Characterizing I/O Optimization Effect Through Holistic Log Data Analysis of Parallel File Systems and Interconnects

Yuichi TSUJITA[1], Yoshitaka FURUTANI[2], Hajime HIDA[3], Keiji YAMAMOTO[1], Atsuya UNO[1]

1. Center for Computational Science, RIKEN
2. FUJITSU Limited
3. Fujitsu Social Science Laboratory Limited

Outline

- Motivation
- Configuration of the K computer
- I/O Activity Analysis Framework at the K computer
- EARTH on K
- Experiments at the K computer
- Summary

Motivation

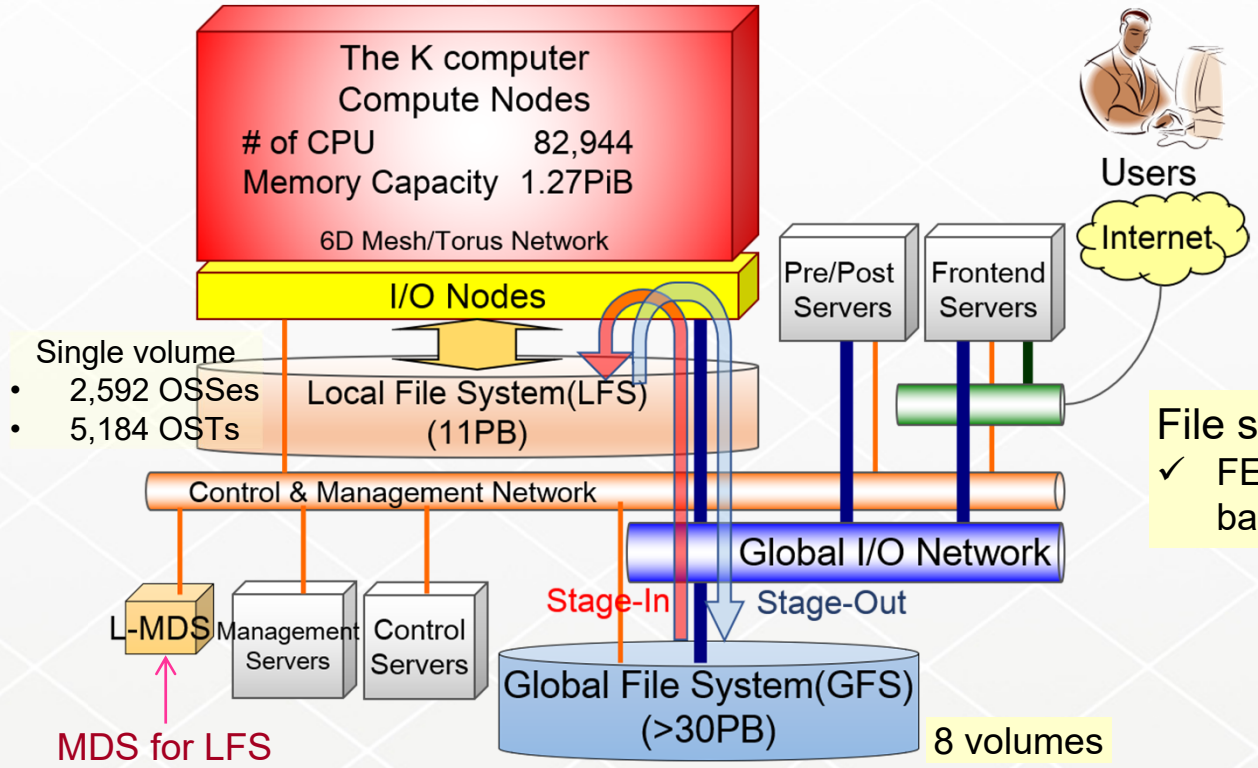
- Performance tuning/monitoring of I/O activities in the HPC systems
  - I/O operation is one of the bottlenecks in data-intensive applications.
  - At the K computer, I/O performance improvements have been studied, but it has been difficult to know the reason for the improvements.
    - Benchmark evaluation is a common way to know I/O performance values.
    - There are no tools to know I/O activities on file systems and data transfer status of interconnects among I/O nodes at user-side.
- Monitoring the following metrics is quite useful for I/O tuning.
  - I/O activities at the file systems
  - Packet data transfer status of interconnects among I/O nodes and file systems

Configuration of the K computer

K computer and its two-layered file systems

- System configuration of the K computer

Operated until Aug. 2019

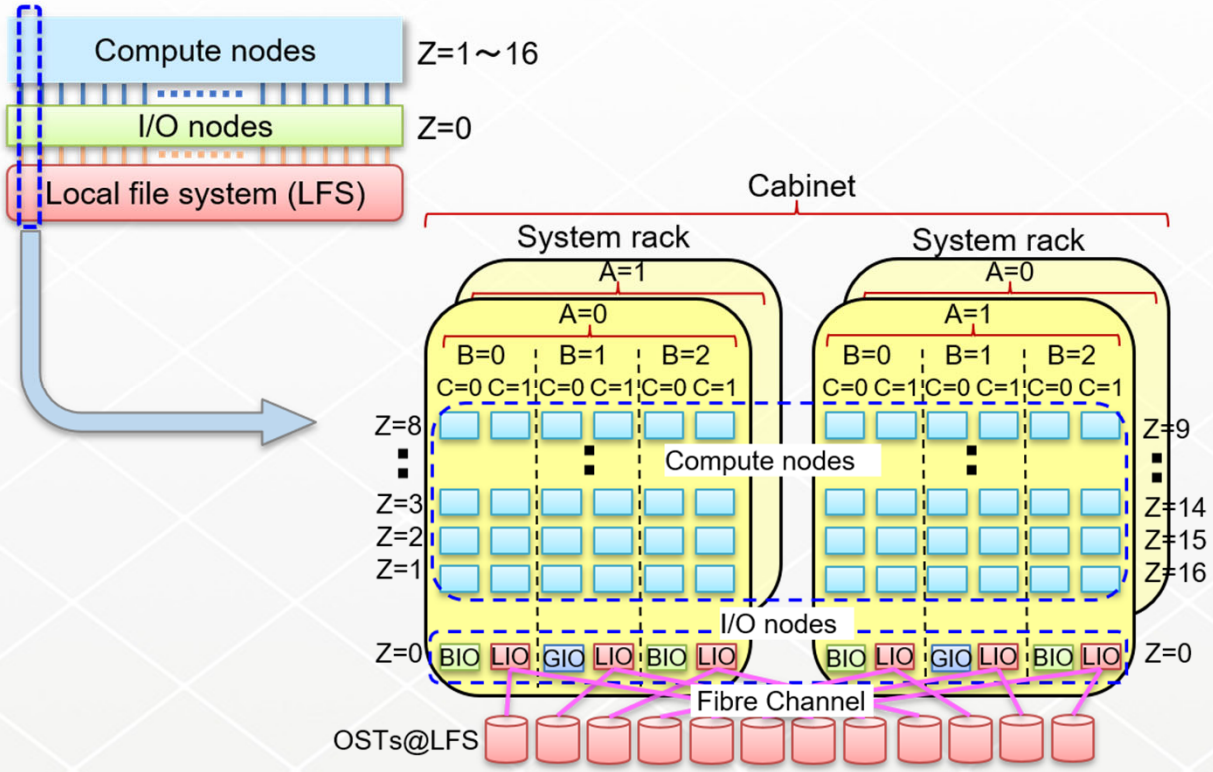


File systems (LFS and GFS)

