shaping tomorrow with you

Fujitsu’s Contributions to Lustre 2.x Roadmaps with Intel

Shinji Sumimoto, Oleg Drokin
Fujitsu/Intel

Apr.18 2013 LUG2013@San Diego

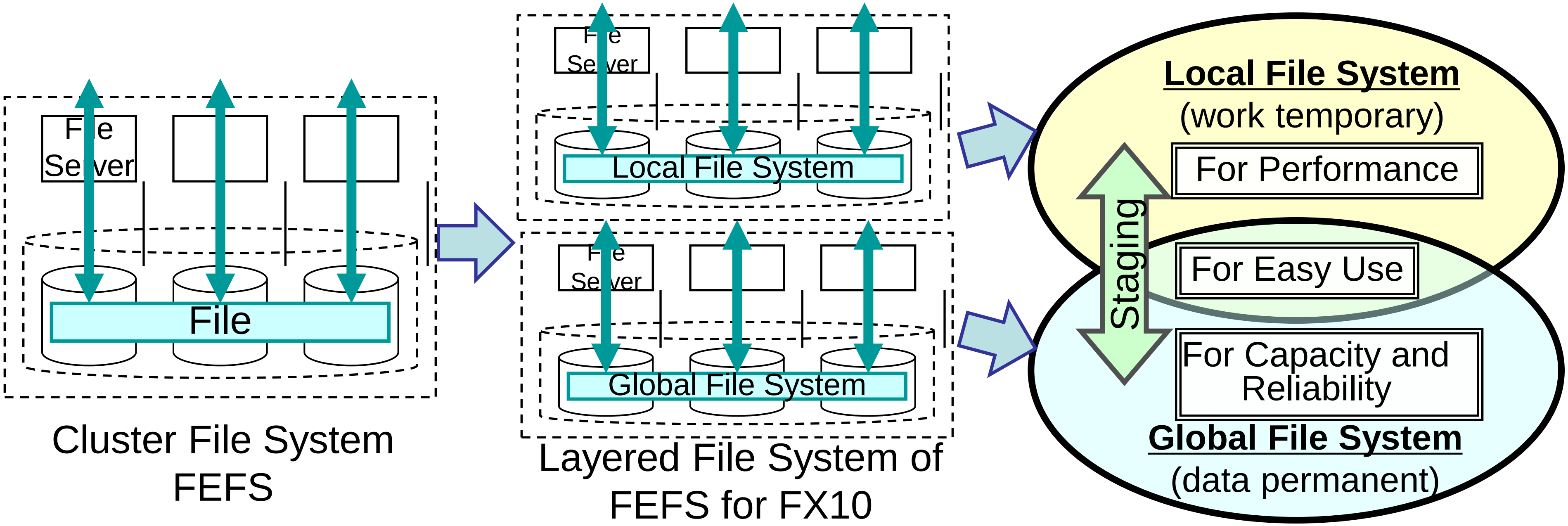
Outline of This Talk

- FEFS* and FEFS extension overview
- Contribution of FEFS extension to Lustre 2.x
  - Current Status and Schedule
- Discussion of Selected items (By Oleg Drokin)

*: "FUJITSU Software FEFS"

Overview of FEFS

- Goals: To realize World Top Class Capacity and Performance File system 100PB, 1TB/s
- Based on Lustre File System with several extensions
- 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



