Evaluating I/O Acceleration Mechanisms of SX-Aurora TSUBASA

Yuta Sasaki[*], Ayumu Ishizuka[*], Mulya Agung[‡], Hiroyuki Takizawa[‡*]

[*] **Graduate School of Information Sciences, Tohoku University,** {ysasaki,ishizuka}@hpc.is.tohoku.ac.jp

[‡] **Cyberscience Center, Tohoku University,** agung@hpc.is.tohoku.ac.jp, takizawa@tohoku.ac.jp

Outline

- Introduction
- Target System
- Performance Evaluation
- Use Case of I/O Acceleration
- Conclusions and Future Work

Introduction

- **Heterogeneous HPC systems**
  - Promising to increase their computational performance
  - Bring a new challenge in achieving high file **I/O performance**

- **I/O operations are processed by the collaboration among…**
  - Host processor is responsible for handling system calls
  - Some processors **not fully supporting OS functions** invoked via system calls
    - Ask the host to manage system calls such as file I/O operations

Motivation

- **File I/O performance could be a performance bottleneck**
  - Intermediate simulation results are periodically stored into files in practical numerical simulations

- **Auto-tuning is required for I/O performance as well as computational performance**
  - Understanding the I/O characteristics of heterogeneous systems is needed

- Introduction
- **Target System**
- Performance Evaluation
- Use Case of I/O Acceleration
- Conclusions and Future Work

Target System

■NEC SX-Aurora TSUBASA (SX-AT)

- A latest vector computing system with heterogeneous configuration
- Vector hosts (VHs) : x86 host processor
- Vector engines (VEs) : NEC’s proprietary vector processor
  - General-purpose processor
    - Programmers can execute the whole application code
  - Rely on the VH to provide OS functionality
    - Need additional PCIe data transfer between VH and VE for I/O operations
    - Potentially cause **non-negligible overhead**

NEC SX-Aurora TSUBASA
Vector Engine*

■Two I/O acceleration mechanisms

- Accelerated I/O (AccIO)[3]
- ScaTeFS VE Direct I/O (DirIO)[3]
- Implemented as libraries to transparently replace file I/O system calls
  - Without any modification of application code

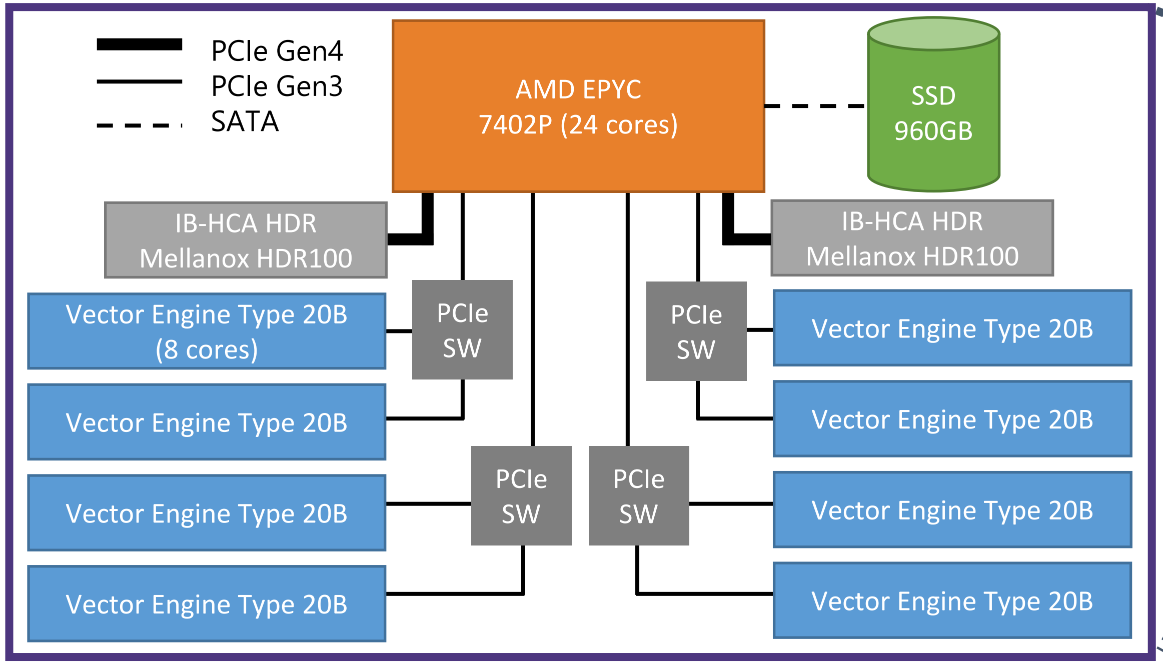
*https://www.nec.com/en/global/solutions/hpc/sx/vector_engine.html

[3] NEC Corporation, “SX-Aurora TSUBASA Program Execution Quick Guide,” https://www.hpc.nec/documentatio

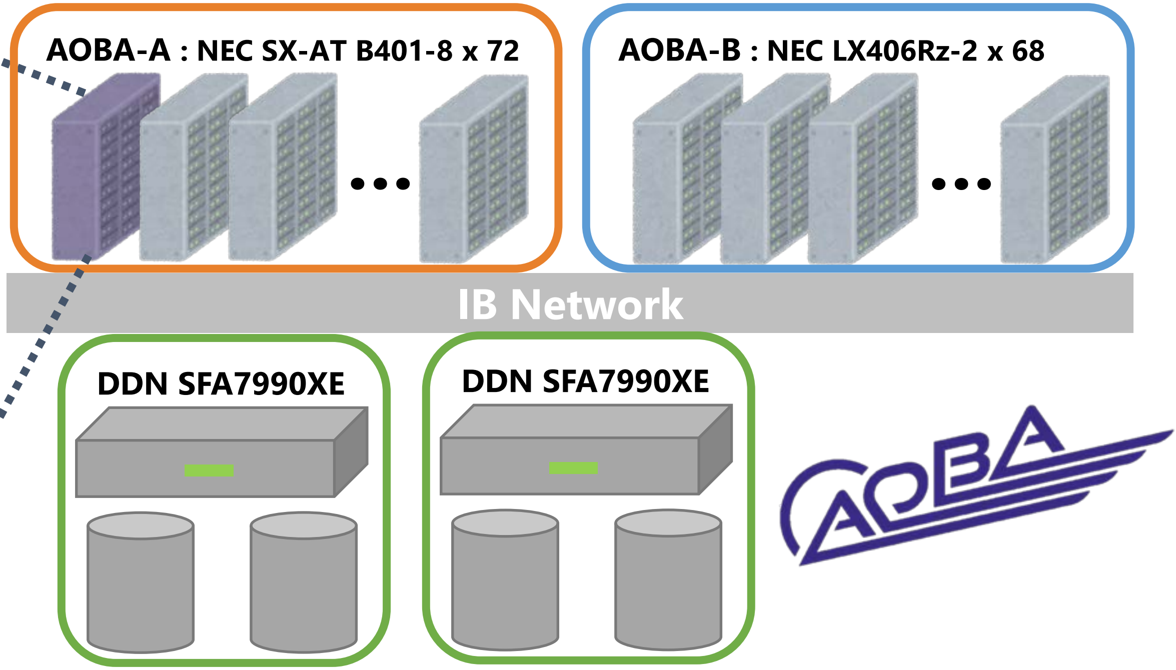
System Configuration

■AOBA system installed at the Cyberscience Center, Tohoku University

- The remote storage system is relatively small of only two DDN SFA7990XE
- Local SSD is also equipped on each node



