

HPC-IODC (July 16, 2015, Frankfurt, Germany)

Activities Towards High Availability of Parallel I/O at the K computer

Yuichi Tsujita^{1,2}, Fumiyoshi Shoji¹, Atsushi Hori^{1,2},
Atsuya Uno¹, Keiji Yamamoto¹, Toyohisa Kameyama^{1,2}, Yutaka Ishikawa¹

1. RIKEN AICS
2. JST CREST

Outline

1. Overview of the K computer
2. R&D Status of High Performance MPI-IO on the K computer
3. FEFS Operation Status
4. Summary

Overview of the K computer

System Configuration of the K computer

40 m x 40 m

Full System

Compute Rack × 864



4000mm x 800mm

2 Cabinets

Compute Rack × 4

Disk Racks × 1



10.6(11.3)PFLOPS

1.27(1.34)PiB

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)

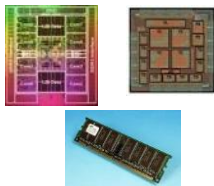
Node × 4



512GFLOPS

64GiB

Node
CPU × 1
ICC × 1
memory



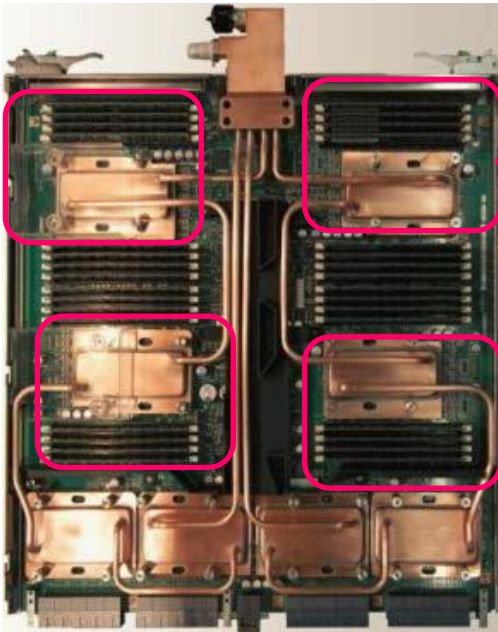
128GFLOPS

16GiB

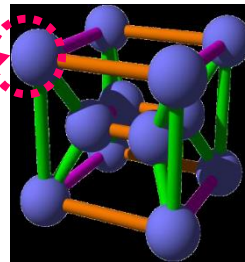
() included IO node performance and memory capacity

Tofu Interconnect

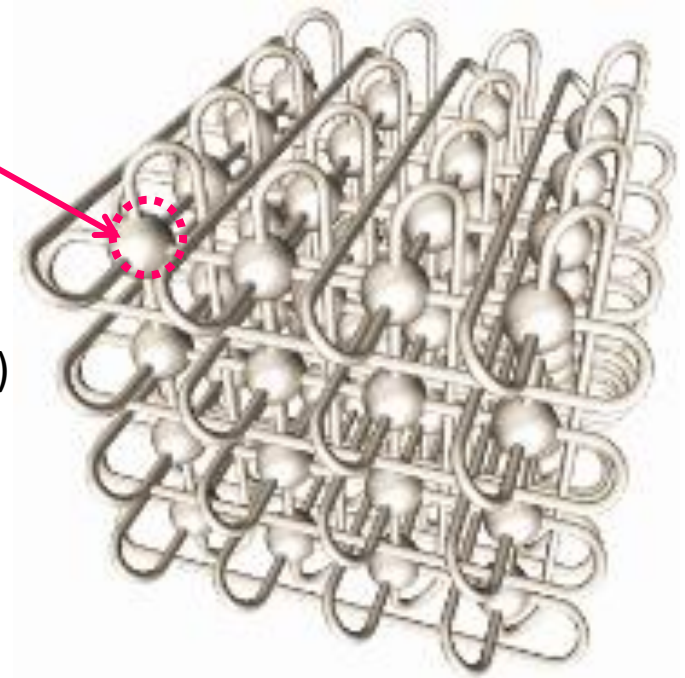
- Computing nodes with the Tofu interconnect
 - Tofu : **T**orus **F**usion



1 system board
(4 computing nodes)



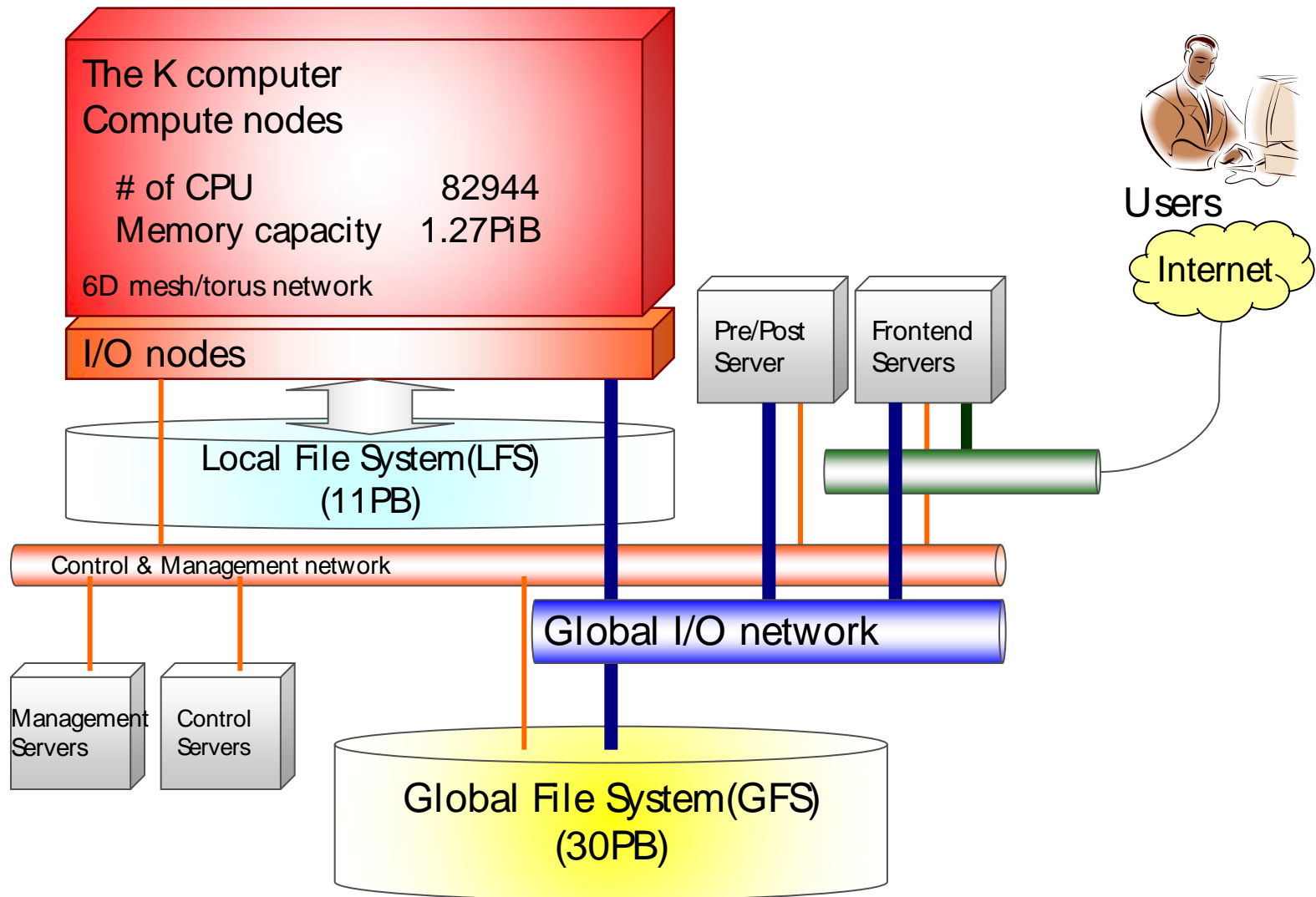
1 Tofu-unit
(2x3x2 nodes)



6D mesh/torus using the Tofu interconnect

- Axis : X, Y, Z, a, b, c
- X,Z,b : torus (Z=0: I/O node), Y, a, c : Mesh
- Network size : (X, Y, Z, a, b, c) = (24, 18, 17, 2, 3, 2)

