Astrophysical Supercomputing Using Particles
IAU Symposium, Vol. 208, 2003
J. Makino and P. Hut, eds.

The GRAPE project: Current Status and Future Outlook

Junichiro Makino
Department of Astronomy, School of Science, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033, Japan

Abstract. I'll briefly overview the present status and the future of the GRAPE project. GRAPE (GRAvity PiPE) project is a project to design, develop and use special-purpose computers for astrophysical N-body simulations to do large-scale N-body simulations. Our first machine, GRAPE-1 was completed in 1989 and offered the speed of 240 Mflops. Since then, we have continued to develop newer and faster machines, and the newest machine, the GRAPE-6, has achieved the peak speed of 32 Tflops. I'll briefly discuss GRAPE-6 and its parallel architecture, and then discuss the possible form of GRAPE-7, the next generation machines.

1. Introduction

In this paper, I give an overview of the present status and the future of the GRAPE project. In section 2 I give a overview of basic concept of GRAPE hardware and the algorithms used with GRAPE hardware. In section 3 I describe the GRAPE-6 system, which is completed recently. In section 4 we discuss possible design and performance of the next generation machine. Section 5 is for summary.

2. Overview of GRAPE hardware and software

2.1. Special-purpose hardware

GRAPE Project (Sugimoto et al. 1990, Makino and Taiji1998) is a project to develop and use special-purpose computers to perform large-scale astrophysical N-body (and SPH) simulations. Figure 1 show the basic concept of the GRAPE hardware.

From the viewpoint of the calculation cost, the rather troublesome nature of the gravitational interaction is that it is a long-range interaction. All particles in a system interact with all other particles in the system, and we cannot apply cutoff.

If we use straightforward direct force calculation, the calculation cost scales as $O(N^2)$, where N is the number of particles in the system. If we can make the special-purpose GRAPE much faster than the general-purpose host for force calculation, we can speed up the overall calculation by a very large factor.

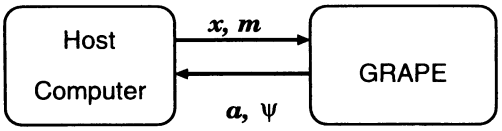

Figure 1. Basic idea of GRAPE

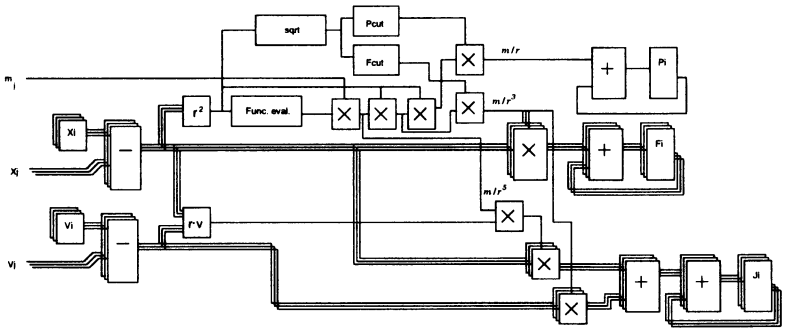

Figure 2. GRAPE pipeline

Figure 2 shows the basic structure of the pipeline processor we developed. In this processor, we organize the arithmetic units following the dataflow of the force calculation. Thus, we can use a large number of arithmetic units connected in a simple way. A single pipeline would contain around 30 arithmetic units. If we choose to calculate the first time derivative, which is need to implement the Hermite integration scheme, this number doubles to around 60. Thus, a single pipeline needs very large number of arithmetic unit.

This need for very large number of arithmetic unit would have been a drawback when the number of transistors we can use to make one computer was small. In 1960s, even the fastest supercomputers had arithmetic units which required multiple clock cycles to produce one result. Thus, increased number of transistors were used to reduce the number of clock cycles needed to perform one operation.

At present, however, the situation is totally different. A typical microprocessor of year 2001 contains more than 10^7 transistors, about 100 times more than what is needed to implement a single floating-point multiplier.

