T2K Open Supercomputer: Inter-University and Inter-Disciplinary Collaboration on the New Generation Supercomputer

Hiroshi Nakashima
Academic Center for Computing and Media Studies
Kyoto University
ACCMS North Bldg., Yoshida Hommachi, Kyoto
606-8501, JAPAN
h.nakashima@media.kyoto-u.ac.jp

Abstract

Our new supercomputer system, which will begin its operation on July in the Academic Center for Computing and Media Studies, is based on the T2K Open Supercomputer Specifications designed by the collaboration with U. Tsukuba and U. Tokyo. The specifications are not only the result of our new concept and procurement principle, but also the conclusion of the discussion between computer scientists/engineers in the center and computational scientists of a wide spectrum of research field in our university. The collaboration of computer and computational scientists is becoming tight for the high-performance computing on the Open Supercomputer and has already shown its creativity with useful and novel software methods.

1 Introduction

The Academic Center for Computing and Media Studies (ACCMS) has various missions on computation and communication infrastructures and services provided through them. Among those missions, the design and operation of a large scale supercomputer system are highly prioritized in order to provide a state-of-the-art high-performance computation power to a wide range of researchers both inside and outside our university. To pursue the mission on supercomputing, both inter-university and inter-disciplinary collaborations are essential. Since modern supercomputers are massively parallelized and deeply layered with respect to both hardware and software aspects, the design and usage of them require a wide spectrum of professional knowledges and skills.

For example, designing a supercomputer system requires professional architects of processors, memory systems, interconnections and storages to cover these hardware layers. As for the software stack, operating systems, compilers, communication and numerical libraries, ready-made and newly programmed applications have to be fully covered. These widely spread technology area of *supercomputer science* cannot be covered by a supercomputer center or even by a university as a whole, and thus we need to assemble researchers in multiple universities for collaborative design.

The parallelized and layered architecture of supercomputer hardware and software is also the source of the necessity of inter-disciplinary collaboration. A computational scientist who has a large scale application should have deep knowledge of the problem to be solved and may know something about its parallelization method. However, it is hardly expected the researcher fully understands how the application program is compiled and executed on modern superscalar processors, how its memory accesses conform the caches and memory hierarchy, how the parallel processes communicate with each other, and so on. Furthermore, even if the researcher recognizes these complicated issues, it is harder to force him/her into being highly skillful to find a state-of-the-art algorithm/implementation quickly. Thus our mission is not only to provide a high-performance system for the researchers but also to collaborate with them to attack their problems jointly.

This paper describes our recent activities of an inter-university collaboration for the system design and an inter-disciplinary one for an efficient parallel implementation of a simulation onto the designed system as follows. Section 2 discusses the architecture of the *T2K Open Supercomputer*[3] which researchers of three universities designed for their new supercomputer systems. Then Section 3 presents an efficient implementation of the particle simulator for distributed memory systems which we jointly developed with the researchers in the Research Institute for Sustainable Humanosphere (RISH) in our Uji campus. Finally Section 4 concludes the paper briefly showing our

prospect of the evolution from the collaborations to the *knowledge society infrastructure* and *knowledge grid computing*.

2 T2K Open Supercomputer

2.1 T2K Alliance

Aiming at collaborative work on the specification design for new supercomputers, the following three university supercomputer centers established the T2K Open Supuercomputer Alliance in May 2006; the Center for Computational Sciences, University of Tsukuba; the Information Technology Center, University of Tokyo; and the Academic Center for Computing and Media Studies, Kyoto University. These universities have led supercomputer science for many years. University of Tokyo and Kyoto University have long histories of 40 years to provide high-performance computing power to nation wide researchers through their computer centers. University of Tsukuba has the sole accomplishment in Japanese universities to place their CP-PACS supercomputer at the top of the TOP500 list as the world fastest supercomputer in 1996.

This activity is for creative procurements at the initiative of universities based on technology market researches, rather than those by choosing machines from the product market and at the initiative of system venders. This creative and university-initiative approach results in open specifications, with respect to the architecture, software stack and usage of the new supercomputers, to give an efficient HPC solution with the most advanced technologies to a wide-spectrum of users in Japanese academe.