Cluster File System FEFS

Layered File System of FEFS for FX10

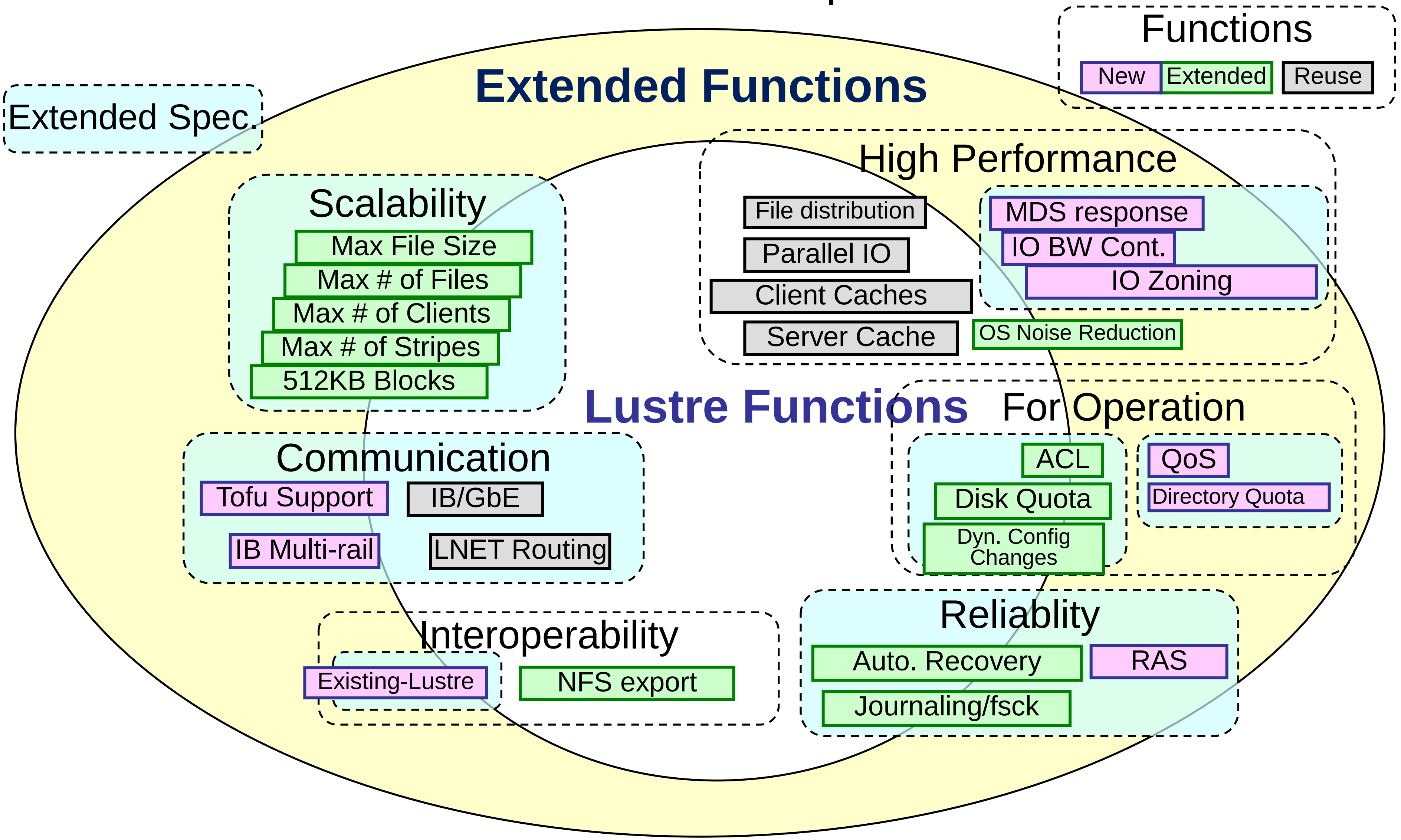
Lustre Specification and Goal of FEFS

Features		Current Lustre	Our 2012 Goals
System Limits	Max file system size	64PB	100PB (8EB)
	Max file size	320TB	1PB (8EB)
	Max #files	4G	32G (8E)
	Max OST size	16TB	100TB (1PB)
	Max stripe count	160	20k
	Max ACL entries	32	8,191
Node Scalability	Max #OSSs	1,020	20k
	Max #OSTs	8,150	20k
	Max #Clients	128K	1M
Block Size of *ldiskfs* (Backend File System)		4KB	~512KB