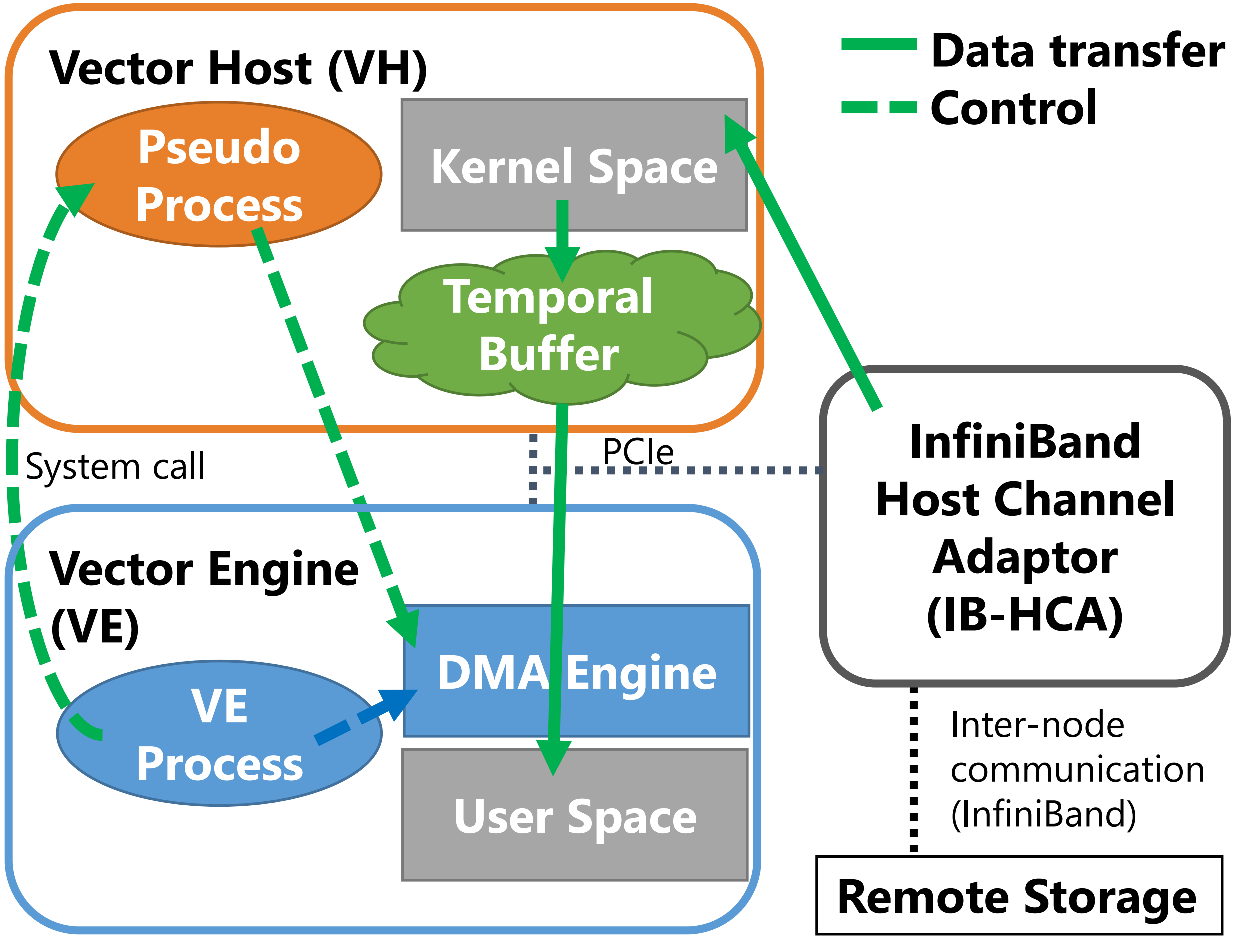
Hardware configuration of NEC SX-AT B401-8



An overview of the AOBA system

Normal I/O (without acceleration)

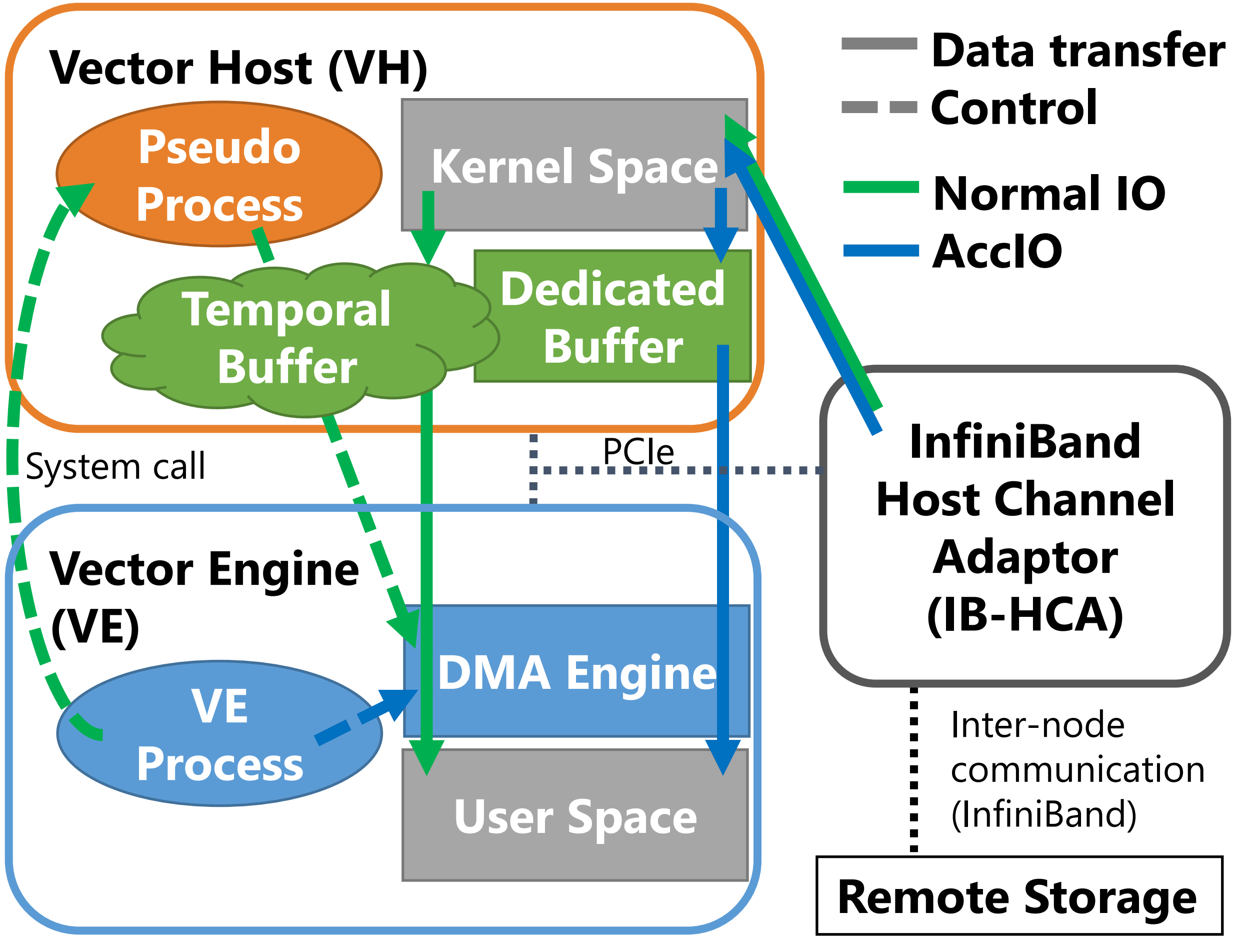
- **Most system calls on VEs are offloaded to user space daemons running on VH (pseudo process)**
  - VE needs additional PCIe data transfer to access a file
- **Data transfers for file read operations**
  - Read from the file to a kernel space buffer in VH
  - Copied to a temporal buffer allocated in the VH user space
  - Sent from the VH to the VE via PCIe interconnect



