Next-Generation Massively Parallel Computers — Development of Massively Parallel Computers for Continuous Physical Systems —

Yoichi Iwasaki (Vice President (Research), University of Tsukuba)

1 Project Organization

Yoichi Iwasaki(Leader)	Vice President (Research), Univ. of Tsukuba	
Akira Ukawa(Core Member)	Inst. of Phys., Univ. of Tsukuba	(AIOV)
Sinya Aoki	Inst. of Phys., Univ. of Tsukuba	(AMOC)
Masayuki Umemura	Inst. of Phys., Univ. of Tsukuba	(AIOV)
Kazuyuki Kanaya	Inst. of Phys., Univ. of Tsukuba	(AIOV)
Taishi Nakamoto	Inst. of Phys., Univ. of Tsukuba	(AIOV)
Tomoteru Yoshie	Inst. of Phys., Univ. of Tsukuba	(AMOC)
Masanori Okawa	High Energy Accelerator Res. Org.	(AIOV)
Arifa Ali Khan	Cent. Comp. Phys., Univ. of Tsukuba	(AMOC)
Taisuke Boku(Core Member)	Inst. of Inf. Sci. and Elec., Univ. of Tsukuba	(IOV)
Tomonori Shirakawa	Inst. of Eng. Mech. and Syst., Univ. of Tsukuba	(IOV)
Shigeru Chiba	Inst. of Inf. Sci. and Elec., Univ. of Tsukuba	(IOV)
Tsutomu Hoshino	Inst. of Eng. Mech. and Syst., Univ. of Tsukuba	(IOV)
Moritoshi Yasunaga	Inst. of Inf. Sci. and Elec., Univ. of Tsukuba	(IOV)
Yoshiyuki Yamashita	Inst. of Eng. Mech. and Syst., Univ. of Tsukuba	(IOV)
Koichi Wada	Inst. of Inf. Sci. and Elec., Univ. of Tsukuba	(MOC)
Shuichi Sakai	Grad. School of Eng., Univ. of Tokyo	(MOC)
Kisaburo Nakazawa	Dept. Inf. Sci., Meisei Univ.	(MOC)
Ikuo Nakata	Fac. of Comp. and Inf. Sci., Housei Univ.	(IOV)
Hiroshi Nakamura	Cent. for Adv. Sci. and Tech., Univ. of Tokyo	(MOC)
Yoshiyuki Watase	High Energy Accelerator Res. Org.	(IOV)
Kenichi Itakura	Cent. Comp. Phys., Univ. of Tsukuba	(IOV)

Research themes;
- AIOV: Parallel I/O and visualization for physics applications
- AMOC: Memory-integrated VLSI architecture for physics applications
- IOV: Parallel I/O and visualization
- MOC: Memory-integrated VLSI architecture

2 Research Objective

Recent development of computational sciences is strongly correlated with enhancement of computing power due to massively parallel computers (MPP). Physical systems in scientific and engineering applications can be broadly classified into continuous systems and particle-based systems. The CP-PACS project succeeded in developing a high performance MPP focusing on the former which appear in many areas of physics, such as particle physics, astrophysics and condensed matter physics.

In the present project we focus on MPP for continuous physical systems, and pursue R&D on the two issues urgently needed for the next-generation of such computers: (i) to realize fast and flexible I/O and visualization mechanisms to deal with enormous amount of data generated by such computers, and (ii) to develop a novel computer architecture required to enhance the computer speed to the range of a hundred TFLOPS in order to meet the demands of computational science applications.

We also examine the feasibility of a heterogeneous multi-computer system for an efficient processing of physical systems having continuous and multi-particle components.

3 Overview of Research Plan

Parallel I/O and parallel visualization We aim to develop a standard for high-performance I/O and visualization system for MPPs, which are flexible and inexpensive, by exploiting recent commodity networking technologies. As a test bed, we shall build a system consisting of the CP-PACS, a parallel disk server and a graphic server, connected by a multiple channels of 100base-TX Ethernet through high-speed switching hubs.

We shall develop an API on this system which provides users with a flexible parallel I/O environment where the channels to be used out of parallel ones are determined automatically without intervention from users to balance the load on parallel networks.

