System Software Research

MPI-IO for Node-local Persistent Memory in HPC applications

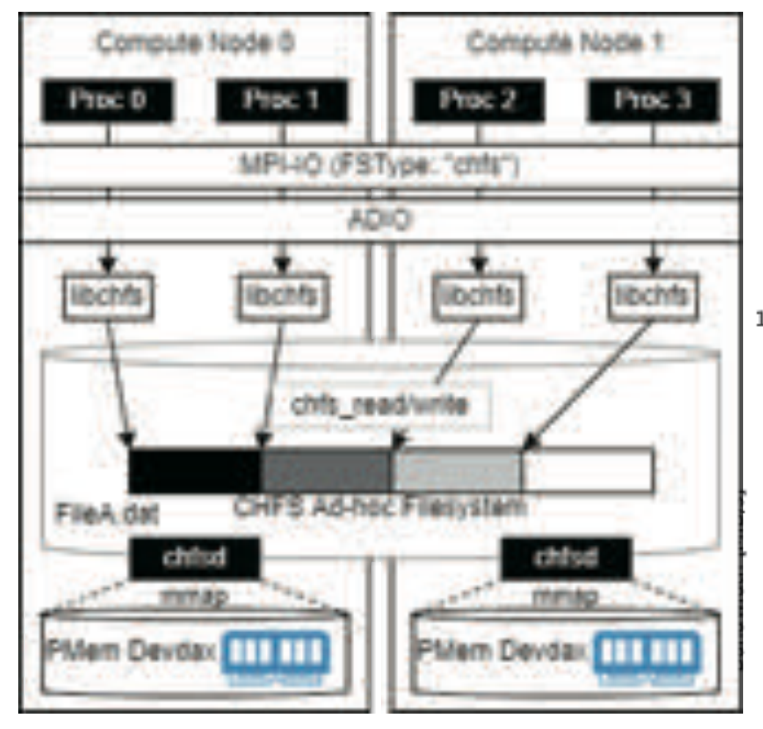
2 type of utilization strategies

MPI-IO/PMem

• Compute node-local Persistent Memory as a burst buffer

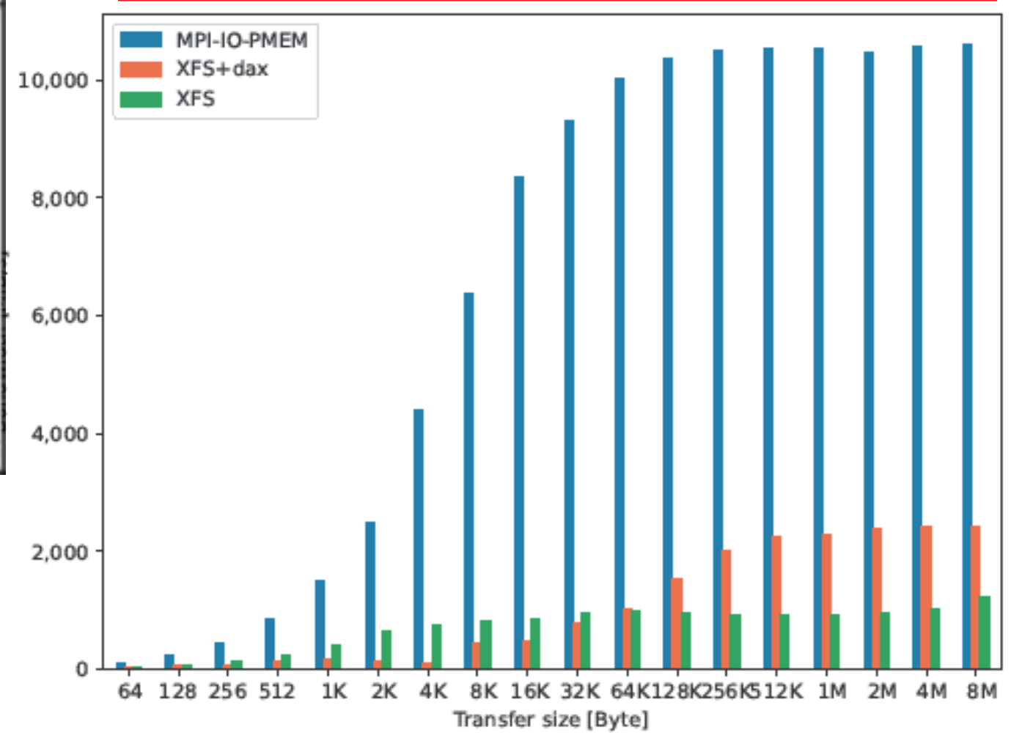
MPI-IO/CHFS

• Utilize node-local PMem via CHFS Ad-hoc distributed filesystem

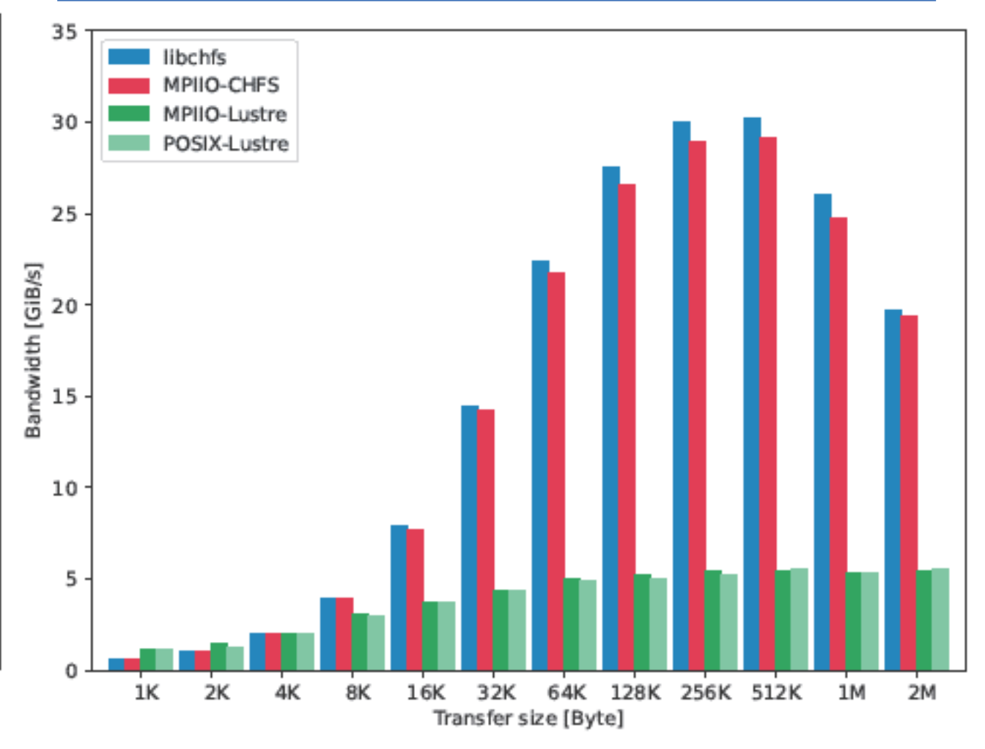


System Overview
(MPI-IO/CHFS)

MPI-IO/PMem
IOR Write BW on a single node
4.39x better than XFS-DAX



MPI-IO/CHFS
IOR Write BW on 4 nodes
5.34x better than Lustre



Accelerating Exploratory Data Analysis with Ad Hoc Parallel Filesystem

Background: Data analysts are looking for high-performance analysis systems to reduce the time to analyze big data. However, workloads are I/O intensive, and each attempt takes time in I/O.
Proposal: We propose the way to use ad hoc parallel filesystem as a fast caching system, focusing on the fact that data analysis, especially exploratory data analysis (EDA), reads the same data many times.
Implementation: We implemented Dask integration with state-of-the-art CHFS as an ad hoc parallel filesystem.