Data read from the remote storage via IB-HCA

Accelerated I/O (AccIO)

- **Acceleration through elimination of two overheads**
  - Pseudo process in VH controls VE’s direct memory access (DMA) engine
    - AccIO allows a user process running on VE (VE process) to directly control the DMA engine
    - Improve the efficiency of data transfers between a VH and a VE via PCIe
  - A temporal buffer in the VH user space is allocated and released whenever a read system call is invoked
    - Page-locked and dedicated buffer is allocated once and reused

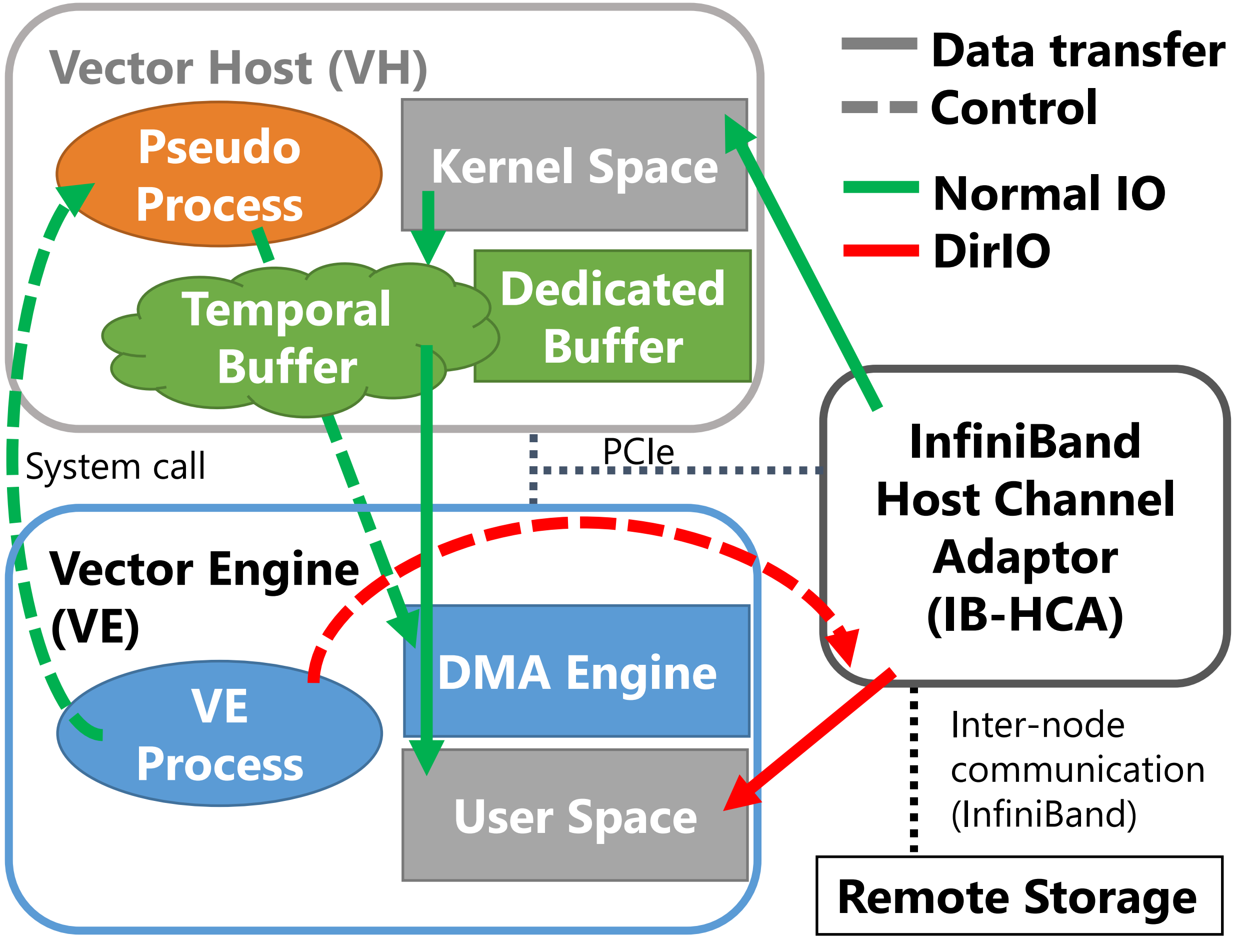


Data read from the remote storage via IB-HCA (AccIO)

ScaTeFS VE Direct I/O (DirIO)

- **Acceleration through direct access to ScaTeFS parallel file system**
  - VE process is capable of directly communicating with IB-HCA
    - Without offloading relevant system calls to VH
    - When the I/O size is greater than 1MB

- **AccIO and DirIO cannot be used at the same time**
  - Need to decide which one should be enabled for a given application



Data read from the remote storage via IB-HCA (DirIO)

- Introduction
- Target System
- **Performance Evaluation**
- Use Case of I/O Acceleration
- Conclusions and Future Work

Evaluation

- **Evaluate the file I/O performance of the ScaTeFS parallel file system[4]**
  - Performance values shown in this work depend on the system configuration
    - i.e. Theoretical peak I/O bandwidth of ScaTeFS scales with the number of I/O servers
  - We do not intend to make comparisons with other systems

- **How does the I/O bandwidth change with the I/O acceleration mechanisms and application behaviors?**
  - Three I/O modes (Normal I/O, AccIO, DirIO)
  - IOR benchmark[1]
    - Widely used to discuss the I/O performance of HPC systems
    - Reproduce the I/O behaviors of various applications by adjusting parameters
      - File size, Transfer size and Single-shared-file/File-per-process etc…