In visualization, a software-based flexible image processing is still required even hardware technology realizes a high speed but limited rendering process. To realize a flexible real-time image processing, we shall implement a parallelized and extendable visualization software combined with parallel I/O system for high throughput processing.

Memory-integrated VLSI architecture In spite of development of MPPs with a speed of 1 TFLOPS, there still remain a number of problems in computational sciences which require a computing power in the range of a hundred TFLOPS for solution. Meeting this requirement demands a high floating point operation capability of CPU and a memory hierarchy structure which ensures a sufficient throughput to supply data from the memory system to CPU.

We propose a novel processor architecture called SCIMA in which a medium-sized SRAM is placed within CPU as a software-controllable memory. Evaluation of the architecture through simulation of Linpack and physics application programs, and LSI design at the register transfer level shall be made to demonstrate the effectiveness and feasibility of the architecture for large-scale scientific and engineering calculations.

We shall also examine how thousands of SCIMA processors should be connected together to form an MPP with a speed of a hundred TFLOPS.

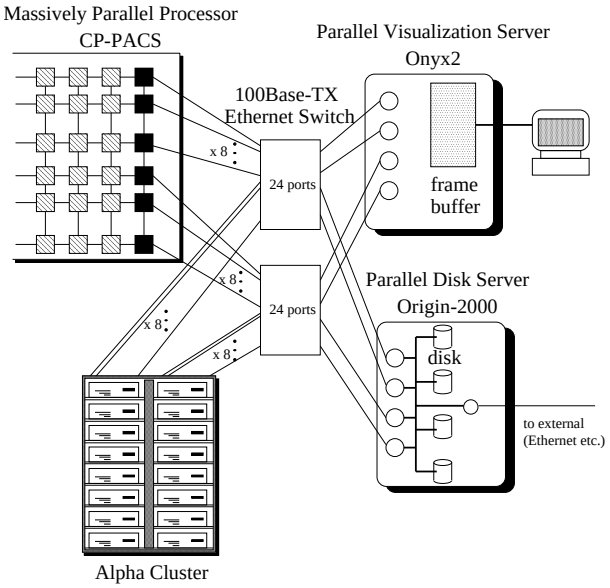


Figure 1: Experimental environment of parallel I/O system

Heterogeneous multi-computer system In the final year of the project we aim to experiment with a multi-computer system in which MPPs for continuous and multi-particle systems are unified. This will provide a new vision of MPP toward future large-scale scientific simulations.

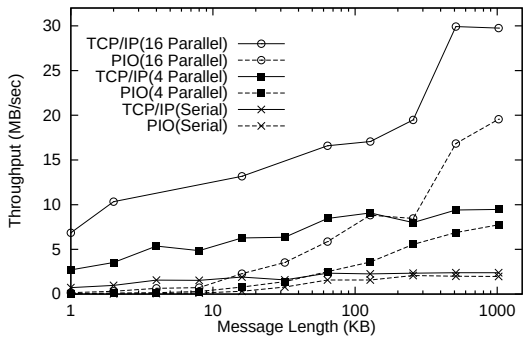
4 Research Achievements in FY2000

4.1 Parallel I/O and Visualization System

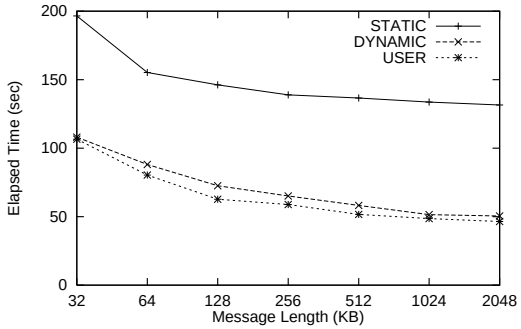
4.1.1 Parallel I/O system for next generation MPP

This year's research on parallel I/O system for next generation MPP was concentrated to build a complex with various types of platform as a surrounding environment of MPP system, based on parallel network connectivity among these systems. In the studies so far, we have established the basic method to connect a distributed memory MPP and a shared memory workstation in a point-to-point manner utilizing parallel Fast-Ethernet channels to provide wide bandwidth and easy programmability to end-users. This year, the parallel I/O system is extended to manage more than two parallel processing systems and to realize flexible and high-speed data exchange among them.

