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

Leader:		
Yoichi Iwasaki	Vice President (Research), Univ. of Tsukuba	
Core members:		
Akira Ukawa	Inst. of Phys., Univ. of Tsukuba	(AIOV)
Taisuke Boku	Inst. of Inf. Sci. and Elec., Univ. of Tsukuba	(IOV)
Members:		
Sinya Aoki	Inst. of Phys., Univ. of Tsukuba	(AMOC)
Shigeru Chiba	Inst. of Inf. Sci. and Elec., Univ. of Tsukuba	(IOV)
Kazuyuki Kanaya	Inst. of Phys., Univ. of Tsukuba	(AIOV)
Taishi Nakamoto	Inst. of Phys., Univ. of Tsukuba	(AIOV)
Hiroshi Nakamura	Cent. for Adv. Sci. and Tech., Univ. of Tokyo	(MOC)
Ikuo Nakata	Fac. of Comp. and Inf. Sci., Housei Univ.	(IOV)
Kisaburo Nakazawa	Dept. Inf. Sci., Meisei Univ.	(MOC)
Masanori Okawa	High Energy Accelerator Research Organization	(AIOV)
Shuichi Sakai	Grad. School of Eng., Univ. of Tokyo	(MOC)
Mitsuhisa Sato	Inst. of Inf. Sci. and Elec., Univ. of Tsukuba	(MOC)
Tomonori Shirakawa	Inst. of Eng. Mech. and Syst., Univ. of Tsukuba	(IOV)
Daisuke Takahashi	Inst. of Inf. Sci. and Elec., Univ. of Tsukuba	(MOC)
Masayuki Umemura	Inst. of Phys., Univ. of Tsukuba	(AIOV)
Moritoshi Yasunaga	Inst. of Inf. Sci. and Elec., Univ. of Tsukuba	(IOV)
Yoshiyuki Yamashita	Dept. of Science and Engineering, Saga Univ.	(IOV)
Tomoteru Yoshie	Inst. of Phys., Univ. of Tsukuba	(AMOC)
Koichi Wada	Inst. of Inf. Sci. and Elec., Univ. of Tsukuba	(MOC)
Yoshiyuki Watase	High Energy Accelerator Research Organization	(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 advance of computational sciences is strongly related with the dramatic increase 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. This project focuses 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 to meet the demands of science applications.

Development of MPP for particle-based systems is pursued in a parallel project. Combining the development of the two projects, we propose the concept of *Heterogeneous Multi-Computer System (HMCS)* for an efficient processing of computations of complex physical systems having both continuous and multi-particle components.

3 Research Achievements in FY2001

3.1 Parallel I/O and Visualization System

In this year we have finalized the development of our parallel I/O and visualization system and applied it to various systems. The total package is named PAVEMENT (Parallel I/O and Visualization Environment).

The parallel I/O component of the package is named PAVEMENT/PIO. We have ported it to various architectures this year, which includes CP-PACS MPP system, shared-memory workstations and their clusters, and various Linux-based PC clusters. As a total system, PIO can connect an MPP system to surrounding computational resources through parallel Fast-Ethernet channels through multiple network switches, keeping wide bandwidth and easy programmability to end-users.

We have developed several application systems based on PIO. One of the applications is HMCS which consists of general purpose and special purpose parallel processors connected with PIO. This system implements a new methodology of computational physics that combines multiple paradigms of computation. The detail of HMCS will be described later in Section 3.4.

Another application is PAVEMENT/PFS (Parallel File System) which provides a high throughput file access from an MPP to local or remote parallel/distributed disks. The concept of PFS is based on the distribution of parallel file access streams to parallel disks on *user level*. Such a technique on OS level is well known as *disk striping*. We have implemented the same idea with a user-level library to utilize PIO with minimum overhead and high programmability for users. The concept of PFS is based on SPMD programming with the space decomposition method. If one makes a simple declaration of the global mapping rule for parallel processes on to a logical array at file opening, suceeding file operations are automatically translated as partial accesses to the logical array (Figure 1). With PIO communication, PFS is applicable both for local parallel disks and for remote parallel disks with high I/O througput.

The basic modules for parallel visualization system named PAVEMENT/VIZ was completed last year. VIZ consists of a collection of program modules designed to embed in AVS/Express, which is a defacto standard GUI-oriented visualization programming system in object oriented manner. The basic components of VIZ are a parallelized 3-D volume rendering module and a high throughput data input module to attach AVS/Express to PIO.

