Computer simulations create the future

Jun. 28, 2018@HPC-IODC'18

I/O Interference Alleviation on Parallel File Systems Using Server-Side QoS-Based Load-Balancing

Yuichi Tsujita[1], Yoshitaka Furutani[2], Hajime Hida[3],
Keiji Yamamoto[1], Atsuya Uno[1], Fumichika Sueyasu[2]

1 RIKEN Center for Computational Science
2 FUJITSU LIMITED
3 FUJITSU SOCIAL SCIENCE LABORATORY LIMITED

Outline

- Research Background
- K computer and Its Filesystems
- QoS Management for Load-Balancing
- Performance Evaluation
- Related Work
- Summary

Research Background & Motivation

- High MDS load or very slow MDS response in a huge scale of parallel file systems at the K computer lead to performance degradation in
  - local I/O by compute nodes
  - data staging, and so forth

Ineffective operation 

- How can we alleviate high MDS load / slow MDS response for further performance improvements?

- Knowing the root-causes of such high MDS load / slow MDS response

- Adopting QoS control for an MDS, which is available on the parallel file systems at the K computer

Effective operation 

K computer

System Configuration of the K computer

40 m x 40 m
Full System
Compute Rack × 864

10.6(11.3)PFLOPS
1.27(1.34)PiB

4000mm x 800mm
2 Cabinets
Compute Rack × 4
Disk Racks × 1

49.2(52.4)TFLOPS
6.00(6.38)TiB

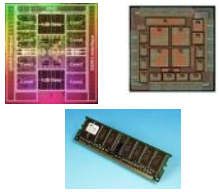
800mm x 800mm
Compute Rack
SB × 24
IOSB × 6

12.3(13.1)TFLOPS
1.50(1.59)TiB

500mm x 500mm
System Board(SB)
Node × 4

512GFLOPS
64GiB

Node
CPU × 1
ICC × 1
memory

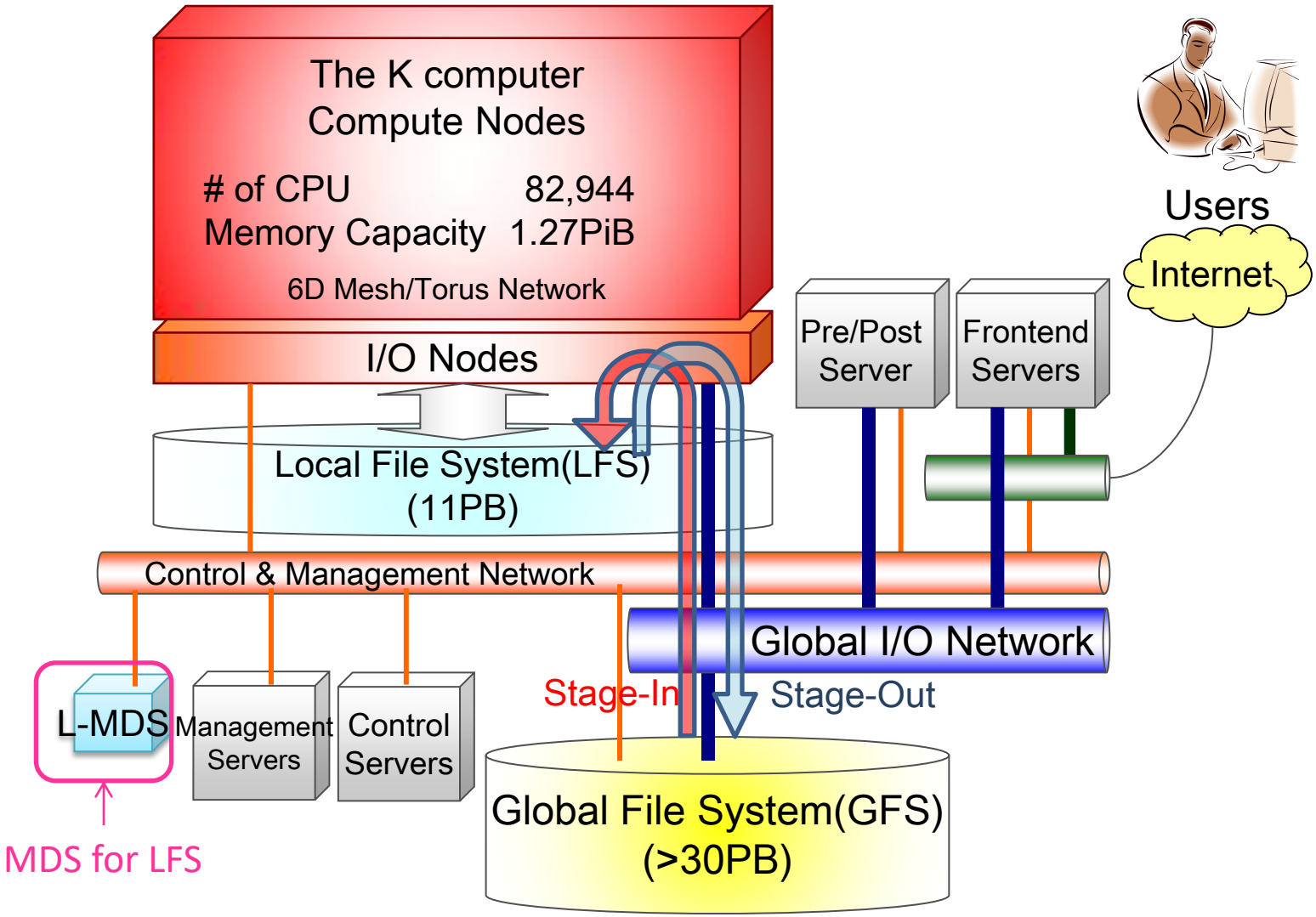


128GFLOPS
16GiB

( ) included IO node performance and memory capacity.