Figure 1 shows the experimental environment. CP-PACS, being a distributed memory MPP with 16 I/O units equipped with Fast-Ethernet interface, is the main data generating system. SGI Origin-2000 with 8 processors and SGI Onyx2 with 4 processors are file server and visualization server, respectively. In addition, a cluster system equipped with 16 Compaq Alpha CPU processors exist as an experimental COTS (Commodity Off-The Shelf) cluster. These systems are connected via parallel 100base-TX Ethernet through a



(a) Throughput on ping-pong transfer



(b) Dynamic load-balancing feature

Figure 2: Performance of PIO system

couple of high-speed switching hubs. We have introduced multiple switching hubs instead of single one with a large number of ports to realize a scalability which is not limited by the maximum number of ports in a switching hub.

On all these machines, a parallel network communication management system named PIO (Parallel I/O) is running. PIO consists of end-user libraries which provide simple, efficient and flexible API for application programmers and a server program named `pioserv` which manages parallel channels usability and data buffering on them. The number of `pioserv` instances to run on a system depends on the system configuration. For instance, on CP-PACS as a distributed memory MPP, a `pioserv` runs on each I/O processor, while a multi-threaded single process of `pioserv` runs with four to eight threads on Origin-2000. The number and type of `pioserv` processes are determined by a master configuration file where all system attributes are recorded. All servers communicate with each other via TCP/IP protocol to maintain the portability of PIO system.

Figure 2(a) shows the performance of data transfer throughput on point-to-point ping-pong communication between CP-PACS and Origin-2000. Since PIO provides multiple levels of data buffering, it introduces additional latency on ping-pong communication and the overall throughput is degraded compared with raw TCP/IP communication. However, when the number of used channels is increased, the total performance also increases in proportion to the number of channels. For messages of large size, the efficiency of multiple channels saturates because the total bandwidth on the Origin-2000 side is not enough to support that of CP-PACS with 16 channels. However, this problem will be solved with multiple servers. The PIO system is now extended to support any-to-any dynamic communication, and it is easy to distribute data among multiple servers.

Figure 2(b) shows the load-balancing feature of PIO. The horizontal axis shows the average message size to be transferred while the size of each message varies, and the size of maximum message is a double of that of minimum one. There are three types

of load balancing cases: `USER`, `STATIC` and `DYNAMIC`. In `USER`, the application program explicitly distributes messages and specifies channels to be used keeping the load-balance among parallel channels. In `STATIC`, there is no dynamic load-balancing, and messages are always transferred with the nearest channels. In `DYNAMIC`, PIO system tries to keep the load-balance dynamically based on channel usage information which is always exchanged among both sides of communicating machines. As a result, the dynamic load-balancing works very well to keep the total data transfer time to be minimized and close to `USER`, which provides a complete load-balancing. The overhead of dynamic load-balancing to `USER` is negligibly small.

4.1.2 Parallel Visualization System

As an application using the PIO system, we have developed a parallelized visualization system on parallel workstation Onyx2. On the next generation MPP, a powerful computing power which enables a large-scale real-time simulation is expected. On such a simulation, a real-time data visualization system with high data transfer bandwidth and high-speed visualization capabilities is required. The PIO system realizes the former feature. Then we combined the PIO feature with a parallelized visualizer based on a multi-threaded general purpose data manipulator.

The basic system used here is a defacto-standard visualization system, AVS/Express which consists of a number of small modules communicating with each other in an object oriented manner. The original system only provides a simple input from local files and sequential processing for each visualization stage. We have extended the data input method to accept parallel channel inputs with PIO, and also parallelized the volume rendering module which consumes most of CPU time for heavy calculation duty. We have implemented the modules into AVS/Express environment, and realized a high throughput volume rendering system which is well balanced both for data transfer and manipulating performance.

