Understanding Parallel I/O Performance Trends under Various HPC Configurations

SNTA 2019, Phoenix

Hanul Sung, Jiwoo Bang, Alexander Sim, Kesheng Wu, Hyeonsang Eom

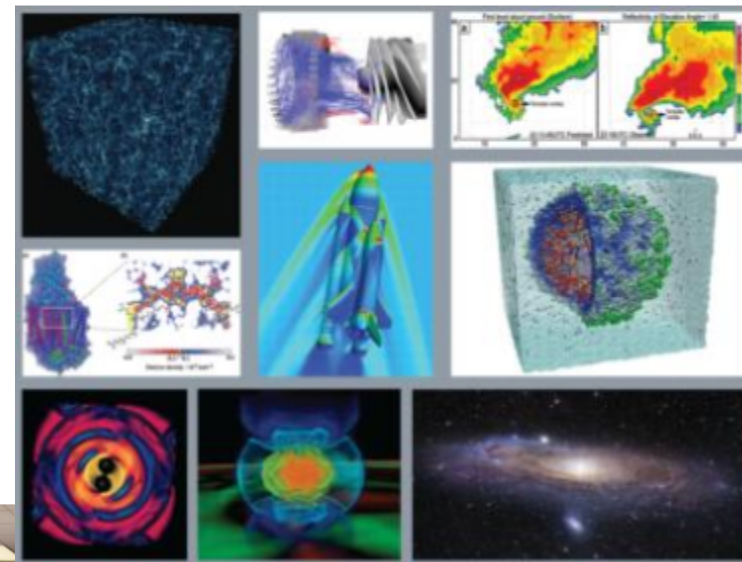
Distributed Computing Systems Laboratory
Department of Computer Science and Engineering
Seoul National University, Korea
Lawrence Berkeley National Laboratory, USA

Table of Contents

- **Introduction**
- **Background**
  - Cori supercomputer
  - Parallel I/O operation
- **Parallel I/O Performance Trend**
  - Independent I/O
  - Collective I/O
- **Evaluation**
- **Conclusion**

Petascale Computing on Super Computer

- **Improving the importance of Distributed File System**



Petascale Computing on Super Computer

- **Improving the importance of Distributed File System**
  - Various tunable parameters
    - # of compute nodes, # of cores, # of OSTs and stripe size

Limitation of Super Computer Usage

- **HPC users are unfamiliar with their HPC environments**
  - They do not use HPC parameter settings properly

Limitation of Super Computer Usage

- **HPC users are unfamiliar with their HPC environments**
  - They do not use HPC parameter settings properly

- **There are too many combinations of parameters to be considered for experiments**
  - I/O characteristics for each application is different
  - I/O performance differs depending on the HPC system

Limitation of Super Computer Usage

- **HPC users are unfamiliar with their HPC environments**
  - They do not use HPC parameter settings properly
- **There are too many combinations of parameters to be considered for experiments**
  - I/O characteristics for each application is different
  - I/O performance differs depending on the HPC system
- **Performance fluctuation is inevitable because hardware resource is shared by multiple users in HPC environment**
  - It is difficult to get expected or correct I/O performance in a single experiment per configuration

Parallel I/O Performance Trend

- **Motivation**
  - Easy way to get the best configuration with minimal efforts
    - For the highest I/O performance

Parallel I/O Performance Trend

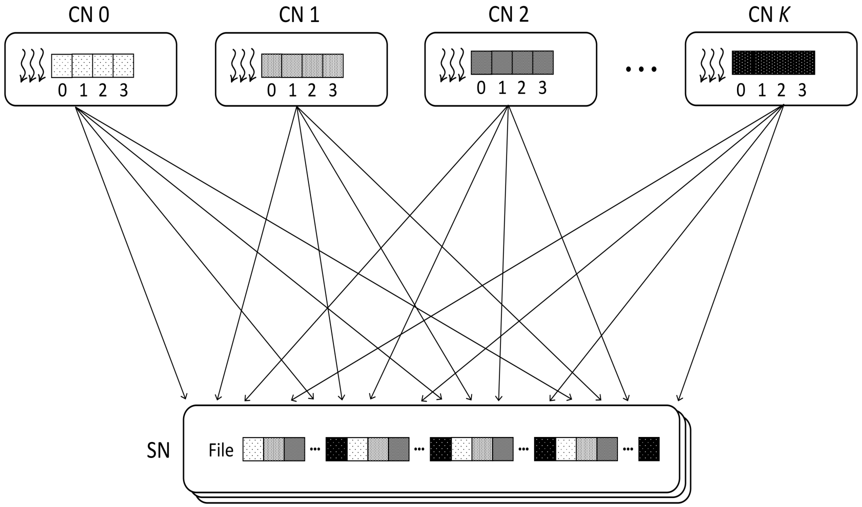
- **Motivation**
  - Easy way to get the best configuration with minimal efforts
    - For the highest I/O performance
- **Goal**
  - Analyzing performance trends by adjusting the tunable parameter settings in Cori system

Parallel I/O Performance Trend

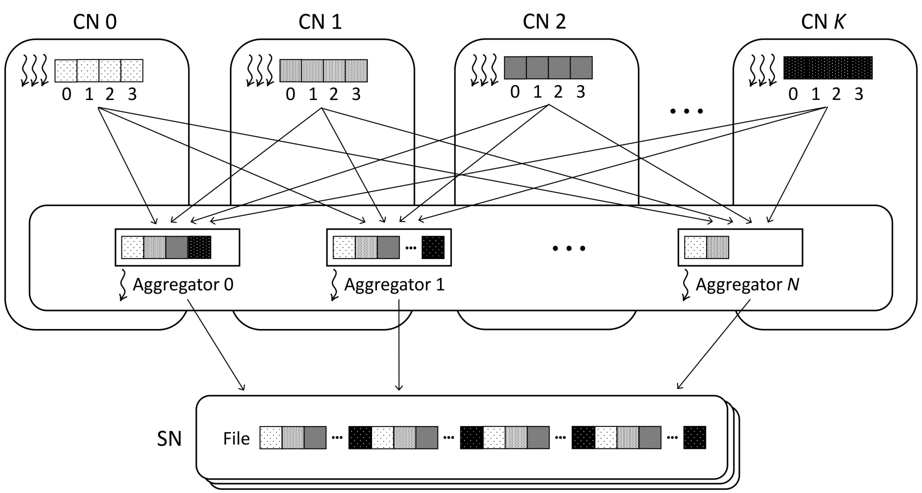
- **Motivation**
  - Easy way to get the best configuration with minimal efforts
    - For the highest I/O performance
- **Goal**
  - Analyzing performance trends by adjusting the tunable parameter settings in Cori system
- **Method**
  - Different HPC I/O characteristics
    - Independent I/O
    - Collective I/O
  - Tunable parameters
    - # of compute nodes
    - # of cores per compute node
    - # of OSTs

HPC I/O Characteristic

- **Independent I/O**
  - Each MPI processes handles I/O operations independently on its own data
- **Collective I/O**
  - Aggregators merge other processes' data and only participate in file I/O