K computer

File Systems of the K computer



FEFS is used for both LFS and GFS.
(FEFS: Fujitsu Exabyte File System based on Lustre technology)

K computer

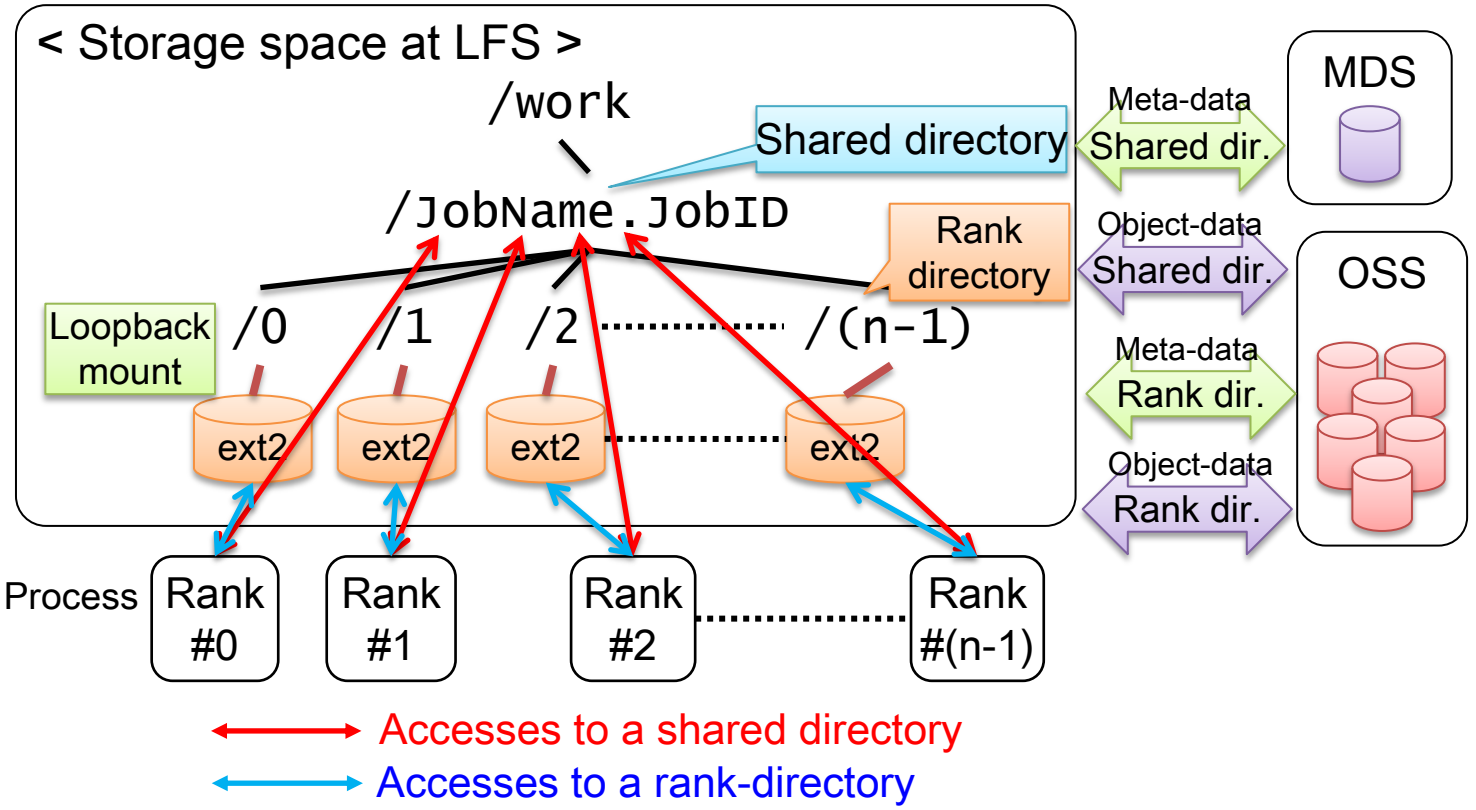
File System Configuration

- Organization of file systems at the K computer
  - LFS : Performance oriented
    - for high performance I/O during computation
  - GFS : Capacity oriented
    - for huge data storing and high redundancy

File system	LFS	GFS
Total volume size	~ 11 PB	> 30 PB
# volumes	1	8
# OSSs	2,592	90
# OSTs	5,184	2,880
Disk system of OST	RAID5	RAID6