The reason why we do not have a personal computers with 100 processors on a chip is that computer architects have not yet find out a way to connect these 100 processors to each other and also to the memory units, and at the same time achieve a good performance for wide range of applications.

If we limit the range of the problems to solve, however, to make use of a large number of arithmetic unit is almost trivial. This is the essential reason why we can achieve a much higher speed with special-purpose designs. In fact,

https://doi.org/10.1017/S0074180900206979 Published online by Cambridge University Press

the present GRAPE-6 chip implements six force calculation pipelines for Hermite scheme, and one predictor pipeline for the individual timestep algorithm, resulting in around 400 arithmetic units on a chip. Here, 400 is larger than 100, the number given above, mainly because we adopted 32-bit format for most of the operations. This is another advantage of the specialized architecture. Since one arithmetic unit is used only for one purpose, we can optimize each of the arithmetic units in the pipeline for the accuracy required for the operation it performs. In general-purpose computers, it would be hard to change the accuracy depending on the operation. More importantly, it would be difficult to achieve any speedup from such a change, since full-accuracy calculation can be done in single cycle anyway.

If we have multiple pipeline processors sharing the same memory unit, we can use them to calculate the forces on different processors in parallel. If we have multiple pipelines with separate memory unit, we can use them either to calculate the forces on different particles from the same particles, or to calculate the forces from different particles to the same particles.

2.2. Barnes-Hut treecode

At least for some applications where the requirement for the accuracy of the calculated gravitational force is not too stringent, we can apply approximate methods such as the Barnes-Hut treecode (Barnes and Hut 1986) and achieve the reduction of the calculation cost from $O(N^2)$ to $O(N \log N)$ per timestep. Even so, calculation time usually limit the number of particles N, and therefore the resolution and accuracy, of the simulations we can do. In cosmologies and galactic dynamics, the accuracy required for the forces on particles is rather low, simply because the overall accuracy of the result is limited by two-body relaxation (Makino et al. 1990, Hernquist et al. 1993). To obtain more accurate result, it is more important to increase the number of particles than to improve the accuracy of the particle-particle force.

It turned out that we can also use the simple hardware as shown in figure 1 to accelerate the tree algorithm(Makino 1991, Athanassoula et al. 1998). In the tree algorithm, we calculate the force on a particle by traversing the tree structure.

To implement the tree traversal itself in hardware would be a challenging task, but we can make use of the special-purpose machine of figure 1, by doing the tree traversal on the host. Of course, if we do the tree traversal for each particle, the calculation cost would not be reduced. However, we can slightly change the algorithm and traverse the tree for a group of particles. The list of nodes and particles constructed during the tree traversal can then be used to calculate the force on all particles in the group. The tree structure itself provide a good way of grouping particles. as we increase the size of the group, the calculation cost of the host is reduced, but the calculation cost of the GRAPE side increases since the length of the list becomes longer. Thus, there is an optimum value for the size of the group.

This algorithm is quite powerful, resulting in the speedup of a factor of 10 to more than 100, depending on the required accuracy. The speedup is higher for higher accuracy. We can even use high-order multipole expansions with GRAPE hardware, using the pseudoparticle multipole method in which we express the

https://doi.org/10.1017/S0074180900206979 Published online by Cambridge University Press

multipole expansions by means of a distribution of particles(Makino 1999, Kawai and Makino 2001).

2.3. Individual timestep

For many astrophysical problems, the use of the individual timestep (Aarseth 1963) is essential to achieve high accuracy and reasonable calculation speed. When the density contrast is high, there could be $O(N)$ difference in the speed of individual timestep algorithm and shared timestep algorithm. Thus, for GRAPE hardware it is clearly necessary to support efficient implementation of individual timestep algorithm.

This can be achieved with relatively small additional cost. The difference between the force calculation in the individual timestep algorithm and that in the shared timestep algorithm is that we need to predict the positions of particles which exert the forces on particles in the present block. By adding the hardwired pipeline to evaluate the predictor polynomial, we can let GRAPE hardware do this prediction (Makino et al. 1997).