Hardware Configuration of the K computer



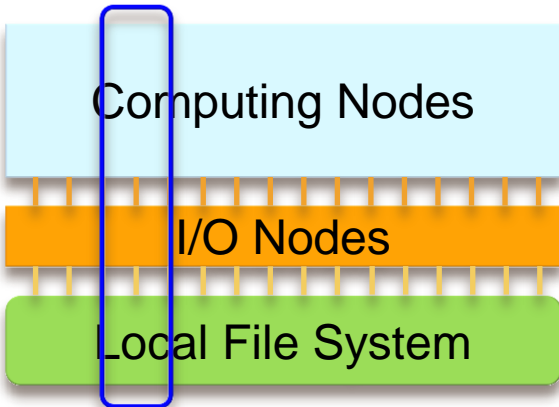
LFS: Performance oriented

GFS: Capacity and reliability oriented

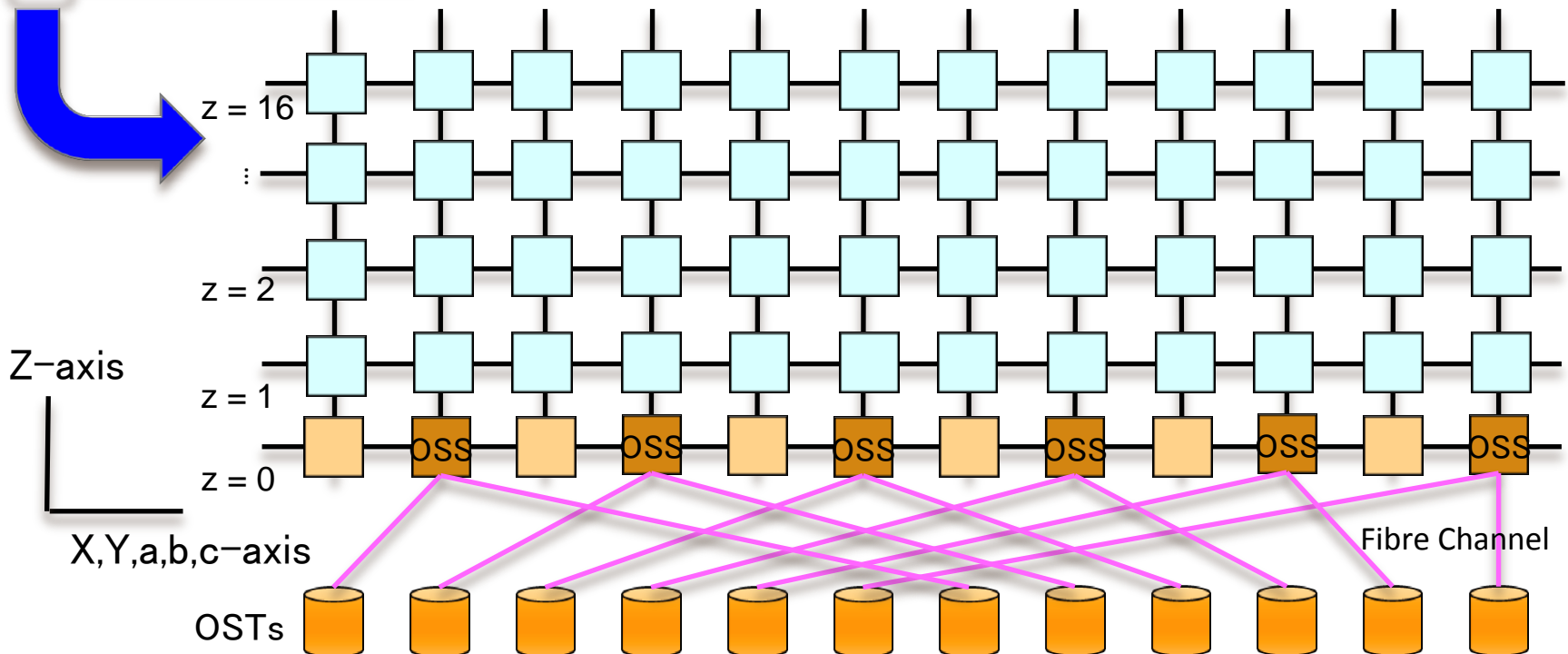
FEFS(LFS) and Tofu in the K computer

- Network Configuration of a Local File System (LFS)

* FEFS : Fujitsu Exabyte File System developed by Fujitsu based on Lustre technology



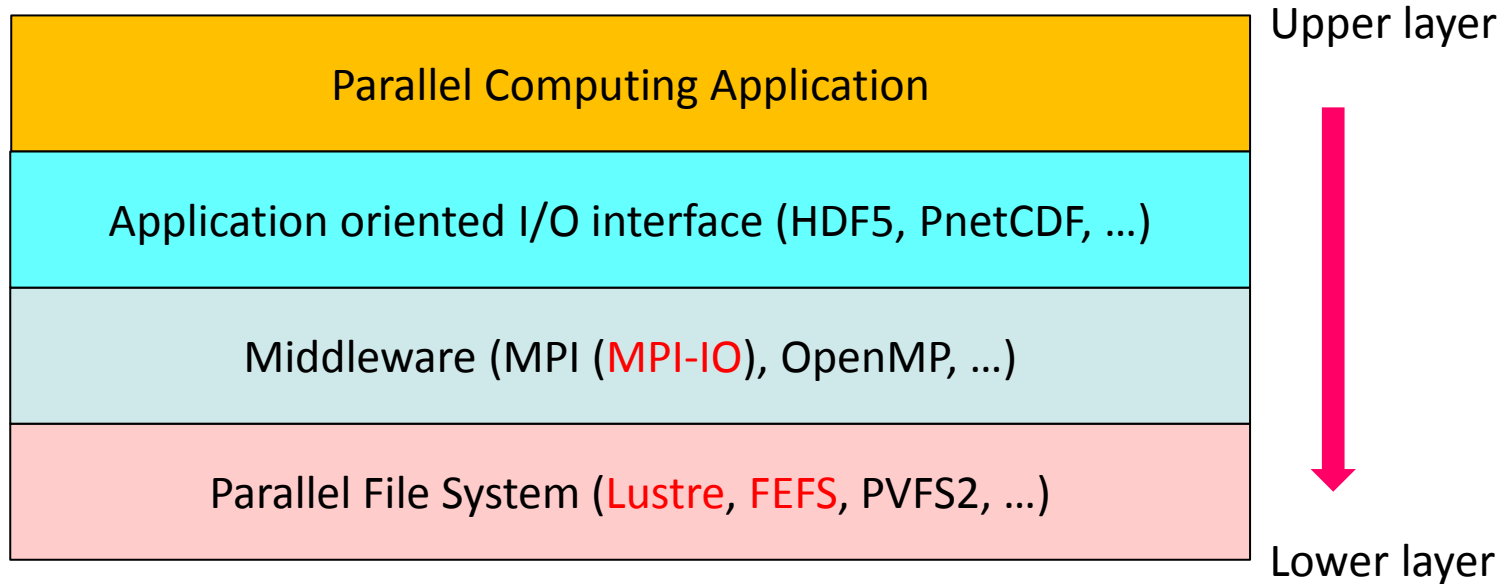
- I/O nodes are located on $z=0$ of z -axis.
- Every OSTs are accessible from I/O nodes via Fibre Channel interconnects
- Accessing a target OST through an OSS on an I/O node





R&D Status of High Performance MPI-IO on the K computer

Software Stack of Parallel I/O

- Our focus on parallel I/O

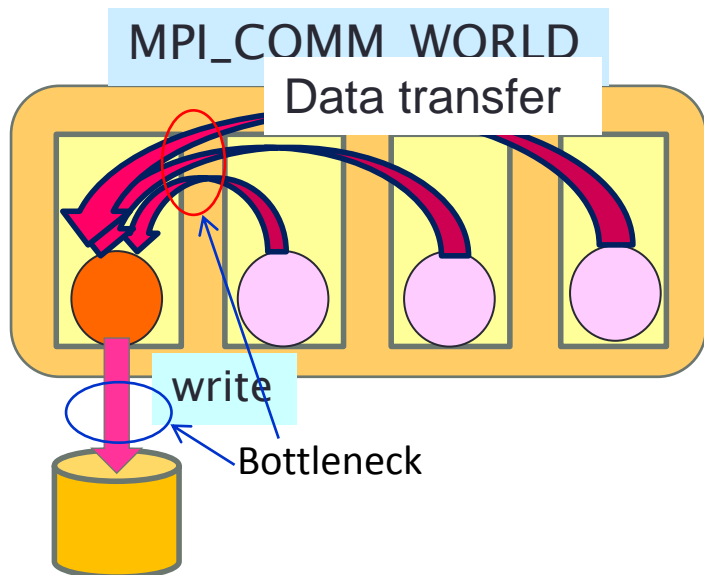


MPI-IO on the K computer

- HDF5, PnetCDF
 - Widely used application oriented I/O libraries (HDF5, PnetCDF, ...)
 - MPI-IO is used in underlying parallel I/O layer
 - MPI-IO performance improvement is essential to cope with a huge scale of data management.
- 
- Performance Improvement of MPI-IO on the K computer
 - Optimization of an MPI-IO implementation named ROMIO
- 
- Affinity-aware optimizations for FEFS and Tofu interconnect
 - Optimizations suitable for FEFS striping layout with topology-awareness for Tofu interconnect