Rank-Directory (Loopback File System) at the LFS

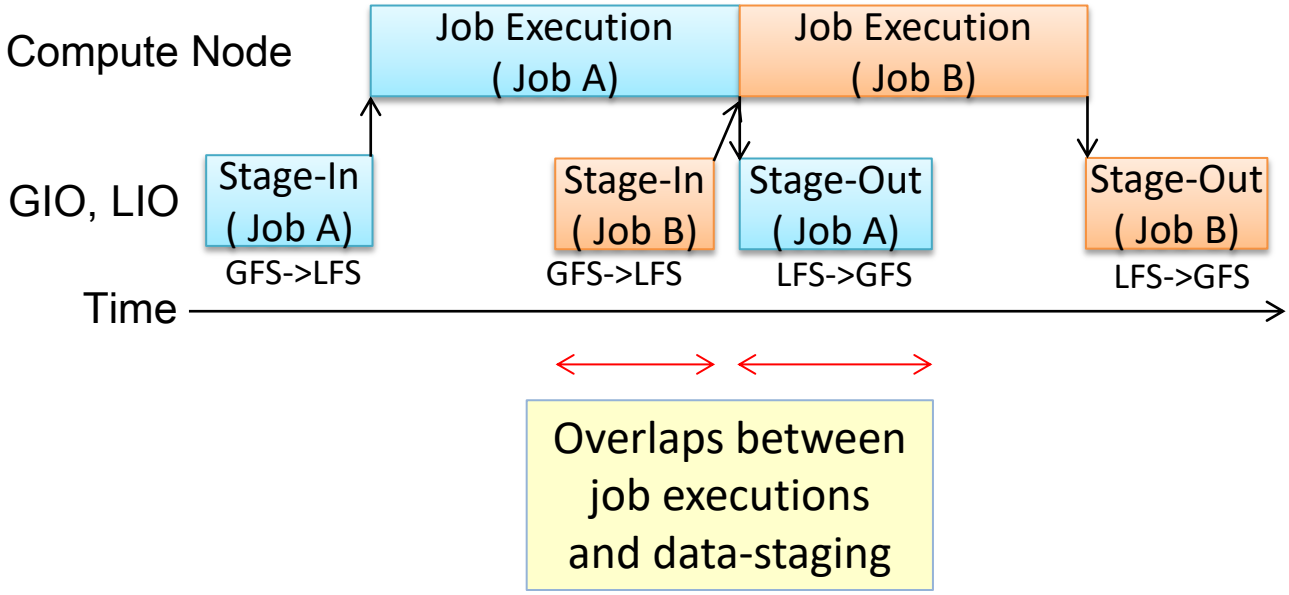
- Mitigation of I/O interference among processes



- Reducing MDS accesses leads to effective utilization of LFS.
- I/O accesses in rank-directories are free from slowdown of MDS performance.

Data-Staging at the K computer

- Asynchronous data-staging



✓ Stage-in phase includes rank-directory creation.

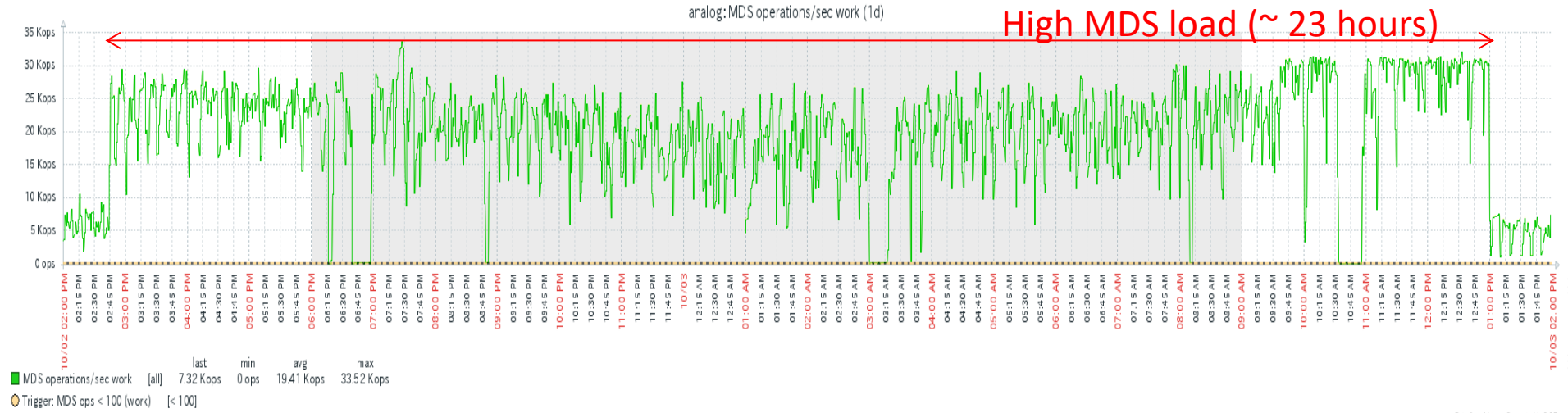
✓ High MDS load or quite slow MDS response may increase times for rank-directory creation. ⇨ An increase in times for stage-in phase