✓ FEFS (Fujitsu Exabyte File System) based on Lustre ver.1.8

Compute nodes and system rack configuration

- System rack configuration in the Tofu 6D layout (X, Y, Z, A, B, and C)

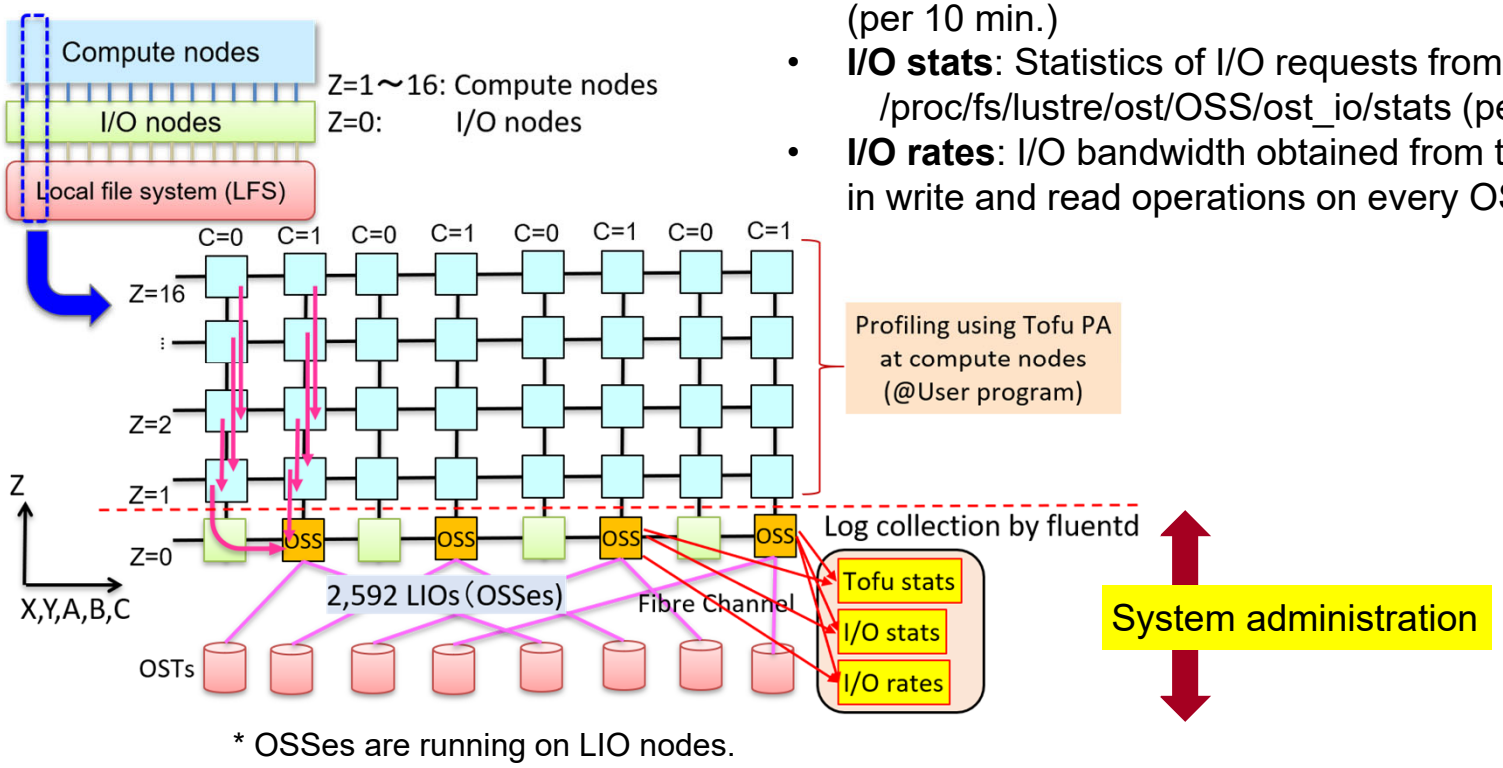


- One cabinet consists of
  - ✓ 192 compute nodes
  - ✓ 12 I/O nodes
- 12 OSTs associated with one cabinet

I/O activity analysis framework at the K computer

Log collection at the K computer

- Log collection from the I/O nodes (LIOs, GIOs, and BIOs)