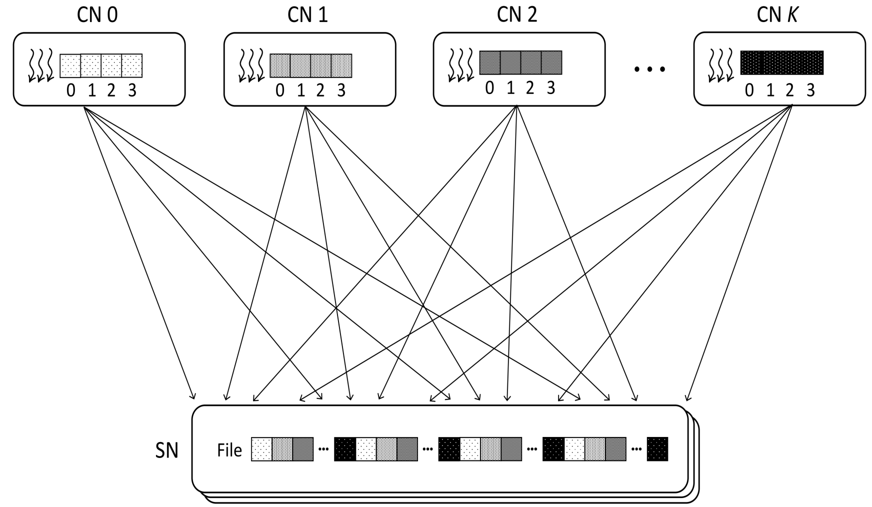
Independent I/O



Collective I/O

Independent I/O

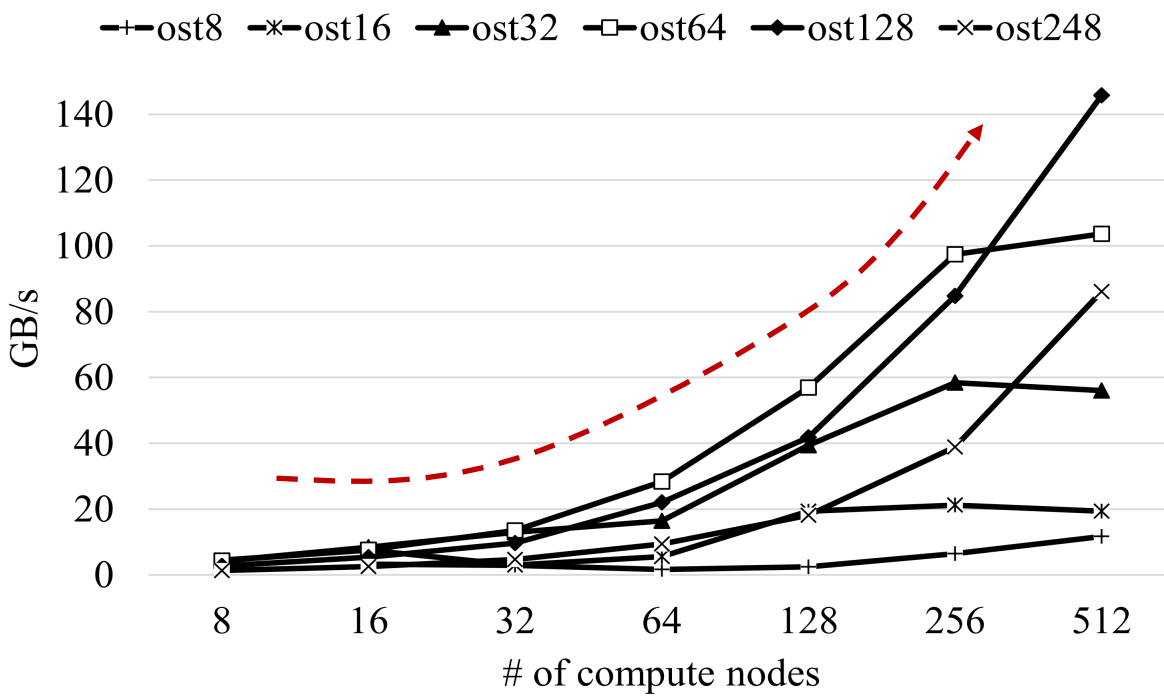
- **Analysis setting**
  - 8 to 512 KNL compute nodes
    - 8, 16, 32, 64, 128, 256, 512 compute nodes
  - 64 cores per compute node
  - 8 to 248 OSTs
    - 8, 16, 32, 64, 128, 248 OSTs
  - 512GB output size
  - 1MB stripe size