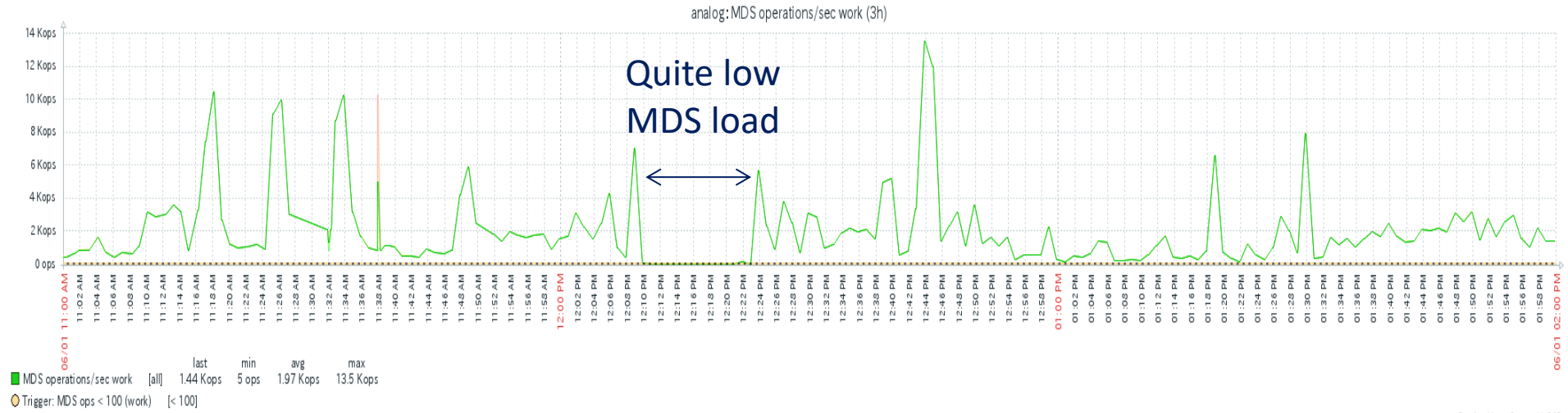
✓ Ineffective job scheduling

K computer

High MDS Load/Slow MDS Response

- MDS Activities

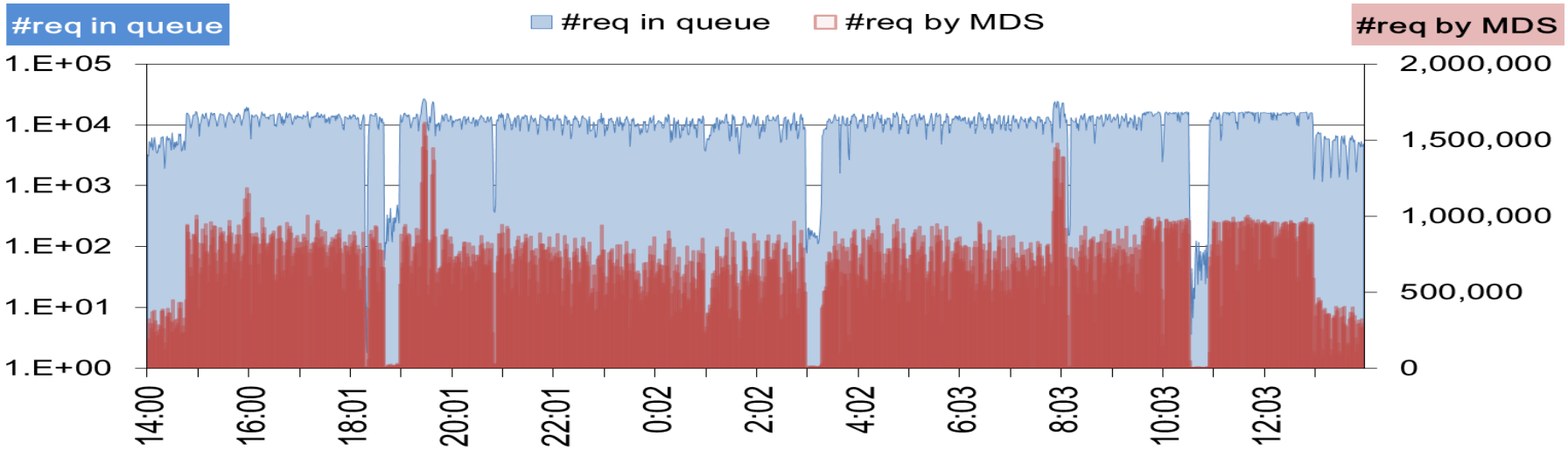




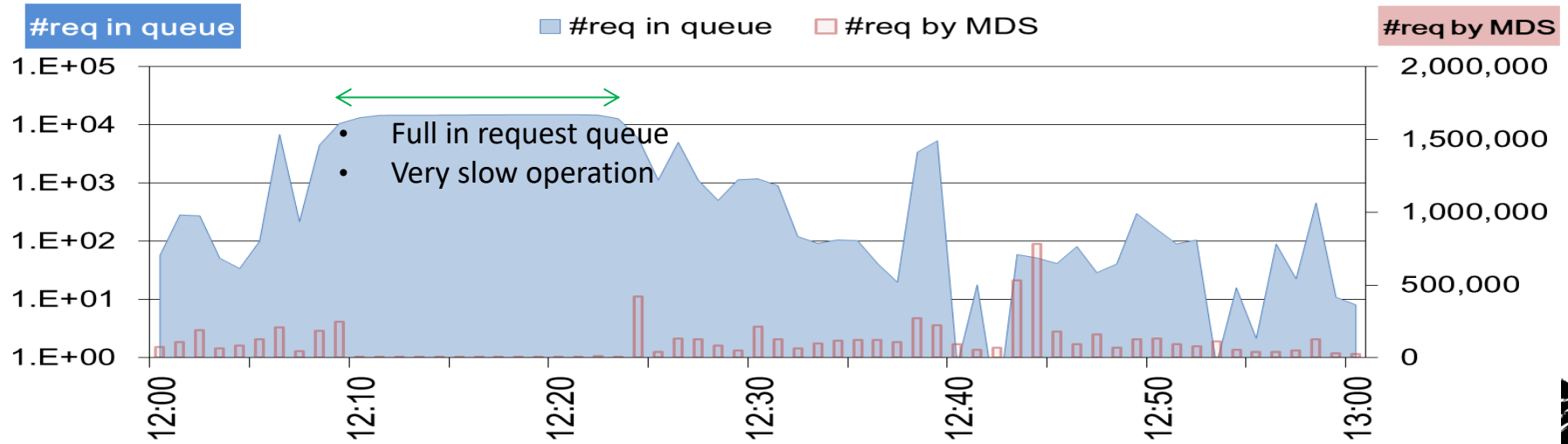
K computer

MDS Request Queue Status

- High MDS load



- Slow MDS response



K computer

Interference due to MDS Problems

- High MDS load/Slow MDS response lead to
  - ✓I/O performance degradation in local I/O at the LFS
  - ✓Increase in times for data-staging, and so on.