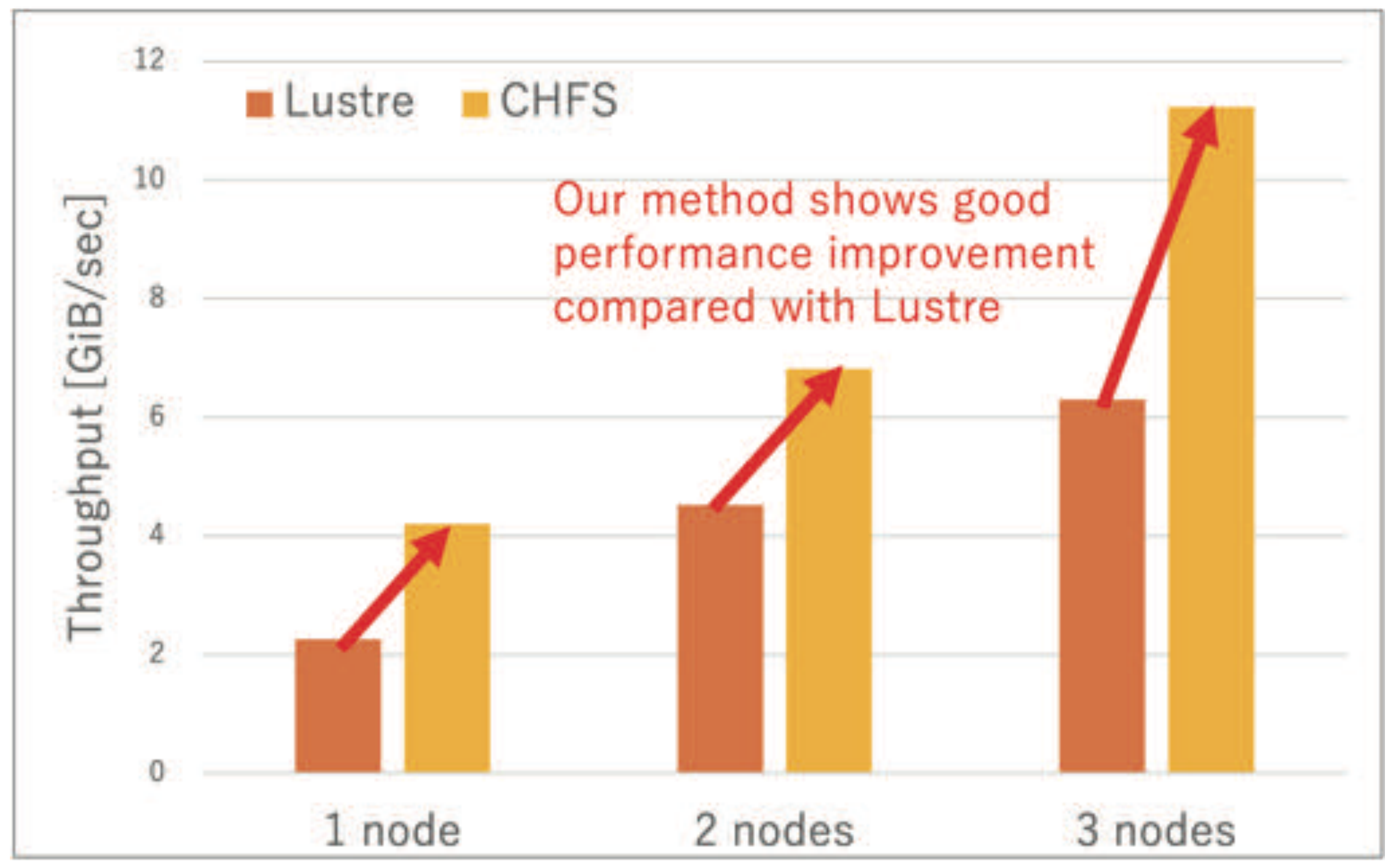