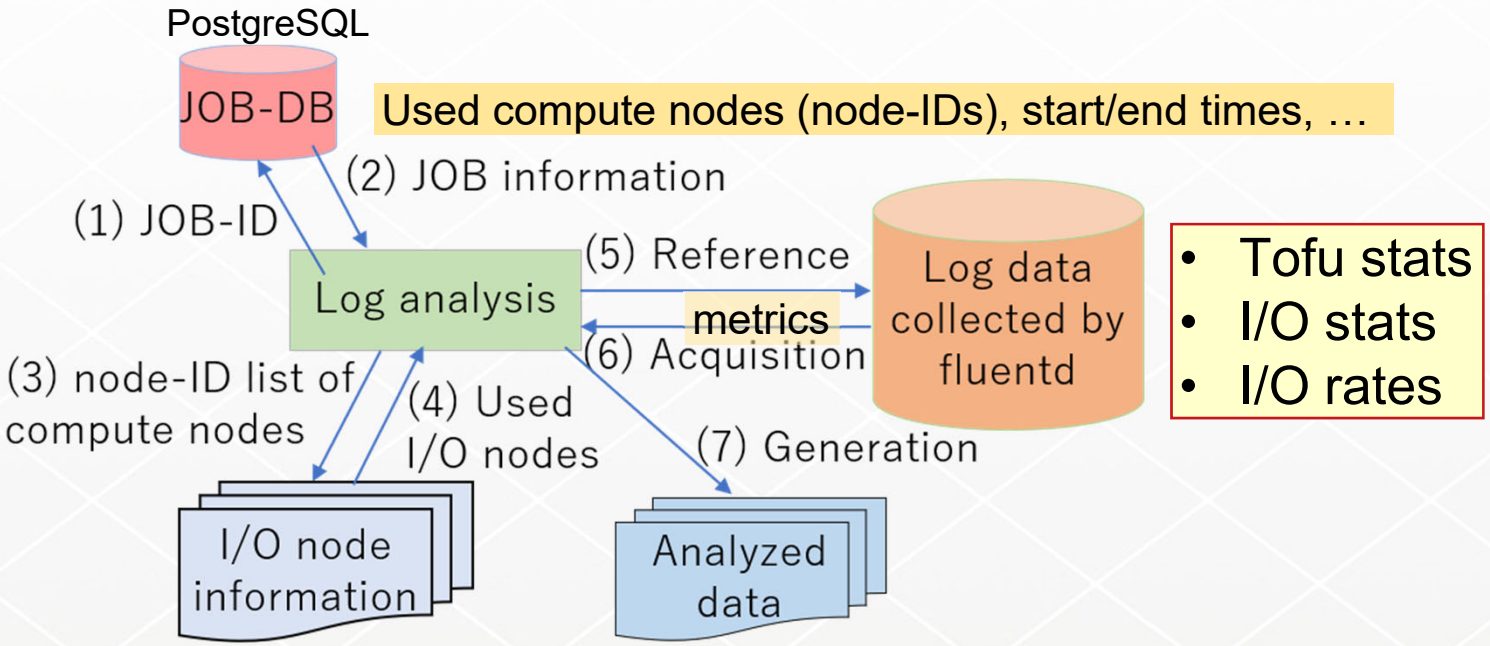


* OSSes are running on LIO nodes.

- **Tofu stats**: Data transfer status of I/O nodes on Tofu links (per 10 min.)
- **I/O stats**: Statistics of I/O requests from /proc/fs/lustre/ost/OSS/ost_io/stats (per 1 min.)
- **I/O rates**: I/O bandwidth obtained from the amount of sizes in write and read operations on every OST (per 10 min.)

Analysis framework

- Analysis framework using log data and job database



Tofu stats: metrics from Tofu interconnects

- Performance counters obtained from Tofu Network Router (TNR) on every I/O node
  - Additional deployment in the remaining few months of the operation
  - 10 minutes interval collection not to disturb I/O nodes (conservative way)
    - ✓ Around two months experimental monitoring until the end of the K computer operation
  - ➢ Cycle counts until target transfer buffer is available (zero-credit count)
    - Cycle counts ➡ Waiting time (Congestion status)
  - ➢ Amount of transferred data size
    - Data transfer bandwidth: *(transfer size) / (monitoring interval)*

    - Bandwidth utilization ratio: *(data transfer bandwidth) / (theoretical bandwidth)*

I/O stats: metrics collected from OSSes

- The following three metrics from /proc/fs/lustre/ost/OSS/ost_io/stats
  - 1 minute interval collection

1. *req_qdepth* ➡ Congestion status
   - Lower number is preferable because high number represents I/O request congestion.
2. *req_waittime* ➡ Congestion status
   - Lower number is preferable because high number indicates I/O request congestion.
3. *req_active* (number of active I/O threads at OSS) ➡ Activities
   - High number is preferable because of high activity of I/O threads.

I/O rates: I/O bandwidth at OSTs

- Monitoring write and read bandwidth at OSTs
  - Amount of size in write/read per 10 minutes from log data collected by fluentd

  - Bandwidth in write/read at each used OST in each 10 minutes interval

  - Heatmap generation in the PNG files from the calculated bandwidth values in 2D layout
    - ✓ 2D layout corresponds to locations of cabinets and OSTs
    - ✓ Easily find I/O bandwidth distribution and unbalanced I/O bandwidth situation

EARTH on K: Enhanced MPI-IO (ROMIO) at the K computer

EARTH on K (EARTH) *

- Enhanced two-phase I/O in ROMIO at the K computer
  1. agg: Striping-aware aggregator layout
  2. rr: Round-robin aggregator layout among compute nodes
  3. req: I/O throttling and associated stepwise data aggregation with a given number of I/O requests per step (e.g., req=4 indicates 4 requests per step on each OST.)
  - Remaining issues
    - A combination of the above parameters outperforms the original MPI-IO at the K computer, however, it has been quite difficult to find the reason for the improvements.
    - It has been difficult to find good parameter configuration only through empirical benchmark evaluations.

* Y. Tsujita et al., “Improving Collective MPI-IO Using Topology-Aware Stepwise Data Aggregation with I/O Throttling” (HPC Asia’18)

I/O characterization by the analysis framework

- Providing evidences why the EARTH outperformed the original MPI-IO in terms of activities in associated system components
  - ✓ File system, I/O nodes, Interconnects (Tofu), …

- The proposed analysis framework shows
  - different activities in I/O operations among the two implementations
  - evidences why the EARTH improves I/O performance

Experiments at the K computer

Evaluation at the K computer

- Performance evaluation of collective MPI-IO using
  - Original MPI-IO implementation at the K computer (Original)
  - Enhanced MPI-IO implementation named EARTH on K (EARTH)
- Used I/O benchmarks
  - IOR
  - HPIO
- Comparisons in the evaluations
  - Averaged values of *req_qdepth*, *req_waittime*, and *req_active*
  - Bandwidth utilization and waiting time of the used Tofu links among I/O nodes
  - I/O throughput and load-balancing among OSTs

Benchmark configuration (IOR and HPIO)

- Benchmark run by 12,288 processes on 3,072 compute nodes
  - Compute node layout: 8x12x32 ➡ 192 OSTs (96 LIOs)
  - 3,072 processes were assigned as aggregators
- IOR

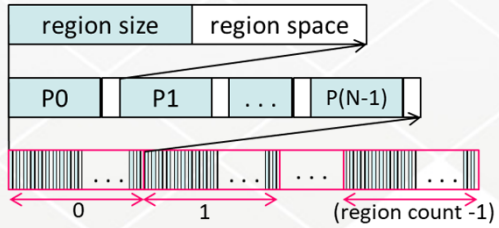
```
$ ior -i 5 -a MPIIO -c -U hints_info -k -m -vvv -w -t 256m -b 256m ¥
    -o ${TARGET_DIR}/test-IOR.dat -d 0.1
```