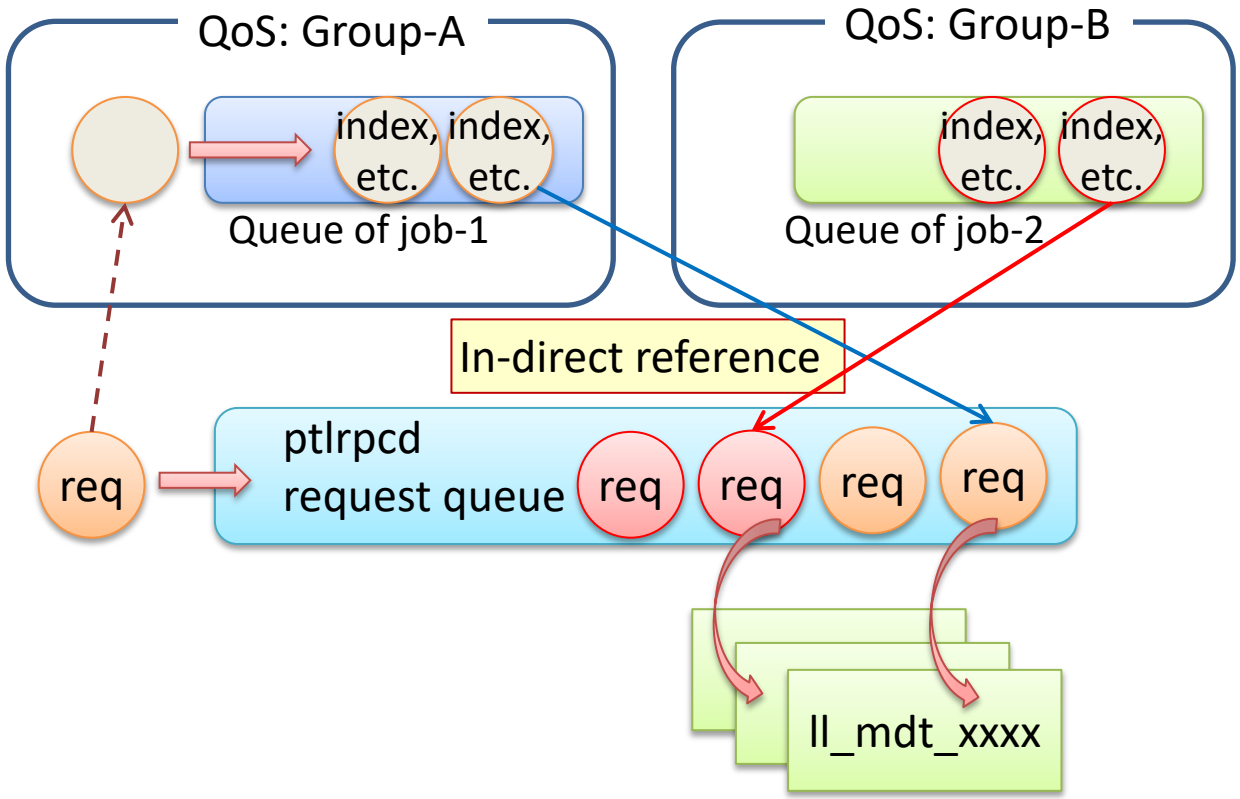
  - High MDS load
    - Fully utilized service threads for dominant MDS heavy job accessing a large number of files concurrently
  - Quite slow MDS response
    - Every service thread was blocked to wait responses from associated OSSes.

Alleviation in I/O interference

QoS control for service threads at the MDS

QoS of FEFS(1)

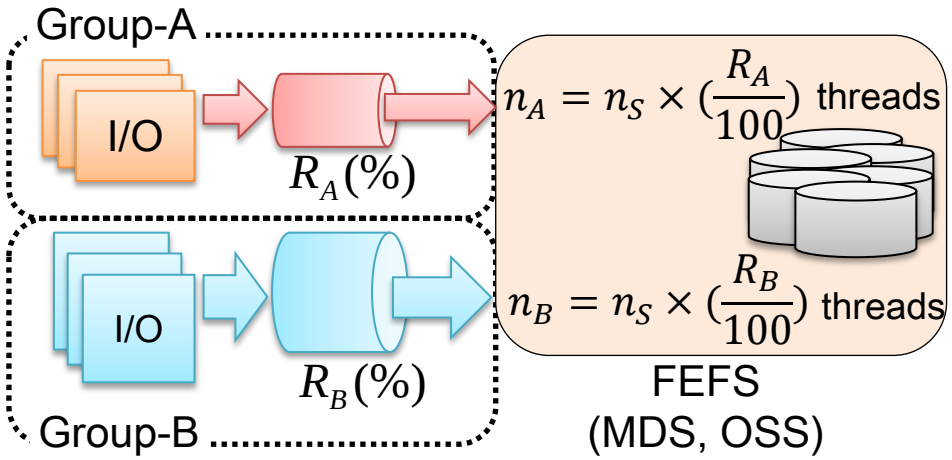
- Requests in queue under QoS control



K computer

QoS of FEFS (2)