This year we extended the functionality of VIZ for more flexible and user-friendly interface.

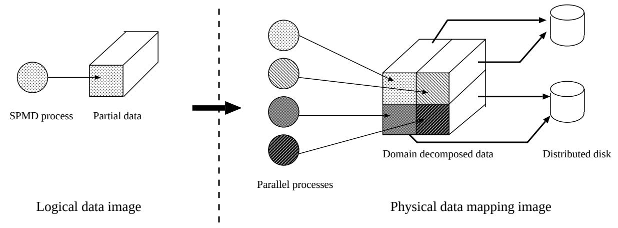
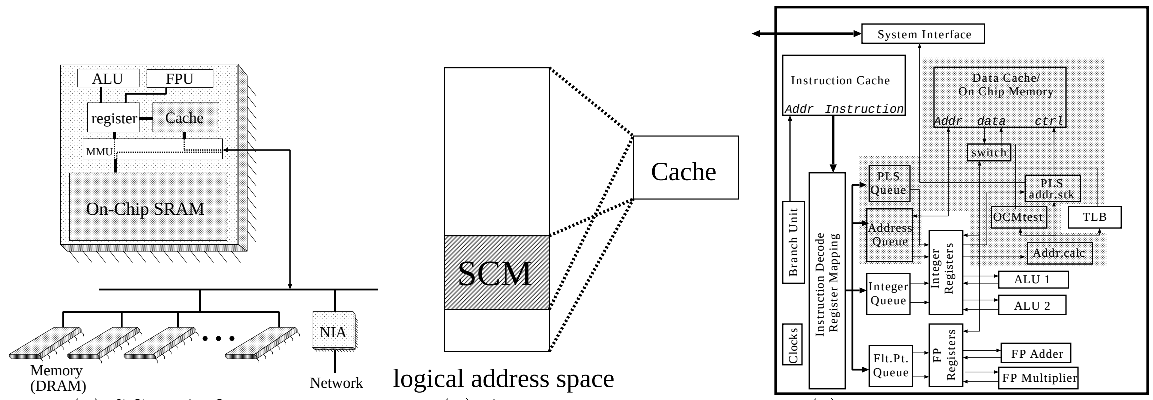


Figure 1: Concept of logical/physical file mapping on PAVEMENT/PFS



(a) SCIMA Overview (b) Address Mapping (c) Data Path Structure

Figure 2: SCIMA

With this expansion, the user can manipulate 3-D objects with mouse dragging, keeping the high speed real time parallelized rendering feature. This function enables the replacement of the original module with our parallelized one keeping the backward compatibility. The parallelized rendering module will be distributed as AVS/Express extension kit from our research partner, Kubota Graphics Technologies.

3.2 Memory Integrated VLSI Architecture

SCIMA (Software Controlled Integrated Memory Architecture) is a new VLSI architecture for high performance computing. This year, we designed SCIMA at register transfer level, proposed an optimization strategy on the usage of the integrated memory, and developed a compiler for SCIMA.

3.2.1 Architectural Overview

Figure 2-(a) shows the schematic view of the proposed SCIMA. Addressable SCM is integrated into a processor chip in addition to ordinary cache. SCM occupies one consecutive and uncacheable part of logical address space as shown in Figure 2-(b).

Two kinds of data accesses are available in SCIMA; (1) register $\leftrightarrow$ SCM $\leftrightarrow$ off-chip memory, and (2) register $\leftrightarrow$ cache $\leftrightarrow$ off-chip memory.

Data transfers between SCM and off-chip memory are invoked explicitly by new instructions called *page-load* and *page-store*. These instructions can identify large amount of data transfer, which reduces the number of off-chip memory accesses for consecutive data access. These instructions also support block-stride data transfer which packs non-consecutive data of off-chip Memory and transfers them into a consecutive area of SCM.

3.2.2 RTL Design

SCIMA is defined as an extension of existing scalar architectures. While the extension consists of only an addition of several new instructions and on-chip memory, it might affect the clock frequency. In order to measure this effect, we designed two processors, a scalar processor and SCIMA. We selected MIPS R10000 microprocessor as the reference architecture, since it is widely used and is one of the current high-performance microprocessors. Since R10000, as usual with high-performance microprocessors, is quite complex, we did not design the whole processor but focused on the difference between R10000 and SCIMA.