  - 256 MiB in stripe size, 192 in stripe count
  - 5 iterations in write, followed by the same iterations in read
  - Transfer size / Block size: 256 MiB ➡ 3 TiB per file per iteration
- HPIO

```
$ hpio -n 0010 -r 6 -B -s 5992 -c 30729 -p 256 -m 01 -O 11 -f 0 ¥
    -S 0 -a 0 -g 2 -H cb_config_list=*:4 -d ${TARGET_DIR} -w 1
```

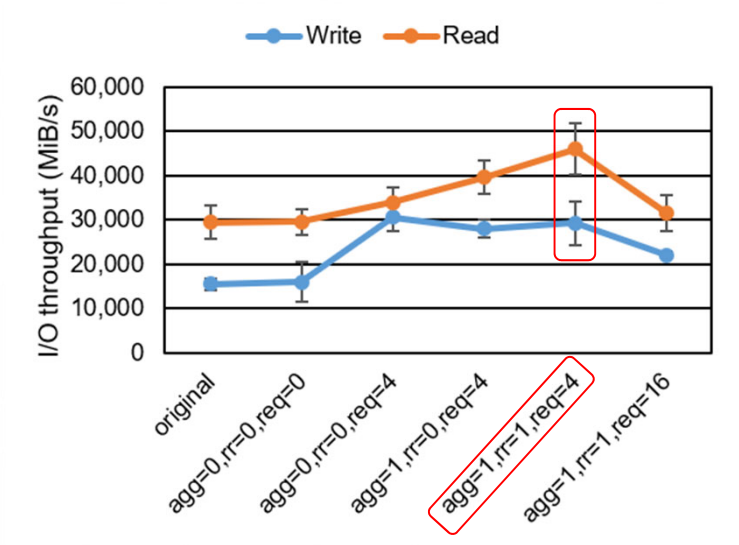
  - 64 MiB in stripe size, 192 in stripe count
  - 6 iterations in write, followed by the same iterations in read
  - Region size: 5,992 B, Region space: 256 B, Region count: 30,729

    ➡ ~ 2.1 TiB per file per iteration

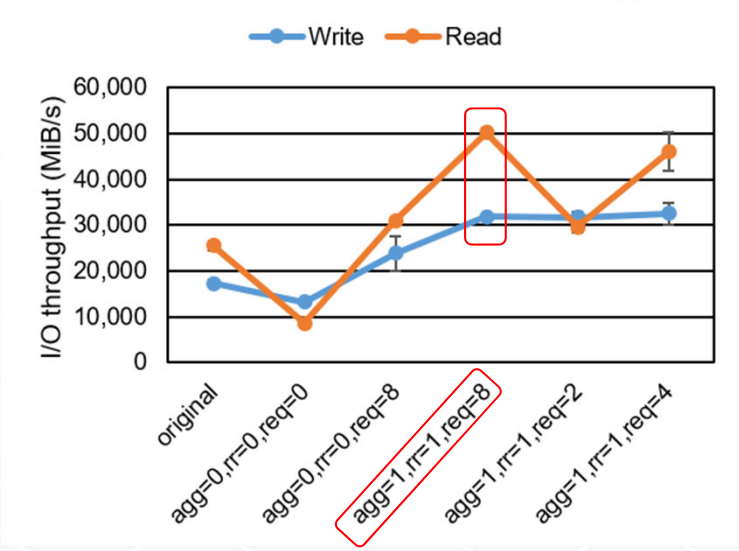


Benchmark results

- I/O bandwidth results



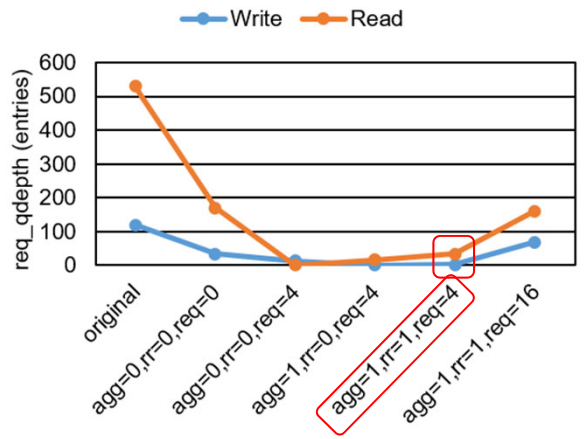
IOR

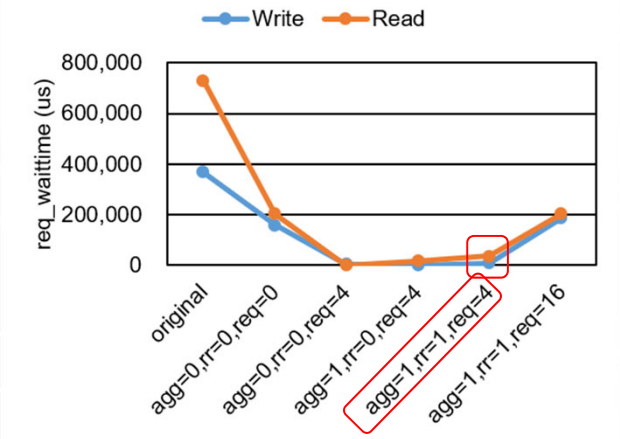


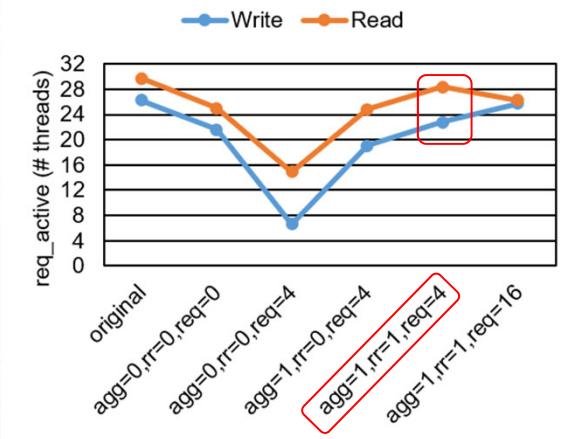
HPIO

I/O stats of OSSes (IOR)