At the last-one-mile point of the supercomputer procurement, we alliance members are planning to the next steps from our open supercomputers as the first base. An important issue which the alliance is targeting is the high productivity of high-performance parallel programs with easy-to-write programming languages, sophisticated and efficient compilers, highly tuned communication and numerical libraries, and so on. We also give a high priority to collaborative work with computational scientists for their parallel program development and tuning, high level education on high-performance computing, and the establishment of virtual research organizations hubbed by the Grid facility connecting three member supercomputer centers.

2.2 Open Supercomputer Specifications

We named our specifications for new supercomputers *Open Supercomputer Specifications* aiming at making our supercomputers *open* in the following three aspects.

Open Hardware Architecture The specifications clearly require that the supercomputers have to be built by commodity devices and technologies, rather than those dedicated only to high-performance computing. For example, their shared memory node must consists of processors of 64 bit x86 architecture which is obviously the mainstreamer of current and near future processor technology, and DDR2 memories also commonly used in servers and high-end PCs. As for the interconnection between nodes, our requirements can be satisfied a bundle of links of Infiniband x4 DDR or Myrinet 10G both of which are leading interconnection technologies for commodity PC clusters. Since these devices and technologies are the most advanced and cost effective in the current IT market, the resulting architecture should be the best solution in the range of our procurement budget.

Open Software Stack We also clearly state that the software stack should be built on top of the open-source operating system Linux. The other major components of the software stack, OpenMP based parallelizing compilers, MPI communication library and Grid middleware and tools, also have open-source solutions commonly used in the PC cluster community. Therefore, it should be easy for PC cluster users to port their programs to the open supercomputers having the open software stack. Moreover, the openness will give us the high portability of software components produced by advance research work of worldwide including ourselves.

Open to User's Needs The supercomputers should much larger and faster than the computers which our users are daily using but have the hardware and software architecture same as the user's own daily environment. Therefore, there are no barriers to use our supercomputers not only for users with numerical applications but also for those who have never used supercomputers for their non-numerical applications. That is, our supercomputers welcome emerging supercomputational scientists in research fields such as information mining, natural language processing, and genomic informatics.

Based on the concept of openness shown above, we specified the following requirements commonly applied to three supercomputer systems.

- The fundamental building block of the system is a shared memory node having 16 processor cores of 64 bit x86 architecture and 32 GB memory of 40 GB/s aggregate bandwidth.
- Each node has a bundle of interconnection links whose aggregate bandwidth is 5 GB.
- The operating system is Red Hat or SuSE Linux based on the kernel v2.6.
- Compilers of Fortran, C and C++ should have OpenMP parallelization API and the capability of automatic parallelization.