Thus, GRAPE hardware can handle two most widely used algorithms, the individual timestep algorithm and the tree algorithm, and therefore can cover most of astrophysical N-body simulations, except for those with very small number of particles. The wide range of scientific results obtained by using GRAPE hardwares are described in a number of contributions to this proceedings. So here I'll concentrate on the description of GRAPE-6 hardware, and possible directions of the next generation machines.

3. GRAPE-6

GRAPE-6 is the successor of the GRAPE-4, which has achieved the peak speed of 1 Tflops in 1995. The processor chip of GRAPE-4 implemented a single force pipeline, which calculates the force between two particles in every three clock cycles. GRAPE-4 chip was made using a 1μm design rule and the number of transistors was around 400K.

3.1. GRAPE-6 chip and board

For GRAPE-6 chip, we used a 0.25μm design rule. As a result, we could use aboout 7 million transistors in one chip to implement six pipelines each of which can calculate one interaction per clock cycle. Also, the clock speed was increased from 32 MHz of GRAPE-4 to 90 MHz. These two improvements combined give a factor of 50 improvement in the speed of a chip. In other words, single GRAPE-6 chip offers the speed slightly faster than that of a single GRAPE-4 board with 48 GRAPE-4 chips.

A single GRAPE-6 chip integrates all basic functions of a GRAPE system, including the interface to memory chips and interface to host. The memory interface has the width of 72 bit (with ECC) and operates at 90MHz. It takes 8 clock cycles to read the data of one particle. Therefore, we use one pipeline as eight virtual pipelines. One chip calculates the force on 48 particles in parallel.

Current configuration of GRAPE-6 consists of 1024 GRAPE-6 chips, for the peak speed of 32 Tflops.

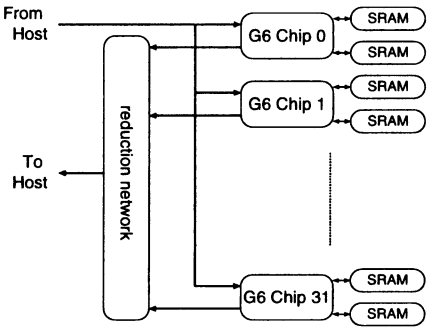

Figure 3. A GRAPE-6 board

Figure 3 shows the structure of single board. 32 GRAPE-6 chips, each with its own memory unit, is connected to the external interface by one broadcast network for data input and one reduction network for data output. GRAPE-6 board is designed so that different chips calculate the forces on the same particles, but from different set of particles. Thus, it is necessary to have some hardware to take summation of forces calculated on 32 different chips. This hardware is implemented using FPGA chips from Altera. For the output data format, we used block floating point format with the exponent pre-specified. By this way, we can obtain exactly the same result on machines with different number of GRAPE-6 chips. Also, the use of the fixed-point format simplified the design of the reduction network.

3.2. The network architecture

The overall architecture of GRAPE-6 is radically different from that of GRAPE-4. The changes are introduced primarily to achieve a good performance with available technology.

GRAPE-6 is around 50-100 times faster than GRAPE-4. This means that even when we use $O(N^2)$ direct summation, the host computer and the communication must be 10-100 times faster than what was used with GRAPE-4. To improve the speed of the communication turned out to be the harder. For GRAPE-4, we originally used DEC TURBOChannel and then moved to PCI. Both offered the speed of around 100 MB/s. Thus, we need at least 1 GB/s, as the throughput for the link between the host and GRAPE.

What we adopted as the interface for GRAPE-6 is still PCI, since practically speaking there was no other choice. PCI is the standard I/O bus for PCs, which at present offer the best price-performance. The problem with PCs is that a single PC is not fast enough, but right now they are so cheap that we can use as many PCs as we like as the hosts for GRAPE-6.

