# Division of High Performance Computing Systems

Chief: BOKU Taisuke, Professor, Ph.D., Director of CCS

The High-Performance Computing Systems Division conducts research and development of various hardware and software for High Performance Computing (HPC) to satisfy the demand for ultrafast and high-capacity computing to promote cutting-edge computational science. In collaboration with the Center' s teams in domain science fields, we aim to provide the ideal HPC systems for real-world problems.

Research targets a variety of fields, such as high-performance computer architecture, parallel programming languages, large-scale parallel numerical calculation algorithms and libraries, computational acceleration systems such as Graphics Processing Units (GPUs), Field Programmable Gate Array (FPGA), large-scale distributed storage systems, and grid/cloud environments.

## Fundamental technologies for parallel I/O and storage systems

Storage performance has become a bottleneck in large-scale data analysis and artificial intelligence (AI) using Big Data on supercomputers. To reduce this performance bottleneck, we are researching and developing fundamental technologies for parallel input/output (I/O) and storage systems. We developed Gfarm/BB, which can temporarily configure a parallel file system when parallel applications are executed, by utilizing the storage system of a compute node. We also designed and implemented a standard library for parallel I/O (NSMPI), which efficiently utilizes the storage system of a compute node. Both of them exhibit scalable performance depending on the number of nodes and contribute significantly to reducing performance bottlenecks.

