III. Reports on Research Activities

1. System Software Research Team

1.1. Team members

Yutaka Ishikawa (Team Leader)
Atsushi Hori (Researcher)
Keiji Yamamoto (Postdoctoral Researcher)
Yoshiyuki Ohno (Research Associate)
Toshihiro Konda (Research Associate)
Toyohisa Kameyama (Technical Staff)

1.2. Research Activities

The system software team focuses on the research and development of an advanced system software stack not only for the "K" computer but also for towards exa-scale computing. There are several issues in carrying out future computing. Two research categories are taken into account: i) scalable high performance libraries/middleware, such as file I/O and low-latency communication, and ii) a scalable cache-aware, power-aware, and fault-aware operating system for next-generation supercomputers based on many core architectures.

Parallel file I/O is one of the scalability issues in modern supercomputers. One of the reasons is due to heavy metadata accesses. If all processes create and write different files, the metadata server receives so many requests by all processes not only at the creation time but also at writing data to each file. Three approaches have been conducted to mitigate this issue. One approach is to introduce a file composition technique that gathers multiple data generated by an application and stores these data into one or a few files in order to reduce the number of files accessed by processes. Another approach is to provide multiple metadata server in which the requests for metadata are sent to a metadata server resolved using hash function. The third approach is to provide a smart MPI-IO implementation for applications using MPI-IO functions.

Increasing number of cores and nodes enforces strong scaling on parallel applications. Because the ratio of communication time against local computation time increases, a facility of low-latency and true overlapping communication and computation communication is desired. A communication library, integrated to the MPI library implementation in K computer, has been designed and implemented, that utilizes DMA engines of K computer. Each compute node of K computer has four DMA engines to transfer data to other nodes. If a communication library knows communication patterns in advanced, it may utilize the DMA engines. Indeed, the feature of MPI persistent communication, standardized in MPI-1.0, allows the runtime library to optimize data transfers

involved in the persistent communication using the DMA engines.

System software stack developed by our team is designed not only for special dedicated supercomputers, but also for commodity-based cluster systems used in research laboratories. The system will be expected to be used as a research vehicle for developing an exa-scale supercomputer system.

1.3. Research Results and Achievements

1.3.1. Scalable File I/O

- File Composition Library

  A file composition technique has been proposed, where lots of file data generated by an application are gathered and stored into one file or a few files. Figure 1 shows the basic concept of the file composition technique. File composition technique composes data, which looks like a file from the viewpoint of the application, into an aggregated larger file. Unlike the existing aggregation mechanisms, the proposed technique follow POSIX file I/O semantics, thus no modification of application programs are required.

  In FY2011, we developed a prototype of file composition library and evaluated the basic performance. Figure 2 shows that the file composition technique is three times faster than POSIX file I/O functions in the case that each of 128 processes creates an individual file.