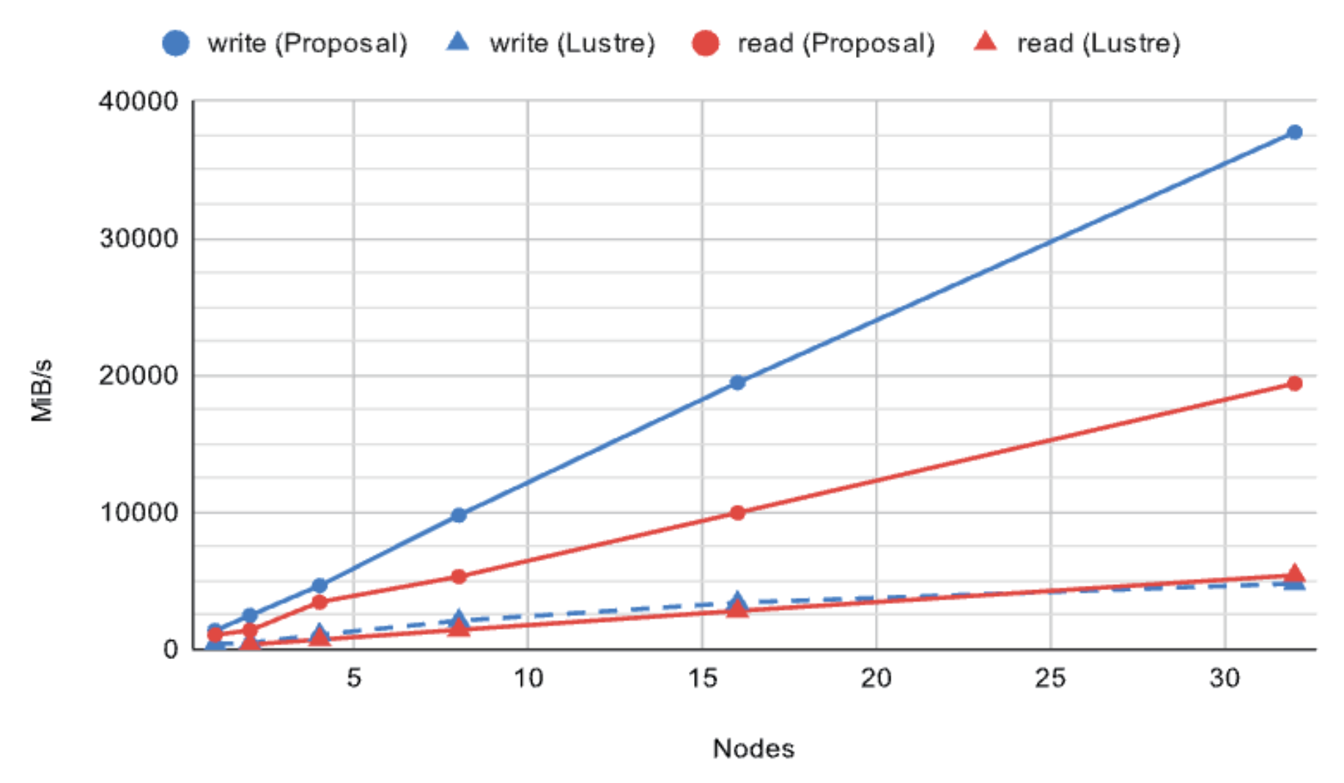