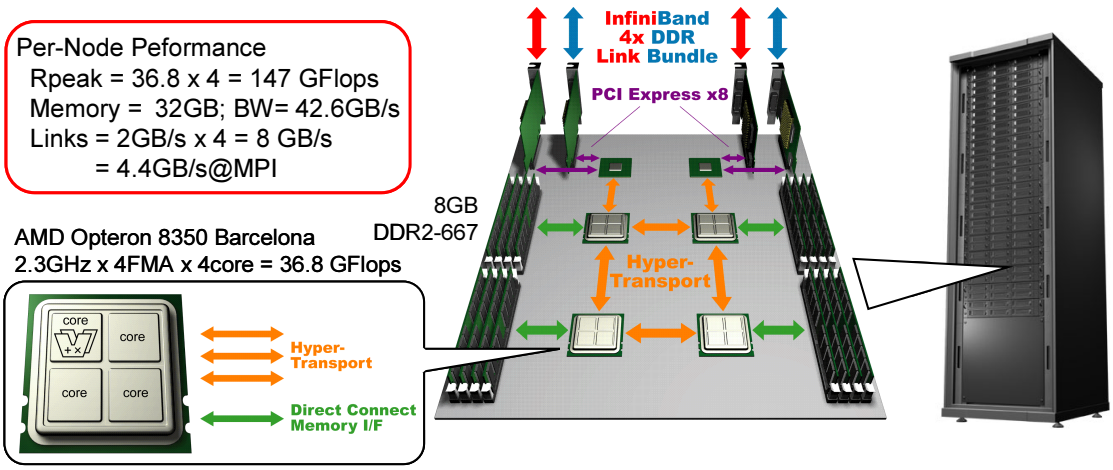

Figure 1. Node Architecture

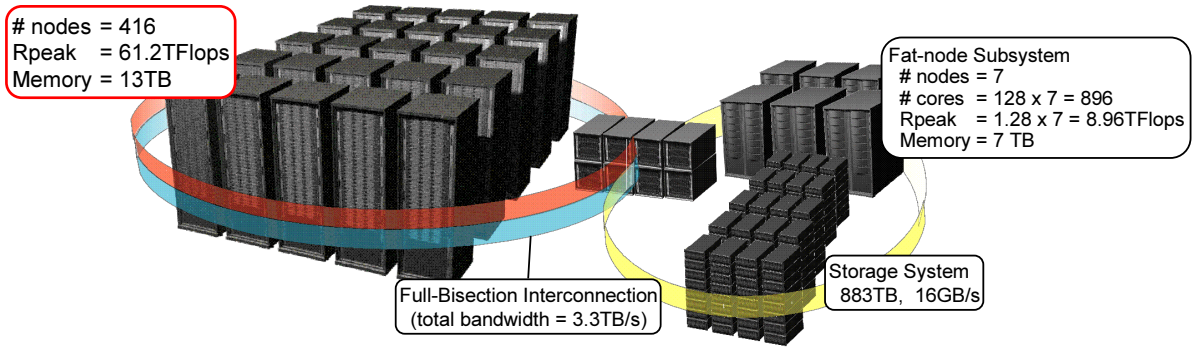

Figure 2. System Configuration in Kyoto University

- The MPI communication library should exerts a high bandwidth of 4 GB/s for a large size message and a short roundtrip latency of 8.5 μs or less for a small size message.
- The performances of a core, a node and a large set of nodes are measured by the standard benchmark set of SPEC CPU2006, SPEC OpenMP and HPC Challenge, in addition to our own benchmark suite for the performance of memory accesses, MPI communications and storage accesses.

Note that the numbers shown above, except for the MPI roundtrip latency, are minimum requirements. Also note that each site has its own requirements not listed above, such as those for system level performance numbers, for batch job schedulers and for site-specific application benchmarks.

2.3 Configuration in Kyoto University

Based on the specifications described above, we called bidding for the supercomputer system in ACCMS, which Fujitsu Ltd. won offering the system configuration shown in Figure 1 and 2.

As shown in Figure 1, a node named HX600 consists of four quad-core processors, AMD Opteron 8350 Barcelona. Since a core can issue two floating point multiply-and-add instructions in its single cycle of 2.3 GHz, the peak performance of a processor is 36.8GFlops. Each processor has a Direct Connect memory interface to an 8 GB DDR2-667 memory and three HyperTransport Links to connect other processors. The HyperTransport links of two processors are also connected with the chip-sets to bridge between each link and two PCI Express x8 links each of which is then connected to a Infiniband 4x DDR host channel adapter. Thus in total, a node has 147 GFlops peak performance, 32 GB memory of 42.6GB/s aggregate bandwidth, and four Infiniband links of 8 GB/s aggregate bandwidth.

As shown in Figure 2, the system has 416 nodes described above and thus has 61.2 TFlops peak performance and a large memory space of 13 TB. As for the interconnections, the system has 138 leaf switches of 24-port and six 288-port spine switches to connect 1664 Infiniband

links from nodes to achieve full bisection bandwidth so that all nodes perform wire-speed communications at once. Besides these T2K Open Supercomputer nodes, our system has other two major components, a fat-node subsystem and a storage system. The fat-node subsystem, a part of which will be provided for radio science as the successor of RISH's A-KDK supercomputer, has seven shared memory SPARC Enterprise MX9000 nodes having 32 processor sockets of quad-core SPARC 64VII and 1 TB memory for each to add 8.96 TFlops peak performance and 7 TB memory space to the system. The storage system have 1024 fast 300 GB disk drives and 768 large 750 GB drives to provide 883 TB disk space accessible through a bundle of 64 Fiber Channel links, the PRIMEQUEST TX580 file server and a bundle of 16 Infiniband links from the Open Supercomputer nodes.