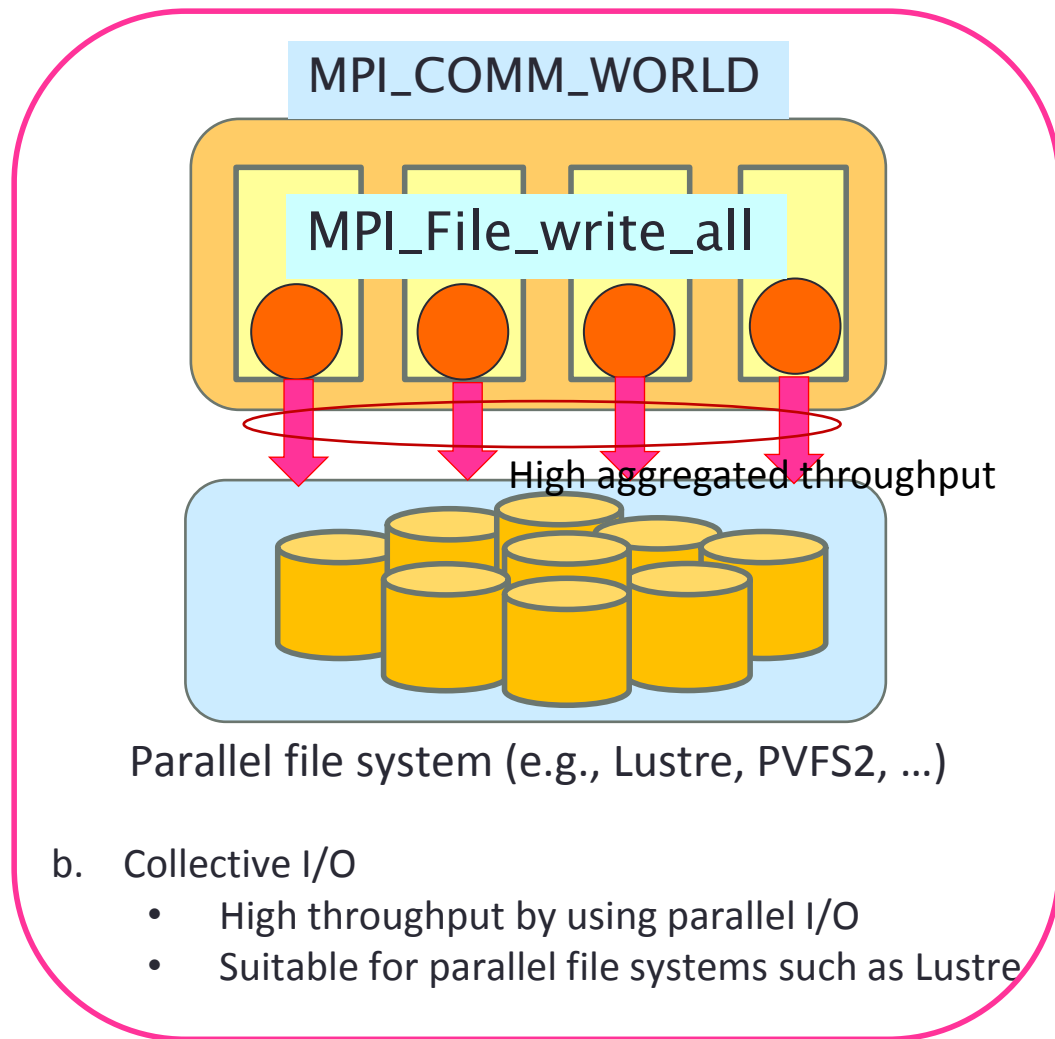
Collective MPI-IO Using Lustre

- Collective I/O



a. Independent I/O

- Large overhead in data communications
- Bottleneck in the representative process to perform local I/O



b. Collective I/O

- High throughput by using parallel I/O
- Suitable for parallel file systems such as Lustre

Target Collective MPI-IO Access Pattern

- Typical MPI-IO access patterns

- APIs are based on the MPI standard.

```
MPI_File_open(..., "filename", ..., &fh)
for (i=0; i<n_local_rows; i++) {
    MPI_File_seek(fh, ...)
    MPI_File_read(fh, row[i], ...)
}
MPI_File_close(&fh)
```

Level 0
(many independent, contiguous requests)

```
MPI_File_open(MPI_COMM_WORLD, "filename", ..., &fh)
for (i=0; i<n_local_rows; i++) {
    MPI_File_seek(fh, ...)
    MPI_File_read_all(fh, row[i], ...)
}
MPI_File_close(&fh)
```

Level 1
(many collective, contiguous requests)

```
MPI_Type_create_subarray(..., &subarray, ...)
MPI_Type_commit(&subarray)
MPI_File_open(..., "filename", ..., &fh)
MPI_File_set_view(fh, ..., subarray, ...)
MPI_File_read(fh, local_array, ...)
MPI_File_close(&fh)
```

Level 2
(single independent, noncontiguous request)

```
MPI_Type_create_subarray(..., &subarray, ...)
MPI_Type_commit(&subarray)
MPI_File_open(MPI_COMM_WORLD, "filename", ..., &fh)
MPI_File_set_view(fh, ..., subarray, ...)
MPI_File_read_all(fh, local_array, ...)
MPI_File_close(&fh)
```

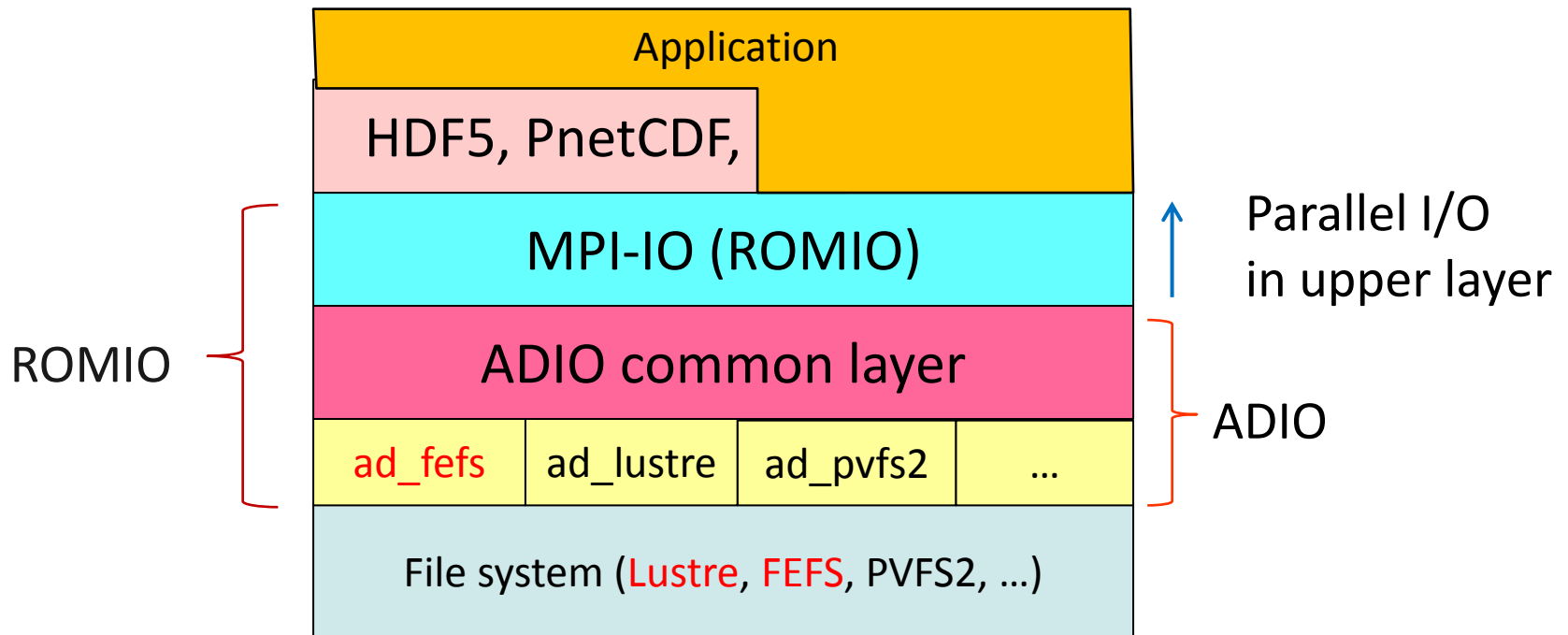
Level 3
(single collective, noncontiguous request)

We are focusing on this pattern.