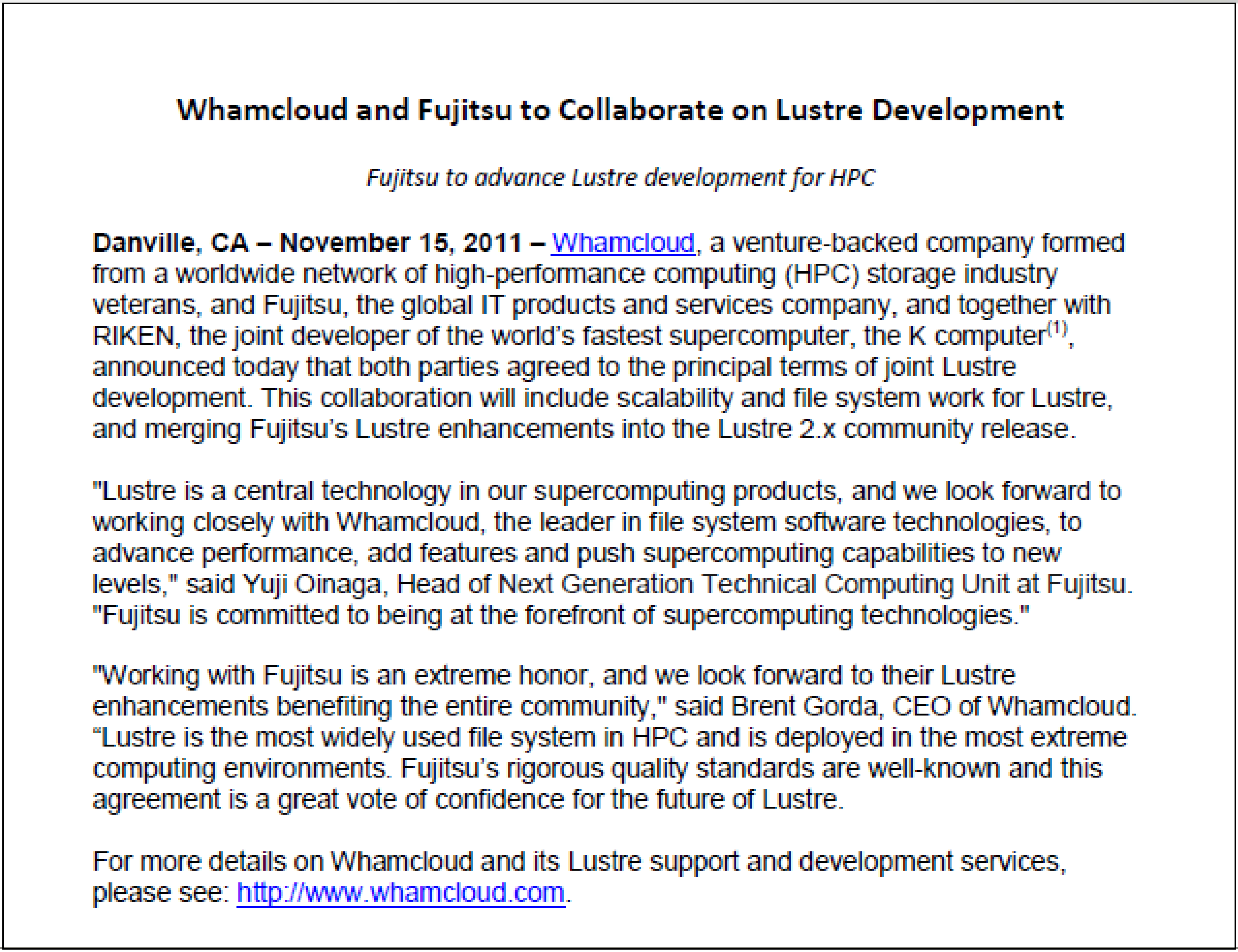
These were contributed to OpenSFS: 2/2011

Lustre Extension of FEFS

- We have extended several Lustre specifications and functions



2011/11: Press Release at SC11



Whamcloud and Fujitsu to Collaborate on Lustre Development

Fujitsu to advance Lustre development for HPC

Danville, CA – November 15, 2011 – Whamcloud, a venture-backed company formed from a worldwide network of high-performance computing (HPC) storage industry veterans, and Fujitsu, the global IT products and services company, and together with RIKEN, the joint developer of the world's fastest supercomputer, the K computer[1], announced today that both parties agreed to the principal terms of joint Lustre development. This collaboration will include scalability and file system work for Lustre, and merging Fujitsu's Lustre enhancements into the Lustre 2.x community release.

"Lustre is a central technology in our supercomputing products, and we look forward to working closely with Whamcloud, the leader in file system software technologies, to advance performance, add features and push supercomputing capabilities to new levels," said Yuji Oinaga, Head of Next Generation Technical Computing Unit at Fujitsu. "Fujitsu is committed to being at the forefront of supercomputing technologies."

"Working with Fujitsu is an extreme honor, and we look forward to their Lustre enhancements benefiting the entire community," said Brent Gorda, CEO of Whamcloud. "Lustre is the most widely used file system in HPC and is deployed in the most extreme computing environments. Fujitsu's rigorous quality standards are well-known and this agreement is a great vote of confidence for the future of Lustre.

For more details on Whamcloud and its Lustre support and development services, please see: http://www.whamcloud.com.