- Load-balancing by QoS



$$n_A = n_S \times (\frac{R_A}{100}),\ n_B = n_S \times (\frac{R_B}{100})$$

(GIO nodes : Compute nodes) = ($\frac{100 \times n_A}{(n_A + n_B)}$: $\frac{100 \times n_B}{(n_A + n_B)}$)

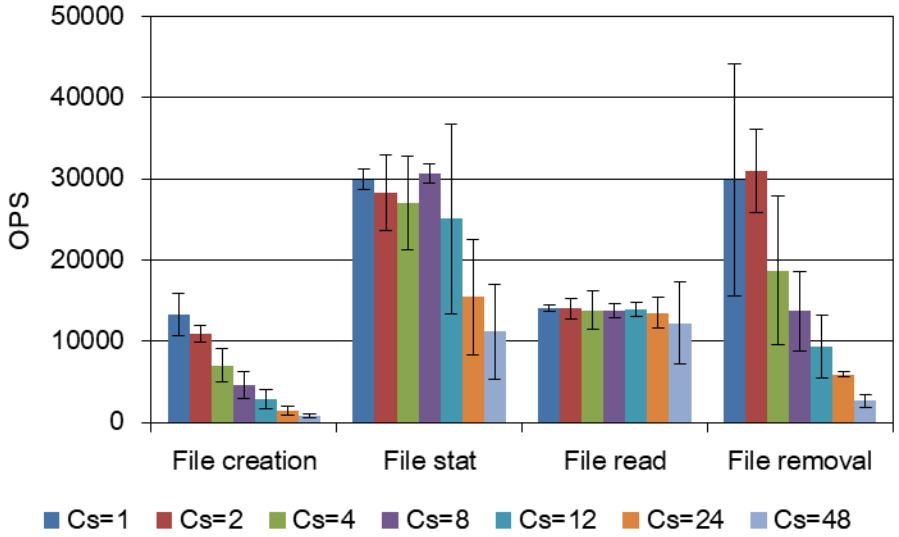
K computer

Performance Evaluation

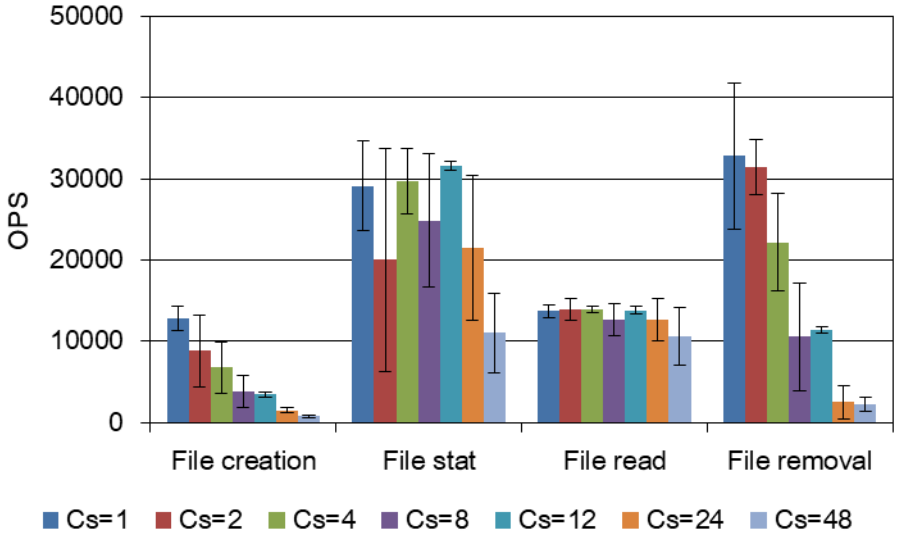
- Stripe count impact in MDS performance
- QoS impact in fair-share execution among concurrent running jobs
- QoS impact in data-staging

MDTEST performance of L-MDS

- 2 sets (768 and 1,536 processes on 192 compute nodes) of MDTEST runs
  - Command: `./mdtest -d ../md_dir -n 100 -i 3 -F -u`
  - Seven sets of stripe counts (Cs): 1, 2, 4, 8, 12, 24, and 48 (12 is default configuration.)
  - Mean values and standard deviations from 6 iterations



768 processes@4x6x8 (4 processes/node)



1,536 processes@4x6x8(8 processes/node)

Cs: stripe count

✓ An increase in stripe count led to performance degradation, especially in "File creation," "File stat," and "File removal."