Software Stack of ROMIO

- ROMIO

- Widely used MPI-IO implementation available in MPICH or OpenMPI
- supports many kinds of file systems such as Lustre, FEFS, PVFS2, and so forth

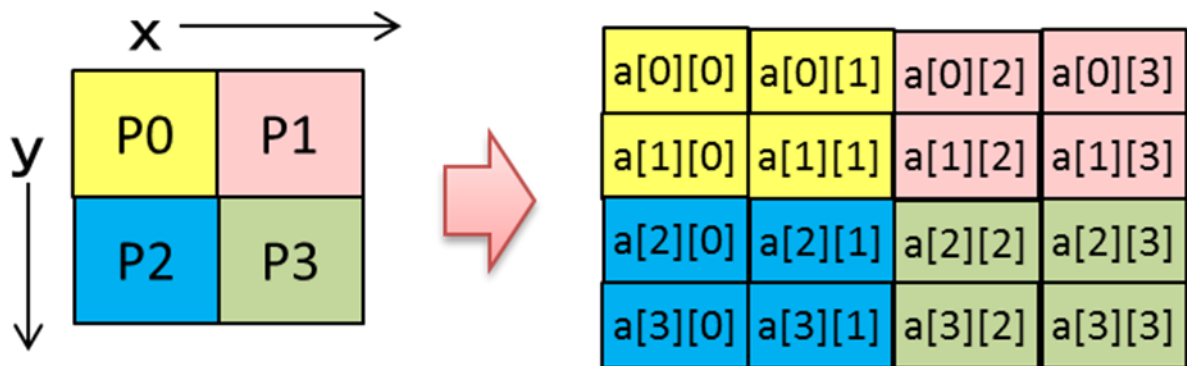


Optimizations for FEFS

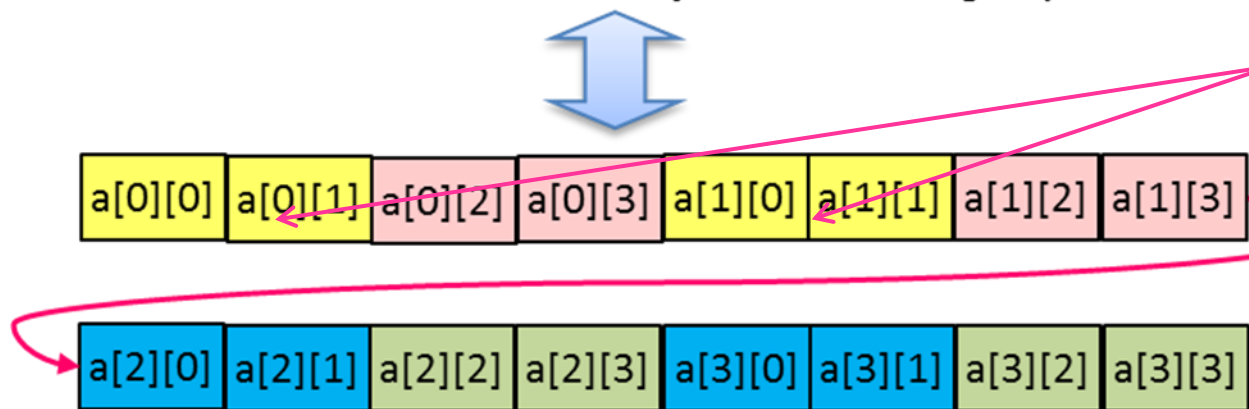
- FJMPI (based on OpenMPI)
 - with Fujitsu's enhanced implementations suitable for the K computer
 - MPI-IO is available by using ROMIO enhanced for FEFS.
 - ADIO for FEFS supports file accesses to FEFS.

2-D Array Data Accesses in Parallel I/O

- Example of I/O accesses for two-dimensional data



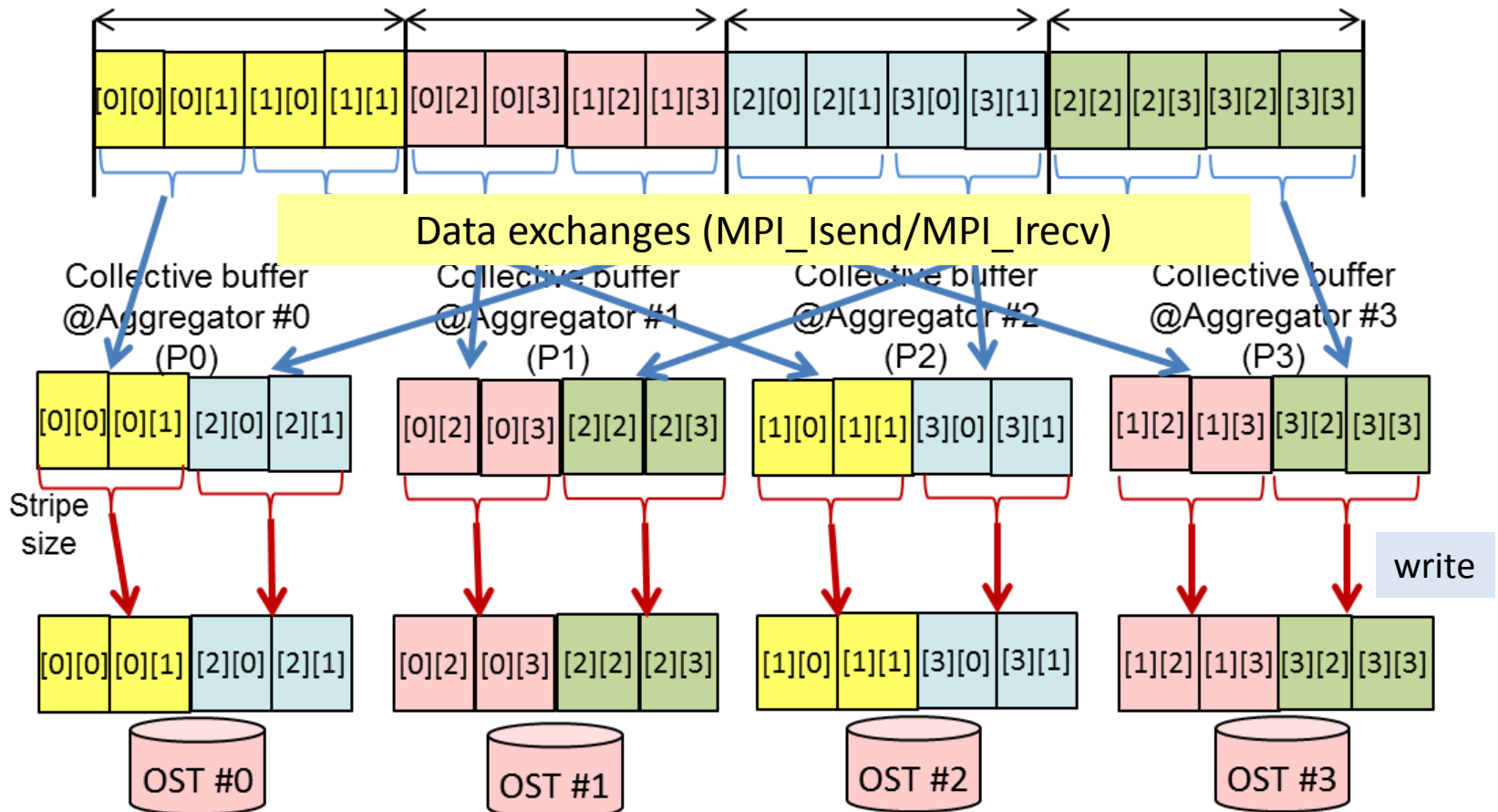
Global view of a 2-D array data among 4 processes