Table 1: delay results of critical path

delay[ns]	R10000 model	SCIMA model
select & issue	6.11	6.48
cache way select (not including array access time)	3.31	3.50

Figure 2-(c) shows the data path of the SCIMA. The shaded region represents the part modified from the original R10000. We have designed the shaded part for both R10000 and SCIMA at the register transfer level in Verilog-HDL. Then, those designs are synthesized into gate level by the "Design Compiler" of Synopsis Inc. and ROHM's 0.35 μm CMOS technology.

We have estimated the delay of critical paths from the obtained designs. There are two possibilities for the location of the critical paths. One is the selection of executable instructions from the queues followed by the issue of the selected instructions. The other is cache way selection logic following cache array access. Table 1 shows the estimated delay of each paths in R10000 and SCIMA. As seen from the table, if the delay of cache array access is longer than 2.8 ns, the cache access path gets the critical path. In either case, the clock frequency of SCIMA degrades only 5% compared with R10000, which is quite small penalty. As is seen from the result of the following section, SCIMA is still much faster than R10000 in spite of this penalty.

3.2.3 Optimization of Integrated Memory Usage

We have developed an optimization strategy on the usage of on-chip memory space, and have made preliminary evaluation of this strategy. Data arrays are classified based on their access characteristics. Figure 3 describes the strategies for making good use of on-chip memory.

Among the strategies, (1) and (2) are more effective than the others. Transferring such non-reusable arrays with large granularity is very helpful for reducing latency-stall. Therefore, we give higher priority to (1) and (2). After they are applied, strategies of (4), (5) and (6) are applied to reusable arrays by using the rest area of on-chip memory.

We used two programs for evaluation. One is kernel FT from NAS Parallel Benchmarks and the other is QCD (Quantum ChromoDynamics) computation. We compared the performance of SCIMA with cache-based architecture in order to evaluate the proposed strategy. High-level optimizations such as loop tiling and loop exchange are applied to both architectures in advance. For SCIMA, the proposed optimization is applied additionally by hand.

The assumptions for performance evaluation are as follows; total integrated memory size

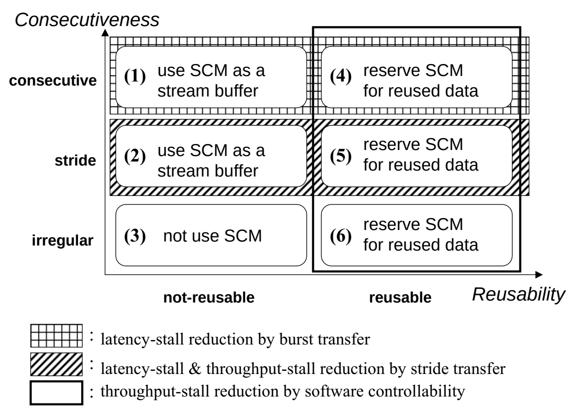
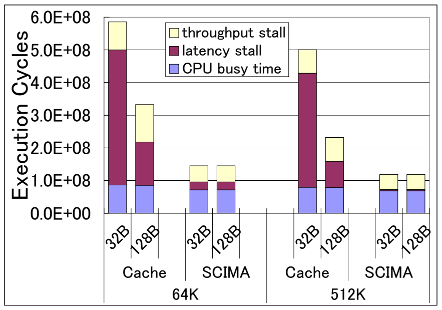
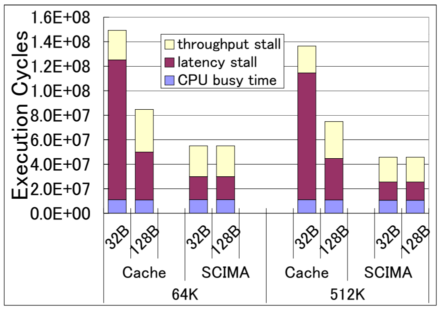

Consecutiveness
consecutive
stride
irregular
(1) use SCM as a stream buffer
(4) reserve SCM for reused data
(2) use SCM as a stream buffer
(5) reserve SCM for reused data
(3) not use SCM
(6) reserve SCM for reused data
not-reusable
reusable
Reusability
: latency-stall reduction by burst transfer
: latency-stall & throughput-stall reduction by stride transfer
: throughput-stall reduction by software controllability

Figure 3: Strategy for Using SCM



(a) FT



(b) QCD

Figure 4: Performance Result

is 64KB or 512KB; cache line size is 32B or 128B; throughput of cache/On-Chip Memory is 8B/cycle; throughput of Off-Chip Memory is 1B/cycle; Off-Chip Memory access latency is 160 cycle; and cache prefetching is available (in Cache model only).