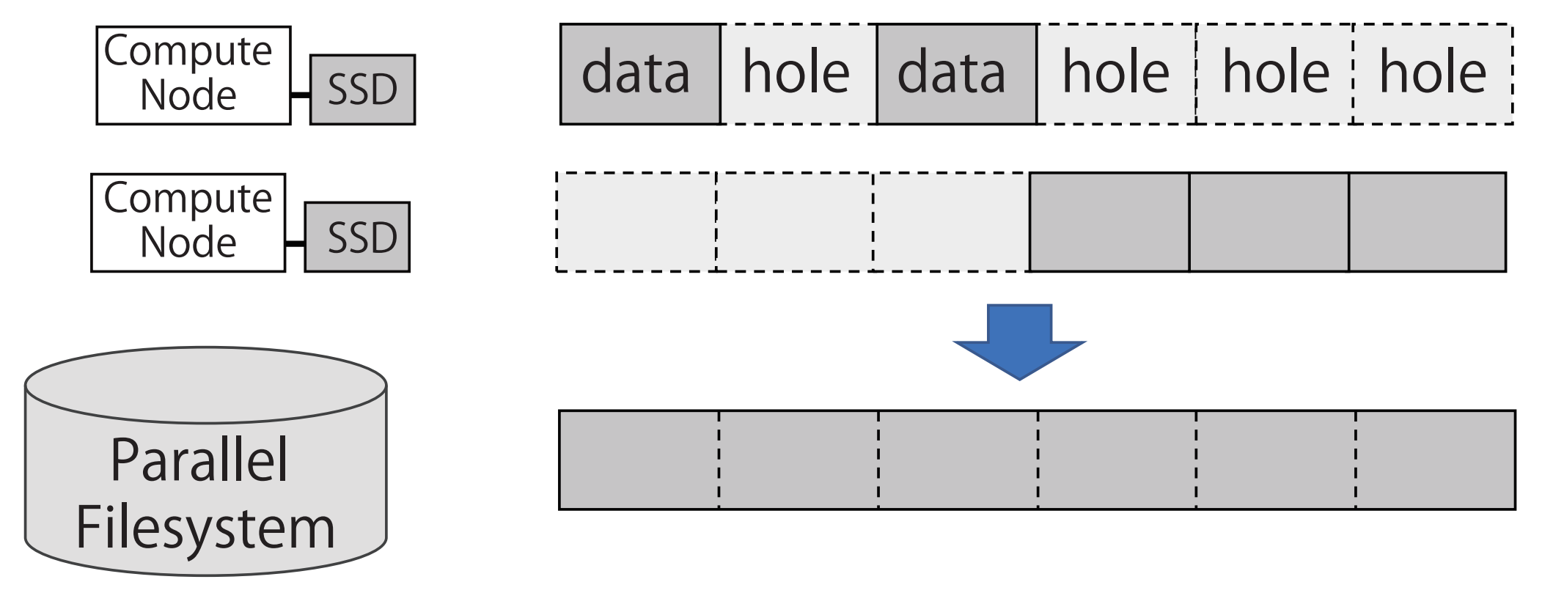