File view of a 2-D array data among 4 processes

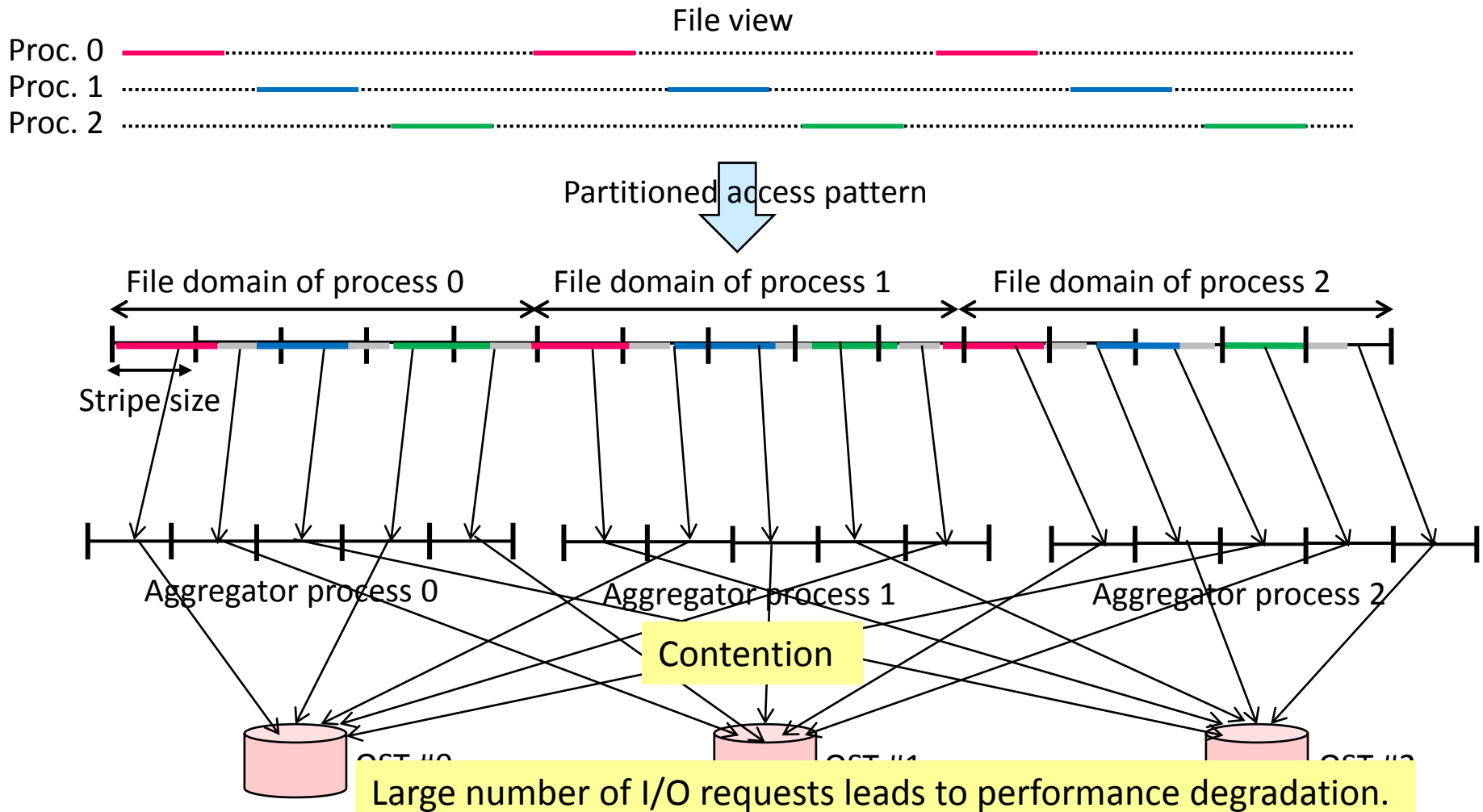
Two-Phase I/O

- Two-phase I/O (TP-IO) in collective write for non-contiguous file accesses
 - Every file access is aligned to a collective buffer (hereinafter, CB).
 - Repetitions of file accesses and data exchanges



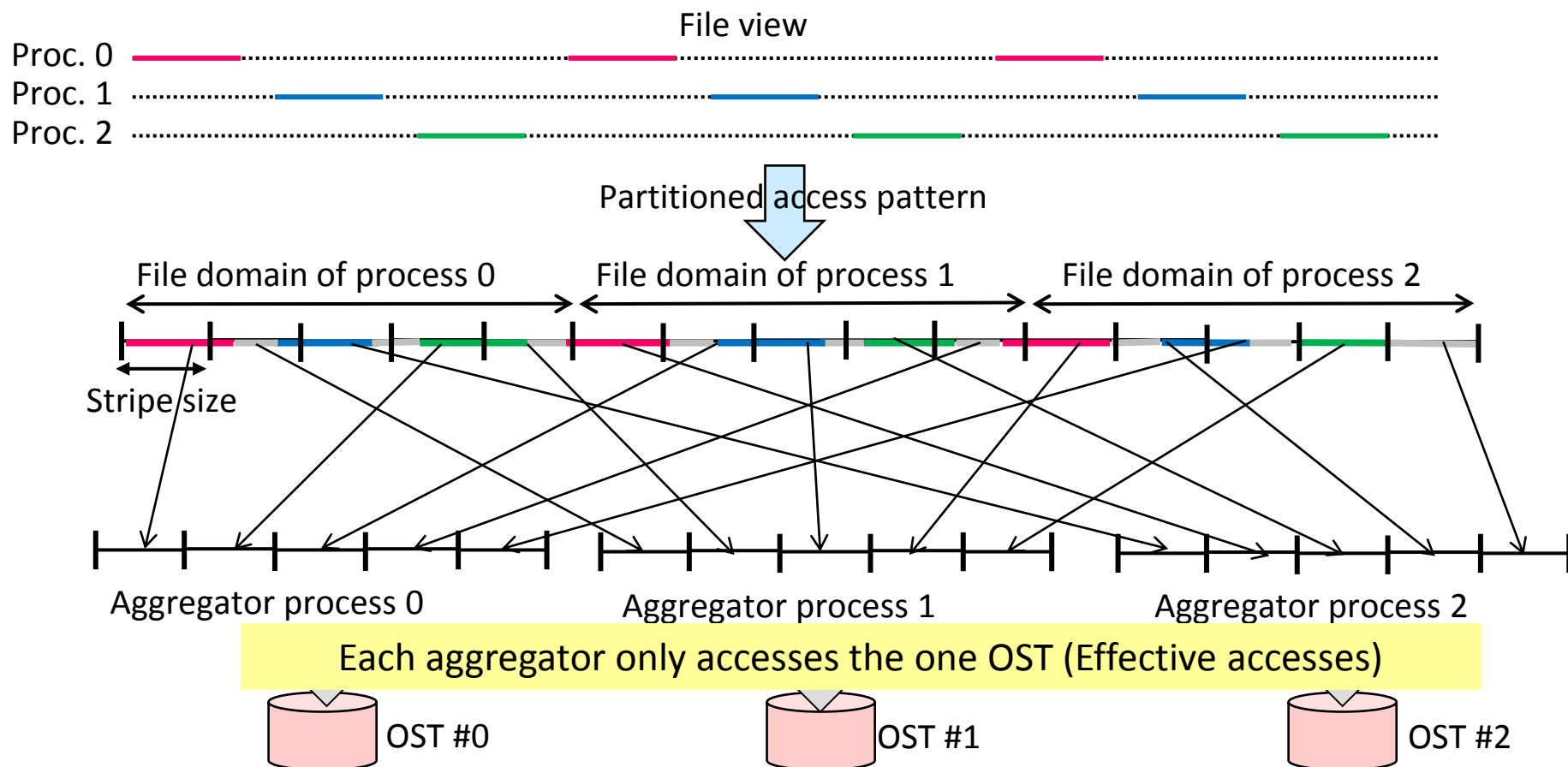
Current ADIO for FEFS

- Previous FJMPI (until Feb. 2015) had the following problems.
 - ✓ Network contention among aggregator processes and OSTs
 - ✓ Performance degradation due to many I/O requests on each OST



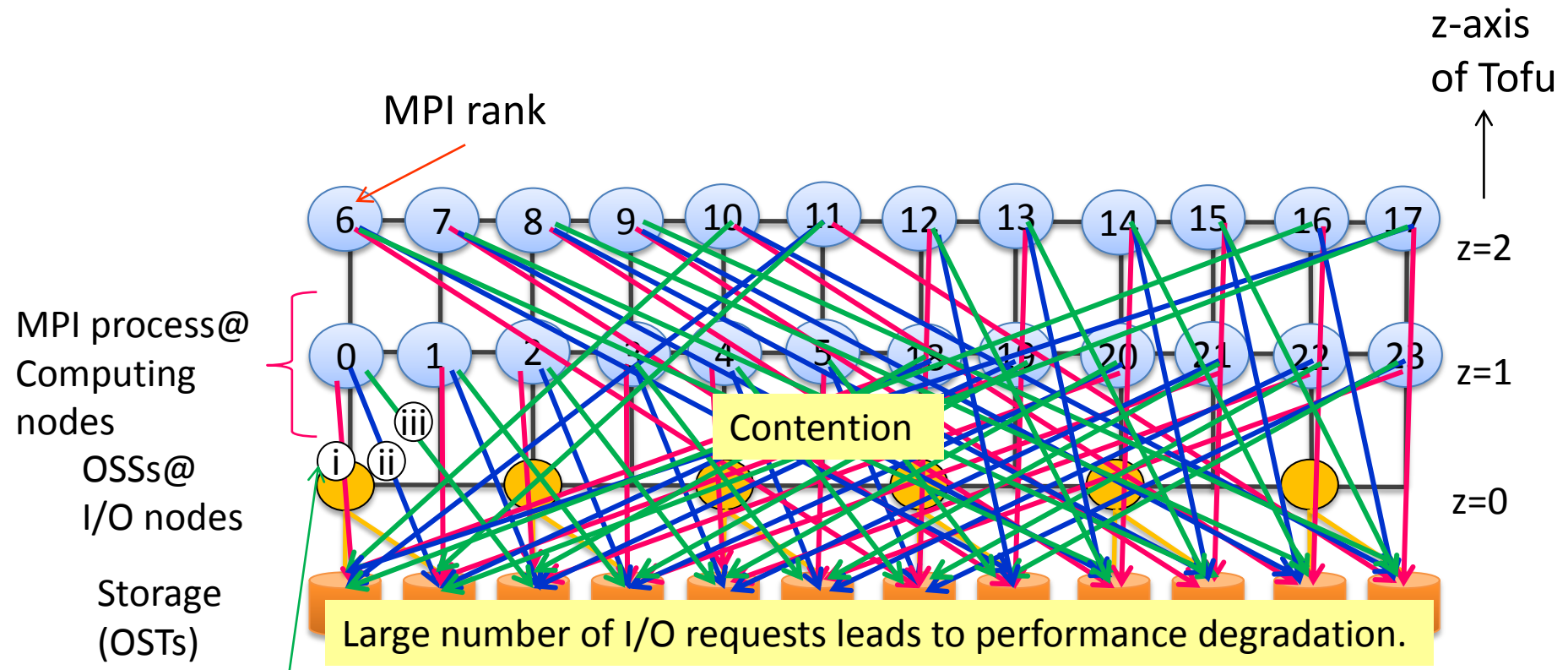
Striping Layout-Aware ADIO for Lustre

- Striping data oriented data aggregation in ADIO for Lustre
 - Pros : Stripe pattern oriented aggregation leads to higher I/O throughput.
 - Cons : Mismatches in aggregation scheme in two-phase I/O