DCSLAB

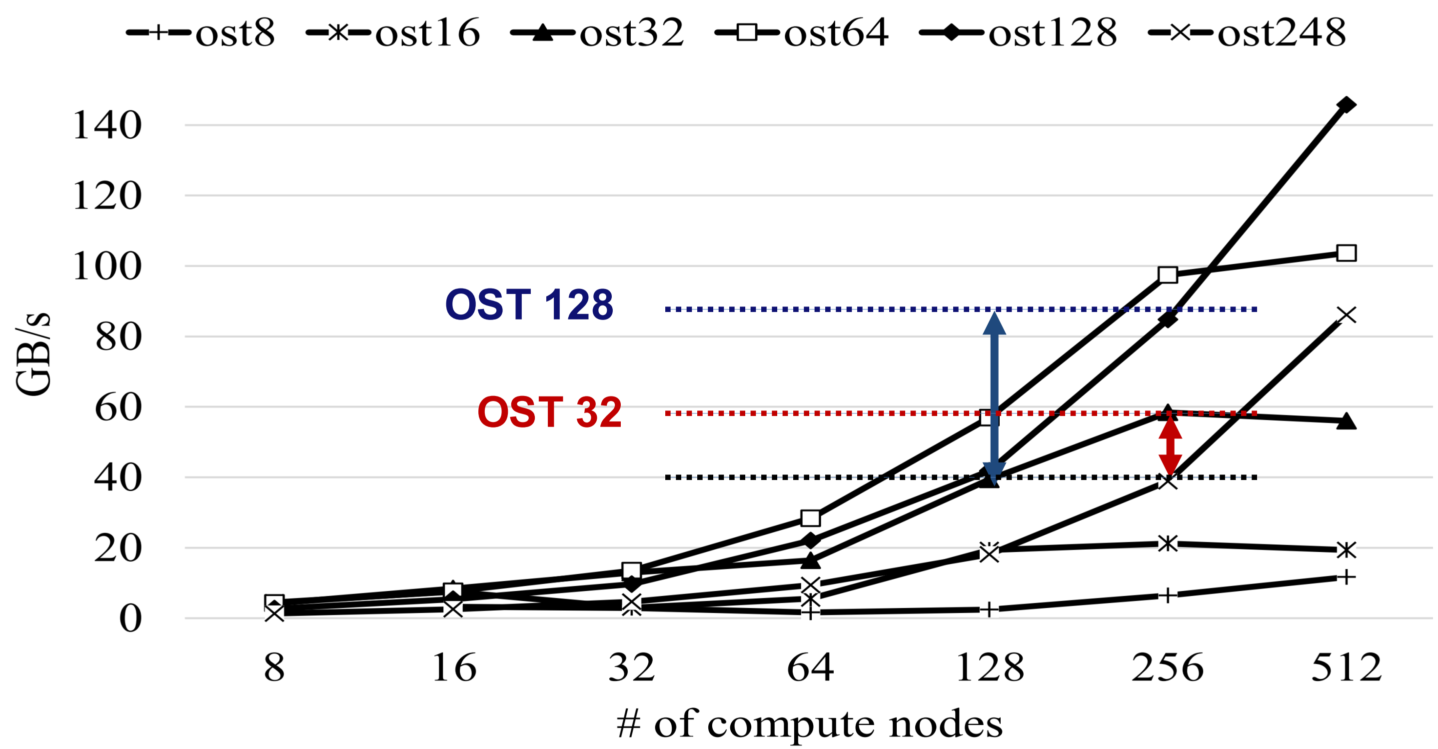
Independent I/O

- **# of compute nodes**
  - # of compute nodes ↑ ➡ MPI throughput ↑



Independent I/O

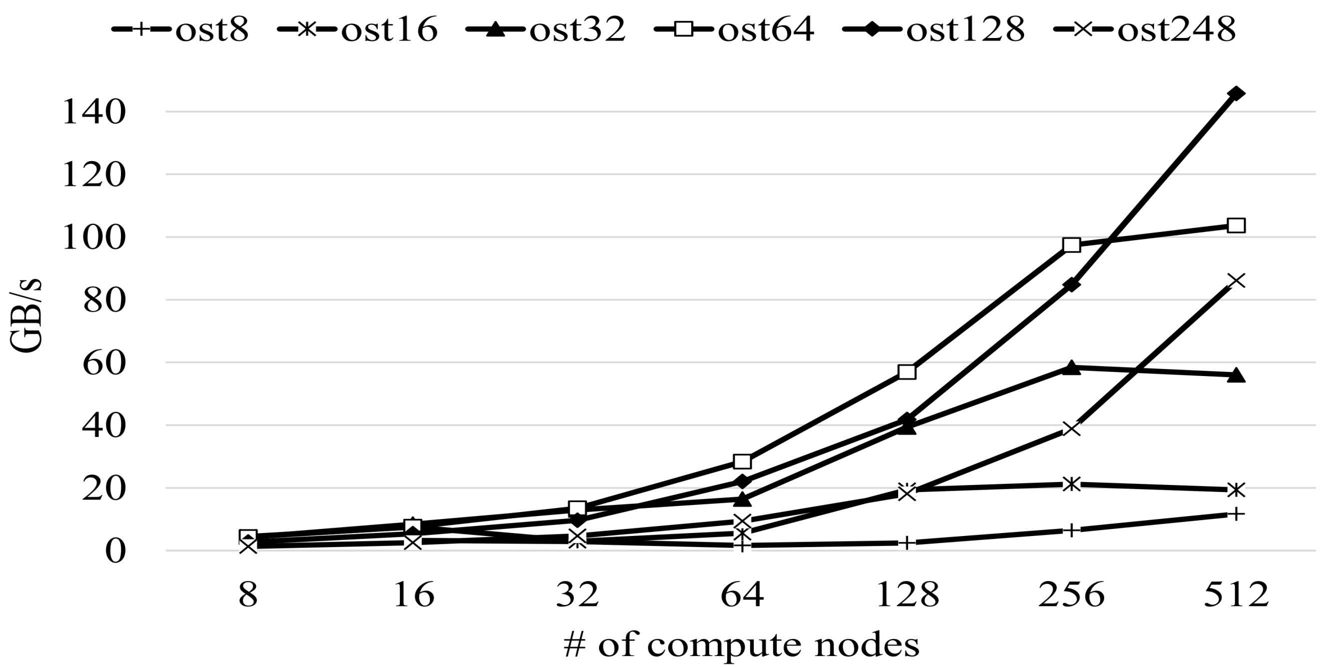
- **# of compute nodes**
  - # of compute nodes ↑ ➡ MPI throughput ↑
  - # of OSTs ↑ ➡ MPI throughput variation ↑



Independent I/O

- **# of compute nodes**
  - # of compute nodes ↑ ➡ MPI throughput ↑
  - # of OSTs ↑ ➡ MPI throughput variation ↑

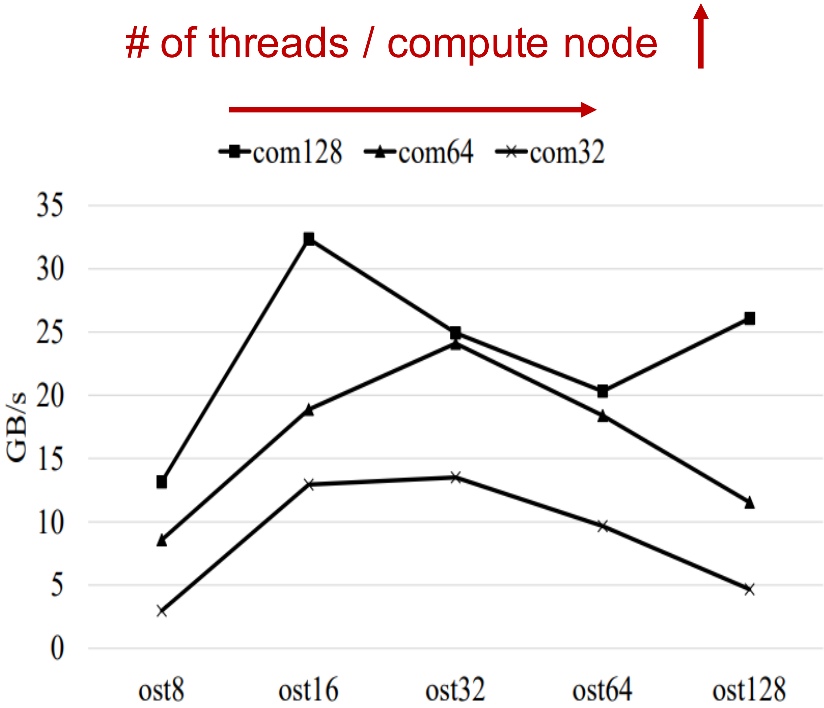
If the number of available OSTs is fixed, it is better to use as many compute nodes as possible for high performance



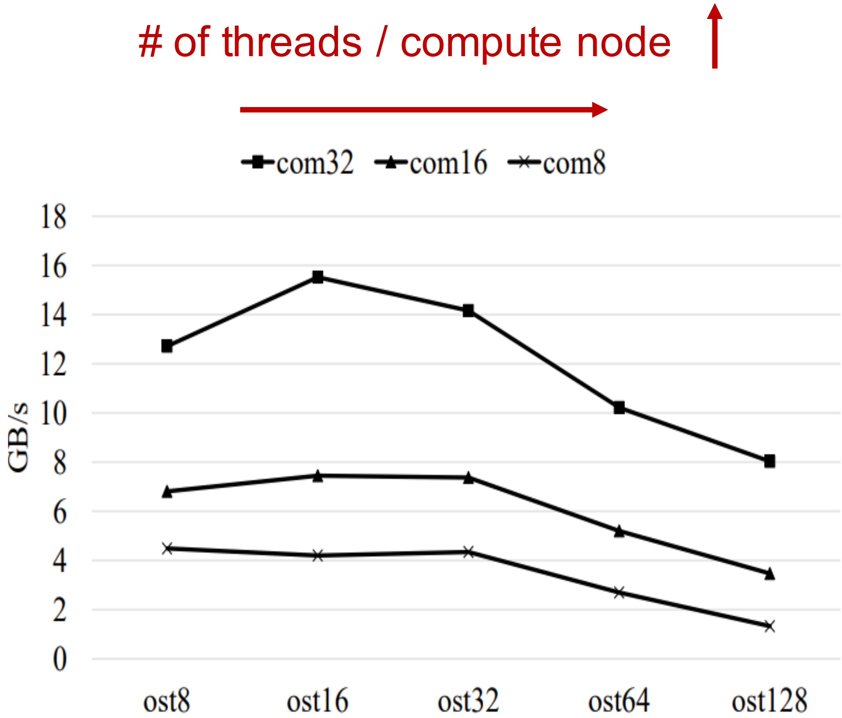
DCSLAB

Independent I/O

- **# of cores per compute node**
  - # of cores per compute node ↑ ➡ MPI throughput ↓