Thus, the simplest way to construct a 100 Tflops GRAPE-6 is to construct 100 copies of small GRAPE-6s (32-chip, single-board system connected to a PC), and connect all the host PCs with fast network (see figure 4).

https://doi.org/10.1017/S0074180900206979 Published online by Cambridge University Press

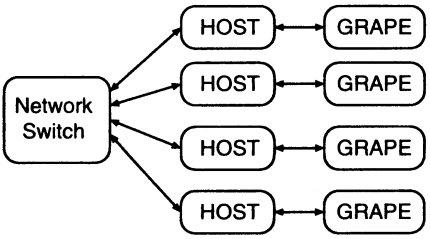

Figure 4. A simple parallel GRAPE system

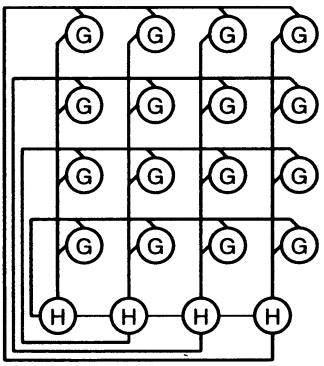

Figure 5. A $n \times n$ GRAPE-6 configuration

One problem with this setup is that it is not well suited to run the simple direct summation algorithm, though it works perfectly for the parallel version of the tree algorithm. In the case of the direct summation, we have basically two possibilities. One is let all host processors to have complete copies of the system. Each processor calculates the force on its share of particles, and updates their orbits. After that, they exchange the updated particles so that all processors have all particles updated. In this case, the communication cost is $O(N)$ and is independent of the number of processors p, while calculation cost is $O(N^2/p)$.

The other is let each processor to have only its share of particles. In order to calculate the force on them, each processor still need to receive the data of all other particles in some way. Thus, the communication cost is again $O(N)$.

With GRAPE-6, we solved this problem by attaching an elaborate communication network to the side of the GRAPE hardware. Figure 5 shows the conceptual view. The problem with the parallel system was that each GRAPE, and therefore each host, needed to receive the data of all particles at each timestep. GRAPEs have to receive the data, but they do not have to receive them from their host. In figure 5, we organize 16 GRAPE boards into 4×4 grid, so that if we regard 4 boards in one column as a whole, it can receive data from all four hosts. We add an additional path, through which host i can broadcast the data

https://doi.org/10.1017/S0074180900206979 Published online by Cambridge University Press

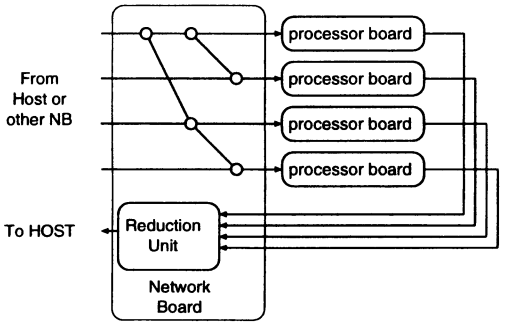

Figure 6. Actual GRAPE-6 network interface

to all GRAPE-6 boards in row i. We use this path to distribute the data from all hosts to all GRAPE-6 boards. GRAPE-6 board with index ij is connected to host i, but it can receive the data from host j as well. Each host can obtain the forces on its particles, by broadcasting the data to GRAPEs in its column and take summation of partial forces. In actual GRAPE-6 hardware, this summation is performed by hardware.

At present, we have two clusters of GRAPE-6, each with 16 GRAPE-6 boards and 4 host computers. The lateral broadcast is actually done through a tree-like network (see figure 6), so that we can use the machine as four independent machines each with 4 boards, 2 hosts with 8 boards, and 4 hosts with 16 boards.

For larger configurations, it is possible to extend the 4×4 grid to 8×8 or even larger grid. However, we plan to make more 4×4 clusters, and connect the hosts by Gigabit Ethernet. We can still use this 4 clusters of 4×4 system as single 8×8 system, by using half of the hosts as the network nodes.

In principle, we could use a system with 64 host-grape pairs as 8×8, by using 56 out of 64 hosts as network nodes. However, we did not take this approach because the host computer, or actually the network interfaces and switches, would be too costly.