Current ROMIO for FEFS

- Aggregator mapping mismatches for FEFS and Tofu
 - [Current status] ascending mapping order from rank=0, which does not care process placement topology on the Tofu interconnects and FEFS

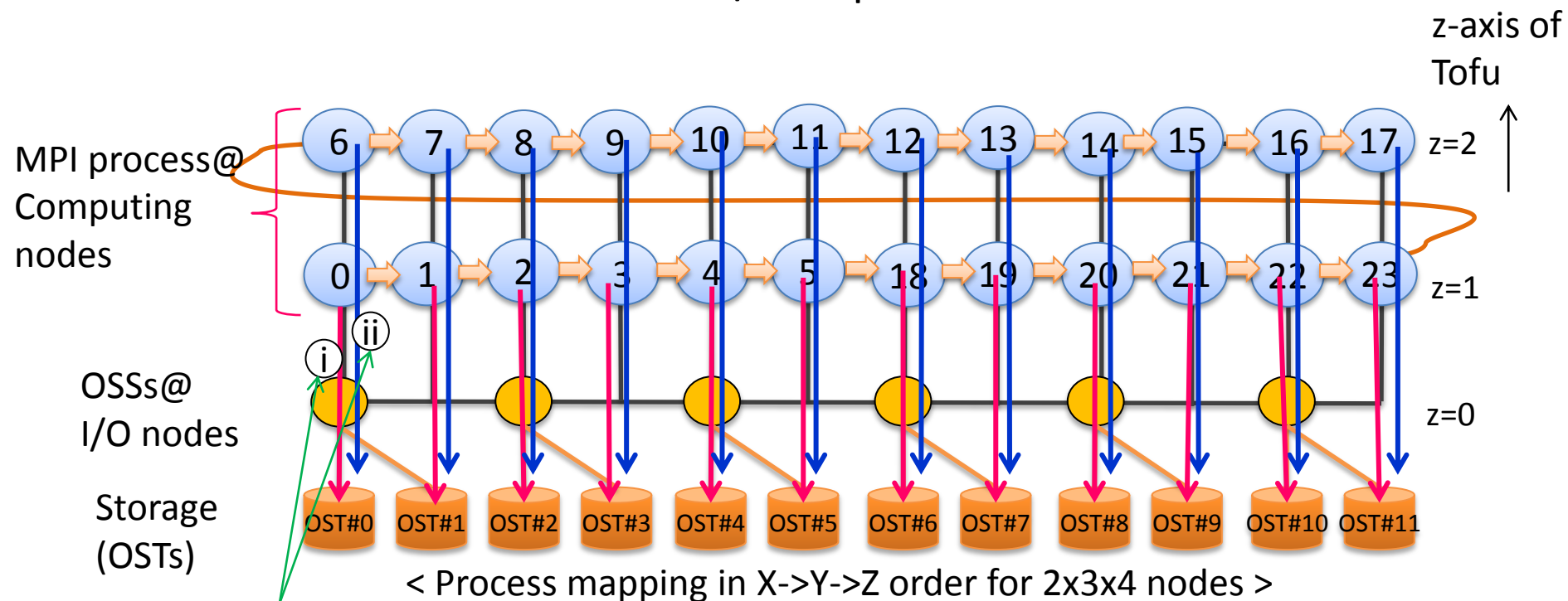


Striping round < Process mapping in X->Y->Z order for 2x3x4 nodes >

- 12 OSTs
- All the MPI processes are assumed to be aggregators.

Aggregator Layout Optimization

- Simplified communication for lesser contention
- Reduction of the number of I/O requests on each OST



Striping round

- 12 OSTs
- All the MPI processes are assumed to be aggregators.

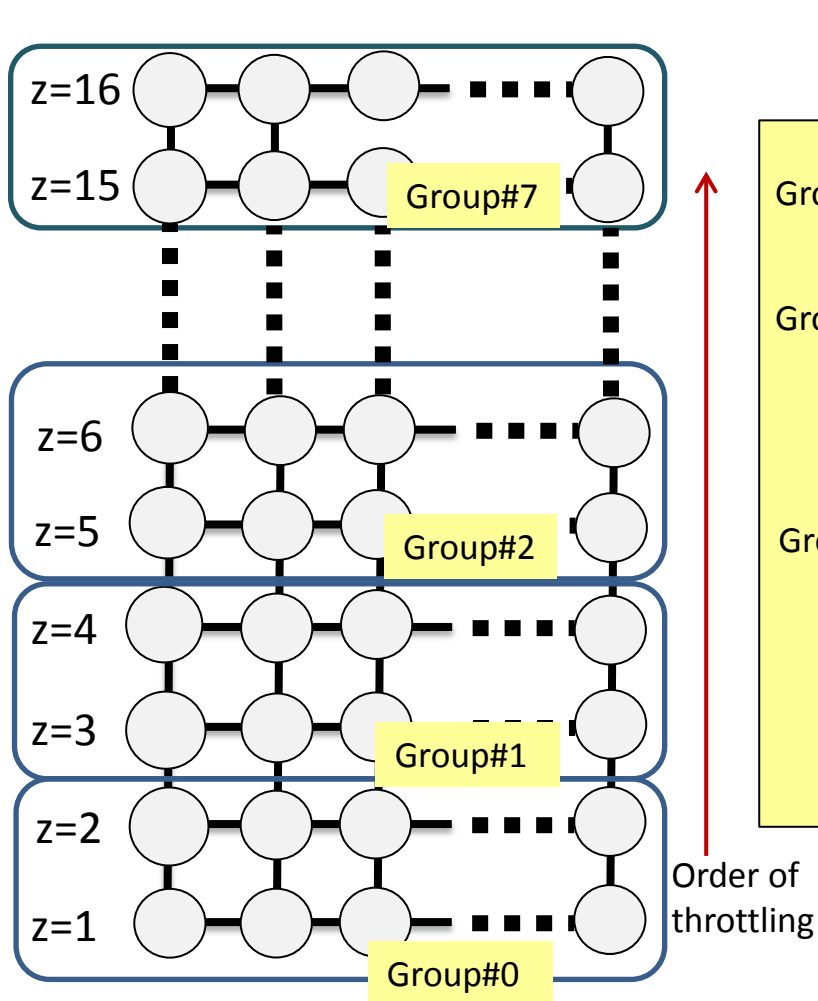
Grouping aggregators which access the same OST on the same z-axis

Further optimization

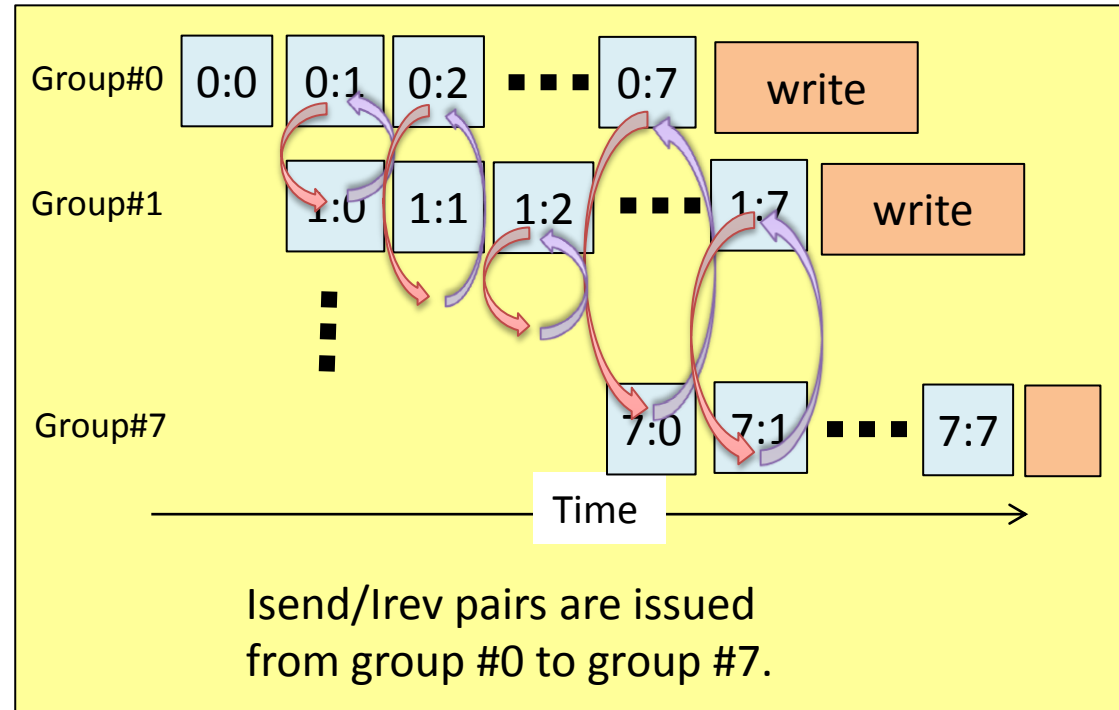


Stepwise Data Exchanges

- Stepwise MPI_Isend/MPI_Irecv associated with I/O throttling



0:1 : data exchanges between group#0 and group#1



The faster data exchange starts, the faster write operation starts.

