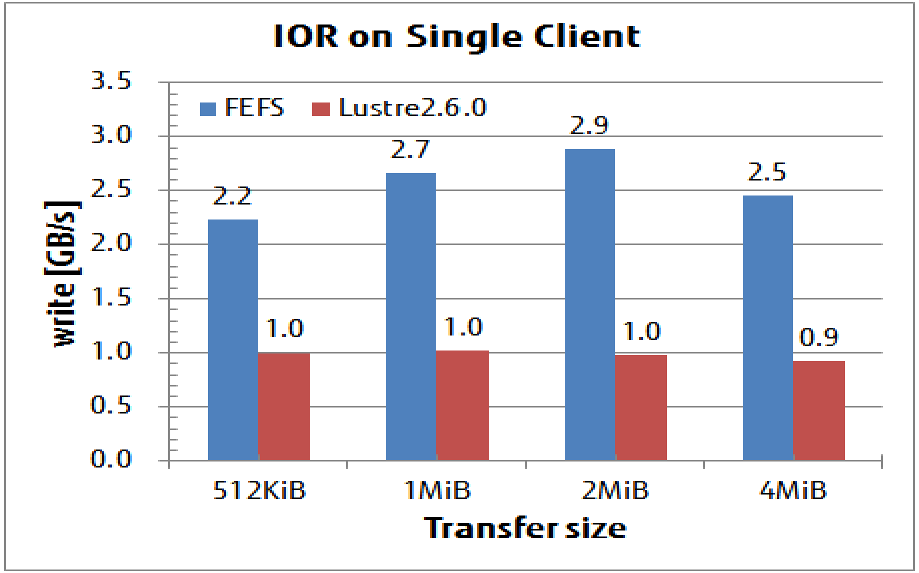
Lustre 2.6.0 vs. prototype (Lustre 1.8 base)