3.3. Measured performance

Here, I present the performance of 4×4 single cluster GRAPE-6 for simple individual timestep code and treecode. The host computer is a 1.7 GHz Intel P4 box with i850 chipset, under Linux Kernel version 2.2.17. All timings are done with g++/gcc compilers version egcs-2.91.66.

Figure 8 shows the speed of the individual timestep algorithm in terms of Tflops, as functions of the number of particles for clusters of different sizes. We used the Plummer model in the standard unit as the initial condition. The softening is 1/64 for all calculations. One can clearly see that the performance scales quite well with the number of hosts and GRAPE boards.

With the network architecture discussed in the previous section, the host computers need not exchange any particle data between them. However, still

https://doi.org/10.1017/S0074180900206979 Published online by Cambridge University Press

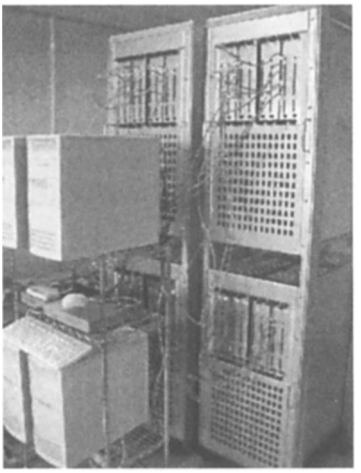

Figure 7. A 32-board GRAPE-6 system

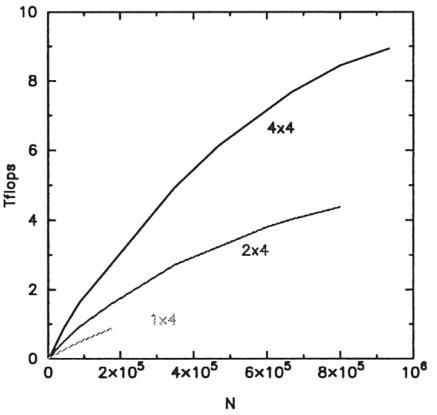

Figure 8. Calculation speed for individual timestep algorithm

https://doi.org/10.1017/S0074180900206979 Published online by Cambridge University Press

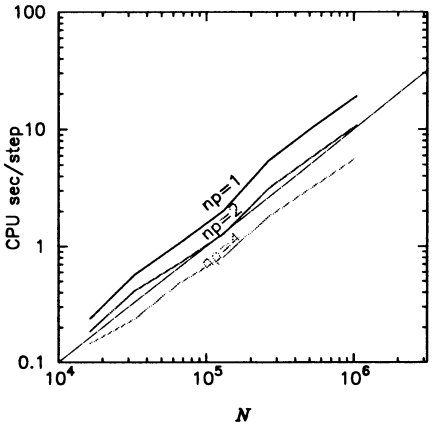

Figure 9. Calculation speed for treecode

they need to synchronize at the beginning of the each blockstep, and they also need to compare the global minimum time and the local minimum time, to decide if it can perform time integration of particles in its current block. At present, the hosts communicate with MPICH/p4 software with TCP/IP on 100M bit fast ethernet. The startup overhead of the communication is visible for very small values of N. Of course, for systems with small core, the average number of particles that share the same time becomes small, and communication overhead becomes somewhat larger.

Figure 9 shows the performance of treecode. This treecode is a newly written version based on orthogonal recursive multi-section, a generalization of widely used ORB tree which allows a division to arbitrary number of domains in one dimension, instead of allowing only bisection. In this particular timing result, however, the result is the same as that for ORB.

The distribution of the particles is again the Plummer model. One can see that the scaling is again pretty good. 4 processor calculation is 1.9 times faster than 2 processor calculation. We can see that the 100 Mb ethernet is fast enough for treecode with accelerated with GRAPE. Clearly, the performance bottleneck is the speed of the host computer. Over the time GRAPE-6 will be used, we expect the speed of the host computer by a large factor, which almost directly will be reflected to the speed of the treecode.

4. Next generation?

Since GRAPE-6 is just finished (well, the full configuration is yet to be delivered), it would sounds too early to talk about the next machine. On the other hand, if we want to have a faster machine in 5 to 6 years from now, we have to start pretty soon since making a big machine is a time-consuming process.