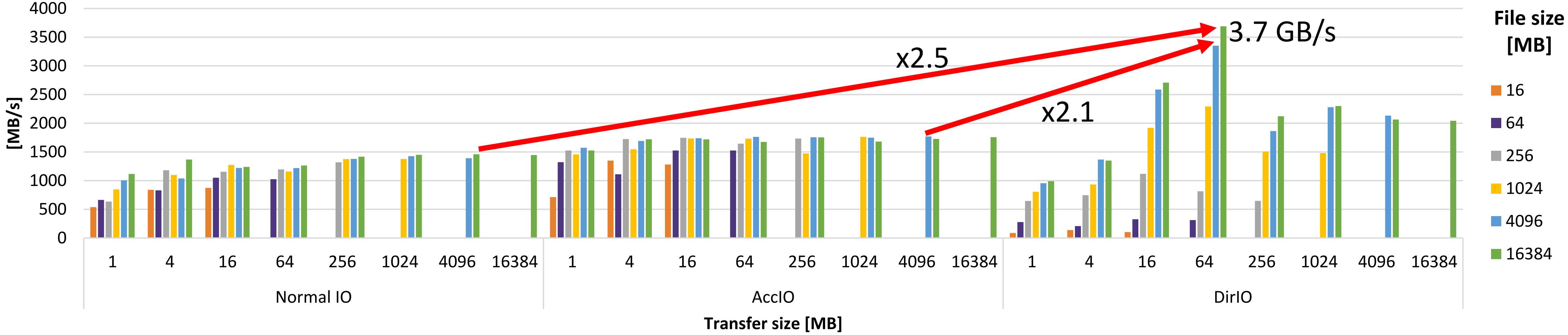
[4] NEC Corporation, "NEC Scalable Technology File System (ScaTeFS) administrator's guide," https://www.hpc.nec/documentation.

[1] H. Shan and J. Shalf, "Using IOR to analyze the I/O performance for HPC platforms," Ernest Orlando Lawrence Berkeley NationalLaboratory, Berkeley, CA (US), Tech. Rep., 2007.

Single-process Write Performance

■**Performance change by the file size *f* and the transfer size *t*** （*f* ≧ *t*）

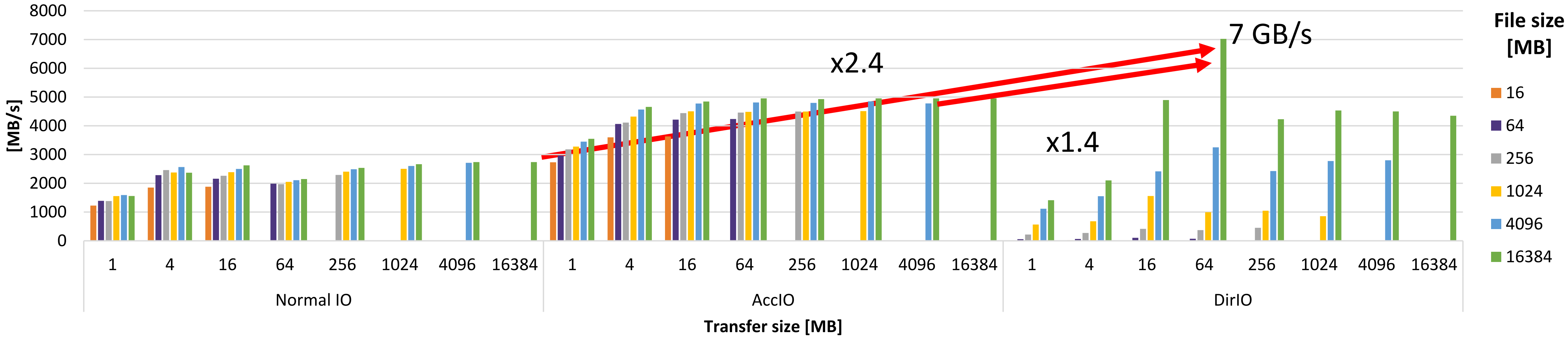
- AccIO performance is higher than the normal I/O performance
  - AccIO improves the efficiency of the data copy between VHs and VEs
- DirIO performance is **much more sensitive to file size *f* and transfer size *t***
  - *f* < 1 GB : DirIO performance is worse even than the normal I/O
  - Significantly increases with f and t and reaches 3.7 GB (*t* = 64MB, *f* = 16GB)
  - **Outperforms the others when writing a sufficiently large file with appropriate *t***



Single-process Read Performance

■The read performance characteristics are similar to those for write

- DirIO needs an even larger file size of 16 Gbytes to outperform AccIO
  - For applications that need to frequently read small files of less than 16GB, it is worth examining to disable DirIO in order to achieve high sustained performance.
- DirIO can achieve the best performance with t = 64 MB
  - ScaTeFS stripe size:4 MB x num of parallel I/O operations for a single process:16 = 64 MB



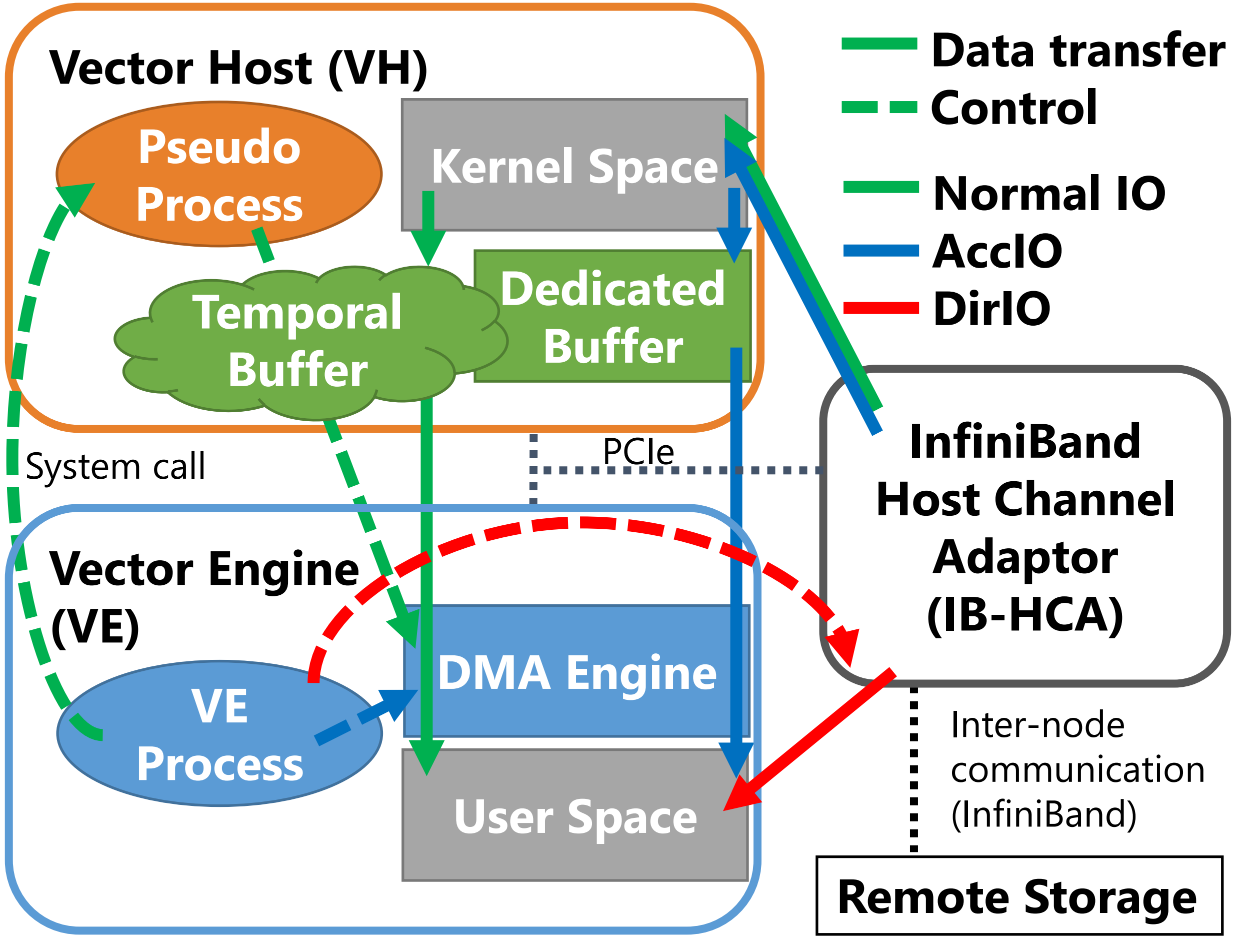
Discussion / Why is DirIO sensitive to *t* ?