Figure 4 presents the required cycles for each program. The cycles are decomposed into three categories, CPU busy time, latency stall and throughput stall.

As seen from Fig. 4, SCIMA achieves better performance than Cache for both programs under all configurations. This is because SCIMA can successfully and drastically reduce the latency-stall time by reducing the main memory traffic. This result reveals that the proposed optimization strategy is very effective.

3.2.4 Compiler

A program running on a SCIMA-based processor must explicitly control data transfer between on-chip memory and off-chip memory. Like other modern processor technologies, this control of memory transfer should be the responsibility of compilers. However, when and what data should be transferred for the best performance during program execution is still a research issue. To provide a research platform for this issue, we have developed a Fortran compiler that accepts directives designed for SCIMA. Programmers can manually specify memory transfer with these directives. This compiler consists of two components: a front-end compiler from Fortran to C, and a back-end compiler from C to assembly.

Front-end Compiler The front-end compiler translates a Fortran program including directives of memory transfer between on-chip and off-chip memory. These directives are newly designed for SCIMA. They are called *SCIMA directives*. The design goal is to provide good abstraction for users but keep the ability for fine-grained control. This would make it easier for researchers to examine new ideas of optimization and develop a general-purpose optimization algorithm for SCIMA.

The SCIMA directives are used for dynamically mapping part of array elements on on-chip memory. This mapping is hidden from users, who do not have to use special syntax for accessing the *on-chip* elements. The compiler translates array access written in regular Fortran to the C code accessing on-chip memory.

A very simple program example using the SCIMA directives is as follows.

```
double precision sum
double precision a(N*2,N*2)
!$scm begin (a, N, N, 0, 0)
!$scm load (a, N + 1, N + 1, N, N)
sum = 0.0
```

```
    do i = N + 1, N * 2
        sum = sum + a(i, i)
    enddo
    !$scm end (a)
```

The lines beginning with `$scm` are directives. In the region surrounded by `begin` and `end`, all the accesses to array `a` are interpreted as accesses to on-chip memory. The directive `load` specifies which part of the elements of the array are to be copied to on-chip memory when the program execution reaches that directive.

The compiler has been already released although some known bugs still remain. We used RWCP Omni Compiler Software as a basis of the compiler. The generated C code can be compiled by either the native MIPS compiler or our back-end compiler.

Back-end Compiler The current version of SCIMA processor assumes the MIPS R10000 ISA. The back-end compiler takes a C code generated from the front-end, and translates it into an object code of the MIPS R10000 processor architecture in SGI IRIX N32 ABI with the SCIMA processor's extensions. The back-end compiler generates a code with SCIMA extended instructions to control data transfer between on-chip and off-chip memories. The generated code is optimized for effective use of extended number of registers in SCIMA, and the compiler is designed for various register structure. On-chip data address is allocated to a particular section corresponding to on-chip memory by the loader.

In the backend compiler, the following phases are implemented to optimize codes as other optimizing compilers:

- Common subexpression elimination and constant-folding, constant propagation.
- Loop optimization including loop invariant elimination and induction variable elimination, operator-strength reduction.
- Global register allocation by register coloring technique.

The optimizing compiler allows programmers and architects to evaluate the performance of several benchmark programs with SCIMA directives under the change of architectural parameters such as the number of registers and extended instructions.

3.3 Node Interconnection and System Design

It is necessary to introduce a full-optical network to support enormous computational power of processors in next generation high-end MPP. However, a full optical switching system is still quite expensive and its switching speed is still not enough for rapid handling of short messages. One of the solutions is to combine optical and electrical links/switches in a clustered hierarchical network. The inter-cluster communication is performed by full-optical switches with wide bandwidth and large latency, while the inner-cluster communication is performed by optical links and opt-electrical switches with medium bandwidth and low latency. Setting the number of clusters to be much smaller than that of processing nodes in a cluster, a collection of short messages for the same destination cluster can be combined into a large message on a special router which connects the cluster to the inter-cluster switch. Such a hierarchical network can balance the trading-off between bandwidth and latency on optical networks.