Fig 1: This benchmark reads 700GiB data per node from CHFS and Lustre through Dask. The throughput is calculated by the amount of data divided by "nanmean" computation time.

Design and Implementation of Distributed Caching Filesystem Using Node Local Storage

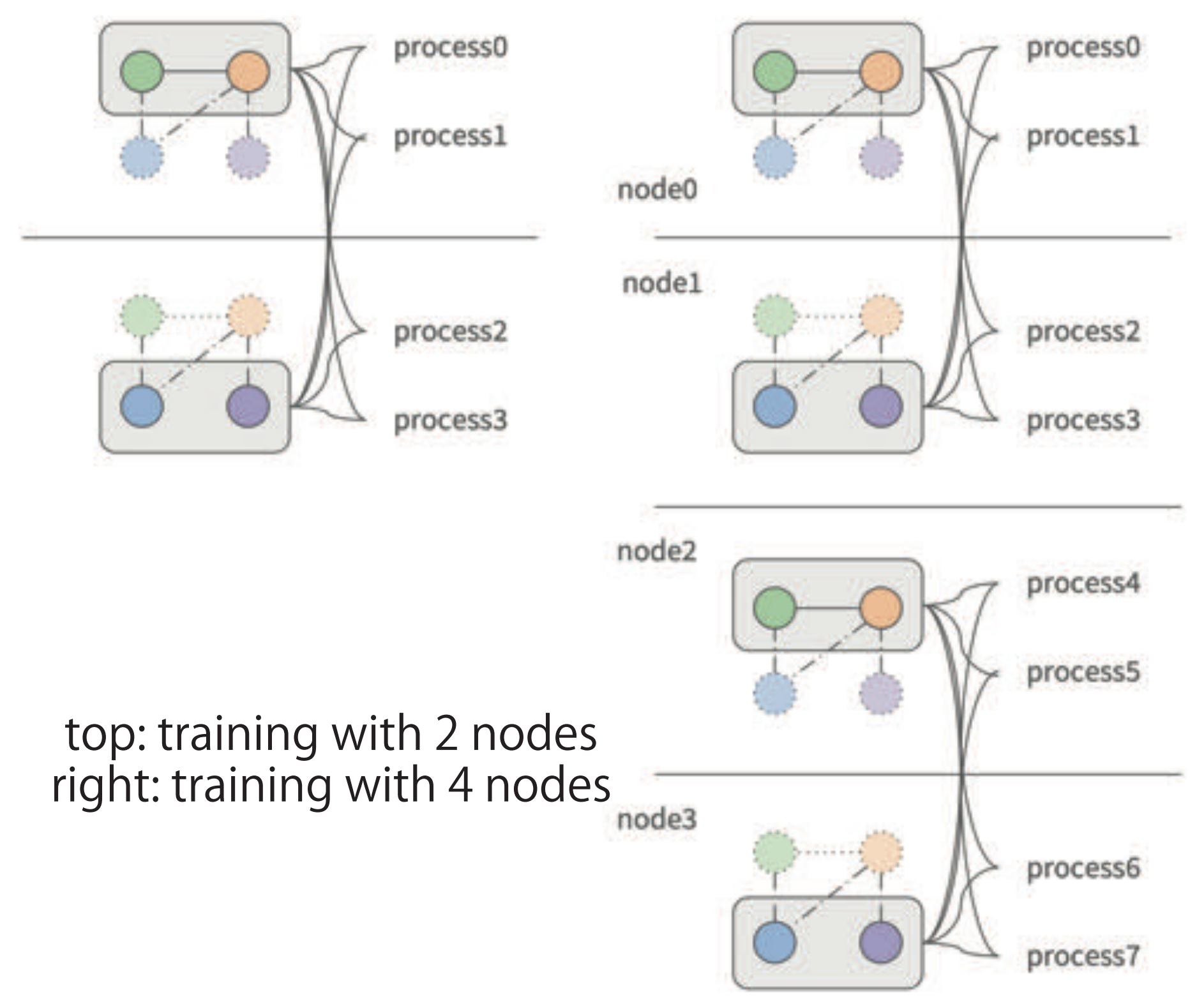
Background: Modern high-performance computing (HPC) systems are introducing large and fast storage device such as NVMe SSDs to improve storage performance.
Proposal: We proposed distributed filesystem using node local storage as a cache of parallel file system. The file system exploits locality of data access and reduces communications between compute nodes.
Implementation: We implemented local write/read and achieved scalable I/O bandwidth with local Single Shared File access of IOR.



