Numerical Computation

Implementation of Parallel 3-D Real FFT with 2-D Decomposition on Intel Xeon Phi Clusters

Background

The fast Fourier transform (FFT) is an algorithm which is currently widely used in science and engineering. A typical decomposition for performing a parallel 3-D FFT is slabwise. This becomes an issue with very large MPI process counts for a massively parallel cluster of many-core processors.

Overview

We proposed an implementation of a parallel 3-D real FFT with 2-D decomposition on Intel Xeon Phi clusters. The proposed implementation of the parallel 3-D real FFT is based on the conjugate symmetry property of the discrete Fourier transform (DFT) and the row-column FFT algorithm. We vectorized FFT kernels using the Intel Advanced Vector Extensions 512 (intel AVX-512) instructions.

Performance

To evaluate the implemented 3-D real FFT with 2-D decomposition, referred to as FFTE 7.0 (2-D decomposition), we compared its performance with that of the FFTE 7.0 (1-D decomposition), the FFTW 3.3.8 and the P3DFFT 2.7.7. The performance results demonstrate that the proposed implementation of parallel 3-D real FFT with 2-D decomposition effectively improves performance by reducing the communication time for larger numbers of MPI processes on Intel Xeon Phi clusters.

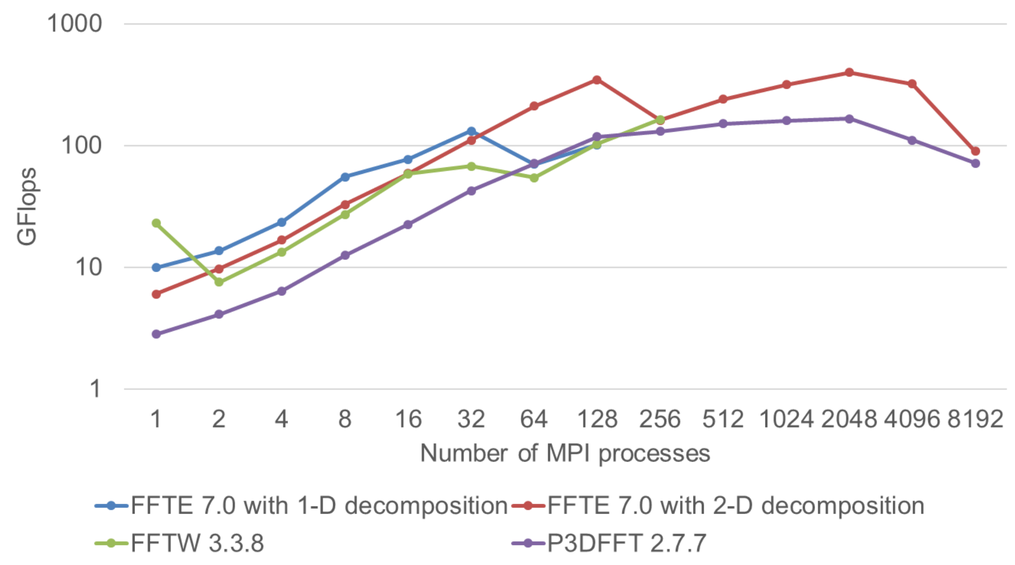

Fig. 1: Performance of Parallel 3-D Real FFTs ($N = 256 \times 512 \times 512$)

Development of the high accurate Block Krylov solver

Linear systems with multiple right-hand sides appear in many scientific applications such as the computation of physical quantity in lattice Quantum Chromodynamics (QCD), inner problems of eigensolvers for sparse matrix, and so on. As numerical methods for solving these linear systems, it is known that Block Krylov subspace methods are efficient methods in terms of the number of iterations and the computation time. However, the accuracy of the obtained solution may often deteriorate due to the error occurs in the computation of matrix-matrix multiplications. To improve the accuracy of the obtained solution, we have developed the new Block Krylov subspace method named Block GWBiCGSTAB method [1]. The Block GWBiCGSTAB method is based on the group-wise updating technique. By using this technique, the matrix-matrix multiplications that cause accuracy degradation can be avoided. As shown in Fig. 1, the accuracy of the obtained solution generated by the Block GWBiCGSTAB method is higher than that by other methods.