For instance, an MPP with 2048 processing nodes is reasonably designed as 8 clusters each of them consisting of 256 nodes, where the inner-cluster network is a 2-dimensional hypercrossbar (16×16 nodes) and the inter-cluster network is a full-crossbar. We have developed a special network simulator which can handle different network link widths and latencies in a system,

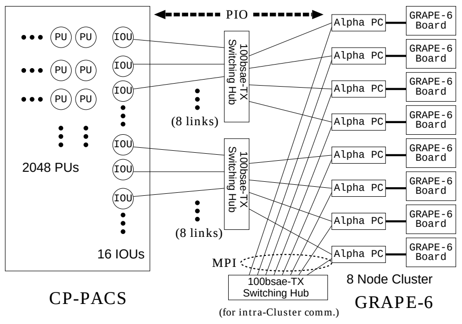


Figure 5: Block diagram of HMCS

and have evaluated the performance of various configurations. The key issue is toughness of the network to inter-cluster communication, that is, locality of data transfer. Assuming the ratio of the network bandwidth of inter-cluster network to inner-cluster network is 100:1, the network achieves almost the same bandwidth with completely local communication for up to 10% of inter-cluster communication ratio on *random* transfer. This ratio is acceptable for various scientific simulations.

As the entire system image based on this network structure, we can design an ideal high performance MPP with SCIMA processors. To connect more than 10000 processors in a system, it is better to introduce SMP node structure which saves the total number of network ports. We have evaluated the SMP configuration of SCIMA processors in a computational node. As a result, 2-way to 4-way SMP configuration is suitable for the computational node. Using 4-way SMP with 8 GFLOPS SCIMA processors and connecting 4096 computational nodes, an MPP with 131 TFLOPS of performance can be constructed.

3.4 Development of Heterogeneous Multi-Computer System

Heterogeneous Multi-Computer System (HMCS) is a platform to combine general purpose and special purpose MPP systems to simulate complex systems having multiple paradigms or multiple scales. In our Research for the Future Project, we have been developing two types of next generation MPP systems, that is, a system for continuum simulations and that for particle-based simulations. HMCS is the final image of the combination of the two systems which provides an ideal large-scale simulation platform.

The basic concept of HMCS is to distribute multiple computation phases to multiple machine functions keeping the problem-machine fitness and the computational efficiency. This year, we have implemented the prototype of HMCS with CP-PACS MPP with 2048 processors and GRAPE-6 cluster with 8 boards, and have applied it to simulate galaxy formation using SPH (Smoothed Particle Hydrodynamics) with radiative transfer and self gravity.

System configuration of HMCS prototype Figure 5 shows the system diagram of the HMCS prototype. We equip 16 of the I/O processors of CP-PACS with 100base-TX Fast Ethernet interface, and connect them to the GRAPE-6 cluster system through multiple switches with PAVEMENT/PIO connection. A parallelized file server with PAVEMENT/PFS and a parallelized visualization server with PAVEMENT/VIZ are also connected to this system. The full configuration of CP-PACS provides the computational power of 614 GFLOPS for general purpose computation, while 8 boards of GRAPE-6 (256 GRAPE-6 chips in total) provides approximately 7 TFLOPS of sustained performance for many body gravity calculation.

In order to manage the 8 boards of GRAPE-6 system and distribute the work load to them, we constructed an Alpha-based cluster as the host computer for GRAPE-6 boards. Before gravity calculation on each GRAPE-6 board, the particle data such as the mass and location of particles have to be stored to its local memory. Then, particle data are distributed to pipeline processors on all boards uniformly. Hence, each host node must keep the data of all particles. For this purpose, particle data prepared by CP-PACS are distributed to host nodes with PIO, then they are exchanged among all the host nodes. Since PIO does not support global data broadcasting, this is the best way to reduce the amount of data transfer between CP-PACS and GRAPE-6 cluster. Calculation results (acceleration for all particles) are extracted from GRAPE-6 boards by host nodes, and sent back to CP-PACS in a reverse way.

The software of HMCS consists of two elements; an application interface library called by the user program on CP-PACS, and a daemon program which runs on each GRAPE-6 host node to communicate with these routines and pass/extract data to/from the GRAPE-6 board. The functions of these routines are based on the original GRAPE-6 manipulating functions, and application programmers on CP-PACS can describe the program as if there exist the GRAPE-6 boards directly connected to CP-PACS.