■System call handling is not vectorizable

- VH is suited for non-vectorizable operations
- **VE's overhead would become larger**

■DirIO enables VE to communicate directly with IB-HCA without offloading

- When the transfer size is small…
  - Overhead is non-negligible and rather dominant in the I/O time
  - If DirIO is disabled, VH does caching and prefetching I/O data while VE does not support
- As the transfer size increases…
  - VE's overhead becomes relatively smaller
  - the benefit of direct communication between VE and IB-HCA outweighs the overhead



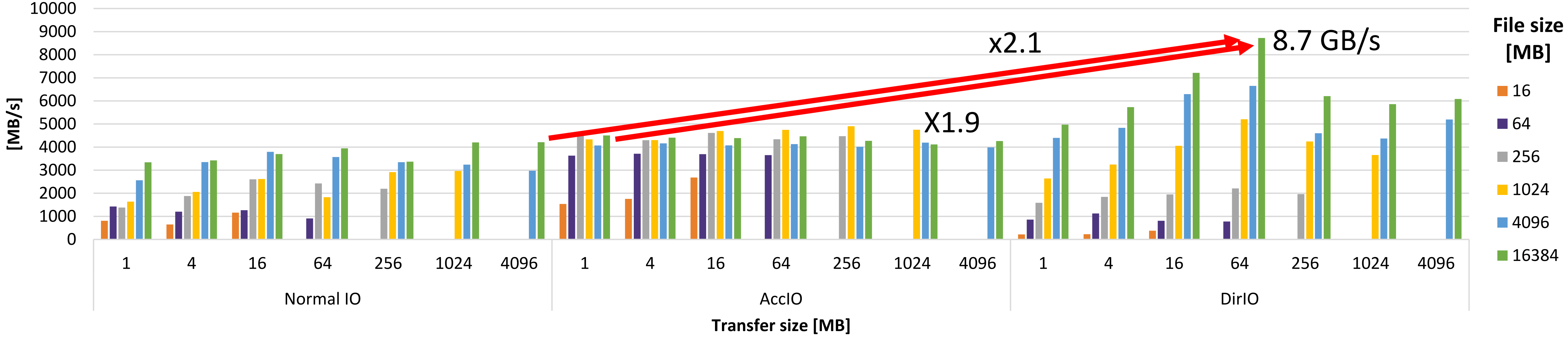
Multi-process Write Performance (1VE)

■Discuss the I/O performance of an MPI application

- All of the 8 cores on a VE are used to access files in parallel
- Each process writes a different file (file-per-process mode)
- File size indicates the size of each file (not the total size)

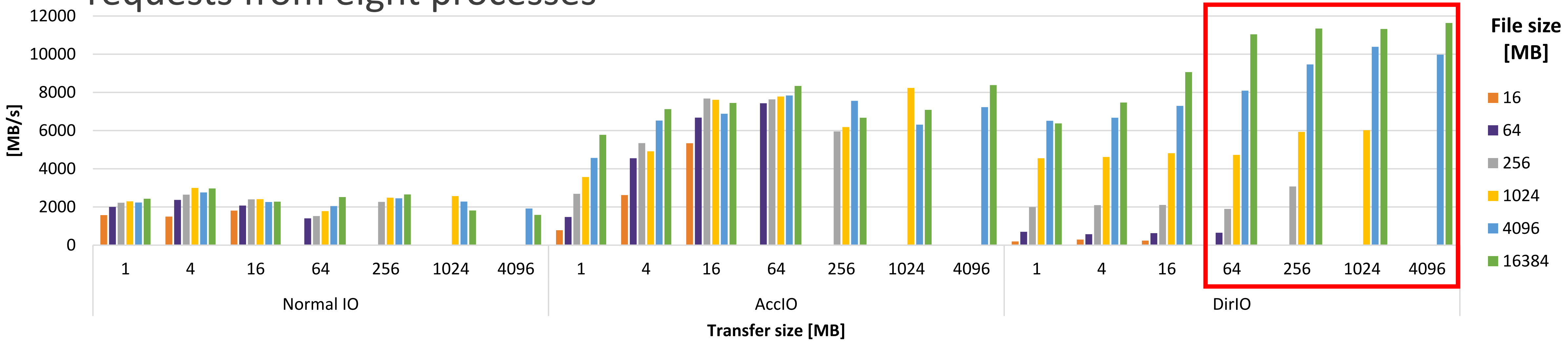
■Aggregated bandwidth of 8 processes

- Reaches about 8.7 GB/s at *t* = 64MB and *f* = 16GB as with the single-process



Multi-process Read Performance (1VE)