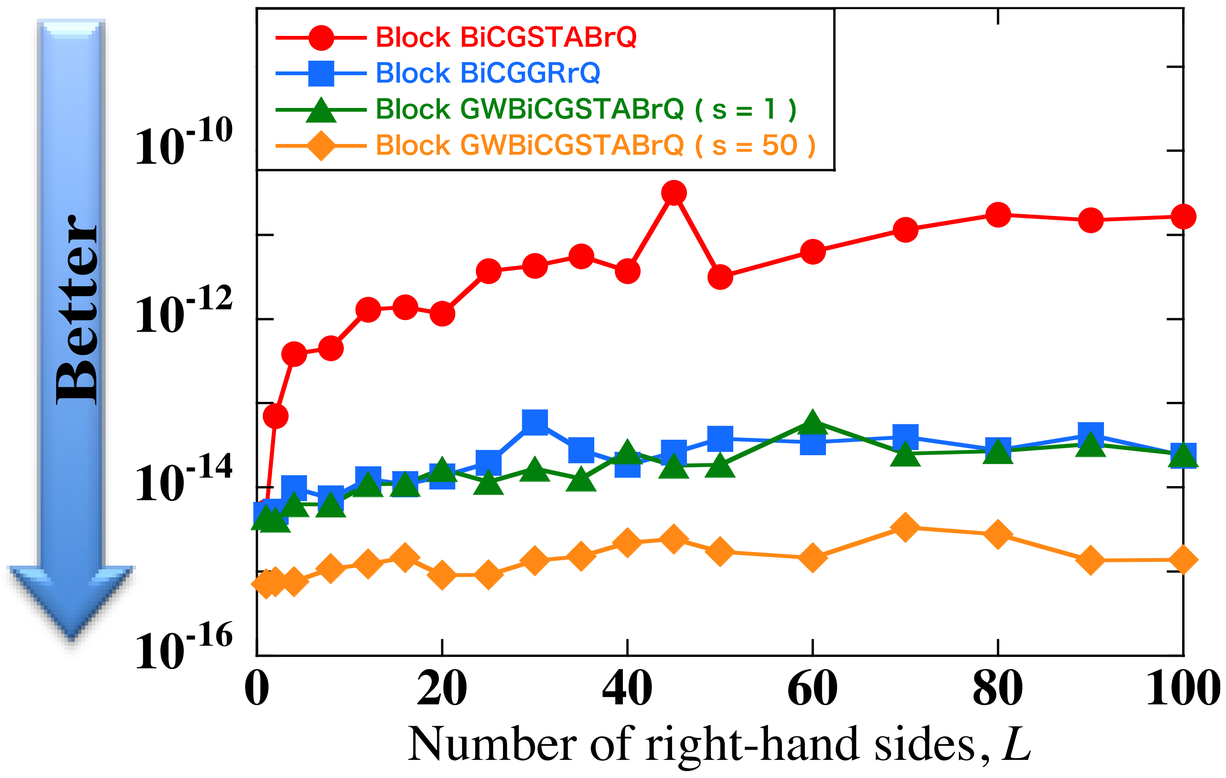

Fig. 2: True relative residual norm as a function of the number L of right-hand sides. The test problem is the linear system derived from the lattice QCD calculation. Problem size: 1,572,864.

[1] Hiroto Tadano and Ryosei Kuramoto, Accuracy improvement of the Block BiCGSTAB method for linear systems with multiple right-hands sides by group-wise updating technique, J. Adv. Simulat. Sci. Eng., Vol. 6, No. 1, pp. 100—117, 2019.