The system has not only high-performance of 70 TFlops in total but also good power-efficiency with 600 kW peak power consumption. Moreover, each computation node has state-of-the-art mechanisms to save electric power by dynamically downscaling its voltage with computation unintensive load and by sleeping with almost no power consumption when it has no load. Thus the system will be greenly operated with 500 kW or less power consumption saving large energy.

3 Efficient Space-Partitioned Implementation of Particle Simulation

Precise and micro-scale simulations of charged particles are indispensable for theoretical and practical research in high-energy physics, space plasma physics, cloud modeling, combustion engineering, and so on. Since the simulation has to cope with a huge number of particles, its parallelization is essentially required. A simple parallelization is to divide the particles into disjoint subsets and to assign each subset to each computation node. Although this method insists that all the nodes share the field information of the whole space the simulation concerns, it is often efficiently implemented on shared memory systems including vector machines[4]. In fact, the A-KEK supercomputer of Fujitsu's PRIMEPOWER HPC2500 nodes has been a powerful mean for the researchers of space plasma physics in RISH and other nation wide institutes.

This simple method, however, has an inherent limitation that the shared memory has to accomodate the huge number of particles. Since a particle is represented by a data structure of about 50 bytes, the number of particles is limited to *only* about 5×10^8 for an Open Supercomputer node with 32 GB memory, and to still insufficient 20×10^9 or less even for a fat-node of our system with 1 TB memory. Thus it is inevitable to find an alternative parallelization for scalable distributed memory environments.

```
for(t=0;t<timesteps;t++){
  bfield();
  position();
  velocity();
  position();
  bfield();
  current();
  efield();
}
```

Figure 3. Kernel Loop of Simulator

A simple distributed memory implementation could be a straightforward extension of the shared memory one, in which each node has a copy of the filed information of the whole space. The obvious drawback of this method is that we need an all-reduce type communication to sum up the contribution of the particle movements to the field in each time-step of the simulation. In addition, a more serious problem is that the field information is too large for a node, more than 500 GB for a three-dimensional simulation with 2048^3 grid points, to accommodate in its local memory.

Therefore, the distributed memory implementation must partition not only particles but also the simulated space. Thus we devised a new space-partitioned load balancing method, named *OhHelp*, in which the space is simply and equally partitioned but the load with respect to both of the number of particles and the size of subspaces is well balanced. This method is the offspring of deep technical discussions between two research groups, one in RISH having deep knowledge and experience about particle simulations and the other in ACCMS having those about parallel processing on distributed memory systems. Thus this research is a good example to prove the importance and effectiveness of inter-disciplinary collaboration of computational and computer scientists.

3.1 Simulation Principle

Figure 3 shows the kernel loop of the baseline serial version of our particle simulator which is based on the KEMPO simulator[2] developed in RISH. The functions `efield()` and `bfield()` update the electrical and magnetic fields, $\boldsymbol{E}$ and $\boldsymbol{B}$, based on the following Maxwell's equations,

$$\begin{aligned} \frac{\partial \boldsymbol{E}}{\partial t} &= c^2 \nabla \times \boldsymbol{B} - \frac{1}{\epsilon_0} \boldsymbol{J} \\ \frac{\partial \boldsymbol{B}}{\partial t} &= -\nabla \times \boldsymbol{E} \end{aligned}$$

where $\boldsymbol{J}$, c and ϵ_0 are the current density, light speed and permittivity, respectively. The functions `velocity()` and `position()` update the velocity $\boldsymbol{v}$ and the position $\boldsymbol{x}$ of each particle of the charge q and mass m by the following equations, respectively.

$$\frac{d\boldsymbol{v}}{dt} = \frac{q}{m}(\boldsymbol{E} + \boldsymbol{v} \times \boldsymbol{B}) \qquad \frac{d\boldsymbol{x}}{dt} = \boldsymbol{v}$$

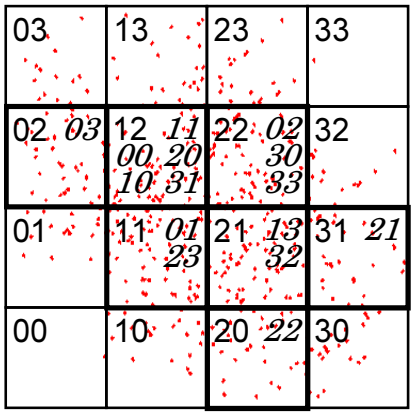

Figure 4. Space Partitioning

Finally, the function `current()` calculates the current density $\boldsymbol{J}$ from the following equation,

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \boldsymbol{J} = 0$$

where ρ is the charge density whose value and its partial derivative are obtained from the position and velocity of particles.