(b) 2048 threads

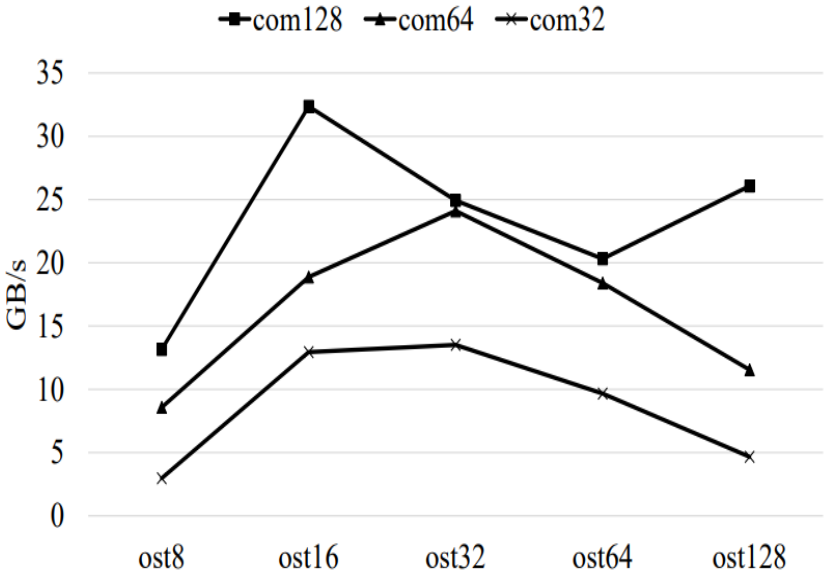


(c) 512 threads

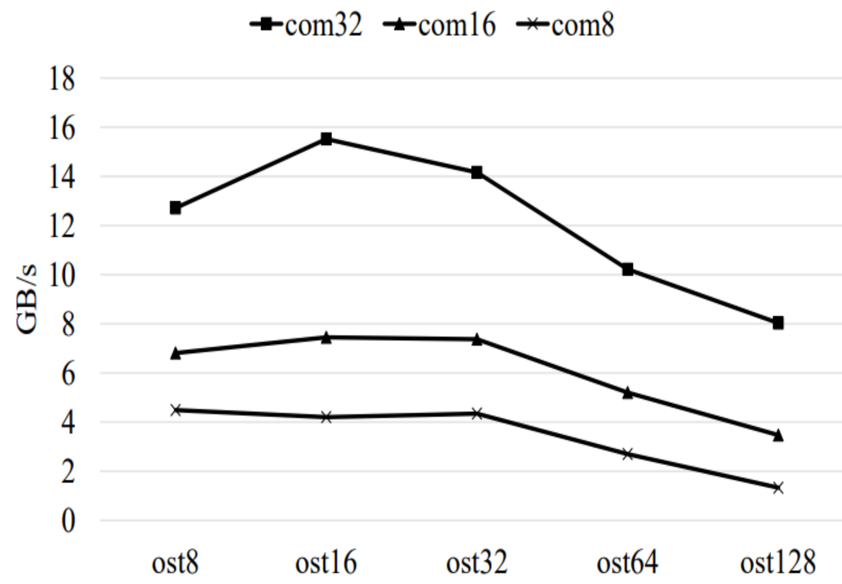
Independent I/O

- **# of cores per compute node**
  - # of cores per compute node ↑ ➡ MPI throughput ↓

If there is enough number of compute nodes, it is better to use fewer cores per compute node for high performance



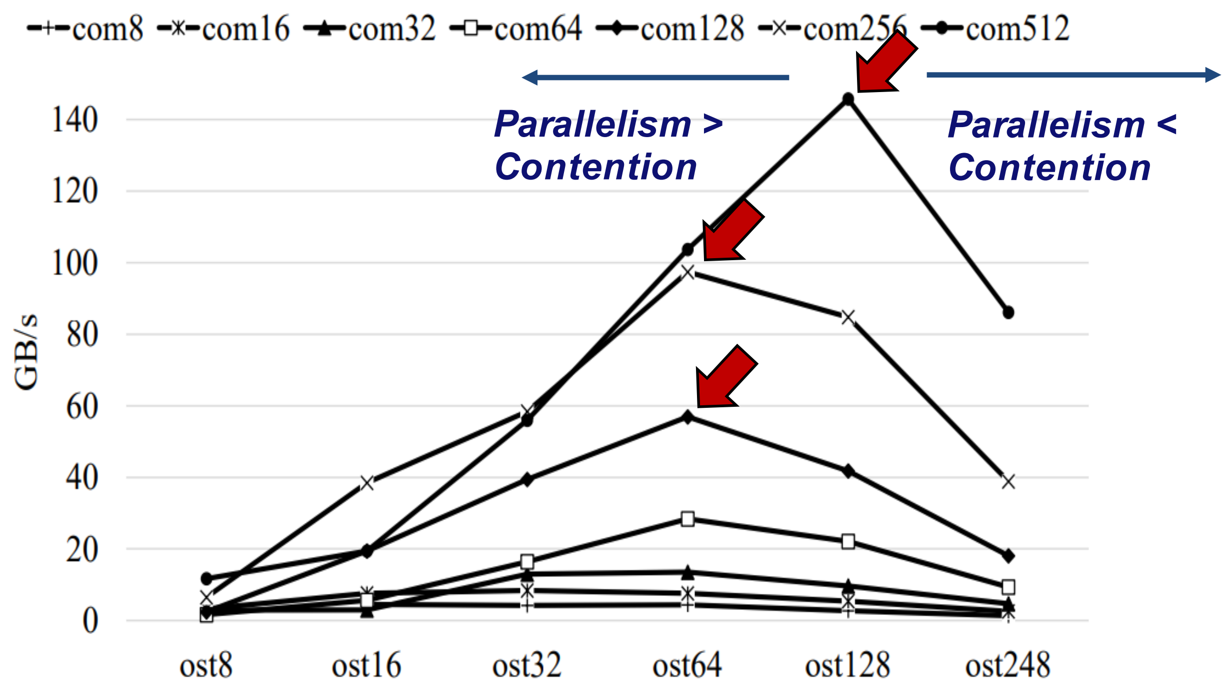
(b) 2048 threads



(c) 512 threads

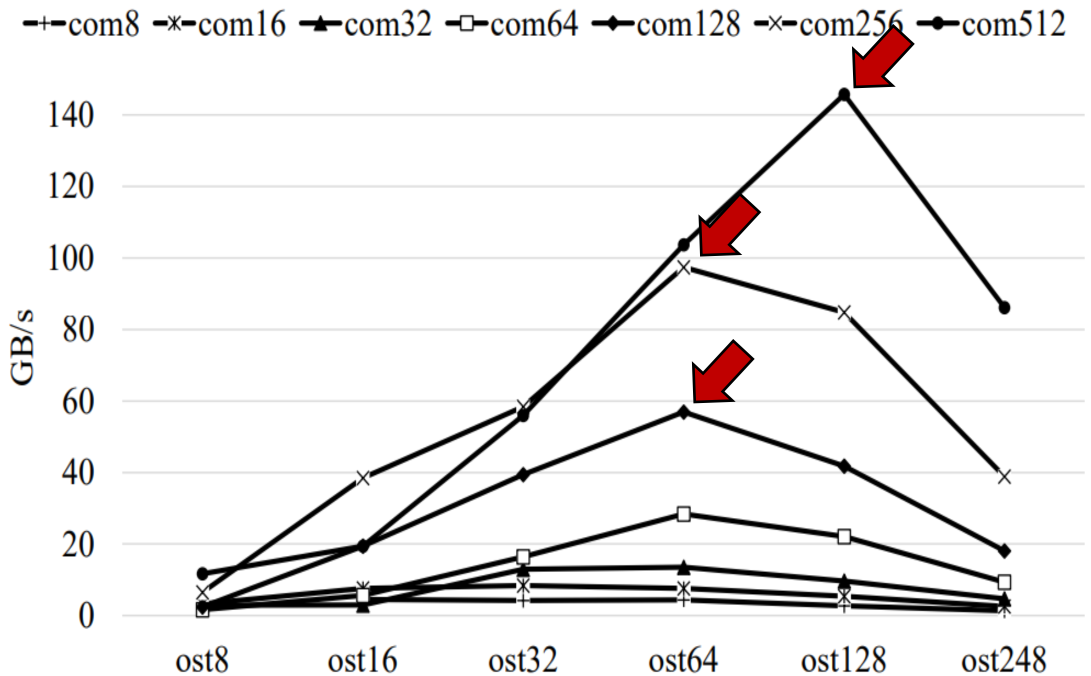
Independent I/O

- **The number of OSTs**
  - # of OSTs ↑ ➡ MPI throughput ↑ until certain point
    - Parallelism vs Contention in shared OSTs



Independent I/O

- **The number of OSTs**
  - # of OSTs ↑ ➡ MPI throughput ↑ until certain point
    - Parallelism vs Contention in shared OSTs
  - # of OSTs giving the best performance differs depending on the number of compute nodes