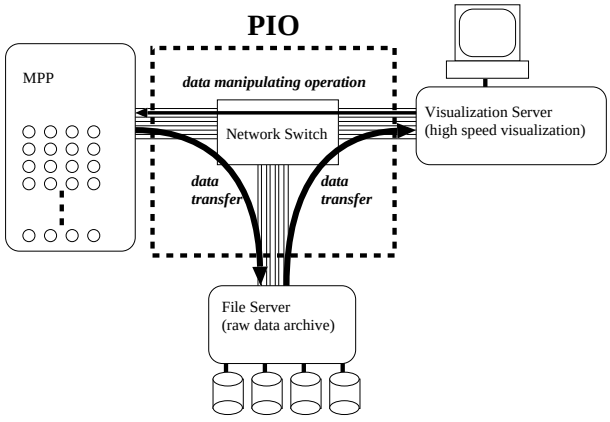


Figure 3: Total system of real-time data manipulation with PIO

Since the extended PIO system can handle any-to-any parallel data transfer, several server systems can be combined into a large complex. Figure 3 shows an example of a file server and a visualization server combined with data generating engine. Here, a large amount of raw data are generated in the MPP, and transferred to the file server for recording. At the same time, this server creates pixel data for each time slice, then transfers them to the visualization server for real-time visualization. In this system, all data are transferred with PIO in parallel and pipelined manner.

4.1.3 Parallel File System

From this year, we have started to develop another application based on PIO, named PFS (Parallel File System). PFS provides a logical view of single file image accessed by parallel processes in SPMD programs for domain decomposition manner. A programmer does not have to be conscious of individual file names nor their location on the file system. PFS library routines automatically generate an individual path names and the best location to exploit the high-bandwidth data transfer by PIO for remote file server accessing.

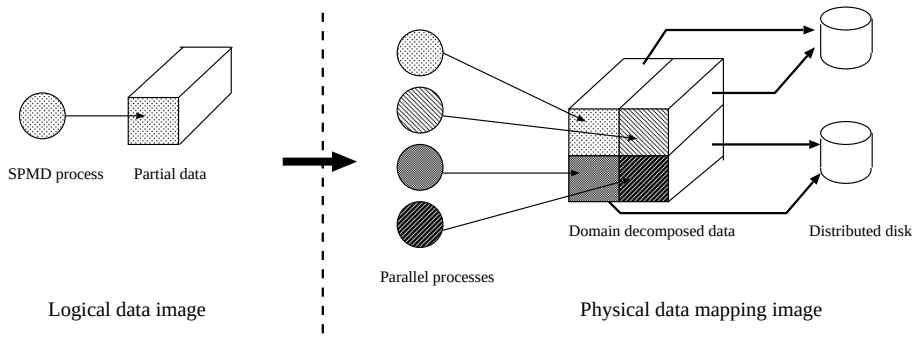


Figure 4: Conceptual image of Parallel File System

We have designed a basic API sets of PFS, and now are designing intermediate data expression and developing several implementations for a couple of systems not only with PIO but also local parallel I/O system on CP-PACS and COTS cluster system.

4.2 Memory Integrated VLSI Architecture

SCIMA (Software Controlled Integrated Memory Architecture) is a new VLSI architecture for high performance computing. In this year, we have developed a detailed simulator, with which a preliminary performance evaluation of SCIMA has been made, and a prototype of SCIMA compiler.

4.2.1 Overview of SCIMA

Figure 5 shows the schematic view of the proposed architecture *SCIMA*. In SCIMA, addressable On-Chip Memory is integrated into the processor chip in addition to ordinary

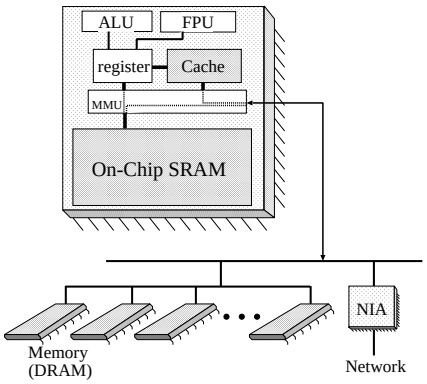


Figure 5: Overview of SCIMA

cache. We employ SRAM as the On-Chip Memory. Since our target is HPC applications, the whole data generally cannot reside in On-Chip memory even if DRAM is used. We put higher priority on the fast access time rather than on large capacity. The main difference between On-Chip Memory and cache is that the location and the replacement of data are controlled by software explicitly in On-Chip Memory, whereas those of cache are controlled by hardware implicitly. Cache is still provided to work for irregular data accesses.