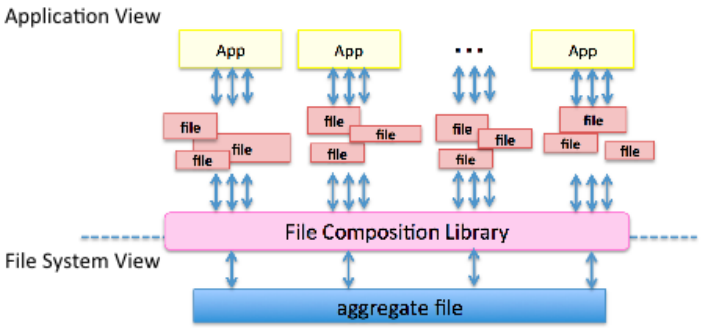

Figure 1. Concept of the file composition technique

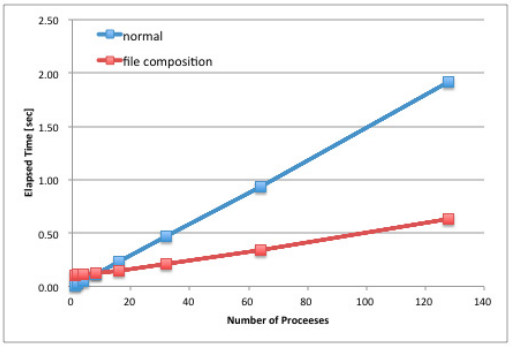

Figure 2. Time of 1 MB file creation by parallel process

- Hash-based Parallel File System

  A parallel file system used in a modern supercomputer is basically composed of MDS (Metadata Server) and OSS (Object Storage Server). A MDS handles file open, create and status operations. An OSS handles file read and write operations. Most current parallel file systems have only one MDS, and thus, such a system causes bottleneck of metadata operations requested by all compute nodes. A new scalable parallel file system based on a hash function was designed and implemented in FY2011. This system consists of multiple MDSs and OSSs as shown in Figure 3. Each MDS is responsible for metadata operations on a part of all files. The metadata set of files is determined by a hash value of the file name with path. The client determines the MDS of a file accessed in the client by a hash value of that file and path, and metadata operations for that file are sent to the MDS. The MDS informs the client to the location of OSSs for write/read operations in the client.

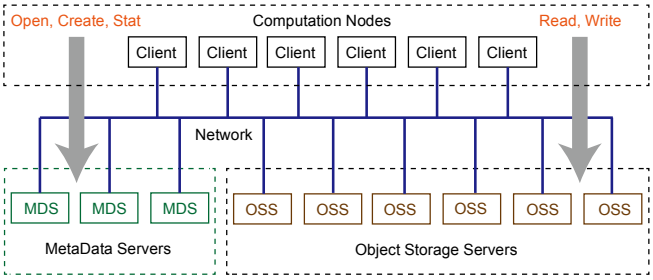

Figure 3. Hash-based Parallel File System

In FY2011, we developed only MDS for prototype file system and evaluated a metadata performance. Evaluations show that throughput of proposed file system is faster than that of current Lustre File System and has good scalability as shown in Figure 4.

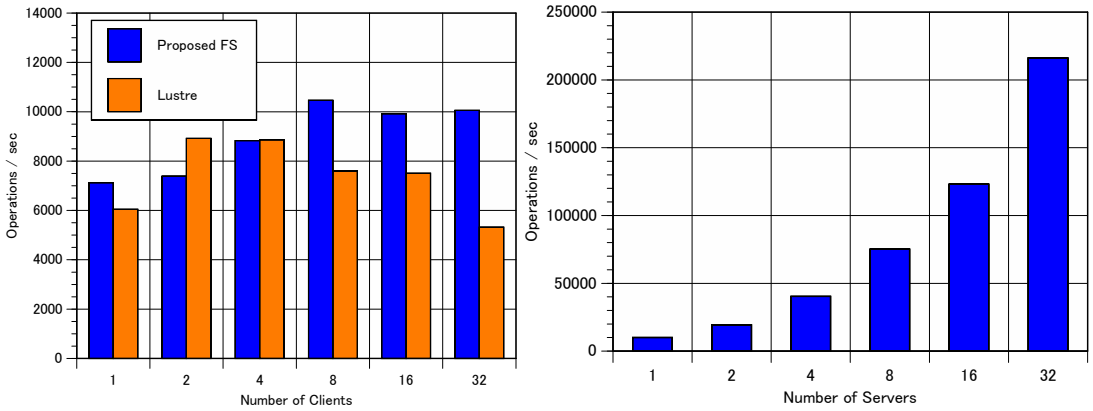

Figure 4. Throughput of the mkdir command execution

- A Smart MPI-IO Library Implementation

  The nature of highly parallelized parallel file access that consists of lots of fine grain, non-contiguous I/O requests in many cases, can degrade the I/O performance severely. To tackle this problem, a novel technique to maximize the bandwidth of the MPI-IO was proposed. This technique utilizes a ring communication topology and was implemented as an ADIO device of ROMIO, named Catwalk-ROMIO, and evaluated. The evaluation shows that Catwalk-ROMIO utilizing only one disk can exhibit comparable performance with widely-used parallel file systems, PVFS2 and Lustre, both of them utilizes several file servers and disks. As shown in Figure 5, Catwalk-ROMIO performance is almost independent from file access patterns, in contrast to the performance of parallel file systems performing only well with collective I/O shown in “Full” cases of Figure 5. Catwalk-ROMIO requires conventional TCP/IP network and only one file server. This is a quite common HPC cluster configuration. Thus, Catwalk-ROMIO is considered to be a very cost-effective MPI-IO implementation.