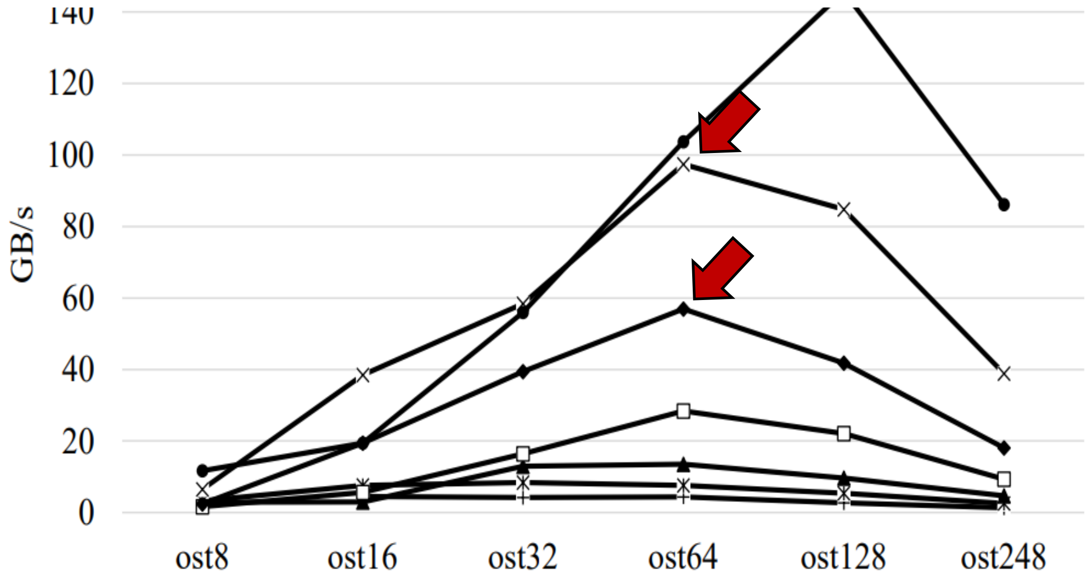


DCSLAB

Independent I/O

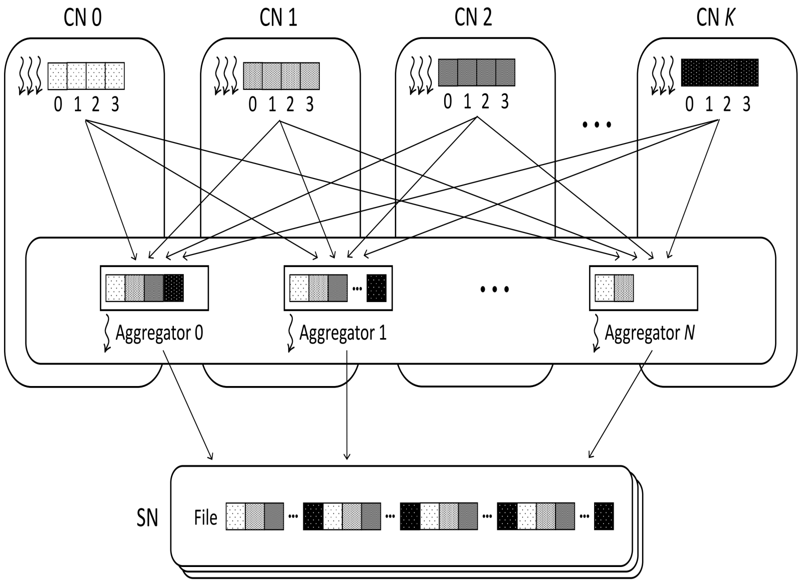
- **The number of OSTs**
  - # of OSTs ↑ ➡ MPI throughput ↑ until certain point
    - Parallelism vs Contention in shared OSTs
  - # of OSTs giving the best performance differs depending on the number of compute nodes

The number of OSTs showing the best performance gets larger as the number compute nodes increases



Collective I/O

- **Analysis setting**
  - 8 to 512 KNL compute nodes
    - 8, 16, 32, 64, 128, 256, 512 compute nodes
  - 32 cores per compute node
  - 8 to 248 OSTs
    - 8, 16, 32, 64, 128, 248 OSTs
  - 512GB output size
  - 16MB stripe size

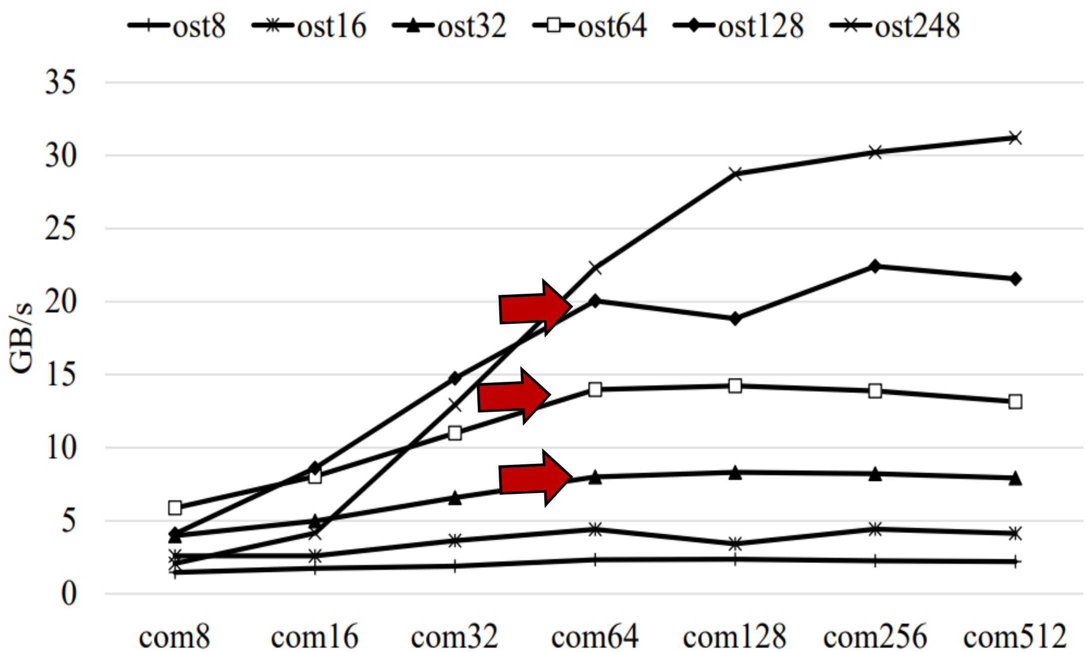


  - $\# \, of \, aggregators = \# \, of \, OSTs$
  - $\# \, of \, aggregators \, per \, compute \, node = \dfrac{\# \, of \, OSTs}{\# \, of \, compute \, nodes}$

DCSLAB

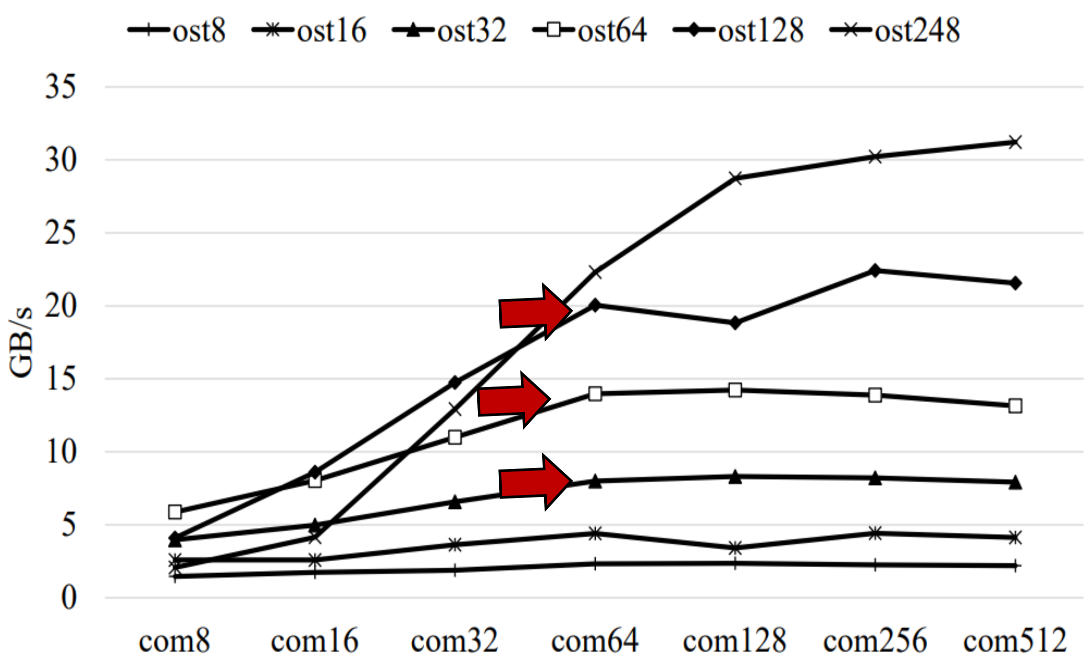
Collective I/O

- **The number of compute nodes**
  - # of compute nodes ↑ ➡ MPI throuhgput ↑ and saturates from the a certain point