Fujitsu’s Contribution work with Intel to Lustre 2.x Roadmaps

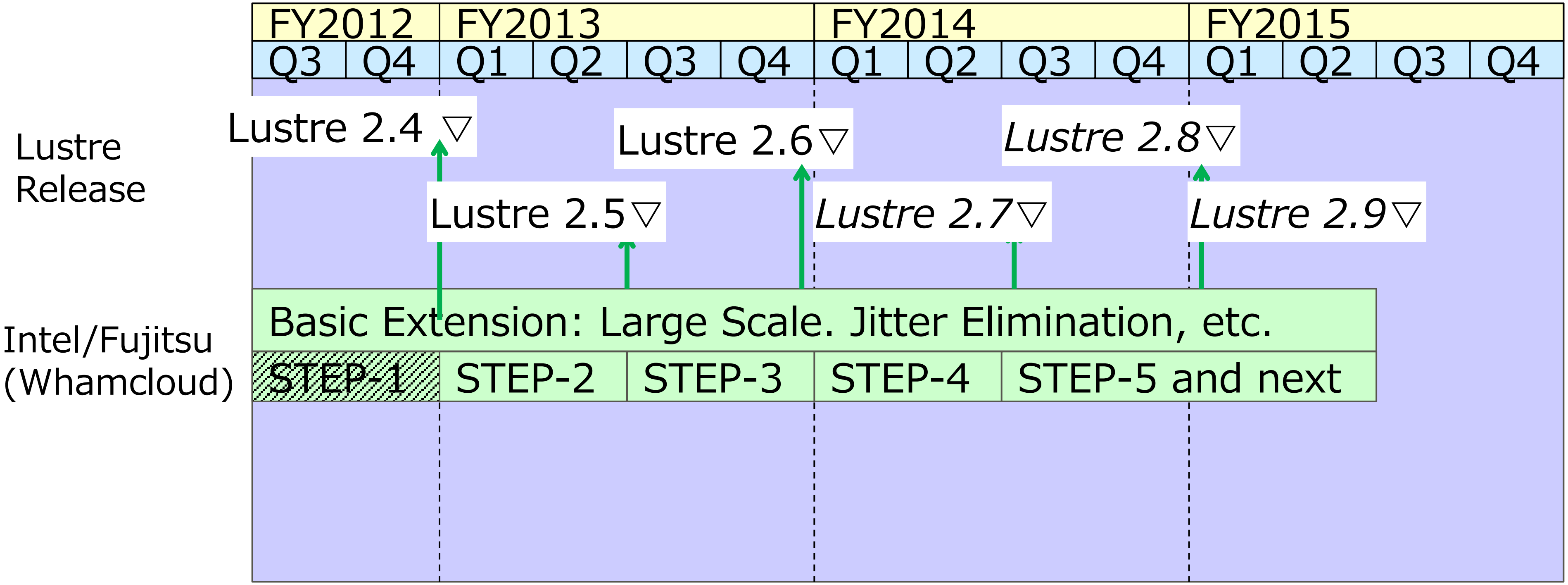
Period	Phase	Topics
2011/12 – 2012/3	Selection of Fujitsu’s Extensions to Lustre 2.x by Intel	20 of 60 items selected
2012/4 – 2012/6	Making Proposals by Intel	Three Phase Proposal
2012/9 – 2013/3	First Phase	Architecture Interoperability, LNET and OS Jitter update
2013/4 – 2013/9	Second Phase	Memory Management
2013/10 – 2014/3	Third Phase	Large Scale Performance, OST management

20 Selected Fujitsu Extensions to Lustre 2.x

No	Subproject / Milestone	Category	Phase
1	LNET Networks Hashing	Performance	1
2	LNET Router Priorities	RAS	1
3	LNET: Read Routing List From File	Large Scale	1
4	Optional /proc Stats Per Subsystem	Memory Reduction	2
5	Sparse OST Indexing	Sparse OST	3
6	New Layout Type to Specify All Desired OSTs	OST selection	3
7	Reduce Unnecessary MDS Data Transfer	Meta Performance	3
8	Open/Replay Handling	Memory Reduction	2
9	Add Reformatted OST at Specific Index	OST Dynamic Addition	3
10	Empty OST Removal with Quota Release	OST Dynamic Removal	3
11	Limit Lustre Memory Usage	Memory Limit	2
12	Increase Max obddev and client Counts	Large Scale	4orF
13	Fix when Converting from WR to RD Lock	Bug Fixes (fcntl)	4orF
14	Reduce ldlm_poold Execution Time	OS Jitter	1
15	Ability to Disable Pinging	OS Jitter	1
16	Opcode Time Execution Histogram	For Debug	4orF
17	Endianness Fixes	Architecture Inter-op.	1
18	OSC Request Pool Disabling	Memory Reduction	2
19	Pinned Pages Waiting for Commit	Memory Reduction	2
20	Errno Translation Tables	Architecture Inter-op	1