Software Researches for Big Data and Extreme-Scale Computing

Gfarm/BB – Gfarm File System for Node-local burst buffer

http://oss-tsukuba.org/en/software/gfarm

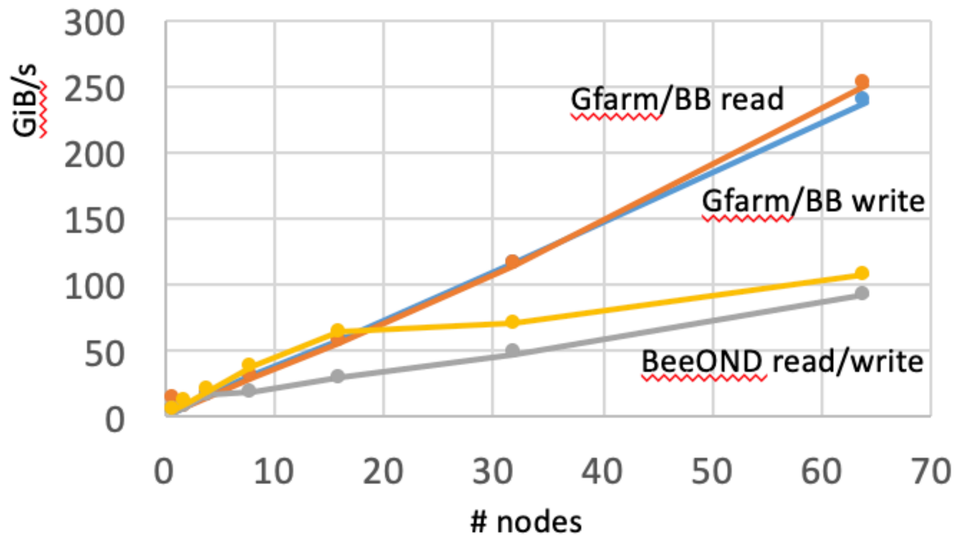

Fig. 1: IOR file-per-process read/write performance on Cygnus supercomputer

```
gfarmbb –h hostfile –m mount_point start
…
gfarmbb –h hostfile stop
```

Features include
- Open source
- Exploit local storage and data locality for scalable I/O performance
- InfiniBand support
- Data integrity is supported for silent data corruption
- Production systems: 8PB JLDG, 100PB HPCI Storage, etc.

Accelerating Python Applications with Persistent Memory

Python is one of the most popular general-purpose programming languages, and persistent memory (PMEM) is a new device which can accelerate data-intensive computing. There is a strong demand to use persistent memory from Python easily. Therefore, we focus on pmemkv, which is a key-value store optimized for persistent memory, and its python bindings. We are currently evaluating pmemkv's python bindings in detail for efficient use of PMEM in Python.

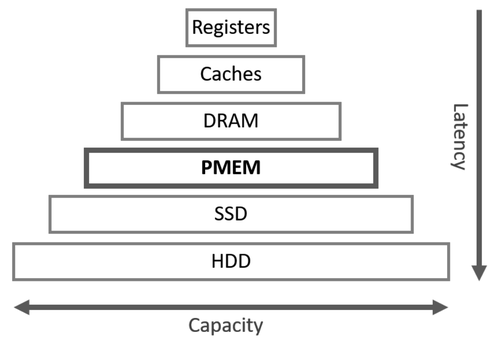

Fig.2a: Memory-storage hierarchy with persistent memory

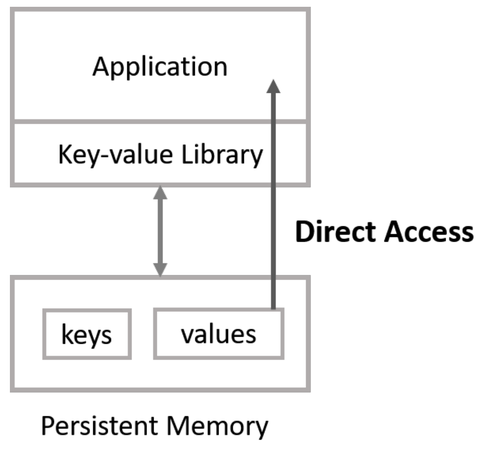

Fig2b: Applications can directly access the persistent memory resident data structures without using buffers.