- Averaged values of *req_qdepth*, *req_waittime*, and *req_active*







Lesser congestion

Shorter waiting time

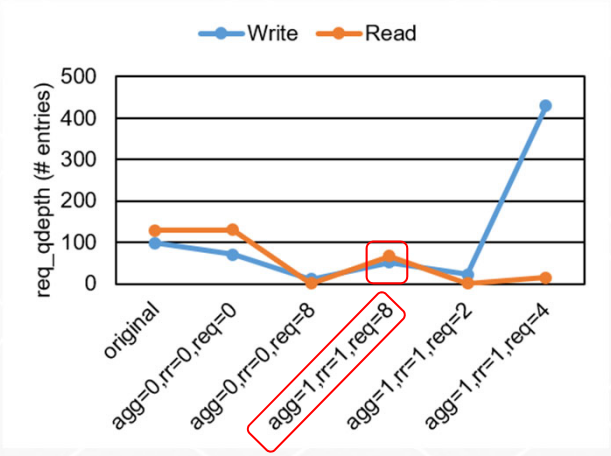
Higher number of active I/O threads

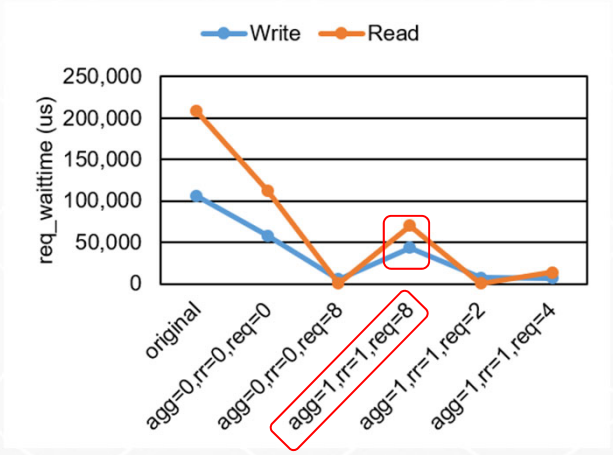
Lower is better

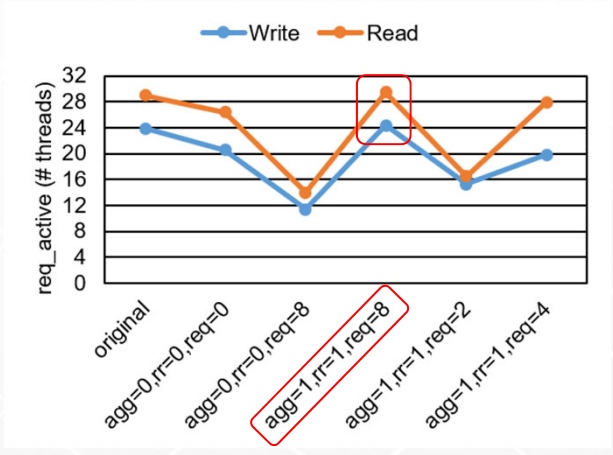
Higher is better

I/O stats of OSSes (HPIO)

- Averaged values of *req_qdepth*, *req_waittime*, and *req_active*







Lesser congestion

Shorter waiting time

Lower is better

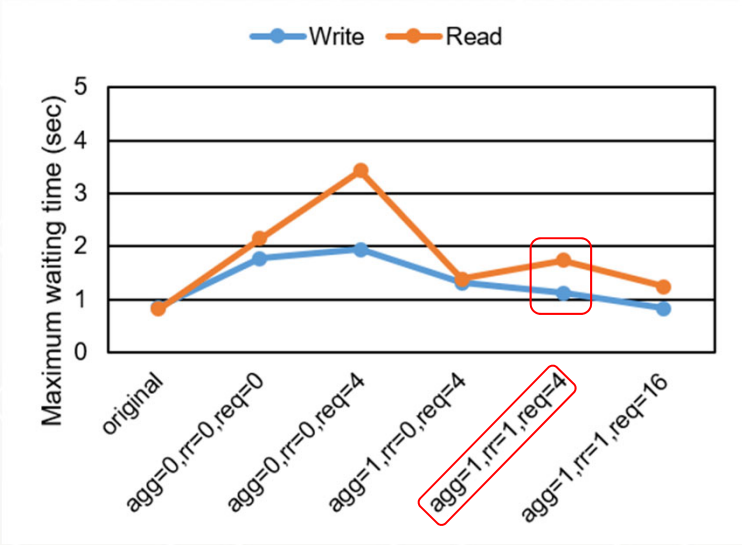
Higher number of active I/O threads

Higher is better

Status of Tofu links among I/O nodes (IOR)

- Averaged values of
  - peak bandwidth utilization
  - maximum waiting time to start data transfer

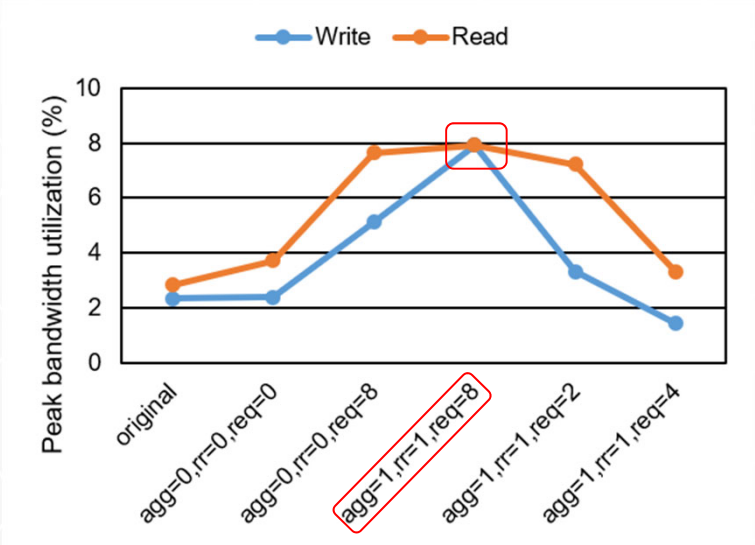
High utilization



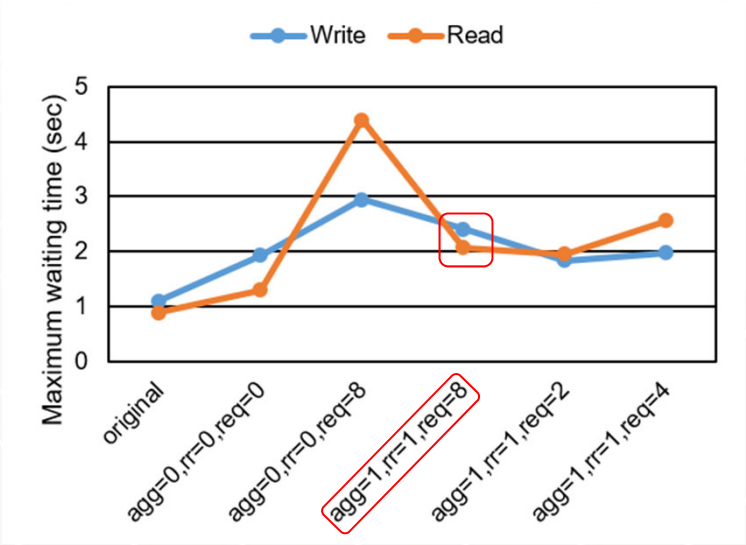
Shorter waiting time

Status of Tofu links among I/O nodes (HPIO)