Collective I/O

- **The number of compute nodes**
  - # of compute nodes ↑ ➡ MPI throuhgput ↑ and saturates from the a certain point
  - # of OSTs ↑ ➡ MPI throughput variation ↑

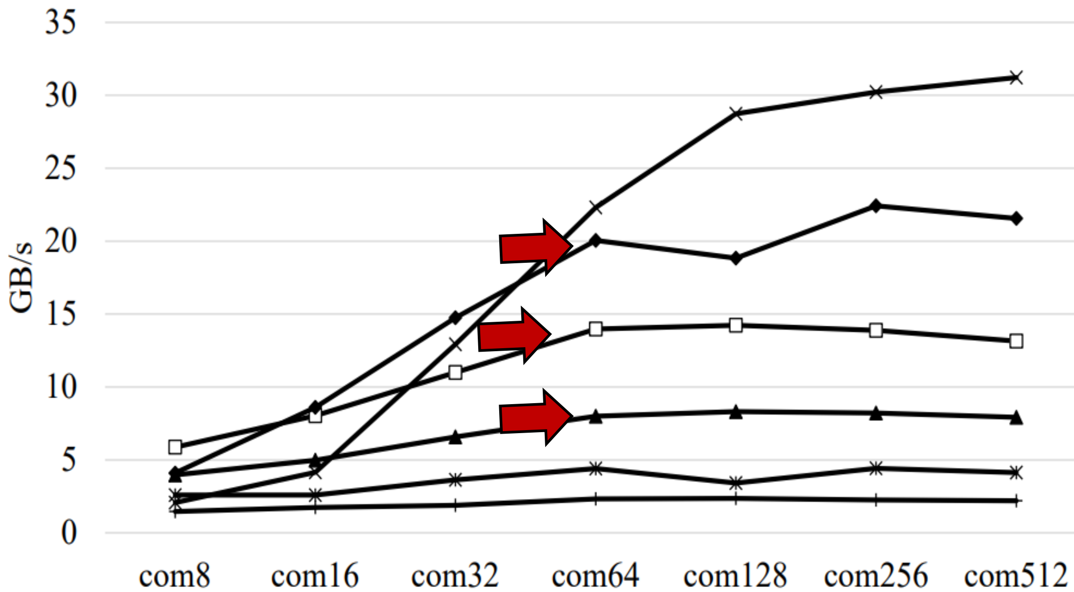


DCSLAB

Collective I/O

- **The number of compute nodes**
  - # of compute nodes ↑ ➡ MPI throuhgput ↑ and saturates from the a certain point
  - # of OSTs ↑ ➡ MPI throughput variation ↑

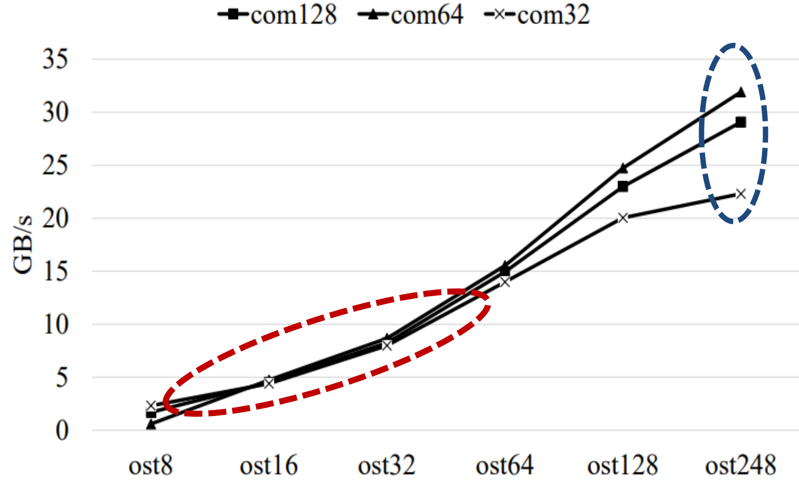
If the number of available OSTs is large enough, the greater the performance improvement can be achieved with more compute nodes



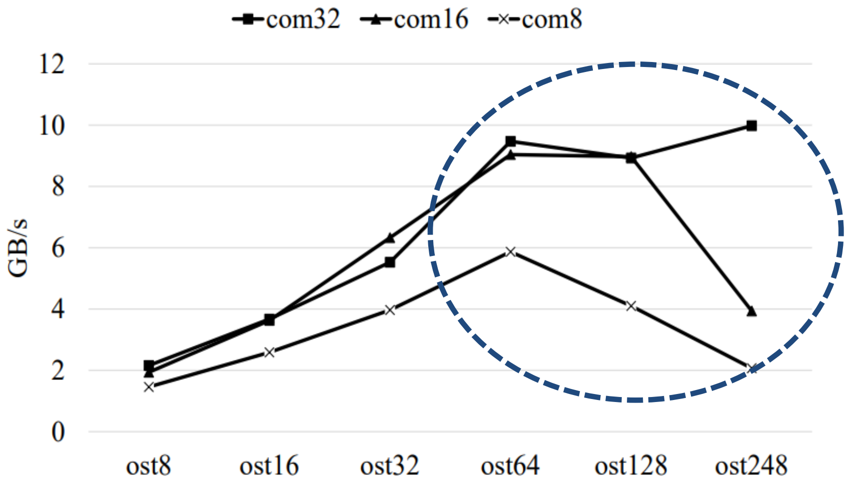
DCSLAB

Collective I/O

- **The number of cores per compute node**
  - Few aggregators per compute node ➡ Similar I/O throughput
  - Larger aggregators per compute node ➡ I/O throughput variation ↑



(b) 2048 threads

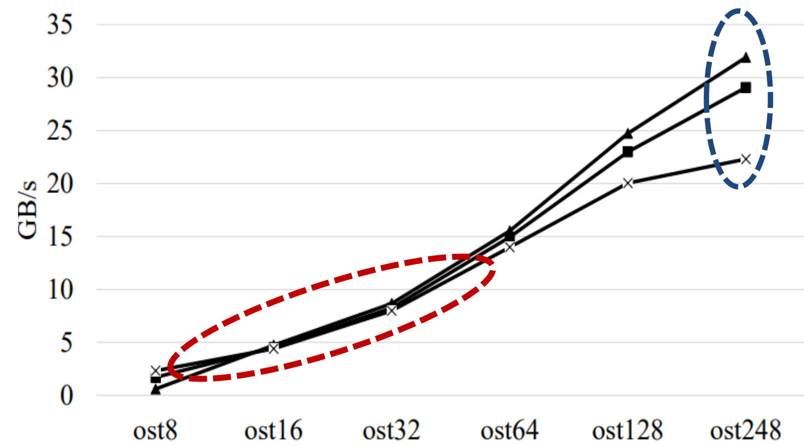


(c) 512 threads

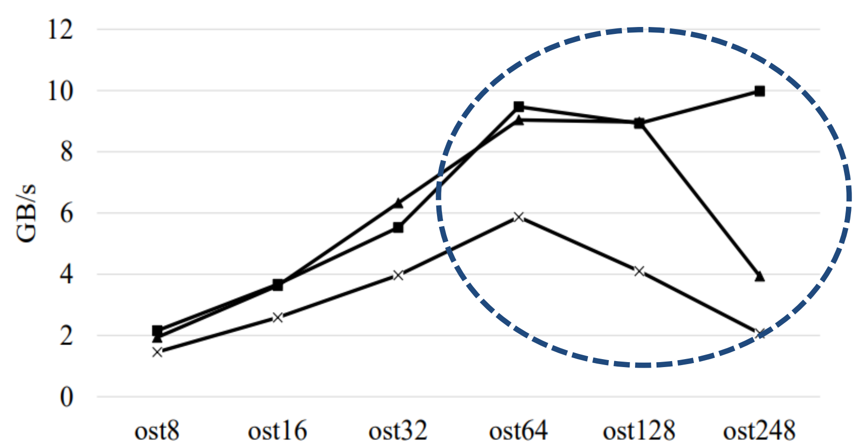
Collective I/O

- **The number of cores per compute node**
  - Few aggregators per compute node ➡ Similar I/O throughput
  - Larger aggregators per compute node ➡ I/O throughput variation ↑