All the address space has a cacheable/uncacheable property. In SCIMA, the On-Chip Memory space is always handled as uncacheable. Therefore, there is no inclusion relation between On-Chip Memory and cache.

Data transfer among memory hierarchy. The following two kinds of data accesses are available:

- register $\leftrightarrow$ On-Chip Memory $\leftrightarrow$ Off-Chip Memory
- register $\leftrightarrow$ cache $\leftrightarrow$ Off-Chip Memory

In order to control data transfer between On-Chip Memory and Off-Chip Memory, special instructions, *page-load* and *page-store*, are newly introduced. These instructions can specify block-strided data transfer. Using this feature, non-consecutive data on Off-Chip Memory can be packed and transferred into a consecutive area on On-Chip Memory. This ability allows effective utilization of On-Chip Memory.

4.2.2 Simulator

SCIMA is defined as an extension of existing architecture. In this research, MIPS IV is selected as the base architecture.

It would be preferable to develop an optimized compiler which can handle the architectural extensions. However, since the compiler is still under development, users currently

declare the following issues in source programs; which data should be allocated on which part of On-Chip Memory, and when the data transfer is invoked.

We developed a preprocessor which inserts these user-directed informations into assembly code after the source code is compiled by ordinary MIPS compiler. We also developed a simulator which can accept the binary object generated by existing MIPS compiler and interpret the informations inserted by the preprocessor. The simulator emulates the behavior of SCIMA cycle by cycle.

The simulator assumes that instruction cache always hits and branch prediction is completely successful. These assumptions are reasonable since time consuming part of HPC applications consist of regular loop structures. As for data cache, the simulator supports lock-up free L1 cache only. Out-of-order execution mechanism by reservation station is also supported.

4.2.3 Compiler

The C compiler exploiting SCIMA is being developed. Based on the commodity C compiler, new expressions dedicated to utilizing On-chip Memory are attached.

In order to make good utilization of On-chip Memory, the following two issues are important. The first issue is allocation, that is, which data should be allocated on which part of On-chip Memory. The second issue is scheduling, that is, when the data transfer between Off-chip memory and On-chip Memory should be invoked. In this year, we focus on the second issue. For a given program with poor scheduling of page-load/store instructions, we developed an algorithm of improving the scheduling quality by using a software pipelining technique. This automatic software pipelining mechanisms has been developed as a part of the compiler.

The backend of the compiler has been reformed to generate codes for the SCIMA simulator, where we have been making performance evaluations. Universe radiation problems and the common benchmarks such as SPEC have been used for performance evaluations. The results show the effectiveness of both the architecture and the compiler.

4.2.4 Performance Evaluation

In this year, we have evaluated the performance of SCIMA when applied to QCD computations. For major advance of QCD, simulations with the lattice size of $48^3 \times 96$ are necessary. Carrying out such simulations requires a performance of at least 64 TFLOPS. A moderate implementation would be a parallel computer with 4096 PUs, where each PU achieves 16GFLOPS and computes a sublattice of a size $6^3 \times 12$.

The following is the assumptions common throughout the evaluations.

- number of execution units: integer=4, floating-point (multiply-add)=4, load/store(cache or On-Chip Memory)=4

Table 1: Combination of cache and On-Chip Memory

	cache size (associativity)	On-Chip Memory size
(a)	2MB(4way)	0MB
(b)	1.5MB(3way)	0.5MB
(c)	1MB(2way)	1MB

- multiply-add operation latency: 4 cycles
- load/store latency: 2 cycle (for both cache and On-Chip Memory)
- throughput of Off-Chip Memory: 8B/cycle
- total on-chip memory (sum of cache and On-Chip Memory) capacity: 2MB