- Averaged values of
  - peak bandwidth utilization
  - maximum waiting time to start data transfer



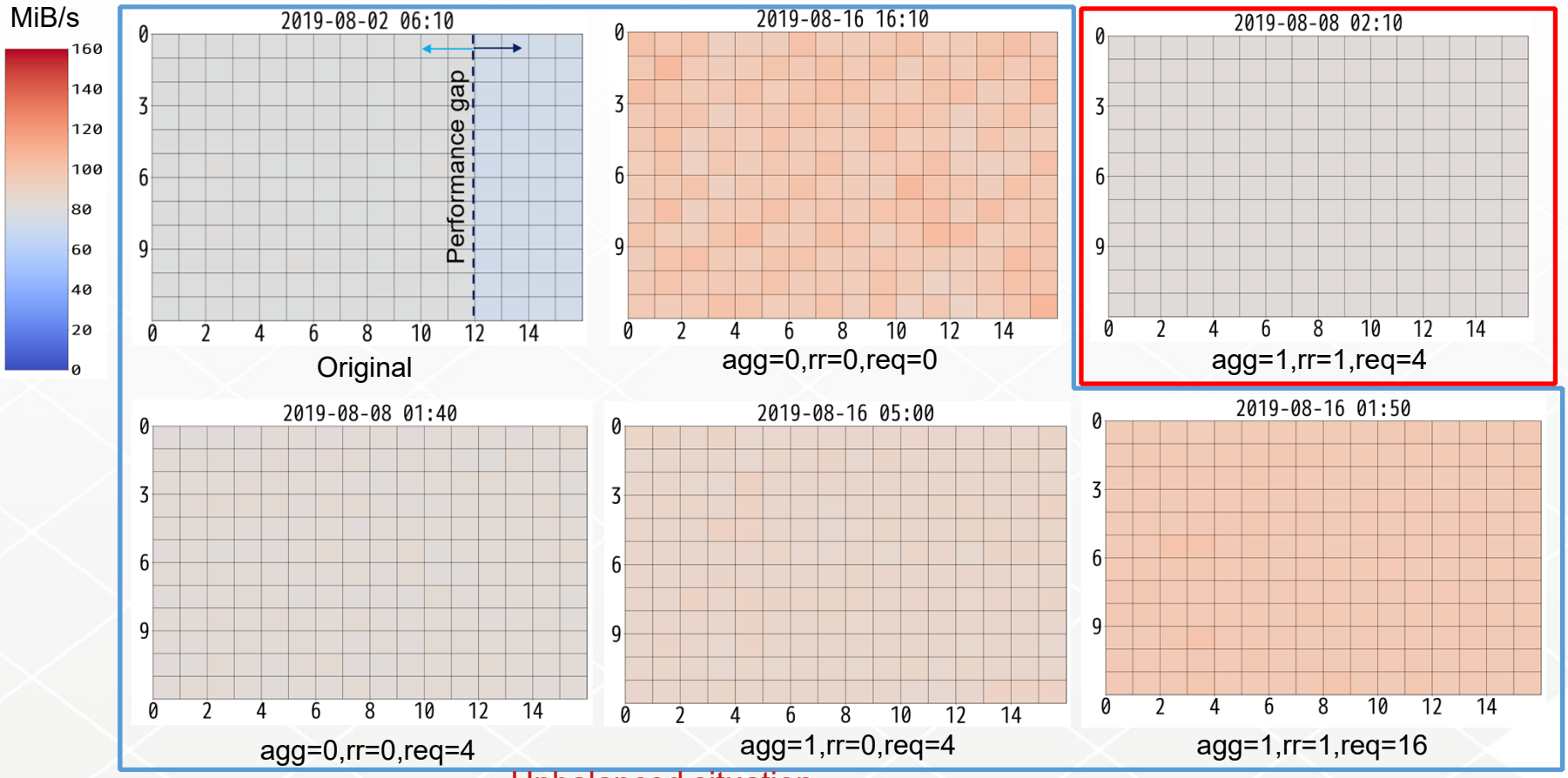
Highest utilization



Shorter waiting time

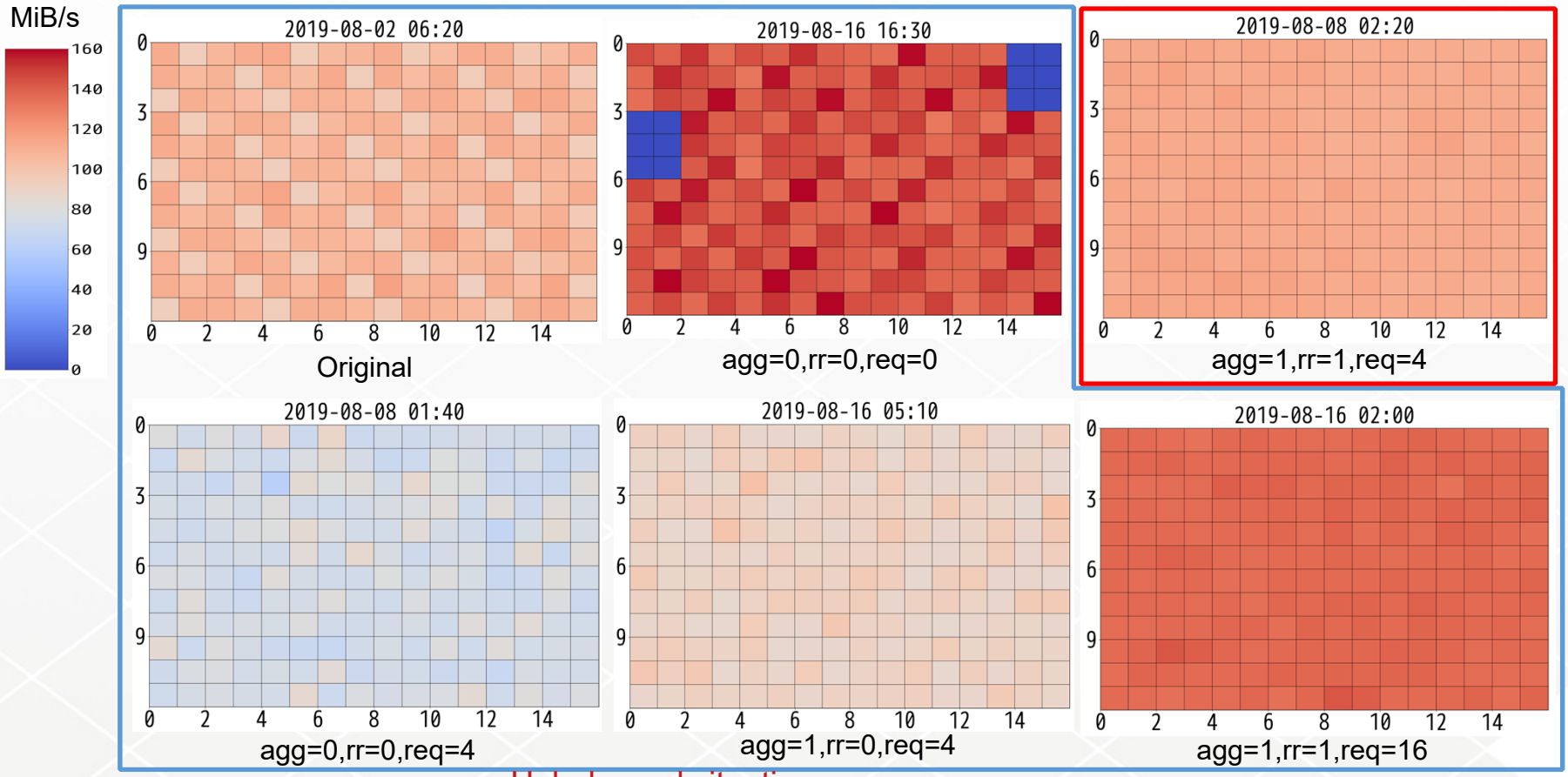