< Example: 2 requests along z-axis case >

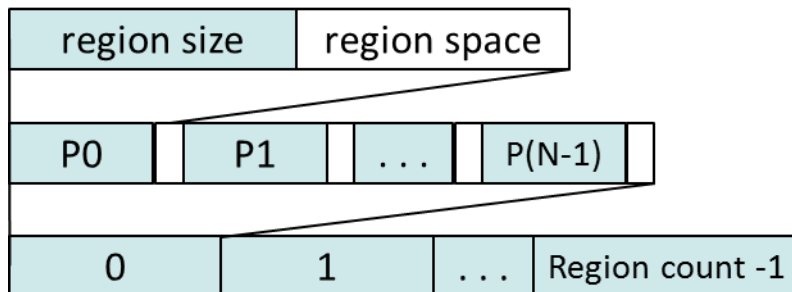
* Length along the z-axis = the number of I/O requests in throttling

Performance Evaluation using HPIO Benchmark

- Collective MPI-IO evaluation using HPIO benchmark
 - Original ROMIO (orig) vs. optimized ROMIO

Implementation	Striping layout aware	Aggregator layout aware	I/O throttling
original	No	No	No
str_agg_aware	Yes	Yes	No
throt_{1,2,4,8,16}	Yes	Yes	Yes

- Evaluation with or without “stepwise data exchanges”



HPIO parameters	
Region size	5,994 B
Region space	256 B
Region count	122,919

< Data layout of HPIO benchmark >

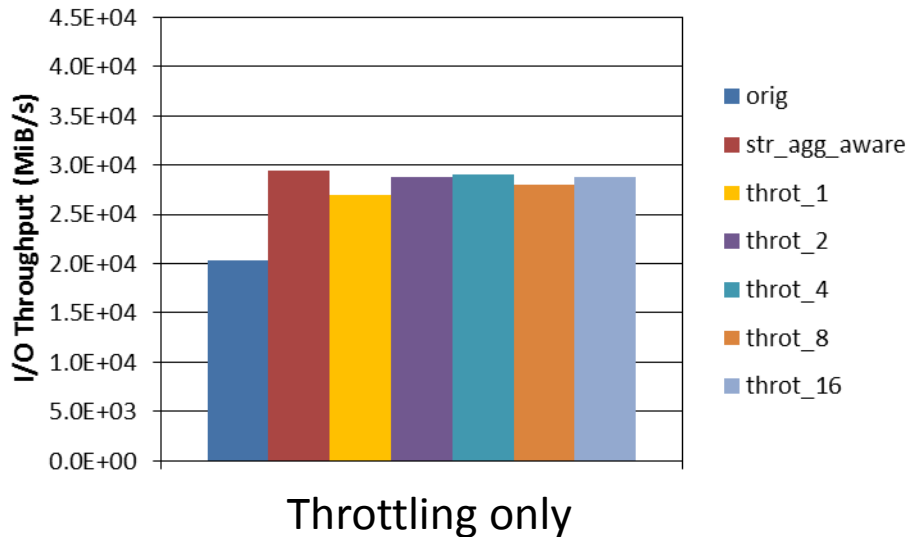
Totally about 730 GiB data was generated per process. (Weak scaling)

HPIO Benchmark Results

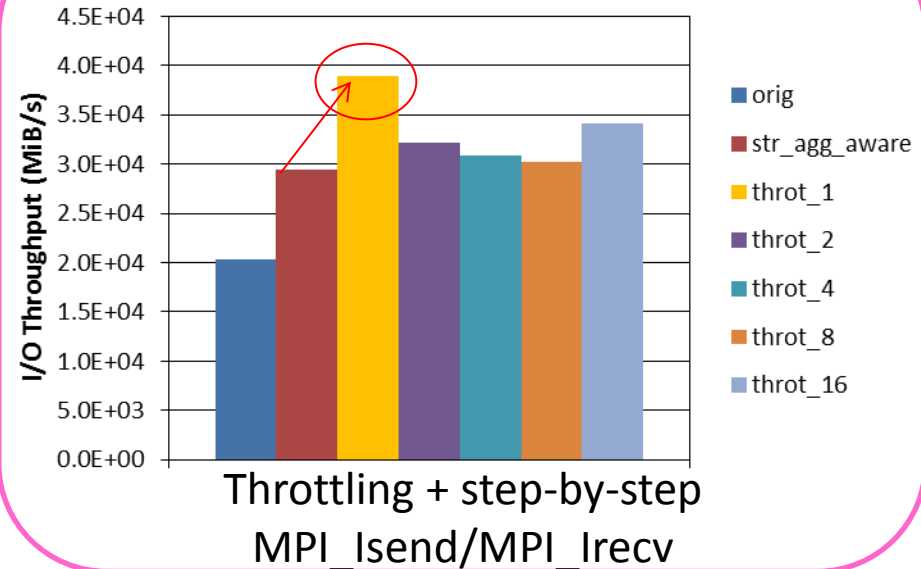
region size: 5,992 B
region space: 256 B
region count: 122,919

- HPIO benchmark result (3,072 processes@8x12x32 nodes)
 - Stepwise MPI_Isend/MPI_Irecv : On / Off

np=3,072(8x12x32,XYZ), stripe size=16 MiB



np=3,072(8x12x32,XYZ), stripe size=16 MiB



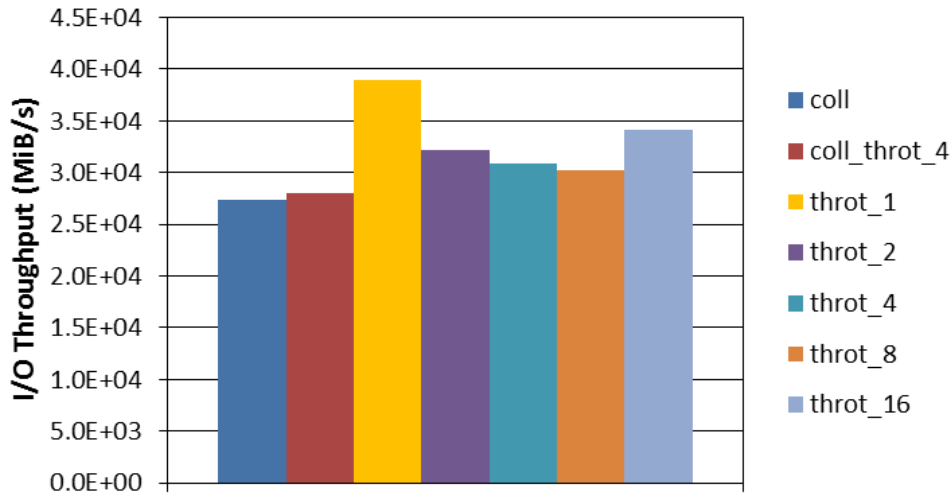
- orig: current ROMIO@K-computer
- str_agg_aware: optimization in striping data layout at aggregators & aggregator layout
- throt_(1,2,4,8): Throttling in FEFS accesses in addition to the “str_agg_aware” optimization (numbers stand for the number of I/O requests in throttling.)

- Applying throttling only is not effective.
- Combination of throttling with step-by-step MPI_Isend/MPI_Irecv leads to higher I/O performance.

HPIO Benchmark Results (cont'd)

- Comparison with the MPI_Alltoallv implementation in data exchange

np=3,072(8x12x32,XYZ), stripe size=16 MiB



< HPIO benchmark setting >

region size: 5,992 B
region space: 256 B
region count: 122,919

* All the cases in the figure are using striping data layout in aggregation and aggregator layout optimization.

- coll: MPI_Alltoallv version instead of MPI_Isend/MPI_Irecv in data exchanges
- coll_throt_4: Same implementation with “coll” with applying throttling in FEFS accesses with 4 I/O requests
- throt_(1,2,4,8): Throttling in FEFS accesses (numbers stand for the number of I/O requests in throttling.)

- Throttling with step-by-step MPI_Isend/MPI_Irecv also outperformed collective data exchange case (coll) and its throttling version (coll_throt_4).
- Step-by-step MPI_Isend/MPI_Irecv data exchanges with throttling in FEFS accesses leads to the most highest I/O throughput at this moment.