3.2 Space Partitioned Load Balancing by OhHelp

As shown in Figure 4, OhHelp simply partitions the simulated space into subspaces of equal size and assign each subspace to each computation node as its *primary* subspace. In the figure, non-italic numbers are the identifiers of nodes and also those of primary subspaces assigned to them. Each node is responsible for its primary subspace, and also all the particles in the subspace if the numbers of particles in subspaces are well-balanced, or more specifically, if the number of particles P_s in a subspace s satisfies the following inequality for all s,

$$P_s \leq (P/N)(1+\alpha) \equiv P_{limit} \qquad (1)$$

where P is the total number of particles, N is the number of nodes, and α is the allowance factor greater than 0. We refer to the simulation phases in this fortunate situation as those in *primary mode*.

Otherwise, i.e., if the inequality (1) is not satisfied for some subspace s as shown in Figure 4, the simulation is performed in *secondary mode*. In this mode, every node, except for one node (12 in the figure), is responsible for a secondary subspace having too many particles to satisfy (1) in addition to its primary one. For example, the subspace 22 has *helper* nodes 02, 30 and 33 shown in italic style in Figure 4. The particles in a heavily loaded subspaces are also distributed to its helper nodes so that the number of particles Q_n assigned to node n is well-balanced satisfying the following inequality similar to (1) for all n.

$$Q_n \leq (P/N)(1+\alpha) = P_{limit} \qquad (2)$$

Note that since all but one nodes have secondary subspaces, a node whose primary subspace is heavily loaded, e.g., node 22, is not only helped by other nodes but also helps another node 20, as the balancing algorithm orders. Also note that the load in secondary mode is balanced not only in the number of particles but also in the size of responsible subspaces, although the latter load is twice as heavy as that in the primary mode.

The examination whether the load is well-balanced and the mode switching possibly with load rebalancing are performed as follows each time the function `position()` is called.

1. If the inequality (1) is satisfied for all subspaces, the mode stays in or turns to primary. In the case of staying, only the particles crossing subspace boundaries are transferred between nodes by neighboring communications.

2. If the current mode is secondary and the inequality (1) is not satisfied but (2) is satisfiable keeping the secondary subspace assignments, the mode stays in secondary without global rebalancing. Particles may be transferred among the helpers and their *helpand* for the local load balancing in addition to the transfer of the particles crossing boundaries. For the satisfiability check for (2) and the local balancing, see [1].

3. Otherwise, the secondary subspace assignments are performed (or modified) so that Q_n is equal to P/N for all n to accomplish complete balancing. The subspace assignment algorithm is fairly simple because it repeatedly moves the particles in a node having load larger than the average P/N to the other node having load less than the average so that the latter's load becomes equal to the average by helping the former node. For the detailed discussion of the subspace assignment, see [1].

3.3 Performance

We implemented a prototype of a two-dimensional KEMPO-based particle simulator with the OhHelp load balancing method using C and MPI 2.0. The simulator is compiled by Fujitsu's C compiler version 5.6 and linked to Fujitsu's MPI V6 library to run on the PrimePower HPC2500 supercomputer with 128 SPARC 64V processors and 512 GB shared memory. Although our aim is an efficient particle simulation on a distributed memory environment, we used the shared memory system in order to com-

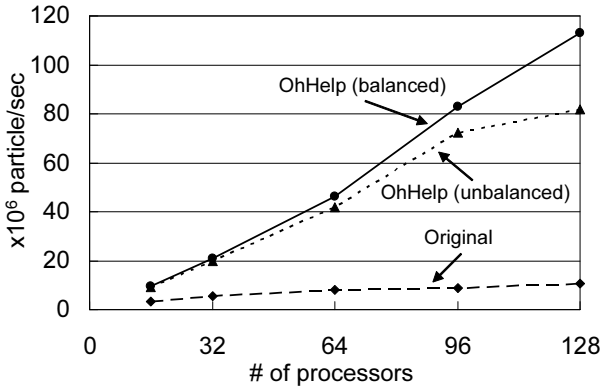

Figure 5. Performance of Simulations

pare our OhHelp version with its original Fortran code automatically parallelized by Fujitsu's compiler version 7.0 so that the particle computation loads are equally distributed among processors.