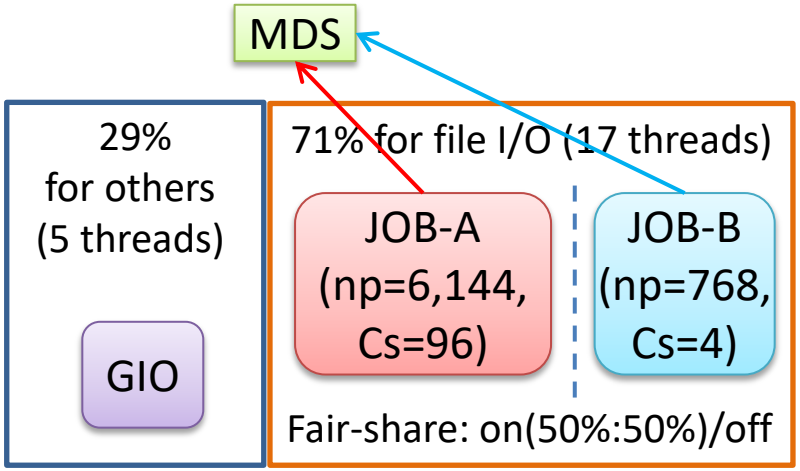
K computer

QoS for I/O Interference Alleviation

- QoS control at the MDS
  - Managing service thread assignment for several groups, such as local file I/O and data-staging
  - Fair-share job execution in order to mitigate I/O interference each other

Examination of Fair-Share Execution(1)

- I/O interference impact among two concurrent jobs (MDTEST runs)
  - JOB-A(6,144 processes): `./mdtest -d ../work -n 100 -i 200 -F -u`
  - JOB-B(768 processes): `./mdtest -d ../work -n 100 -i 3 -F -u`



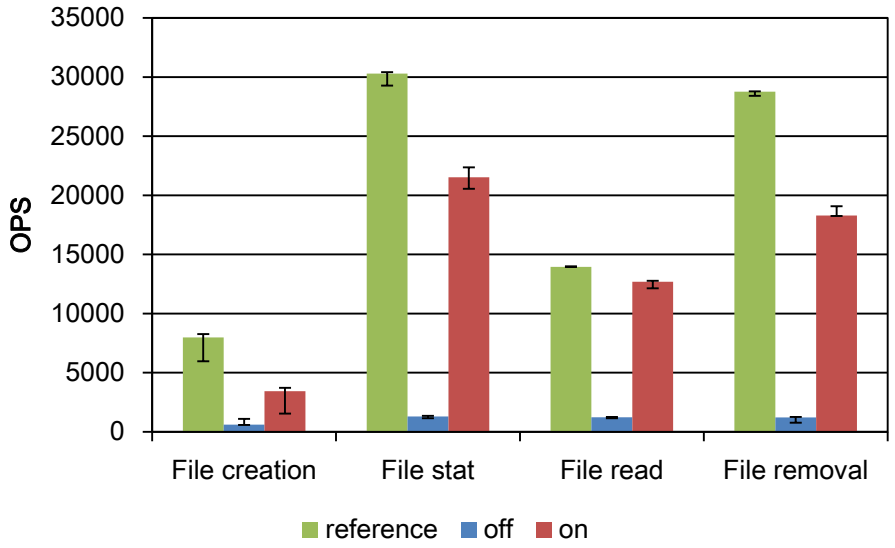
- GIO group and compute node group utilized up to 29% and 71% of 24 service threads of the MDS.
- Fair-share function split available service threads evenly among JOB-A and JOB-B.
- Performance measurements at JOB-B

Notation	Executed jobs	np	Cs	QoS rate	Fair-share
reference	JOB-B	768	4	None	None
off	JOB-A	6,144	96	71%	None
	JOB-B	768	4		
on	JOB-A	6,144	96	71%	JOB-A:JOB-B=50%:50% (up to 90% each if available)
	JOB-B	768	4		

K computer

Examination of Fair-Share Execution (2)

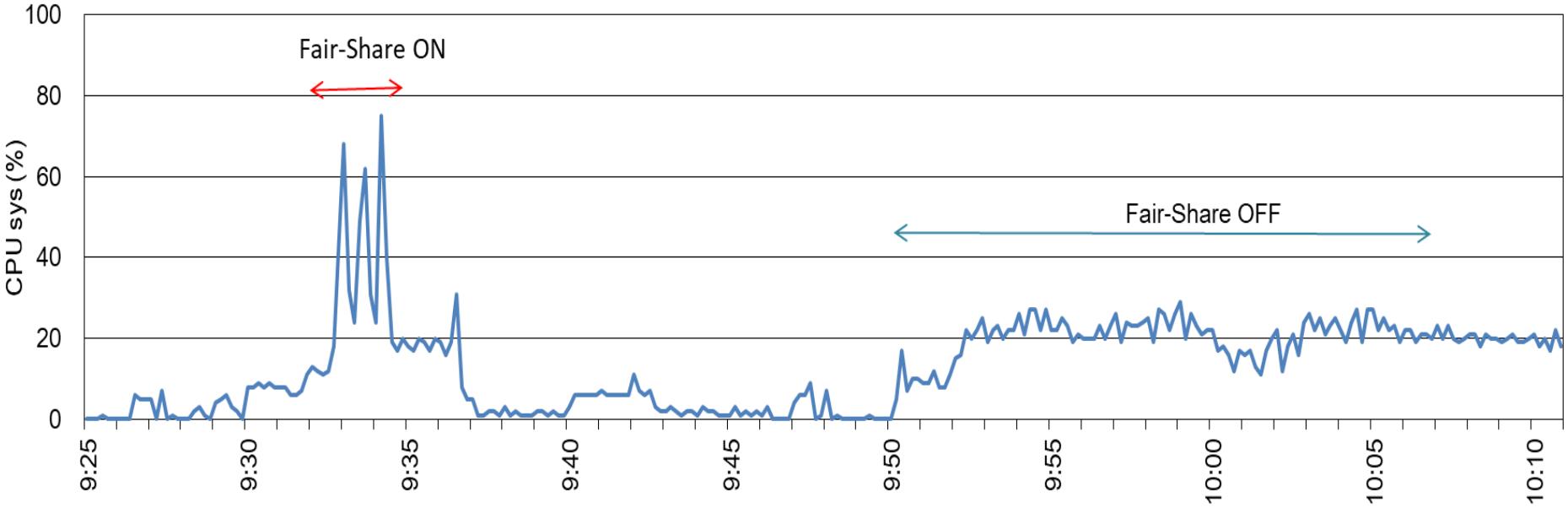
- MDTEST results
  - Big performance degradation without fair-share control (“off”)
  - Big interference mitigation under fair-share control (“on”)
    - Although performance degradation was observed compared with the “reference” case, improvement ratio relative to the “off” case was bigger than minimization ratio relative to the “reference” case.