The larger the number of aggregators in the single compute node, the better performance can be achieved by spreading the aggregators to more compute nodes



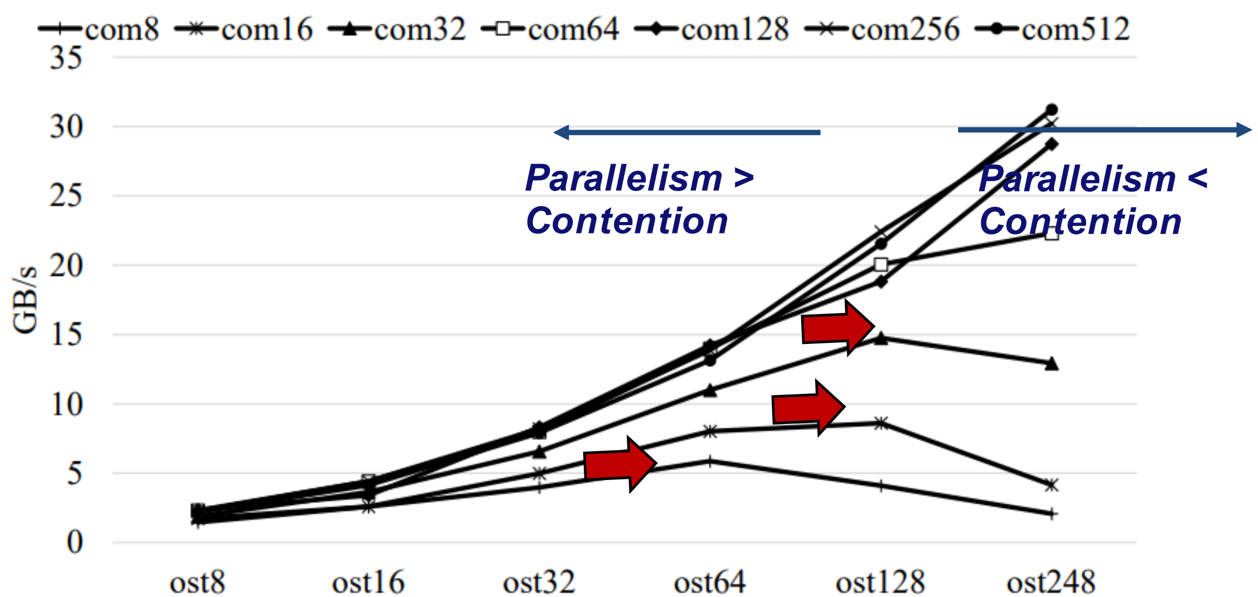
(b) 2048 threads



(c) 512 threads

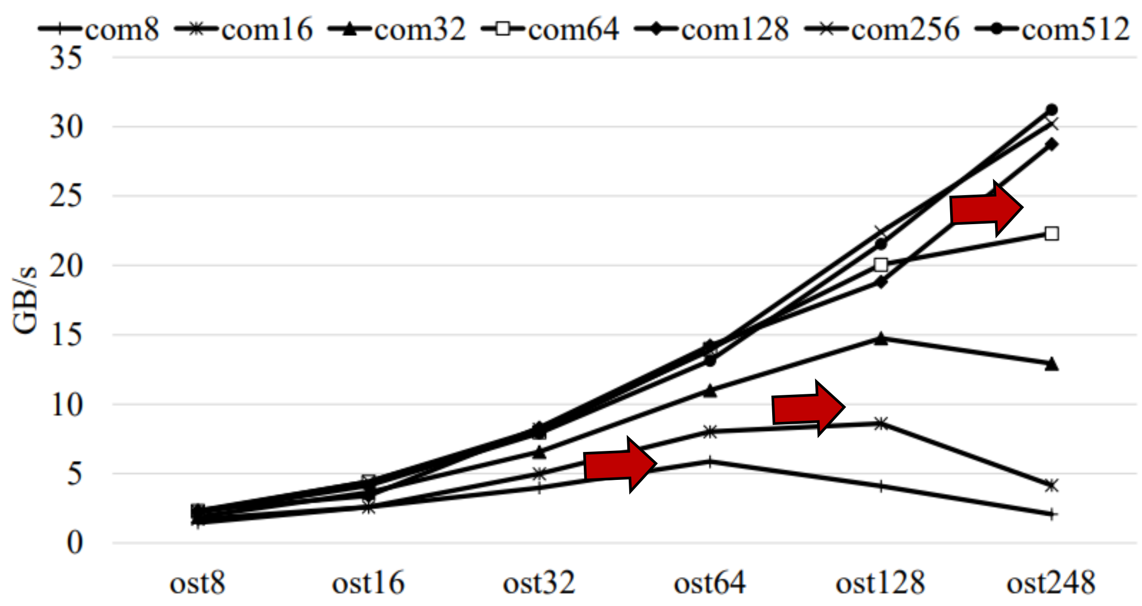
Collective I/O

- **The number of OSTs**
  - # of OSTs ↑ ➡ MPI throughput ↑ until a certain point



Collective I/O

- **The number of OSTs**
  - # of OSTs ↑ ➡ MPI throughput ↑ until a certain point
  - # of OSTs showing the best performance is different depending on the # of compute nodes

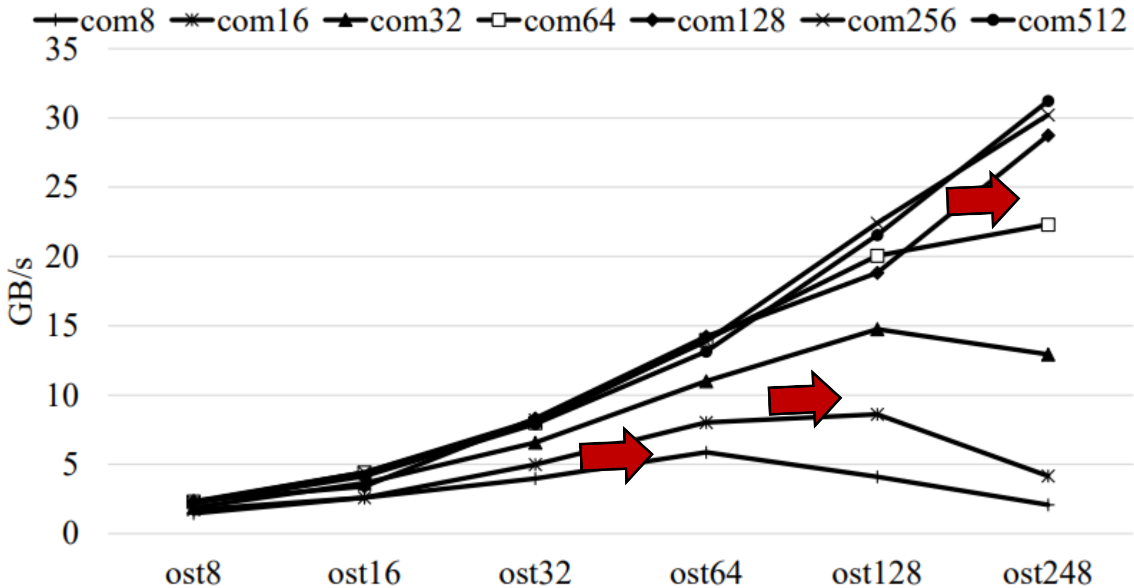


DCSLAB

Collective I/O

- **The number of OSTs**
  - # of OSTs ↑ ➡ MPI throughput ↑ until a certain point
  - # of OSTs showing the best performance is different depending on the # of compute nodes

When the number of aggregators in each compute nodes is small, it is better to use more OSTs for better performance