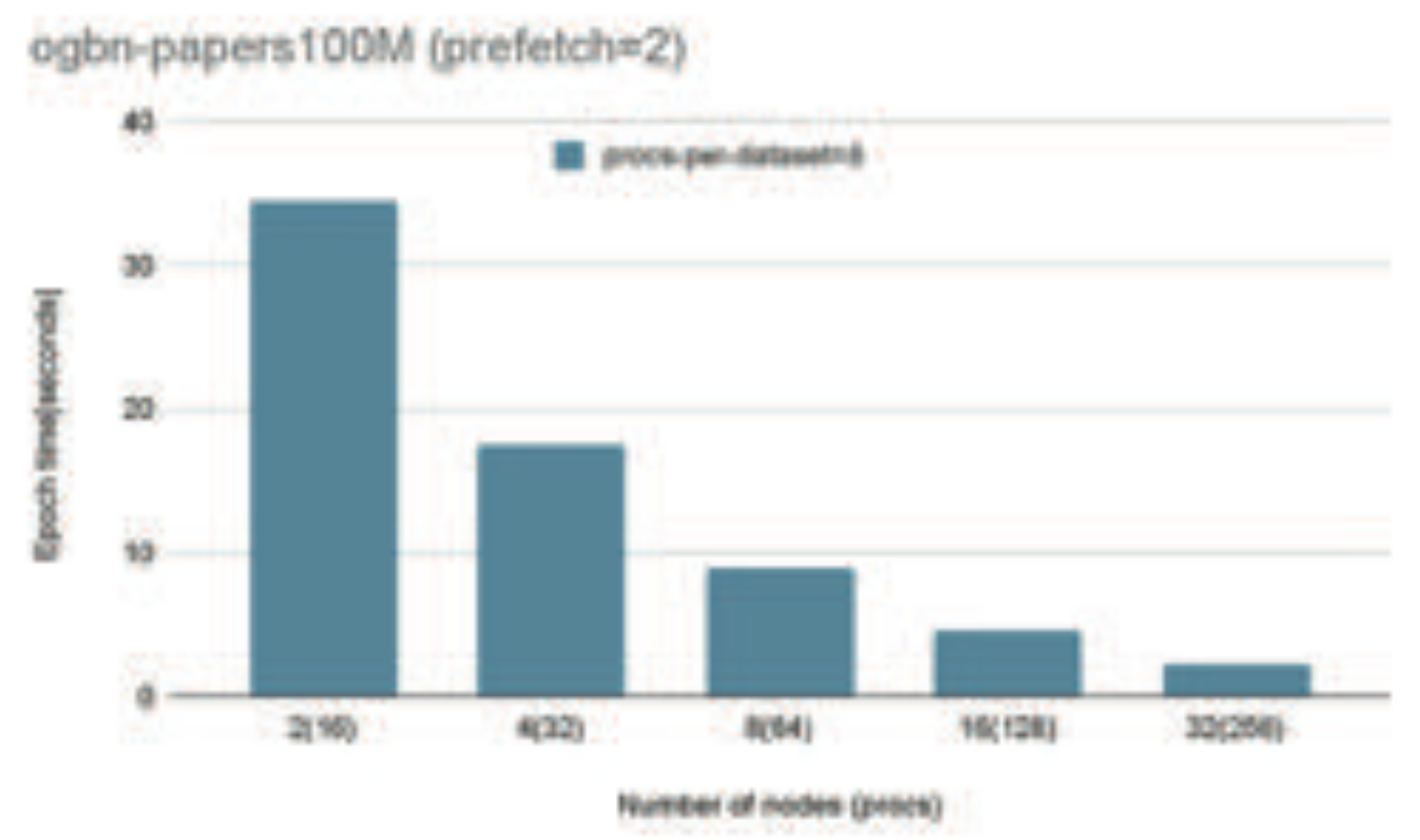


Scalable Data Parallel Distributed Training for Graph Neural Networks

Background: Existing distributed GNN training systems partition a graph among all compute nodes to train for large graphs; however, this results in a communication overhead to degrade training performance.
Design and Implementation: To solve these two problems, we propose a scalable data-parallel distributed GNN training system designed to partition a graph redundantly. It is implemented using remote direct memory access (RDMA) and nonblocking active messages to efficiently utilize network performance and hide communication overhead by overlapping with the training computation.
Performance Evaluation: Experimental results are presented to show the strong scalability of the proposed approach, which achieved parallel efficiencies of 0.95 based on two compute nodes using 32 compute nodes for the ogbn-papers100M dataset.



top: training with 2 nodes
right: training with 4 nodes



Sohei Koyama, Osamu Tatebe, "Scalable Data Parallel Distributed Training for Graph Neural Networks", Proceedings of Workshop on AI for Datacenter Optimization (ADOPT'22), pp.699-707, 10.1109/IPDPSW55747.2022.00121, 2022