With these simulators, we carried out simulations of five time-steps with 1.5 and 2.5×10^9 particles residing in a two-dimensional space of 2048×2048 grid points with the periodic boundary condition, i.e., for the surface of a torus. For the OhHelp version, we evaluated its performance with two types of initial particle distribution; one is a *balanced* setting in which particles are uniformly distributed in the 2048×2048 space, while in the other *unbalanced* one particles are packed in a 2048×1100 region in the space.

Figure 5 shows the performance of simulators with various numbers of processors. It is clear that the OhHelp version greatly outperforms the original version and it exhibits a good scalability. That is, the OhHelp balanced 128-processor simulation is about ten times as fast as the original which is hardly scaled up due to the limited shared memory bandwidth for random accesses. The OhHelp's speedup from 16- to 128-processor simulations is about 12-fold and super-linear probably because of the locality improvement of the memory accesses to the field information. These results prove the importance of the space-partitioning not only for distributed memory systems which essentially require it, but also for shared memory systems to improve the regularity and locality of memory accesses.

The memory access locality could also explain the performance difference between the balanced and unbalanced simulations. The difference is quite small when the number of processors is 64 or less to prove that the computation for field data update, doubled in the unbalanced case, is insignificant. However, the differences in 96- and 128-processor cases are significantly large probably because the unbalanced simulation could not benefit from the access locality improvement due to the double-sized field data. In fact, the 128-processor speedup of the unbalanced simulation is just linear rather than the super-linear in the balanced one.

4 Conclusions

In this paper, we described our recent activities of inter-university and inter-disciplinary collaborations carried out in ACCMS. Although both projects are still ongoing, we have already shipped important products, the Open Supercomputer Specifications and the OhHelp load balancing method for particle simulations.

From these accomplishments we now start to the next steps for *knowledge society infrastructure* and *knowledge grid computing* which our global COE project aims at. As the project plan clearly states, the goal of the project is "to facilitate the smooth circulation of knowledge among societies, communities, organizations, and individuals by organizing inter-disciplinary research teams." Although it is obvious that our collaborations definitely match the objective, we need to enhance them further to reach the goal. As stated in Section 2.1, three universities has already begun tighter cooperative work on software development and Grid computing which should directly lead us to the goal of the knowledge grid computing. The work also tightly related to the inter-disciplinary collaboration to circulate the knowledge of computer scientist among computational scientists, for example, in the form of *methodology library*.

Acknowledgments The author would like to express his sincere appreciation to the following people for their contributions to the work presented in this paper; Prof. Mitsuhisa Sato and Prof. Taisuke Boku of U. Tsukuba, and Prof. Yutaka Ishikawa of U. Tokyo who are core members of T2K Open Supercomputer Alliance; Prof. Yoshiharu Omura and Assoc. Prof. Hideyuki Usui who jointly worked with us for the improved implementation of their particle simulator.

References

[1] H. Nakashima, H. Usui, and Y. Omura. Ohhelp: A simple but efficient space-partitioned dynamic load balancing for particle simulations. In *IPSJ SIG Notes*, 2007-HPC-113, Dec. 2007.

[2] Y. Omura and H. Matsumoto. Kemop1: Technical guide to one-dimensional electromagnetic particle code. In *Computer Space Plasma Physics: Simulation Techniques and Software*, chapter 2. Terra Scientific Publishing Co., 1993.

[3] T2K Open Supercomputer Alliance. http://www.open-supercomputer.org/.

[4] H. Usui and Y. Omura, editors. *Advanced Methods for Space Simulations*. TERRAPUB, 2007.