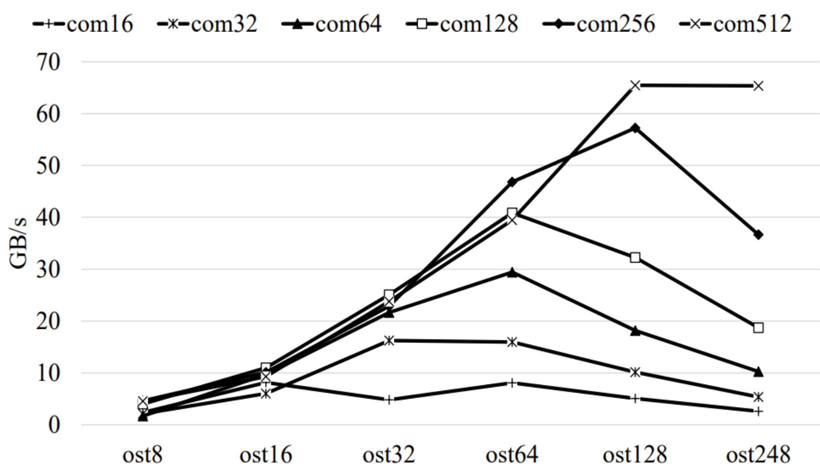
DCSLAB

Prediction Accuracy

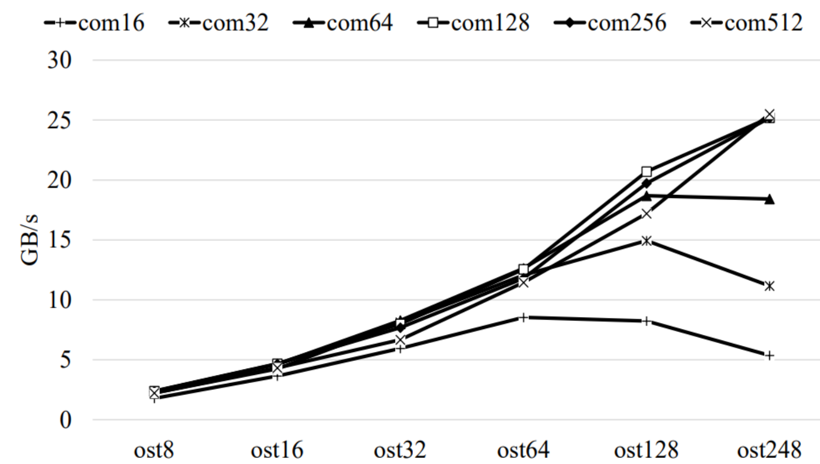
- **Comparing the performance trends with other HPC workloads under similar evaluation setting**
  - VPIC-IO for independent I/O
    - Plasma physics simulation's particle data write phase
  - Chombo for collective I/O
    - Adaptive mesh refinement scientific applications

Performance Trends of HPC Applications

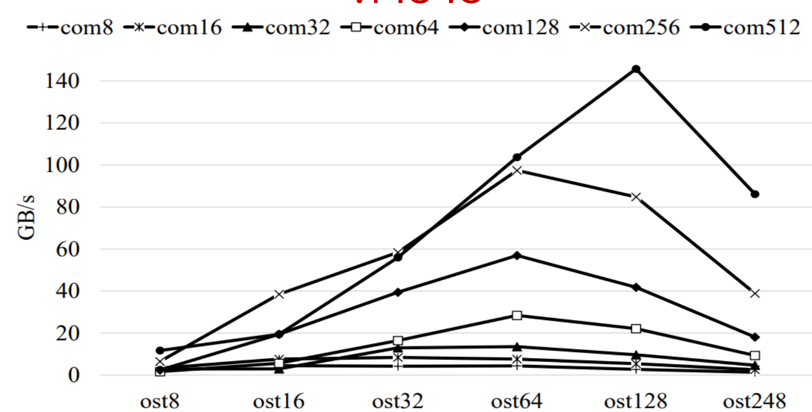
- **Result**
  - Both VPIC-IO and Chombo show similar performance trends as that of analysis using IOR

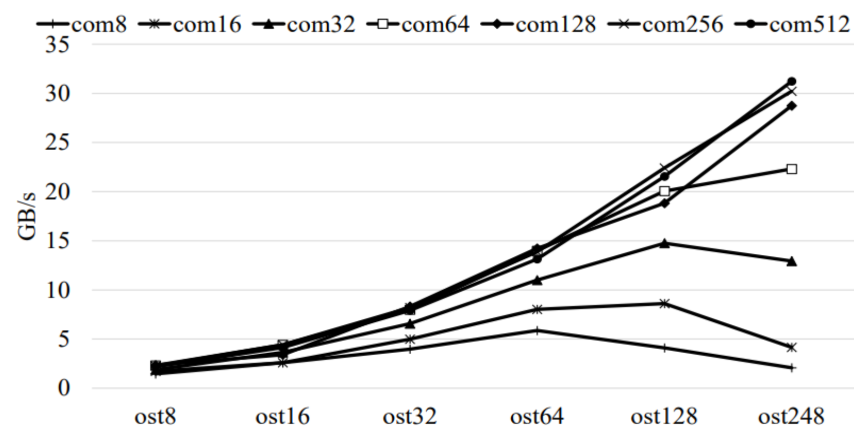


VPIC-IO



Chombo



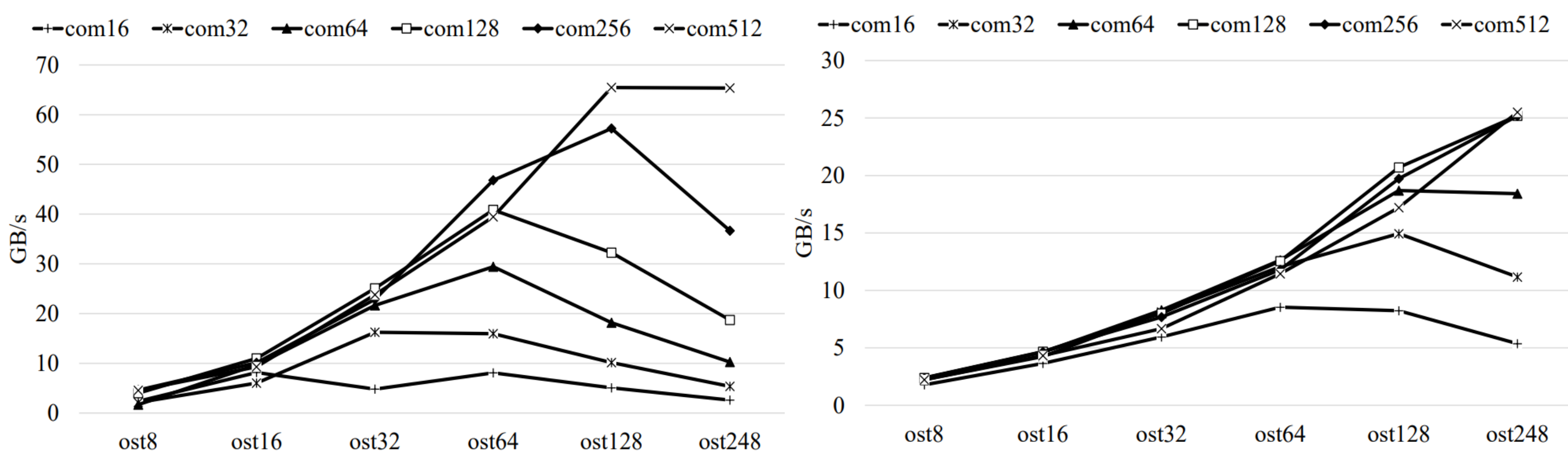


IOR with Collective I/O

IOR with Indenpend I/O

Performance Trends of HPC Applications

- **Result**
  - Both VPIC-IO and Chombo show similar performance trends as that of analysis using IOR
  - Performance trends of the other HPC workloads can be predicted by the analysis through the synthetic workload IOR
  - It is expected to be able to predict the best configuration that gives the highest performance



(a) VPIC-IO (b) Chombo

DCSLAB

Conclusion

- **Summary**
  - We analyzed the performance trends by changing the tunable parameters using the synthetic workload IOR
  - The different I/O characteristics show different performance trends depending on the configurations
  - The performance trends change due to the parallelism and the contention for the shared resource
  - Other HPC workloads, VPIC-IO and Chombo, showed similar performance trends to that of IOR
- **Future work**
  - Considering other I/O characteristics
  - Creating a recommend platform