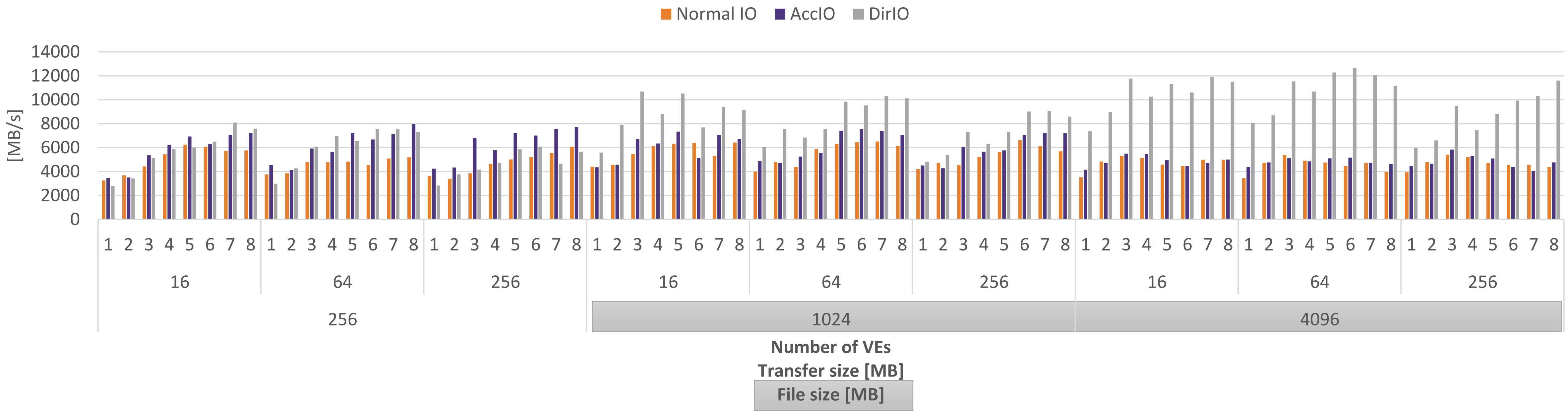
- **No remarkable peak at the transfer size of 64 Mbytes**
  - Performance remains high for a larger transfer size
- **Sustained bandwidth of about 12 GB/s when reading the largest file**
  - The theoretical peak bandwidth of PCIe Gen3 is 16 GB/s
  - **The interconnect bandwidth is almost saturated** by a lot of concurrent file access requests from eight processes



Write Performance with Multiple VEs

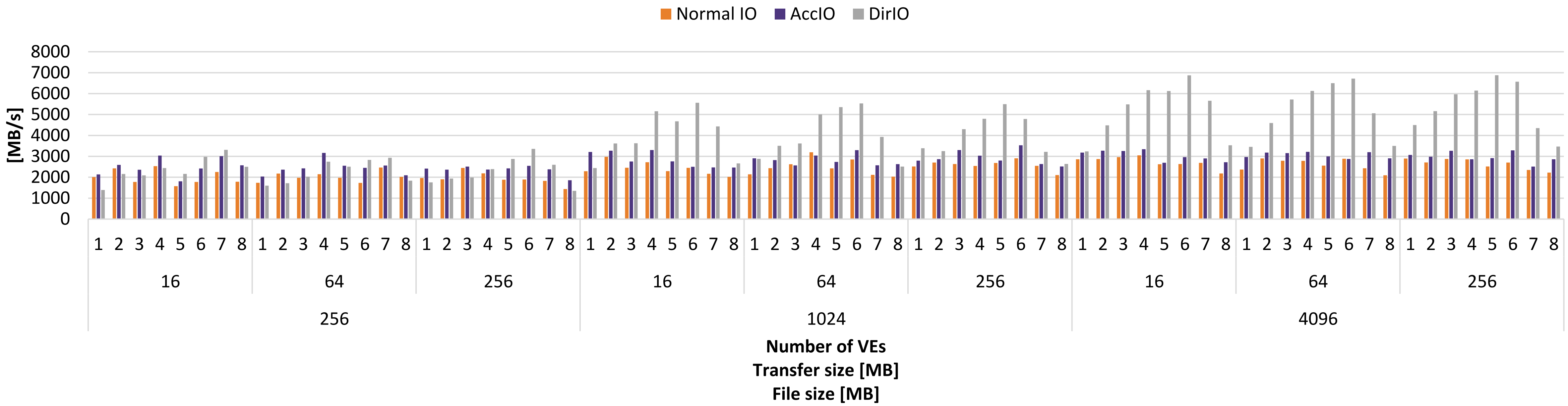
■**When increasing the number of VEs (each VE executes 8 processes)**

- DirIO shows the best performance for the file size of 1 GB or larger
- When the file size and the number of MPI processes are large,
DirIO shows outstanding performance compared to AccIO.



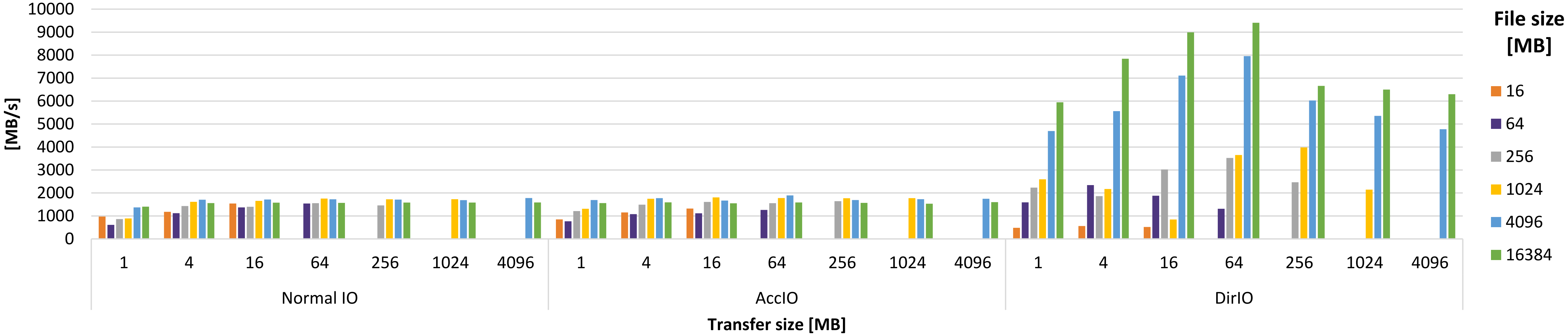
Read Performance with Multiple VEs

- When the number of VEs is large,
  DirIO shows the best performance even for a small file of 1 GB
- The impacts of skipping data copy between VH and VEs
  become more significant when using multiple VEs



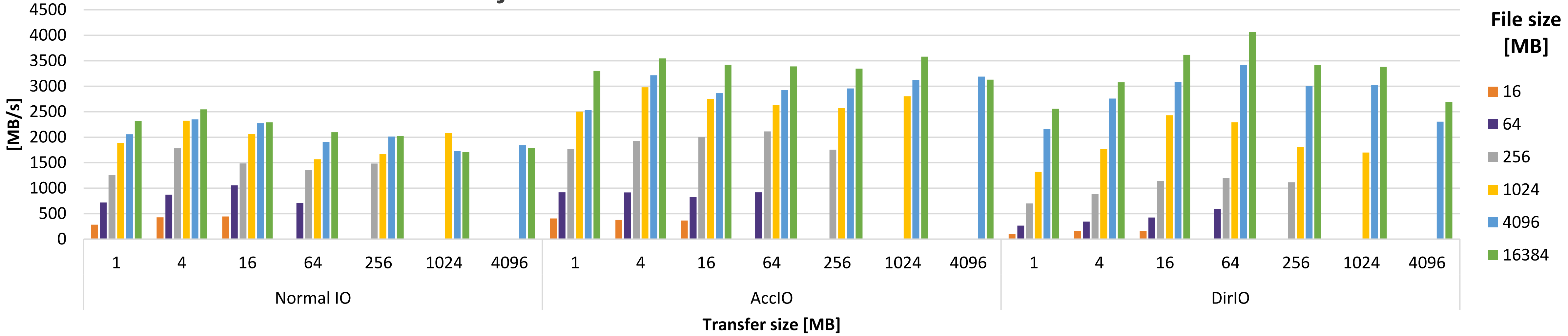
Single-shared-file Write Performance

- **All 8 processes share a single file (“single-shared-file” mode)**
  - File size represents the size that each process writes
- **AccIO performance is lower than that in the “file-per-process” mode**
  - May be due to file blocking
  - The performance degradation of DirIO is small