The CP-PACS and GRAPE-6 cluster perform their calculations individually and alternatively. However, they can be partially overlapped by particle data transfer on the CP-PACS side. In our current application (described in the next part), we have succeeded to hide the communication overhead partially. On the GRAPE-6 side, the data set-up and actual computation time is very short compared with that on CP-PACS. Using 1024 PUs of CP-PACS and four GRAPE-6 nodes, a single step of radiation transfer SPH with self-gravity for 131072 particles takes 12.3 seconds. Approximately 40% of the processing time is consumed for data transfer between the two systems, but it is partially overlapped with SPH computation on CP-PACS.

A typical simulation consists of 25000 steps with this problem size, and it can be performed within 3 days. It is impossible to perform the self-gravity calculation for 131072 particles with CP-PACS only. Therefore, even if a large fraction of total calculation time is consumed by data transfer, this prototype of HMCS has provided a great computational performance for complex computational physics problems.

Application with HMCS prototype There exists a hierarchical structure in the universe such as stars, black holes, star clusters, galaxies, galaxy clusters, and a large-scale structure. Three basic processes control this wide variety of systems; they are hydrodynamical process, gravity, and radiative process. Since astronomical phenomena are highly nonlinear, a large dynamic range for hydrodynamics is required in numerical approaches. A very effective method for this purpose is the Smoothed Particle Hydrodynamics (SPH), in which physical quantities are expressed by a superposition of smoothed particles whose individual sizes vary with the local density.

The gravity part of the SPH calculation requires N^2 operations for N particles. With our HMCS prototype, these gravity calculations are processed very efficiently with multiple GRAPE-6 boards. The SPH calculation with GRAPE is often called GRAPE-SPH.

The most reliable treatment of radiation is to solve radiative transfer (RT). This, however, is a highly complex physical process which requires a large amount of computations. Astrophysical hydrodynamics coupled with full radiative transfer has never been made. We have developed an effective scheme of radiative transfer for SPH simulation, and implemented it on CP-PACS. By coupling it with the self-gravity calculation on GRAPE-6 in our HMCS prototype, we have carried out a RT-SPH calculation with self-gravity for the first time.

With the RT-GRAPE-SPH scheme, we have simulated the galaxy formation in the early history of the universe. It is widely believed that the universe was reionized at redshift epochs higher than 5, and most galaxies formed after this reionization. Thus, we should investigate the galaxy formation in background ultraviolet (UV) radiation. The permeation of background UV

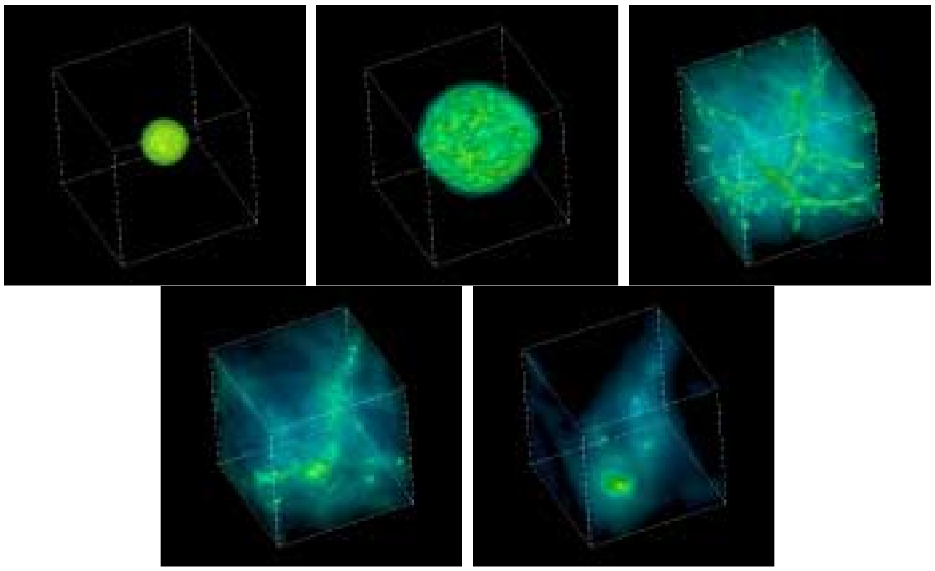


Figure 6: A Simulation of Galaxy Formation by RT-SPH with HMCS

radiation into protogalactic clouds is controlled by radiative transfer. Hence, RT-GRAPE-SPH is the most suitable approach for this issue. We have pursued the evolution of primordial density fluctuations in a dark matter-dominated universe using 65536 dark matter particles and 65536 SPH particles.