I/O rates from used OSTs (IOR)

- Write bandwidth heatmaps among used OSTs



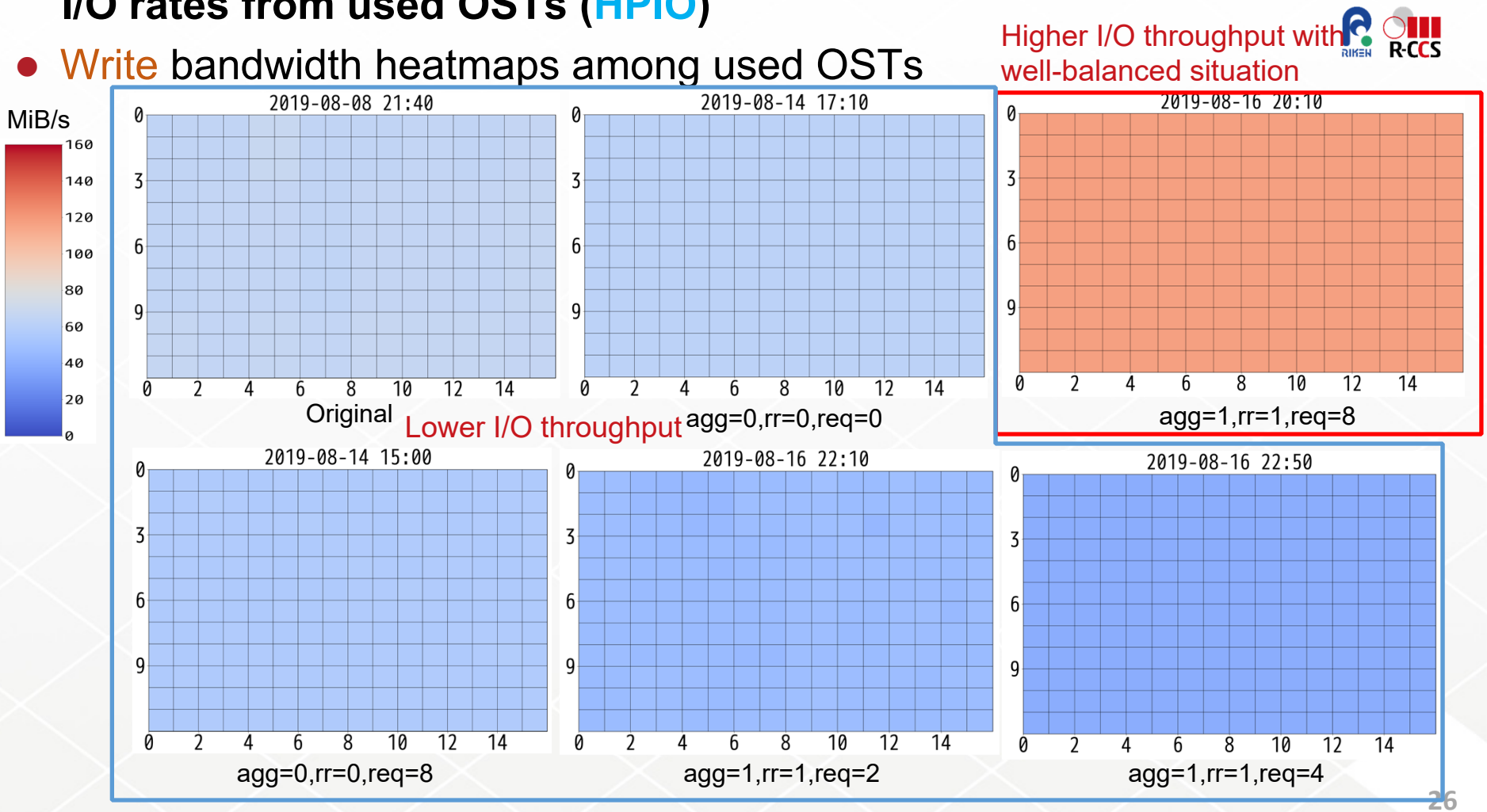
I/O rates from used OSTs (IOR)

- Read bandwidth heatmaps among used OSTs



I/O rates from used OSTs (HPIO)