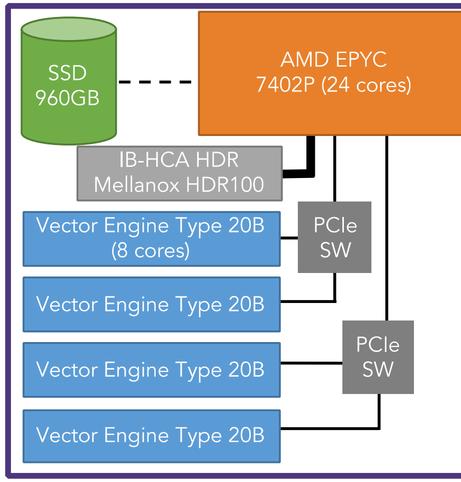
Single-shared-file Read Performance

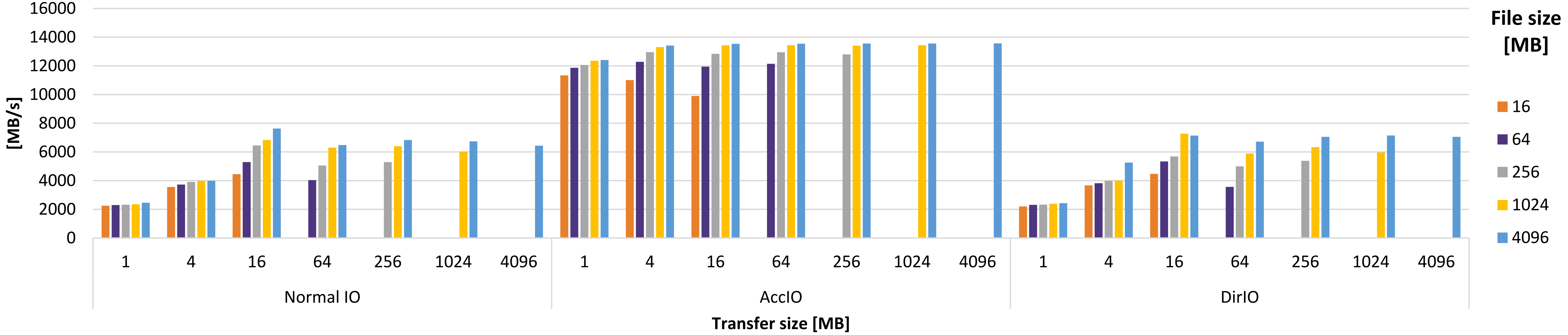
- **Unlike the write performance, the performance of AccIO and DirIO become lower than those in the file-per-process mode**
- **DirIO is likely to be effective when a lot of concurrent I/O operations are executed simultaneously, even if the size of each file is small**
  - This property would be necessary for the operation of HPC systems because a lot of various jobs run at the same time.



Write Performance to Local SSD

- **Each node of AOBA is equipped with a local SSD**
  - All VEs in the node share and access the SSD via the PCIe
- **Only AccIO accelerates write performance**
  - AccIO can improve the data transfer performance between a VH and a VE via via the PCIe interconnect
  - DirIO does not significantly affect the I/O performance because the access to local storage does not go through the IB-HCA

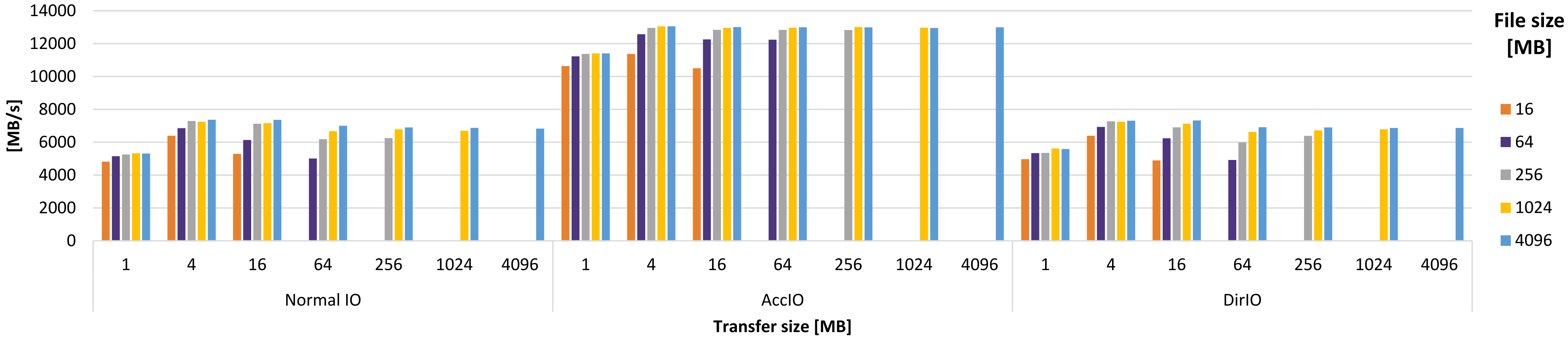




Read Performance from Local SSD

- **As with write, only AccIO accelerates read performance**

- **If an application accesses local storages more intensively than the parallel file system**
  - AccIO should be used to achieve higher I/O performance