- 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



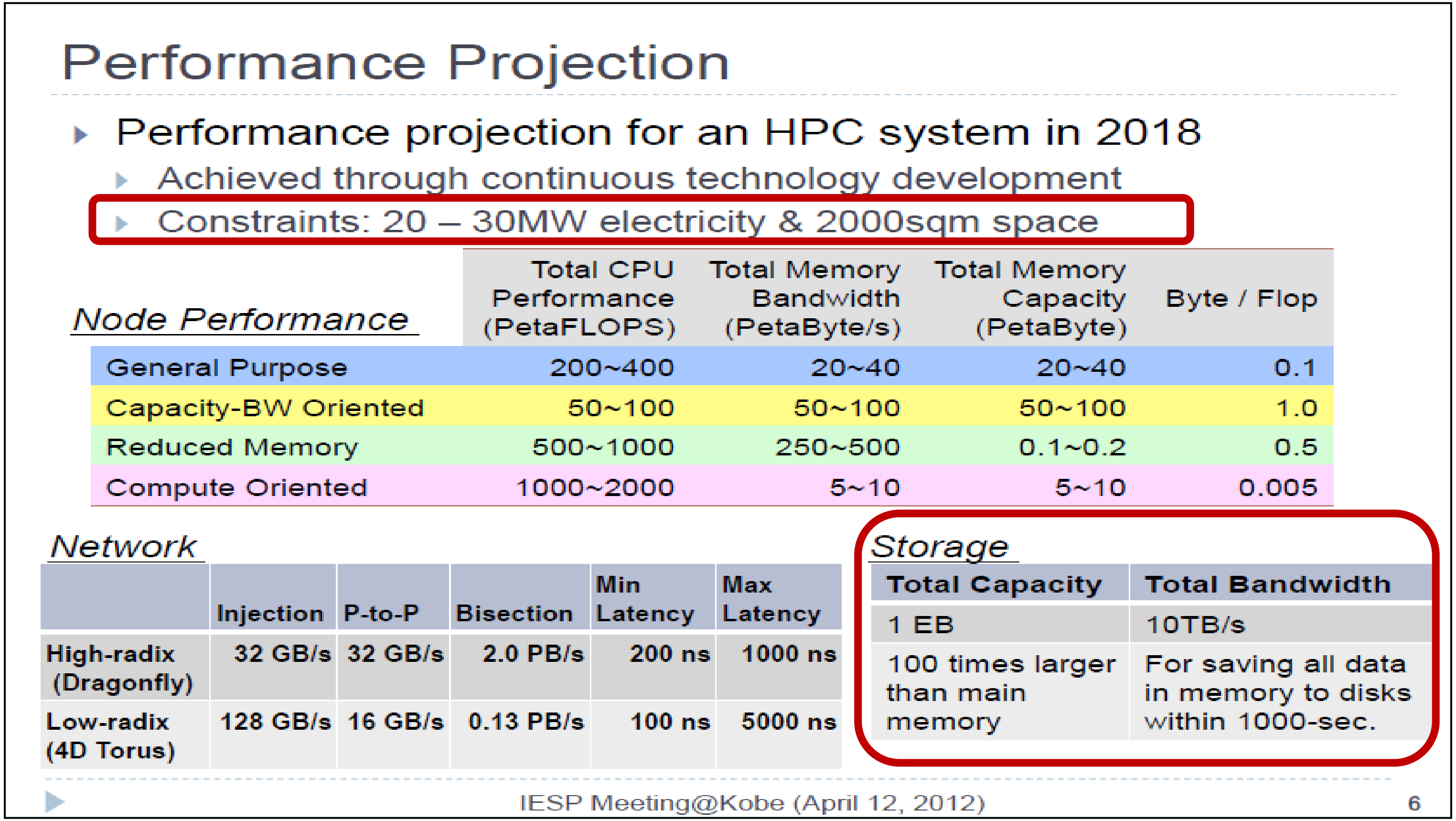
Development Status

- Lustre 2.6 based version was implemented and shipped

- Lustre 2.8 based version will be re-implemented because of base code changes

Toward Exascale Computing

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)

Issues of File System for Exascale Systems

Discussed at Lustre Developer Summit 2014 in Reims

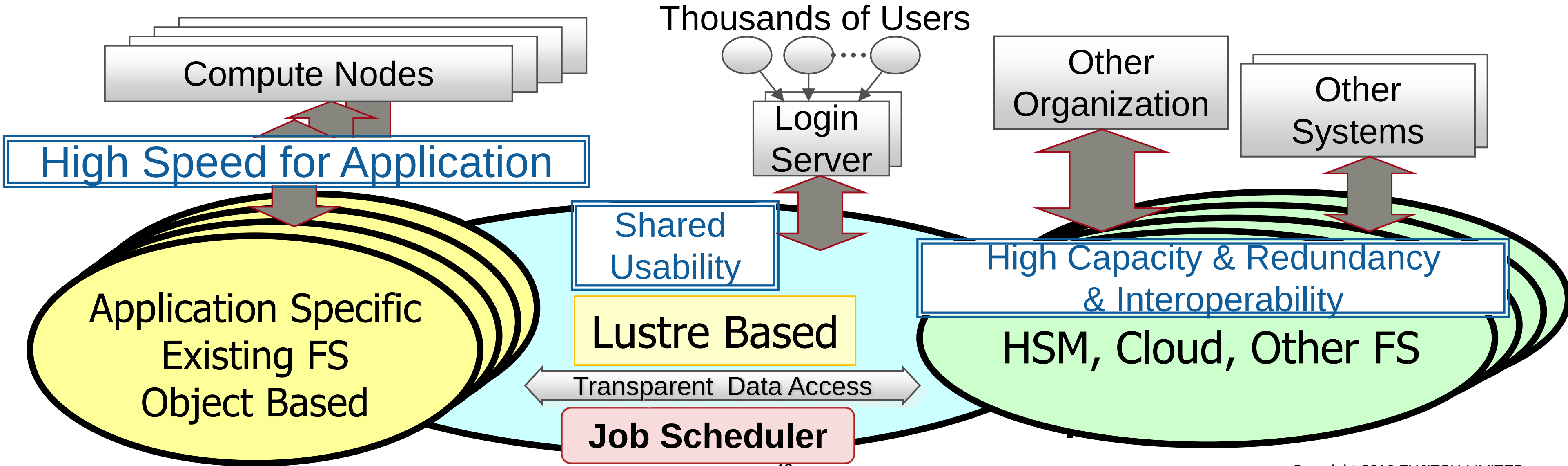
- System Limits: Increase the logical upper limits (capacity, # of clients, # of OSTs, etc…)
- Memory Usage: Required memory should not proportional to # of OSTSs
- Meta Data Performance: Reduce metadata access. Lustre DNE improves metadata performance, but requires additional hardware resource, MDS 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

Discussed at Lustre Developer Summit 2015 in Paris

- 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:

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



Exascale Concerns (Summit 2014)

- 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)

Exascale Concerns (Summit 2014) (Cont’d)

- Meta Data Performance
  - Concern: Meta performance will hit the limit for exascale-applications which create several billions of files in a single job
    - e.g. NICAM creates 1.8 billion files per job
  - Approach: Not only adding MDSs by DNE, but also reduce meta access to Lustre by inserting intermediate layer between compute node and file system
    - e.g. “File composition library” under development by RIKEN AICS manages many files as a single file on Lustre
- I/O Throughput and Capacity
  - Concern: Achieve higher throughput (10TB/s~) and larger capacity (~1EB) in limited power consumption and footprint
  - Approach: Hierarchical storage system architecture
    - e.g. Burst buffer, Fast forward I/O, etc.

Exascale Concerns (Summit 2014) (Cont’d)

System Noise

- Concern: Eliminate OS jitter to maximize performance of massively parallel applications
  - We took great effort to reduce system noise in K computer (reported at LAD12)
- Approach: Introducing dedicated cores for daemons (OS timer, file I/O, MPI, etc)
  - e.g. Fujitsu’s SPARC64 XIfx CPU for Post-FX10 provides with 2-assistant cores

Exascale Concerns (Summit 2015)

FUJITSU

- 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 RAID0 requires doubled space. RAID-5 maybe?
  - One Approach: File Services should not be stopped even if some storages are offline.

FUJITSU

shaping tomorrow with you