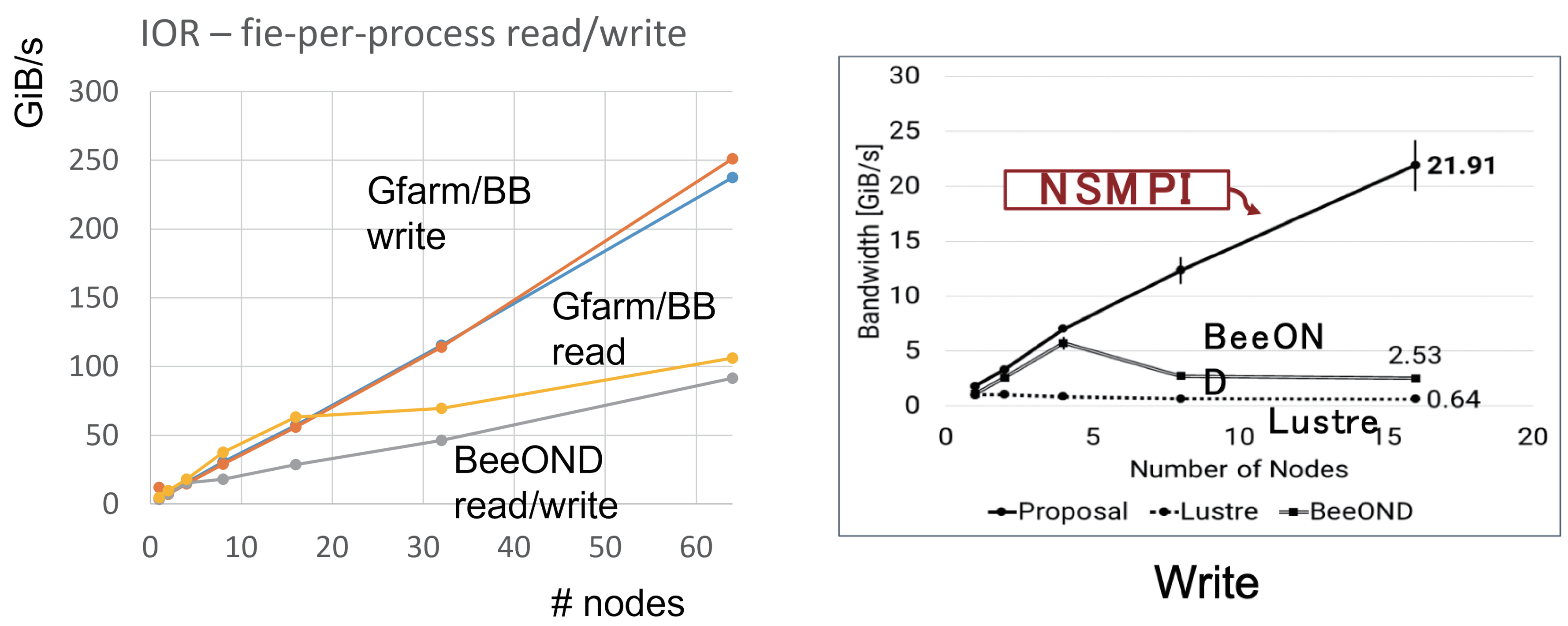


**Fig. 1 left: Gfarm/BB read and write performance**
**right: NSMPI write performance**

## High-performance, massively parallel numerical algorithms

FFTE—an open-source high-performance parallel FFT library developed by our division—has an auto-tuning mechanism and is suitable for a wide range of systems (from PC clusters to massively parallel systems). We are also developing large-scale linear computation algorithms. We constructed a method using the matrix structure of saddle point problems that enables faster solutions than existing methods.

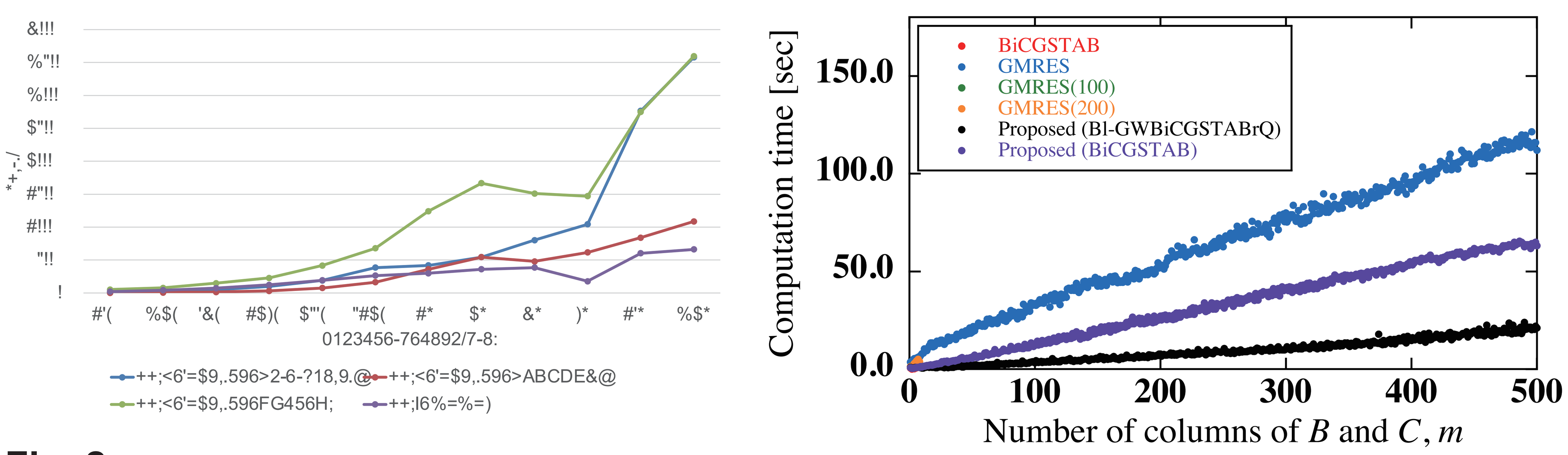


**Fig. 2**
**left: Performance of parallel one-dimensional FFT on Oakforest-PACS (1024 nodes)**
**right: solution time of saddle point problems**

## GPU computing

GPUs were originally designed as accelerators with a large number of arithmetic units for image processing, but they are also able to perform general-purpose calculations. GPUs have been used to accelerate applications but are now used to perform larger-scale calculations, often with longer execution times. In shared systems such as supercomputers, there are execution time limits, but time limits can be exceeded using a technique called checkpointing, which is used to save the application state to a file in the middle of an execution so that the application can be resumed later.

Because the checkpointing technique for applications using only the central processing unit (CPU) is well established, we extended it to the GPU. The GPU has its own memory separate from the CPU, and GPU applications control the GPU with dedicated application programming interface (API) functions from the CPU. We monitor all the calls to these API functions, collect the necessary information to resume execution, and store it in the CPU memory. Additionally, in the pre-processing of saving the checkpoint, the data in the GPU' s memory is collected on the CPU side so that they can be saved together in a file. Because we cannot increase the execution time to enable checkpoints, we must minimize the overhead of monitoring the API. It is important to minimize the number of API functions to be monitored and to save only the data that are needed.