Design tradeoff in GRAPE systems

Junichiro Makino
Department of Information Science and Graphics,
College of Arts and Sciences, University of Tokyo,
Tokyo 153, Japan

Abstract

In this paper, We discuss several aspects in design consideration of massively-parallel GRAPE systems. In order to give practical argument, we first discuss the architectures of GRAPE-3 and GRAPE-4. For GRAPE-3, we adopted a very simple architecture. As a result, the time to develop the hardware and the software was rather short. The performance of the GRAPE-3 was limited by that of the host computer. For GRAPE-4, we tried to design a system much faster than GRAPE-3. Thus, in order to relax the limitation of the performance, the architecture of GRAPE-4 became more complex than that of GRAPE-3. We discuss the effects of crucial design decisions in detail. We also discuss the possible architecture of the follow-up machine for the GRAPE-4. The next-generation machine (GRAPE:TNG) will have the peak performance of 30-100 Tflops, if the project will start 2-3 years from now and if the budget for it will be similar to that for the present GRAPE-4. We discuss the requirement for the host computer, the design of the interface to the host, and the connection topology in the GRAPE system.

1 Introduction.

In this paper, we discuss several aspects in the design consideration of massively-parallel GRAPE system. The increase in the speed in GRAPE system in the last four years has been quite dramatic. Our first machine was the GRAPE-1 with 120 Mflops (Ito, et al., 1989), and we now have a 6-boards GRAPE-3AF with 36 Gflops (Okumura, et al. 1993), achieving a speed improvement of a factor of 300. Early in next year, we will have a one-teraflops machine, the GRAPE-4 (Makino, et al., 1993), realizing the increase of the speed of another factor of 30. In other words, we are going to achieve the improvement in the performance of a factor of 10,000, in just five years.

This tremendous increase in the speed was achieved mostly by exploiting higher degree of parallelism. The GRAPE-1 system had one pipeline to calculate interparticle force, running on 4 MHz clock. The GRAPE-3AF has 48 pipelines with 25 MHz clock, and GRAPE-4 will have 1920 pipelines with 30 MHz clock. Thus, of the increase of 10,000 in speed, only a factor of eight is due to the increase in the clock speed and another factor of 1000 is due to the increased parallelism.

This increase in the parallelism resulted in a more complex overall architecture. This increased complexity causes a longer design time, which offsets the cost-performance ratio of massively-parallel systems. In particular, the GRAPE-4 system requires nearly two years to complete the

first prototype. This period is much longer than the time used for any other machine we have ever build.

A long development time is the most dangerous thing for a special-purpose computer. The reason is quite simple: The performance of general-purpose computer doubles every 1.5 years. Therefore, if we design a machine which offer the performance per cost 10 times better than a general purpose computer at that time and spend five years to develop the real system, when it is completed it would have the performance per cost similar to that of general-purpose computers at that time. In other words, it would not be particularly better than general-purpose computers.

When we take into account the advance in the device technology, the development time becomes even more important. The number of transistors available in a single chip quadruples in every 3 years. At the same period, the switching speed roughly doubles. Thus, the potential performance of a chip, defined as the product of the number of available transistors and the switching speed, increases by a factor of eight in every three years, or doubles every year. Thus, one more year for the design of the hardware/software means 50% loss of the total performance.

In this paper, we first describe the architecture of GRAPE-3A and GRAPE-4, and evaluate the decisions in the design of the architecture that has large impacts on the development time and the peak performance. Roughly speaking, GRAPE-3 was designed as simple as possible, even if the simpler design resulted in the degradation of the performance of the system for real problems. Thus, full-size machine would be not very useful for real problems. In contrast, GRAPE-4 is designed in such a way that it would offer a reasonable performance for real problems. As a result, both the hardware and the software became more complex than that for GRAPE-3, and it took much longer time to develop the system. We discuss how we can find an optimal point in the tradeoff between the design time and the peak performance, for the followup machine for GRAPE-4.