Investigate DAOS architecture for metadata operation

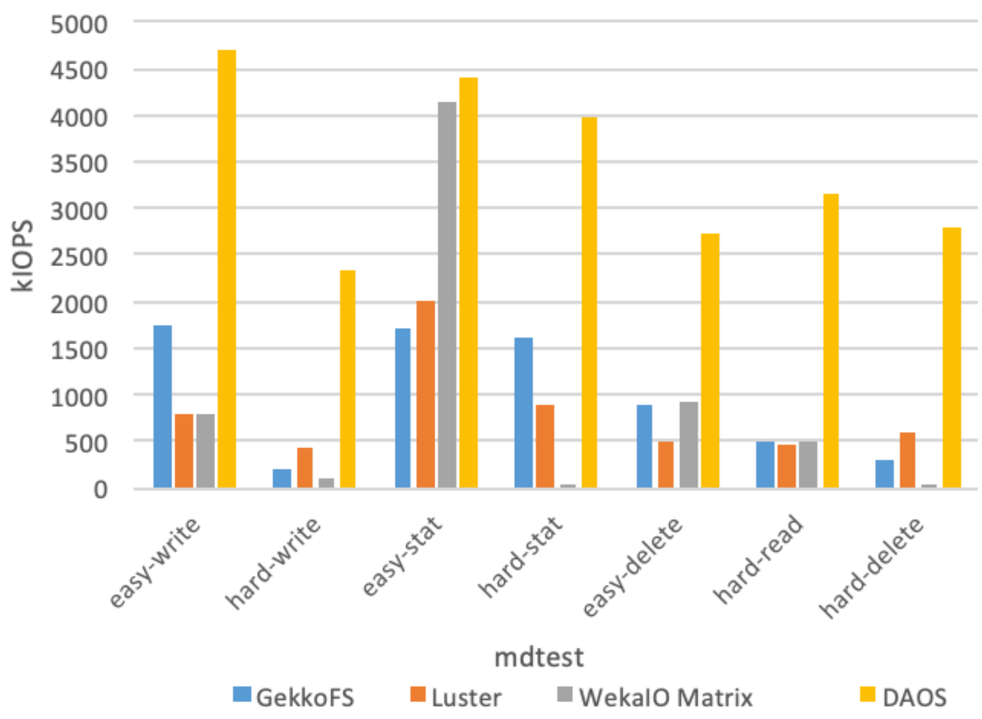

Fig. 3: mdtest performance comparison of IO-500 10 node challenge scores

The open-source DAOS – Distributed Asynchronous Object Storage – is notable for its rank on the IO-500 list and its use of Intel® Optane™ Persistent Memory. In particular, metadata performance is remarkable compared to other systems.

We investigate the reason for DAOS remarkable metadata performance on its architecture and consider to integrate DAOS ways to an existing system or develop a new storage system with persistent memory.

Research of caching file system to exploit node local storages

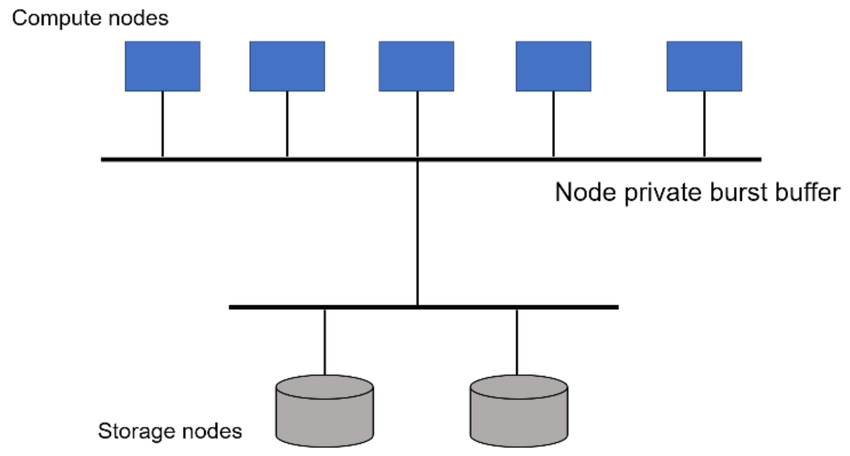

Fig. 4: Automation of construction/destruction a swarm cluster

The performance gap between processors and disk-based storage is growing in modern HPC systems. To reduce the gap, SSDs attached to compute nodes has been used as a "node local burst buffer". We are implementing distributed file system that uses local SSDs as a caching layer of the storage nodes. The system uses fuse-library for system call replacing and mochi-framework for RPC data transfer.

Acceleration of Deep Learning using pytorch with persistent memory

Persistent memory offers greater capacity than DRAM and significantly better performance than storage. We use it for deep learning with pytorch. Usually, before performing deep learning using the GPU, the training data is copied to the main memory from the storage. We exploit the persistent memory to improve the performance.

Acknowledgment
This work is partially supported by Multidisciplinary Cooperative Research Program in CCS, University of Tsukuba, New Energy and Industrial Technology Development Organization (NEDO), and Fujitsu Laboratories Ltd.