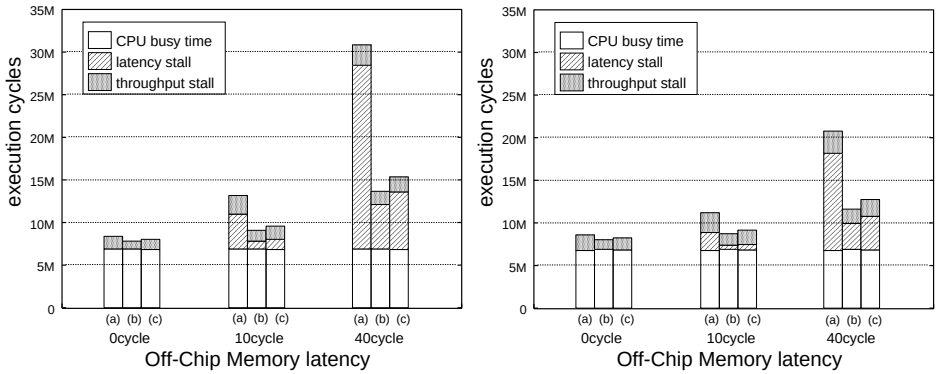
We alter the configuration of cache and On-Chip Memory in three ways as shown in Table 1. In case (a) all data are accessed through cache. Two kinds of cache line size, 32B and 64B, and three kinds of Off-Chip Memory latencies, 40, 10 and 0 cycle, are selected in the evaluation.

Figure 6 illustrates the execution cycles and their breakdown under each configuration for three kinds of Off-Chip Memory latencies and two kinds of cache lines. We break down the execution cycles into CPU busy time, latency stall, and throughput stall. The total cycle indicates the execution cycles under a given assumption. The throughput stall is defined as the cycles which can be saved from the total cycles if Off-Chip Memory bandwidth were infinite. The latency stall is defined as the cycles which can be saved further if Off-Chip Memory latency were 0 cycle. The CPU busy time is obtained under the hypothetical assumption in which Off-Chip Memory bandwidth were infinite and latency were 0 cycle.

As shown in Figure 6-[A] for the cache line size of 32B, the cases (b) and (c) having On-Chip memory achieve 2.2 times and 2.0 times higher performance than the case (a) with cache only, respectively, when the latency is 40 cycle. This is because the data transfer size from Off-Chip to On-chip Memory is quite large, which leads to a significant reduction of latency stall.

Changing the cache line size from 32B (Figure 6-[A]) to 64B (Figure 6-[B]), the latency stall decreases for larger line size. However, the throughput stall slightly increases for the larger cache line size. This is because more line conflicts are likely to occur for larger cache line size. Therefore, larger cache line sizes do not always bring higher performance.

Considering the future direction of the semiconductor technology, Off-Chip Memory latency is expected to increase and the relative Off-Chip Memory bandwidth is expected to decrease. Therefore, it is indispensable to reduce Off-Chip Memory traffic and to



[A] cache line size = 32B [B] cache line size = 64B

Figure 6: Breakdown of execution cycles

make data transfer size larger. SCIMA achieves high performance by realizing both requirements.

5 Future Plan

Parallel I/O and Visualization We shall refine and complete the parallel I/O and visualization system developed in the present project. This will include (i) optimization of PIO performance for each of the wide variety of platforms including MPP, shared memory workstations and COTS clusters, (ii) further development of PFS into a fully equipped parallel file management environment to be used by PIO, and (iii) a total end-user environment for parallel visualization system that combines file and visualization servers for easy data manipulation and flexible control of animated visualization.

Memory Integrated VLSI Architecture The simulator will be expanded to handle detailed characteristics of memory structure including memory interleave and a variety of DRAM interfaces, with which the SCIMA shall be evaluated for a wide range of real applications including astrophysics and condensed matter physics.

The SCIMA C compiler shall be further optimized in the backend to show that a compiler which allows an effective usage of SCIMA is possible without expert manual coding incorporating the SCIMA features. We also plan to develop FORTRAN compiler for SCIMA in the final year.

Finally we plan to design SCIMA at the register transfer level and verify the feasibility of implementation of SCIMA-based processor with future semiconductor technology.

Next generation MPP for continuous simulation Connecting thousands of SCIMA-based processors, an MPP system exceeding 100 TFLOPS of peak performance is possible.

For such systems designing an interconnection network with high bandwidth and low latency to support inter-processor data transfer is a major issue. We examine a hybrid network architecture combining electrical connection for local communication and optical connection for global communication. We will design the best network topology and methodology for hybrid network as well as node architecture to connect multiple SCIMA processors in an SMP manner. This will lead to a total design for the next generation MPP for continuous physical systems.