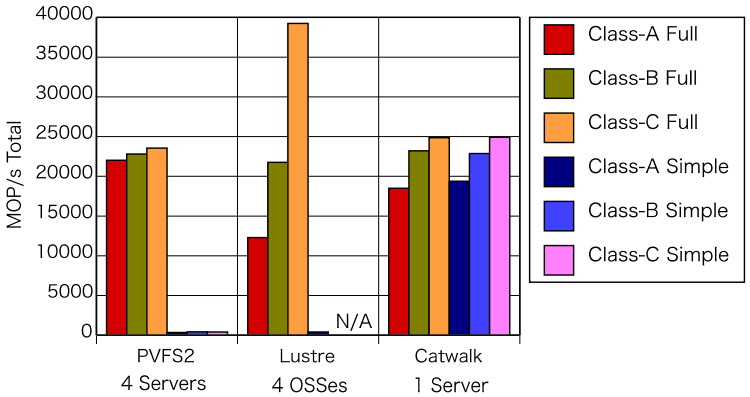

Figure 5. BT-IO Performance with PVFS2, Lustre and proposed Catwalk-ROMIO

1.3.2. Communication Library

- Persistent Remote DMA

  The implementation of persistent communication provided in MPI was reconsidered to provide low latency and true overlapping communication and computation. In the persistent communication facility, the end-points of both the sender and the receiver are set up by issuing MPI_Send_init and MPI_Send_recv primitives prior to actual communication triggered by the MPI_Start or MPI_Startall primitive. The same communication pattern is reused without reissuing the initialization. Thus, at the start of actual communications in persistent communication, the runtime system already knows all the communication patterns, i.e., peers and message sizes if both sender and receiver have issued persistent communication primitives. Such situations have a chance for the communication library to utilize four DMA engines equipped in K computer and carry out true overlapping communication and computation.

  A new communication protocol and an implementation for persistent communication, called PRDMA (Persistent Remote Direct Memory Access), was designed and implemented in FY 2011.

- Communication Library

  The power wall issue of current and future supercomputers is gathering attentions. The technique of user-level communication to achieve high communication performance is widely used by parallel applications, and processes are spinning-wait for the incoming messages. This spinning loop in the absence of incoming messages is simply wasting energy, and thus it increases the power consumption of a parallel computer. The proposed technique is implemented in two ways; 1) combination of spin-loop and blocking system call, and 2) combination of spin-loop and using the Intel x86 *monitor*/*mwait* synchronization instructions which put computational core into a low-power mode. Unlike the techniques using the DVFS (Dynamic Voltage and Frequency Scaling) function of CPU, our proposed technique does not sacrifice application performance but can save energy. Figure 6 shows that 7% total system power can be saved with the FT application of NAS parallel benchmarks.

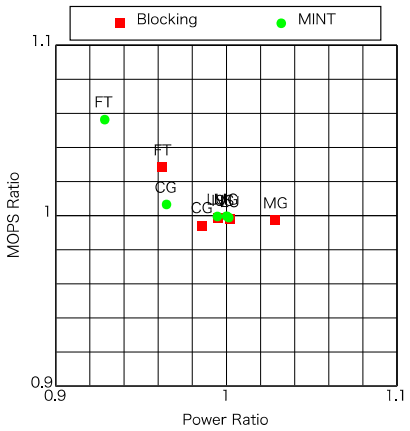

Figure 6. Power and Performance, NAS Parallel Benchmark Class B

1.3.3. Multi-Threaded Library

Towards the Exa-scale computing, hiding the latencies of memory and communication is one of crucial issues. To provide such a capability, the thread management must be fast enough in the order of sub-micro seconds. The thread library, named **Shadow Thread**, is developed to utilize Simultaneous Multi-Threading mechanism which schedules threads by hardware in a very fast way, and utilizes the *monitor* and *mwait* instructions supported by some Intel processors. Figure 7 shows that the two-phase synchronization technique combining the conventional spin-wait method and the pair of the *monitor*/*mwait* instructions can satisfy the requirement of speed and low-power consumption simultaneously. Figure 8 shows that a memory copy function using the proposed Shadow Thread library can exhibit better performance up to 20% compared with the normal *memcpy()* function.