K computer

Examination of Fair-Share Execution(3)

- CPU utilization at the MDS
  - Higher CPU utilization around 70% was realized under fair-share control.
    - Similar CPU utilization compared to the “reference” case (~70%)
  - Without fair-share control, CPU utilization was low (around 20%).



K computer

QoS Impact in Data-Staging (1)

- Evaluation of I/O Interference impact in data-staging

1. Submit a job for data-staging
2. Start-up of stage-in phase
3. Rank-directory creation at each rank on OSTs
4. Start-up of data transfer from GFS to LFS
5. End of stage-in phase

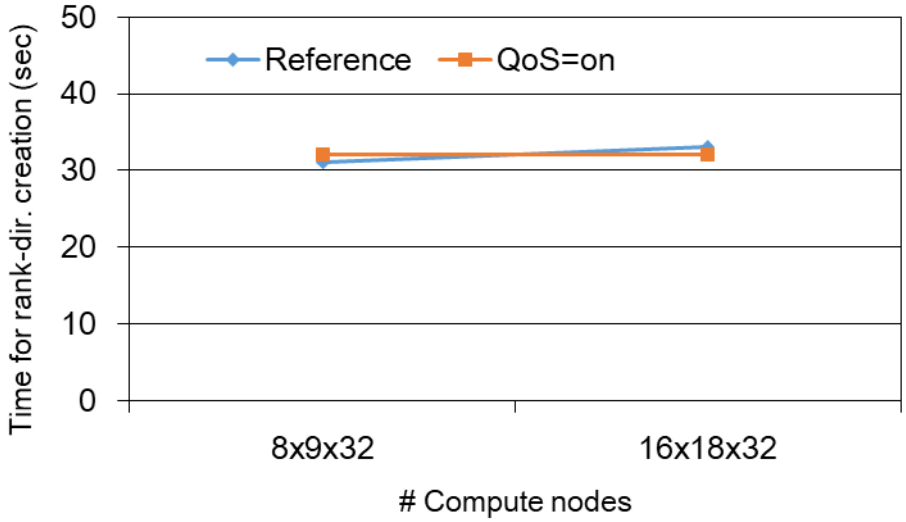
Picking up the most earliest time stamp to start (t_S) and the most slowest time stamp to end (t_E) for rank-directory creation

Figuring out time for rank-directory creation: $t_{rnk\_d} = t_E - t_S$

K computer

QoS Impact in Data-Staging (2)

- Times for rank-directory creation in data-staging (stage-in phase)
  - Cs=96
  - Measurement of stage-in times under MDS high load due to an MDTEST run by 6,144 processes on the same number of compute nodes.
    - QoS off: Rank-directory creation was not completed within 5 minutes.
    - QoS on: Comparable in times for rank-directory creation of “Reference” case without the MDTEST run



➢ QoS management has much impact in interference alleviation.

K computer

Related Work

- Performance optimization in Lustre:
  - Many works have been addressing to tune parameters based on empirical study or operation profiles.
- Monitoring tools played an important role for performance tuning.
  - Lustre Monitoing Tools (LMT) (C. Morrone, LUG 2011)
    - LMT reports server-side performance metrics such as CPU utilization, memory usage, disk I/O bandwidth.
    - However, it does not provide detailed I/O information such as file system statistics.
- Load-balancing or contention-aware optimizations
  - QoS setup on PVFS2 using machine learning (Zhang et al., SC'11)
  - Dynamic I/O congestion control at Lustre (Qian et al., MSST'13)
  - Token bucket filter in NRS (Qian et al., SC'17)
    - Does not guarantee free service threads
- QoS at FEFS guarantees free service threads by limiting the number of threads to each pre-assigned group (client IP-address based or user-ID based) for fair-share utilization. ⇨ Suitable for operation of a huge scale of file systems

Summary

- We have investigated root-causes of (1) high MDS load and (2) quite slow MDS response at the K computer.
  - High MDS load was due to a large number of concurrent file accesses.
  - Quite slow MDS response was caused by a large number of concurrent file accesses under a large stripe count.
    - Large stripe count configuration caused congestion on associated OSSes.
    - Service threads at an MDS were unable to proceed new requests due to slow response from associated OSSes.
- QoS control at the MDS has been introduced to mitigate such MDS performance degradation.
  - I/O interference among user jobs has been mitigated under fair-share service thread allocation for each job.
  - Minimization in times for rank-directory creation has been achieved even if an another job which caused high MDS load was running at the same time.
  - QoS management performed very high impact in I/O interference alleviation.