The first question to ask is what kind of systems are primary target for the full-size system. The second question would be what algorithm should we

https://doi.org/10.1017/S0074180900206979 Published online by Cambridge University Press

use. Of course, to some extent what system we want to model depends on what hardware we have. So, in the following we discuss primarily what kind of hardware we can develop.

Let us start with predictions for the available technology. For GRAPE-4, we have used 0.25μm design rule. Assuming that we will get the sample chip by 2003 or 2004, 0.13μm or somewhat finer rules should be available. Assuming the standard scaling, this offers a factor of 6 improvement in the gate density (another 50% comes from improved wiring etc)and a factor of two improvement in speed, and about the same power consumption. Actually, there were a few problems with the physical design of GRAPE-6 chip, which limited its clock frequency below 100 MHz. With appropriate physical design it should be possible to achieve at least 300 MHz with the next generation chip. This, in theory, will give us a chip with the speed of 600 Gflops, 20 times faster than GRAPE-6 chip.

This improvement is somewhat less than what was achieved with GRAPE-6, which is a factor of 50 faster than GRAPE-4. This is simply because we have used rather conservative technology with GRAPE-4, and fairly aggressive one with GRAPE-6. Even so, by putting together 3,000 chips, we can reach 2 Petaflops. This speed will allow us to perform 1M particle simulation of thermal evolution of globular clusters, for cosmological simulations, around 10^{10} particles would be possible.

There are a number of design decisions at all levels of architecture which we have to make correctly to complete a usable machine within the project period and budget. In the following, we'll briefly discuss a few of them.

The first question is how to interface the memory and pipelines. With GRAPE-4, we can let 48 chips to share a single memory unit. With GRAPE-6, we changed this design so that each chip has its memory. This change was necessary to keep the number of pipelines visible from the software becoming too large. With individual timestep algorithm, it is necessary that the hardware gives acceptable performance when asked to calculate forces on relatively small number of particles. GRAPE-4 needed the minimum of 96 particles to keep all pipelines on board busy. A single GRAPE-6 chip needed 48 particles. A GRAPE-6 board also needed 48 particles, since all chips on a board calculate the forces on the same set of 48 particles. However, boards connected to different hosts calculate the forces on different particles. Thus, a 64-board system needs about 400 particles.

A GRAPE-7 (well, for the time being, I'd call the next machine GRAPE-7) chip gives 20 times faster computing speed. Thus, to keep the visible number of pipelines to be the same as GRAPE-6, we will need 20 times faster memory interface, or the speed of about 15 GB/s compared to 720MB/s of GRAPE-6. Even if we could make the interface work at 300MHz, we'll need around 500 pins and minimum of 15 memory chips per processor chip, which will push up the production cost by a fair factor.

Other possibilities are (a) to reduce the memory chips to two or so, to achieve similar pin counts as GRAPE-6 or (b) integrate memory and processor to a single chip. From pure technological point of view, option (b) is the best, since we can achieve very high memory bandwidth without any problem. However, SRAM (or even DRAM) cells which can be integrated with logic gates are large, and it will be difficult to integrate more than a few Mbits. This size will be too

small for small systems with 10 or less chips. Thus, practically speaking, the only option would be to use a small number of separate memory chips, as we did with GRAPE-6. This will push the number of particles to around 200.

Next concern is the interface to the host. GRAPE-6 chip has one input port and one output port, and both operate at 90MB/s. Currently, this speed is not really a bottleneck, since calculations on the host take more time than the communication. However, if the host is sufficiently fast, the communication would become the bottleneck.

In the timescale of 5 years or so, we can expect the calculation speed of the single CPU host to be improved by a factor of 5 to 10. Thus, though a factor of 10 improvement in the communication speed is not a real necessity, a factor of 5 or so would be desirable.

The question is what interface protocol we should use. Right now, PCI is okay, since there is no real alternative. However, the speed of standard PCI is limited to 133MB/s, and 64-bit, 66MHz PCI is not popular yet. Also, there are several other potential alternatives like PCI-X, next-generation IEEE-1394, Infiniband, 3GIO etc, etc. It would be perhaps better to not to choose a particular interface now, but just to assigning a some level of speed to the host interface of the single chip.