In Figure 6, the evolutionary sequence of the simulated galaxy formation is shown. At the initial stage, a density fluctuation is generated to match a cold dark matter spectrum. This fluctuation expands with the cosmic expansion, and simultaneously smaller-scale density fluctuations develop inside to form filamentary structures. Tiny filaments evaporate due to the heating by background UV radiation, whereas larger filaments shrink to coalesce into a condensed rotating cloud. This rotating cloud would evolve into a galaxy. This simulation has revealed that the background UV radiation plays an important role for the final structure of the galaxy. We expect that further work of RT-GRAPE-SPH with HMCS would lead to significant advance in our understanding on the origin of the hierarchical structure in the universe.

4 Summary and Future Perspectives

In our five-year project, we have carried out research and development on key elemental technologies for next generation MPP with emphasis on high performance processor archtecture, new methodology of interconnection network and high throughput visualization and I/O processing. To solve the serious problem of processor-memory gap, we have developed a novel processor architecture SCIMA which utilizes data locality to minimize the data traffic among on-chip/off-chip circuits. PAVEMENT/PIO and VIZ provide a high throughput I/O feature for various types of parallel processing systems and high speed visualization for large amount of 3-D data, respectively.

We wish to emphasize, however, that enhancing the computational power of processors and network, as has been traditionally pursued so far, would not be sufficient to achieve real high performance in the next generation of scientific simulations. They have to treat complex phenomena in which both short-ranged and long-ranged interactions and multiples of scales are involved, and hence require efficiency in processing more than a single type of computations. HMCS is a novel approach to combine a general purpose supercomputer and a special purpose one for such complex physics simulations. The former system handles continuum simulations which requires flexibility in programming, and the latter one achieves very high speed computations of routined calculations in particle-based simulations. Thus HMCS combines high flexibility and high performance, distributing the calculation load in a best way possible for each sub-system.

Looking ahead, we envisage further development of the HMCS concept. Coupling two heterogeneous computers, while powerful, still suffers from potential problems from communication between the systems and load balancing. These problems are resolved if we combine general purpose and special purpose processors within each processing node, and connect them with high performance opt-electrical hybrid network to configure an MPP system.

It is our assertion that this kind of methodology will be necessary, and indeed will lead to an ideal platform, for the next generation of large-scale scientific simulations characterized by multiple interactions and scales.

5 Presentations and Publications

5.1 Presentations at International Conferences

[1] T. Boku, M. Matsubara, and K. Itakura, "PIO: Parallel I/O System for Massively Parallel Processors", LNCS-2110, European High Performance Computing and Network Conference 2001, pp.383-392, Jun. 2001 (Springer-Verlag).

[2] M. Kondo, M. Fujita, H. Nakamura, "Software-Controlled On-Chip Memory for High-Performance and Low-Power Computing", HPCA-8 Work-in-progress Session, 2002 (to appear)

5.2 Presentations at Domestic Conferences

[1] K. Itakura, T. Boku, and M. Matsubara, "Parallel visualization for parallel data stream", Proceedings of Joint Symposium on Parallel Processing (JSPP 2001), pp.189–196, Jun. 2001.

[2] T. Boku, J. Makino, H. Susa, M. Umemura, T. Fukushige, and A. Ukawa, "Space radiation transfer and hydrodynamics calculation with self-gravity on Heterogeneous Multi-Computer System", Proceedings of High Performance Computing and Computational Science Symposium (HPCS 2002), pp.17-24, Jan. 2002.

[3] T. Ohneda, M. Kondo, and H. Nakamura,"A Study of Memory Access Organization on SCIMA", IPSJ SIGARC, ARC-144-29, pp.165-170, 2001.

[4] M. Fujita, M. Kondo, H. Nakamura, S. Chiba, and M. Sato,"A framework of the Optimize Compiler for Software-Controlled On-Chip Memory",IPSJ SIGARC, ARC-146-6, 2002.

5.3 Journal Papers

[1] M. Kondo, H. Nakamura, T. Boku, "Performance Optimization Techniques on SCIMA and its Evaluation", IPSJ Transactions on High Performance Computing Systems, Vol.42, No. SIG 12(HPS 4), pp.37-48, 2001.