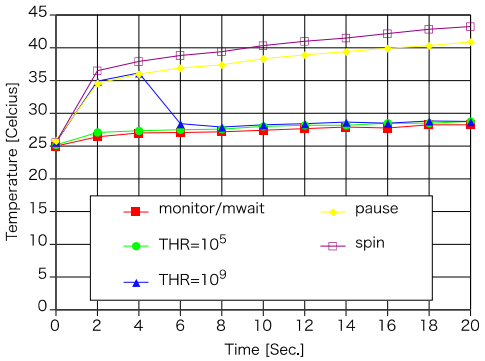

Figure 7. CPU temperature over time while synchronization

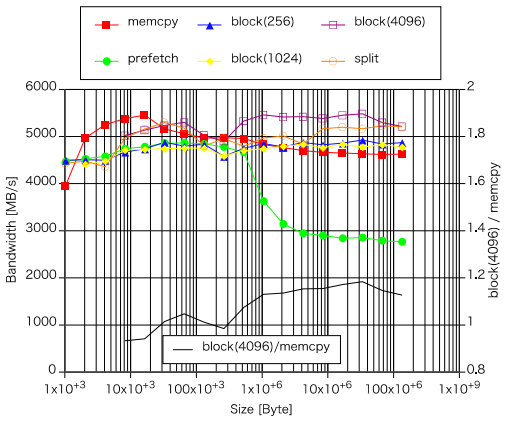

Figure 8. Multi-threaded memcpy bandwidth over region size

1.4. Schedule and Future Plan

The prototype implementation of File I/O and Communication libraries/middleware will be deployed to K computer in FY2012. Due to limited functionalities, the users will be restricted. The enhanced version of those libraries/middleware will be developed and deployed in FY 2013. In FY2013, those libraries/middleware will be deployed for users.

A scalable cache-aware, power-aware, and fault-aware operating system for next-generation supercomputers based on many core architectures are being designed and implemented by collaboration with University of Tokyo, NEC, Hitachi, and Fujitsu. The first prototype system will be distributed in FY2012 and the basic system will be available in FY2013.

1.5. Publication, Presentation and Deliverables

(1) Journal Papers

1. Atsushi Hori, Jinpil Lee, Mitsuhisa Sato, “Audit: A new synchronization API for the GET/PUT protocol”, In Journal of Parallel and Distributed Computing, 2012.

(2) Conference Papers

1. Atsushi Hori, Jinpil Lee, Mitsuhisa Sato, “Audit: New Synchronization for the GET/PUT Protocol,” In the 1st Workshop on Communication Architecture for Scalable Systems, 2011.
2. Atsushi Hori, Keiji Yamamoto, Yutaka Ishikawa, “Catwalk-ROMIO: A Cost-Effective MPI-IO,” In Proceedings of the 2011 IEEE 17th International Conference on Parallel and Distributed Systems, IEEE Computer Society, 2011.
3. Keiji Yamamoto, Atsushi Hori, Shinji Sumimoto, Yutaka Ishikawa, “Xruntime: A Seamless Runtime Environment for High Performance Computing,” In the 2011 International Workshop on Extreme Scale Computing Application Enablement - Modeling and Tools, 2011.

4. Atsushi Hori, Toyohisa Kameyama, Mitarou Namiki, Yuichi Tsujita, Yutaka Ishikawa, “Low Energy Consumption MPI Using Hardware Synchronization,” In IPSJ-SIGHPC 2011-HPC-132(7), 2011. (In Japanese)
5. Atsushi Hori, Keiji Yamamoto, Yoshiyuki Ohno, Toshihiro Konda, Toyohisa Kameyama, Yutaka Ishikawa, “A Ultra-light Thread Library Using Hardware Synchronization,” In IPSJ-SIGHPC 2011-HPC-130(6), 2011. (In Japanese)
6. Yoshiyuki Ohno, Atsushi Hori, Yutaka Ishikawa, “Proposal and preliminary evaluation of a mechanism for file I/O aggregation to one file in a parallel job,” In IPSJ-SIGHPC 2011-HPC-132(34), 2011. (In Japanese)

(3) Invited Talks

1. Yutaka Ishikawa, “HPCI and SDHPC: Towards Sustainable High Performance Computing in JAPAN,” PRAGMA21 Workshop, October, 2011.

(4) Posters and presentations

1. Atsushi Hori, Yutaka Ishikawa, “MINT: a fast and green synchronization technique,” In Proceedings of the 2011 companion on High Performance Computing Networking, Storage and Analysis Companion, ACM, 2011.
2. Keiji Yamamoto, Yoshiyuki Ohno, Atsushi Hori, Yutaka Ishikawa, “Hash-based Metadata Management for Parallel File System,” Symposium on High Performance Computing and Computational Science, 2012.
3. Keiji Yamamoto, Yoshiyuki Ohno, Atsushi Hori, Yutaka Ishikawa , “Current Issue in Parallel File System,” The 4th Forum on Data Engineering and Information Management, 2012.