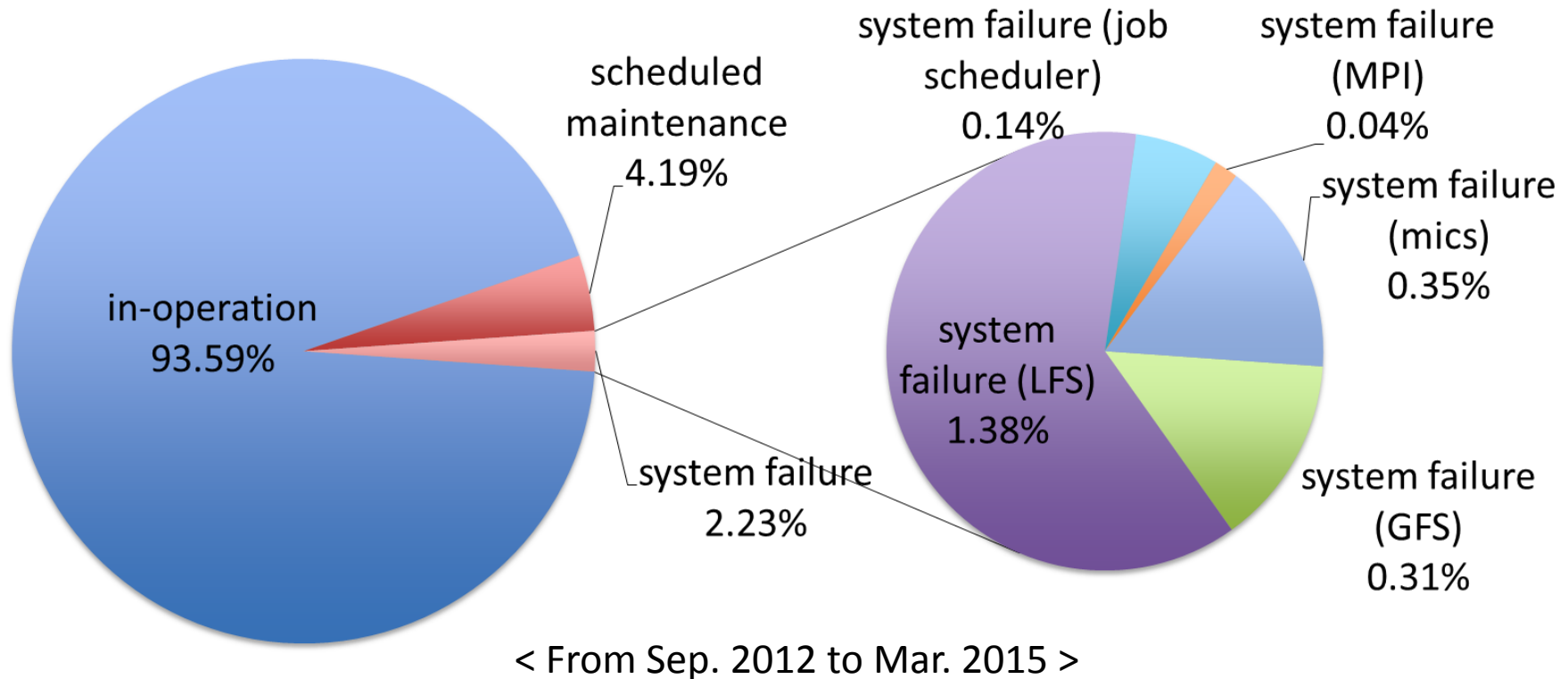
R&D Status and Future Plan

- Throttling and stepwise MPI_Isend/MPI_Irecv seems to be useful to achieve high I/O throughput in collective I/O using ROMIO on the K computer.
- Future Plan
 - Implementation of the optimized ROMIO into HDF5 or PnetCDF for high performance I/O
 - Deployment of the proposed I/O throttling with cooperative stepwise data exchanges at other platform (e.g. large scale of InfiniBand PC cluster system)

FEFS Operation Status

Availability of the whole system since Sep. 2012

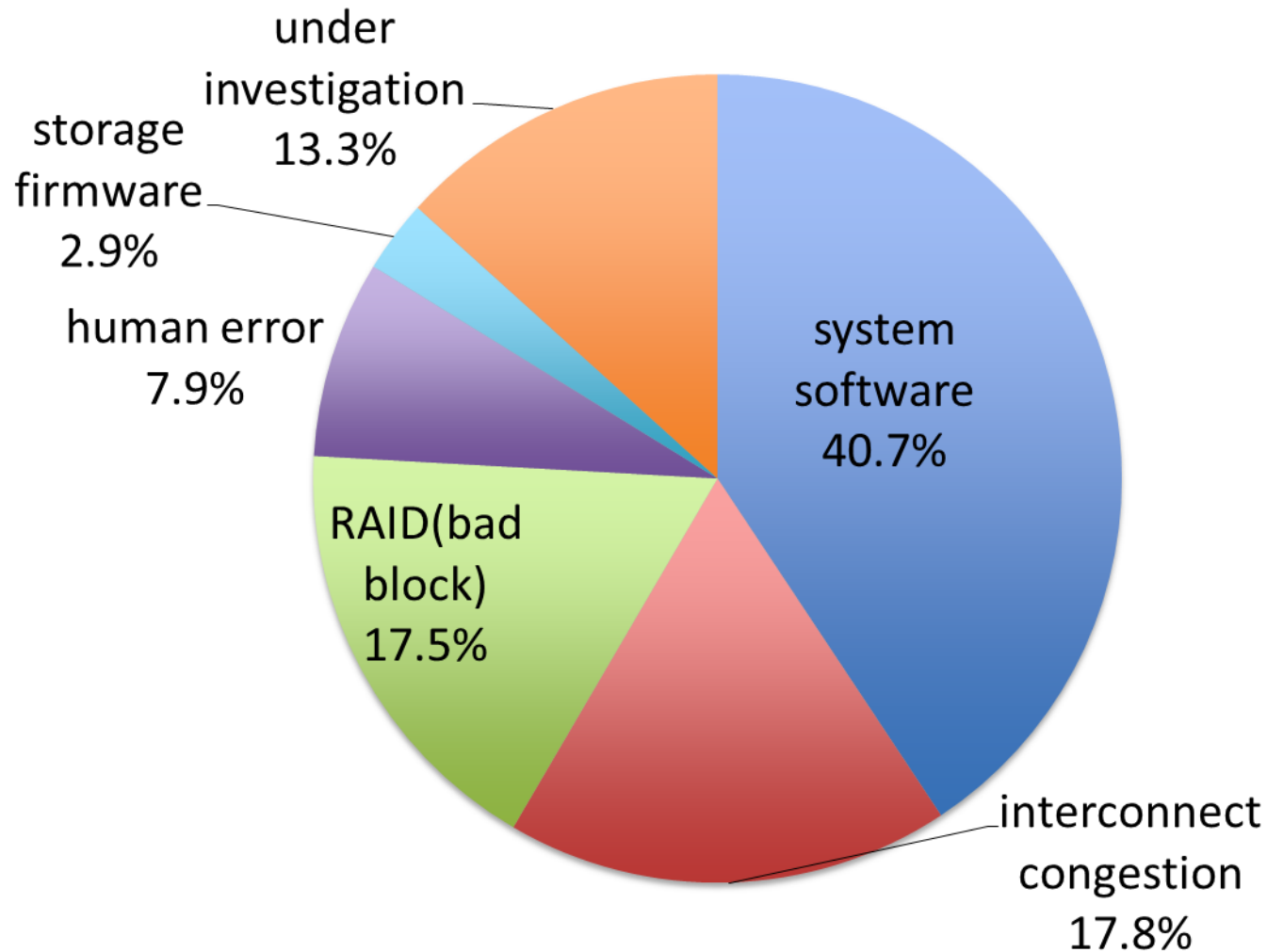
The operation of the K computer is highly stable.



- In-operation and scheduled maintenance amount to 97.78%
- File system failures cause irregular stops.

(F. Shoji et al., "Long term failure analysis of 10 petascale supercomputer," (Best poster awarded) poster presentation in HPC in ASIA session of the ISC'15 (2015))

System failures of LFS in detail



< From Sep. 2012 to Mar. 2015 >

Summary

- Affinity-awareness in collective MPI-IO outperformed the original one by taking care of FEFS striping access layout and the Tofu interconnect configuration.
- Furthermore I/O throttling with stepwise MPI_Isend/MPI_Irecv improved I/O performance on FEFS.
- Most of failures in the K computer comes from disk failures in the local file system built on FEFS. However, the failure rate is quite small (1.38% in more than 2 year operation time). At this moment, operation of the K computer is very stable with many efforts by operation management group.

Acknowledgment

- The authors would like to thank Fujitsu for providing ADIO software for FEFS and useful technical information.
- The authors also would like to thank members of System Software Research Team at RIKEN AICS.
- This research work is partially supported by JST CREST.

References

- Y. Tsujita, A. Hori, and Y. Ishikawa, “Locality-Aware Process Mapping for High Performance Collective MPI-IO on FEFS with Tofu Interconnect,” In Proceedings of the 21th European MPI Users' Group Meeting (Challenges in Data-Centric Computing), ACM (2014).
- Y. Tsujita, A. Hori, T. Kameyama, and Y. Ishikawa, “I/O Throttling and Cooperative Stepwise Data Exchanges in Two-Phase I/O Towards High Performance Collective MPI-IO,” (submitted to ICPADS'15).
- K. Yamamoto et al., “The K computer Operations: Experiences and Statistics,” ICCS2014 (2014).
- K. Yamamoto et al., “Analysis and Elimination of Client Eviction on a Large Scale Lustre Based File System,” LUG2015 (2015).
- F. Shoji et al., “Long term failure analysis of 10 petascale supercomputer,” poster presentation in HPC in ASIA session of the ISC'15 (2015).