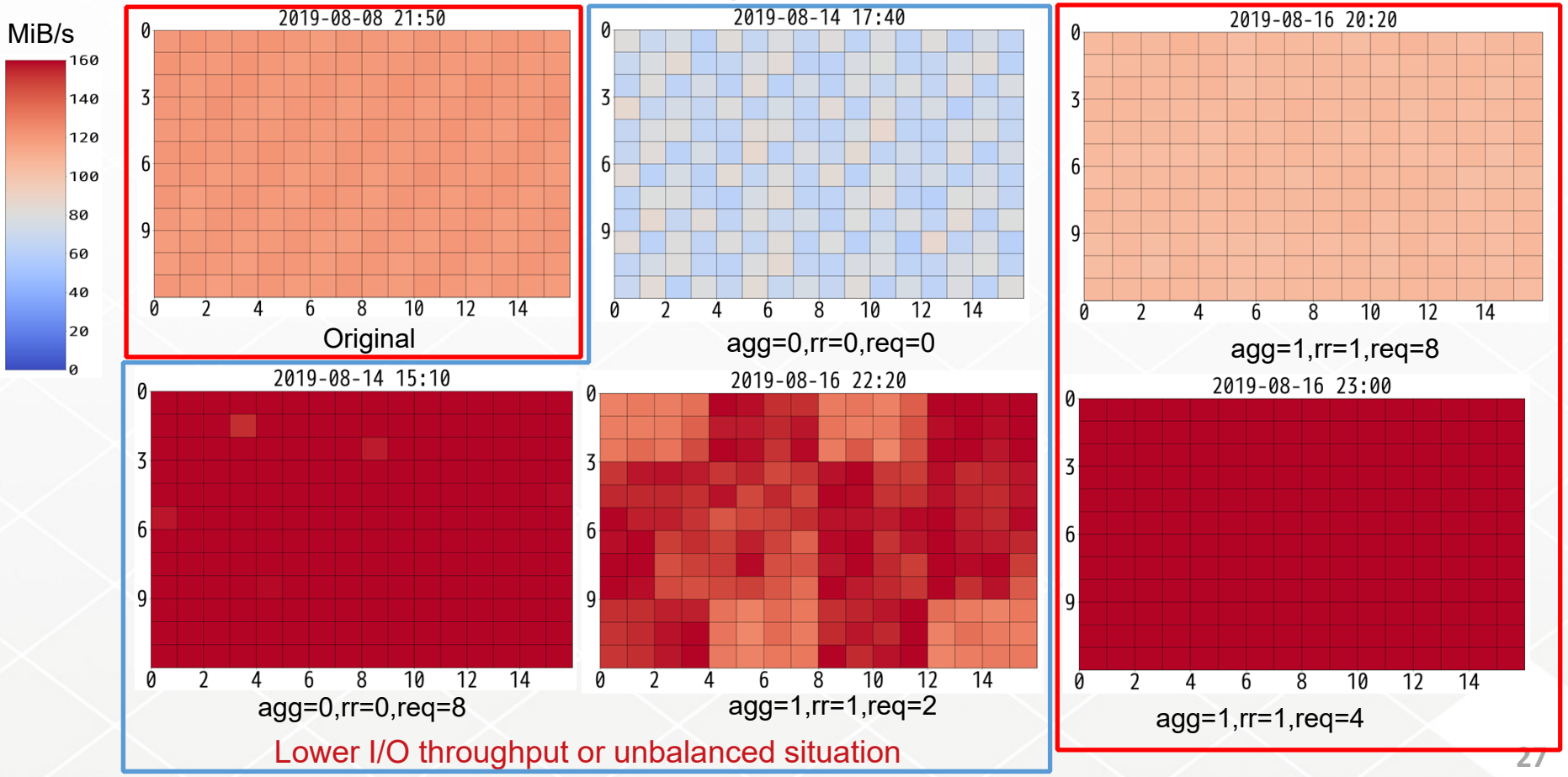
- Write bandwidth heatmaps among used OSTs



I/O rates from used OSTs (HPIO)

- Read bandwidth heatmaps among used OSTs

Higher I/O throughput with well-balanced situation



Characterization of I/O (IOR)

- Scoring summary from profiled data

I/O case	I/O stats (ranks from 1)			Tofu stats (ranks from 1)		Stats SCORE (lesser is better)	I/O rates at OSTs	
	req_qdepth	req_waittime	req_active	BW util.	waiting time		write	read
Original	6	6	1	6	1	**4**	Unbalanced	Unbalanced
agg=0,rr=0, req=0	4	4	4	5	5	**4.4**	Unbalanced	Unbalanced
agg=0,rr=0, req=4	1	1	6	4	6	**3.6**	Unbalanced	Unbalanced
agg=1,rr=0, req=4	2	2	5	2	3	**2.4**	Unbalanced	Unbalanced
agg=1,rr=1, req=4	3	3	3	1	4	**2.4**	Balanced	Balanced
agg=1,rr=1, req=16	5	5	2	3	2	**3.4**	Unbalanced	Unbalanced

The lower score number with balanced I/O at OSTs at “agg=1,rr=1,req=4” shows the good parameter setting.

Characterization of I/O (HPIO)

- Scoring summary from profiled data

I/O case	I/O stats (ranks from 1)			Tofu stats (ranks from 1)		Stats SCORE (lesser is better)	I/O rates at OSTs	
	req_qdepth	req_waittime	req_active	BW util.	waiting time		write	read
Original	**5**	**6**	**2**	**5**	**1**	**3.8**	Low	Balanced
agg=0,rr=0, req=0	**4**	**5**	**4**	**4**	**2**	**3.8**	Low	Unbalanced
agg=0,rr=0, req=8	**1**	**1**	**6**	**2**	**6**	**3.2**	Low	High
agg=1,rr=1, req=8	**3**	**4**	**1**	**1**	**4**	**2.6**	Balanced	Balanced
agg=1,rr=1, req=2	**2**	**2**	**5**	**3**	**3**	**3.0**	Low	Unbalanced
agg=1,rr=1, req=4	**6**	**3**	**3**	**6**	**5**	**4.6**	Low	High

The lower score number with balanced I/O at OSTs at “agg=1,rr=1,req=8” shows the good parameter setting.

Summary

- For I/O characterization, we have implemented an analysis framework to use the three groups of log data in conjunction with a database storing job information at the K computer.
  - I/O stats of OSSes obtained from /proc/fs/lustre/ost/OSS/ost_io/stats
  - Tofu stats (bandwidth utilization and waiting time in each link at used I/O nodes)
  - I/O rates at used OSTs
- The analysis framework showed improved activities at file systems, interconnects, and OSTs associated with the improvements by the enhanced MPI-IO named EARTH on K in benchmark runs.
- Similar approach will be done on our next HPC system based on experiences through this work with the following improvements.
  - More fine-grained monitoring to support detailed analysis
  - Sophisticated database organization for effective utilization of metrics
  - Scoring scheme to evaluate I/O throughputs at OSTs