Current Schedule

- Already started with Intel applying FEFS extension to Lustre 2, and will plan to finish by mid FY2015



FY2012		FY2013				FY2014				FY2015			
Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4

Lustre Release

Lustre 2.4 ▽

Lustre 2.5▽

Lustre 2.6▽

Lustre 2.7▽

Lustre 2.8▽

Lustre 2.9▽

Intel/Fujitsu (Whamcloud)

Basic Extension: Large Scale. Jitter Elimination, etc.

STEP-1

STEP-2

STEP-3

STEP-4

STEP-5 and next

FUJITSU

shaping tomorrow with you

Fujitsu Contributions to Lustre*

High Performance Data Division

Oleg Drokin

April 16, 2013

* Other names and brands may be claimed as the property of others.

Network jitter: Doing away with pings

- On large systems pings are expensive:
  - Clients * targets pings every obd_timeout/4 interval (default 25 sec)
- Main purposes of pinging:
  - Lets clients detect restarted/recovering servers in reasonable time
  - Proactively weeds out unreachable/dead clients
- With Imperative Recovery we’ve got #1 covered
- Many existing systems already know about dead clients from cluster management tools
  - Lustre provides a way for those systems to tell it about dead clients for immediate eviction
- Now servers have a way to tell clients to avoid idle pinging

Lnet routes hashtable

- It was noticed that Lnet stores routing entries in a single linked list
- As number of routes increases on large systems, iterating the list becomes more and more expensive
- Hash table is a pretty natural solution to this problem

Limiting OS jitter – ldlm poold

- On FS with 2000 OSTs ldlm_poold was using 2ms of cpu every second on every client
  - Investigations revealed it was walking a linked list of all ldlm namespaces (one per connected service) every second to update lock stats
- The lock statistics on empty namespaces do not change
  - So no need to walk empty namespaces at all
- An updating action is performed every 10 seconds on clients
  - So no need to wake up every second, just see how much time left till next action and sleep this much
- A lot of the calculations don't need to be periodic and could be predicted, making ldlm_poold pointless (TBD)

SPARC* architecture support

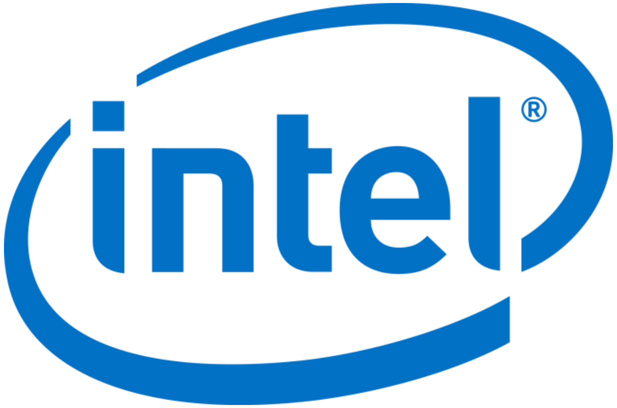
- SPARC architecture is big-endian
  - Fujitsu performed a full Lustre* source audit for endianness issues and contributed the results back to the community
- SPARC Linux has "different" error numbers (Solaris* compatible)
  - This highlights a bigger problem of assuming the error numbers being compatible on different nodes in network which is not true.
  - Fujitsu came with an errno translation table solution that it contributed back to community
    - Intel is working on integrating this solution into 2.x releases
- Fujitsu also contributed access to a SPARC system test cluster

Memory usage improvements

- /proc statistics on clients tends to use a lot of RAM
  - Esp. if you have thousands of targets connected, it could use hundreds of megabytes
- Fujitsu developed and contributed a way to disable such statistic tracking
  - Being adopted by Intel for inclusion into Lustre 2.x

More fine-grained control of striping

- Current Lustre striping of "starting at X, Y wide" is not always adequate
- Fujitsu developed and contributed code to allow very-fine-grained stripe allocation on per-OST basis
  - This is currently being adopted by Intel into inclusion into Lustre 2.x
- Additionally, assumption about contiguous OST numbering is also removed which would allow for flexible OST-numbering schemes


intel®