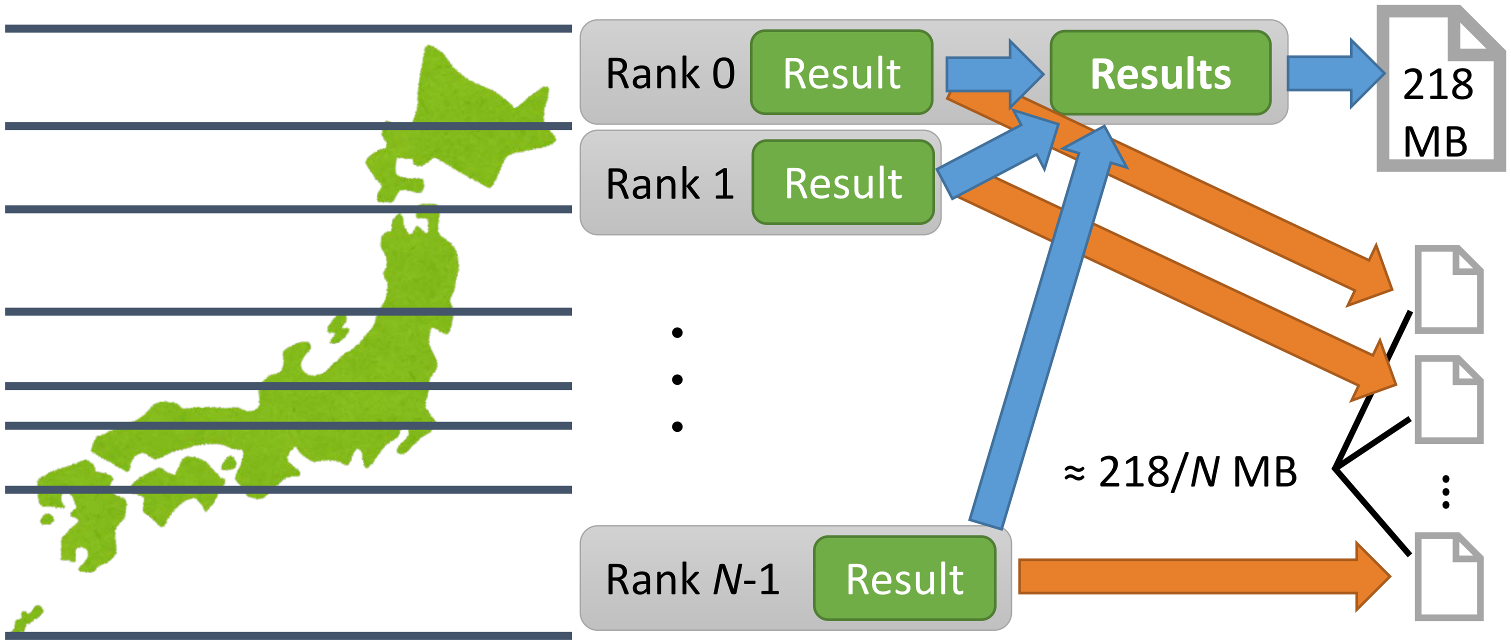


- Introduction
- Target System
- Performance Evaluation
- **Use Case of I/O Acceleration**
- **Conclusions and Future Work**

Use Case of I/O Acceleration

- **Use a real-world MPI application of flood simulation[9] [10] to discuss the performance benefit of I/O acceleration in practical use**
  - Divide the land areas into sections
  - Predict flood damage in parallel with MPI

- **Gathered mode (Original)**
  - Gather intermediate results to Rank 0 and writes the gathered results to a file
- **Parallel mode**
  - Write partial results to a different file

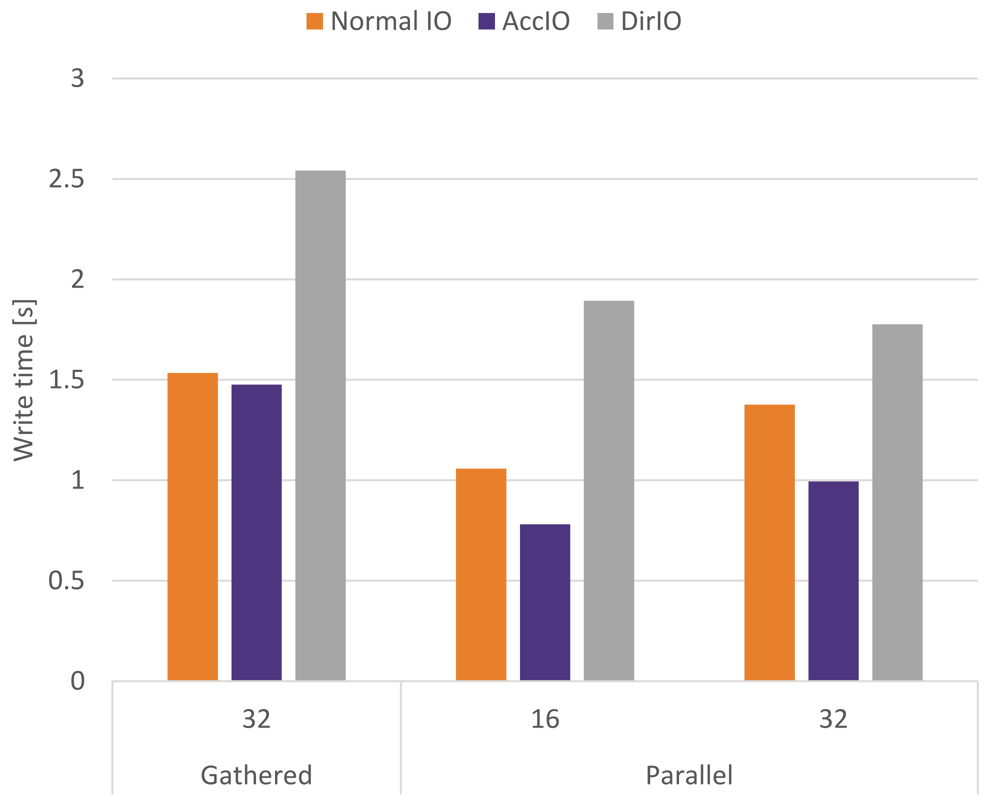


[9] S. Tezuka, H. Takiguchi, S. Kazama, A. Sato, S. Kawagoe, and R. Sarukkaliged, “Estimation of the effects of climate change on floodtriggered economic losses in japan,” International Journal of Disaster Risk Reduction, vol. 9, pp. 58–67, 2014

[10] S. Kazama, A. Sato, and S. Kawagoe, “Evaluating the cost of flood damage based on changes in extreme rainfall in japan,” Sustainability Science, vol. 4, no. 1, pp. 61–69, 2009.

Discussion

- **Write time of the three I/O modes with 16 or 32 MPI processes**
  - In parallel mode, the write time is the total time spent on writing all files
- **Users may consider that DirIO can achieve higher I/O performance in any cases**
  - DirIO could degrade the I/O performance for accessing small files as suggested by the evaluation results with the IOR benchmark
- **Need to carefully select either AccIO or DirIO and its parameters**
  - Considering the file access behaviors



Write performance in flood simulation

Conclusions and Future Works

- **The first investigation into effects of AccIO and DirIO on I/O of SX-AT**
  - Discussed proper use of AccIO and DirIO for a real-world application

- **Our evaluation results clearly show that the two I/O acceleration mechanisms have their own pros and cons**
  - Appropriately used considering the application behaviors and system configuration.
  - Clarified the demand for auto-tuning technology to appropriately select either of the two I/O acceleration mechanisms of SX-AT
  - Abstraction and auto-tuning of those mechanisms will be discussed in our future work

Acknowledgments

■**This work is partially supported by**

- MEXT Next Generation High-Performance Computing Infrastructures and Applications R&D Program “R&D of A Quantum-Annealing-Assisted Next Generation HPC Infrastructure and its Applications”
- Grant-in-Aid for Scientific Research(A) #20H00593

■**The authors would like to thank Takashi Sato and Yuta Watanabe of NEC corporation for their technical supports and fruitful discussions**