Most of the next-generation I/O standards offers the speed in the range of 500MB/s to 1GB/s. In order to match to this speed with a single-chip system, a single chip must have the I/O port with the speed of 500MB/s or so. With currently available technology, it is not too difficult to achieve this level of speed. The simplest is to use the LVDS signal level standard, which works easily with 1GHz or higher clock. With 1.2 GHz clock, 600 MB/s needs only 4 or 5 signal pairs, or just 10 wires. With such a high-speed transfer, however, error detection and correction will become necessity.

The next thing we need to consider is the way we assemble hundreds or thousands of chips to a single machine. Our "traditional" approach has been to design a board with fairly large number of chips (16-48), and assemble relatively small number of these boards to achieve massively parallel system. The implicit assumption we took for granted is that the number of host computers (or the I/O port of the host) is much smaller than the number of the boards even if we design large boards. In other words, a single-CPU host is significantly more expensive than a board with, say, 32 chips.

This assumption is however no longer true. Now we can buy an Intel P4 box with the theoretical peak floating point speed of 4 Gflops for around 1000 USD, and no matter how much money we pay, we cannot get any single-processor system which is faster than this box by a factor of two. Thus, if the connection between hosts is fast enough, we can use a rather small board with 4-8 chips attached to host. It may be possible to design the board as standard PCI(-X) card, which leaves the work of designing the enclosure and power supply system unnecessary. Also, small boards without complex interconnection network makes the development process simpler.

To summarize, the overall design of GRAPE-7 will be rather different from that of GRAPE-6. The basic building block will be a set of an inexpensive frontend and a small board with around 8 chips, with the peak speed of 5 Tflops. A petaflops system is constructed as 256-node Beowulf-type cluster, connected

by Gigabit or faster network. Faster system can be constructed by adding more nodes. The development cost for a petaflops system will be around 3-4 M USD, with additional cost of around 1.5 M USD per petaflops for larger machine.

A potential problem with this architecture is that the number of particles for which the forces must be calculated in parallel would be too large for petaflops or larger configuration, since it would reach 3,000 for 256-node system. This, however, is probably not a serious limitation. It would be unlikely to use the whole machine for just one simulation for very long time. Therefore, for most of production runs, the machine will be divided into 4 or 16 small pieces, for which the required degree of parallelism is factor 2 or 4 smaller.

5. Summary

I overviewed the present state of GRAPE-6, highlighting the performance of parallel-host configurations on both the individual-timestep direct summation code and treecode. For both application, parallel-host system proved to be quite effective.

We also present a conceptual study for the design of GRAPE-7, a multi-petaflops next-generation GRAPE. It will further extend the concept of parallel-host architecture, by putting most of the communication network from GRAPE side to host side. This change, in theory, allows us a fast development of the machine and more flexible use, with additional possibility of improving the performance of overall system by upgrading the host computers only.

References

Aarseth, S. J. 1963, MNRAS, 126 223

Athanassoula E., Bosma A., Lambert J. C., and Makino J. 1998, MNRAS, 293 369

Barnes J. and Hut P. 1986, Nature, 324 446

Hernquist L., Hut P., and Makino J. 1993, ApJ, 411 L53

Kawai A. and Makino J. 2001, ApJ,550 L143

Makino J. 1991, PASJ, 43 621

Makino J. 1999, Journal of Computational Physics, 151 910

Makino J., Ito T., and Ebisuzaki T. 1990, PASJ, 42 717

Makino J. and Taiji M. 1998, Scientific Simulations with Special-Purpose Computers — The GRAPE Systems, John Wiley and Sons, Chichester.

Makino J., Taiji M., Ebisuzaki T., and Sugimoto D. 1997, ApJ, 480 432

Sugimoto D., Chikada Y., Makino J., Ito T., Ebisuzaki T., and Umemura M. 1990, Nature, 345, 33