Heterogeneous Multi-Computer System In the final year of the project, we will realize a unified system which is equipped with both MPPs for continuum simulation and particle simulation. For this experiment, CP-PACS and Grape-6 systems will be connected through PIO as high-bandwidth communication channels.

With this integrated system, we aim to solve gravitational radiation-hydrodynamics in which self-gravity, radiative transfer and multi-dimensional hydrodynamics are self-consistently treated. This problem is hard to solve either by CP-PACS or Grape-6 alone, and hence this experiment shall provide a new vision of heterogeneous multi-computer systems as MPPs for future large-scale scientific simulations.

6 Presentations and Publications

6.1 Presentations at International Conferences

[1] M. Kondo, H. Okawara, H. Nakamura, T. Boku, and S. Sakai, "SCIMA: A Novel Processor Architecture for High Performance Computing", Proceedings of HPC-Asia 2000, pp.355-360, Beijing, May 2000.

[2] M. Kondo, H. Okawara, H.Nakamura, and T. Boku, "SCIMA: Software Controlled Integrated Memory Architecture for High Performance Computing", ICCD-2000, pp.105-111, Austin, September 2000.

[3] H. Nakamura, M. Kondo, and T. Boku, "Software Controlled Reconfigurable On-Chip Memory for High Performance Computing", 2nd Workshop on Intelligent Memory Systems (pre-workshop of ASPLOS-IX), Cambridge, November 2000.

[4] T. Yoshié, "Recent lattice QCD results from massively parallel computers", Conference on Computational Physics 2000 "New Challenges for the New Millennium", Queensland, Australia, December 2000.

6.2 Presentations at Domestic Conferences

[1] M. Matsubara, H. Numa, K. Itakura and T. Boku, "Parallel I/O system on distributed

memory parallel processors", Proc. on JSPP2000, pp.75-82, 2000.

[2] K. Itakura, T. Boku and M. Matsubara, "Parallelization of general purpose visualization tool AVS/Express and its performance evaluation", IPSJ SIGHPC, HPC-82-31, pp.179-184, 2000.

[3] H. Okawara, M. Kondo, H. Nakamura, and T. Boku, "Preliminary Performance Evaluation of New Memory Architecture for High Performance Computing", IPSJ SIGARC, ARC-136-3, pp.13-18, 2000

[4] N. Hattori, D. Iizuka, S. Sakai and H. Tanaka, "Memory Access Reduction by Inter-Procedural Register Promotions" 60th IPSJ Annual Conventions, No.5H-4, Vol.1, March 2000.

[5] N. Hattori, D. Iizuka, S. Sakai and H. Tanaka, "Compiler Support for Load/Store Pressure", IPSJ SIGHPC, HPC82-19, pp.107-112, August 2000.

[6] M. Nakamura, M. Iwashita, S. Sakai, and H. Tanaka, "Software Optimization Methods for SCIMA Architecture", IPSJ SIGHPC, HPC-82-20, pp.113-118, 2000

[7] M. Iwamoto, R. Watanabe, M. Kondo, H. Nakamura, and T. Boku, "Performance Evaluation of SCIMA for NASPB Kernel CG, FT", IPSJ SIGHPC, HPC-83-6, pp.31-36, 2000

6.3 Journal Papers

[1] M. Matsubara, H. Numa, K. Itakura and T. Boku, "Parallel I/O system on distributed memory massively parallel processors", IPSJ Transactions on High Performance Computing Systems, Vol.41, No. SIG 5(HPS 1), pp.58-69, 2000.

[2] K. Kise, S. Sakai and H. Tanaka "Performance Potential of Two-Level Stride Value Predictor", IPSJ Journal, Vol.41 No.5, pp.1340-1350, May 2000.

[3] H. Nakamura, M. Kondo, H. Okawara, and T. Boku, "SCIMA: A New Architecture for High Performance Computing", IPSJ Transactions on High Performance Computing Systems, Vol 41, No. SIG 5(HPS 1), pp.15-27, 2000

[4] CP-PACS Collaboration, A. Ali Khan *et al.*, "Dynamical Quark Effects on Light Quark Masses", Phys. Rev. Lett. 85 (2000) 4674.

[5] CP-PACS Collaboration, T. Manke *et al.*, "Sea Quark Effects on Quarkonia", Phys. Rev. D62 (2000) 114508.