2 The GRAPE-3 System.

2.1 Architecture

Figure 1 shows the architecture of GRAPE-3, which is identical to that of GRAPE-3 except for the number of pipelines per board. GRAPE-3 has 24 pipelines per board, while for GRAPE-3AF we reduced the number of pipelines to 8, in order to reduce the wiring complexity. GRAPE-3 was built using the wire-wrapping, while GRAPE-3AF uses the printed-circuit board.

The architecture of GRAPE-3/3A is very simple. On one board, all pipelines share one common memory unit. This shared memory unit broadcast to all pipelines the positions and masses of particles that exert the gravitational force. Different pipelines calculate the force on different particles.

Each of GRAPE-3/3A board has the interface to VME bus. In the standard configuration, multiple GRAPE-3/3A boards are attached to a single VME bus. Several workstations has the VME bus as the extension bus. For machines which has the I/O bus different from the VME bus, commercial product to provide the VME interface is available.

From the point of view of the host computer, the multi-board GRAPE-3/3A system in one VME backplane still looks essentially same as one-board system, since the memory units of all boards share one same address space, using the broadcast function of VME (revision D) standard. Thus, when the host computer sends the position or mass of one particle to be stored to the memory, these data are stored to the the memory units of all board. Again, the pipelines on different boards calculate forces on different particles. If one system has the total of 48 pipelines, it calculates the forces on 48 different particles from the same set of particles.

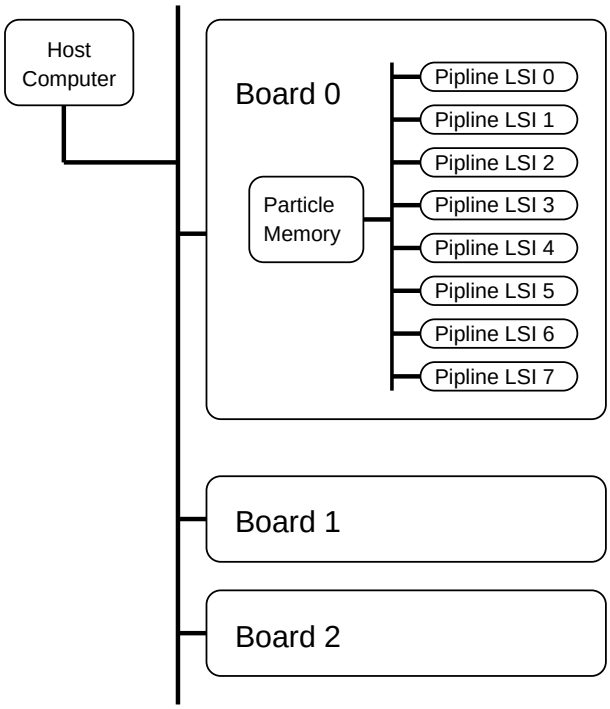

Figure 1: The GRAPE-3 system

2.2 Scalability

Here, we use the word "scalability" in the following meaning: How large a system we can construct. The scalability of a machine can be limited by many reasons. In the case of the GRAPE-3, one obvious limitation is the number of the processor boards that can physically fit to the VME chassis. The VME specification limits the total number of boards in one backplane to 20.

A more important factor that limits the scalability is the performance. The performance we can get is not a linear function of the number of boards, because of Amdahl's law. When we increase the number of boards, the speed of the host computer remains unchanged. As a result, the incremental performance we can get by adding more board becomes smaller and smaller as we increase the number of boards. For example, when we use a simple direct summation algorithm, the calculation time per one timestep is expressed as

$$T_{\rm dir} = \max[T_{\rm comm1}, T_{\rm force}] + T_{\rm comm2} + T_{\rm misc}, \tag{1}$$

where $T_{\rm comm1}$ is the time for the data transfer which occurs concurrently with the force calculation on GRAPE, $T_{\rm force}$ is the time to calculate forces on GRAPE, $T_{\rm comm2}$ is the time for data transfer that is not concurrent with the force calculation, and $T_{\rm misc}$ is the time for computation on the host computers. These terms are expressed as

$$T_{\rm comm1} = 11\frac{b-1}{b}Nt_{\rm comm}, \tag{2}$$

$$T_{\rm comm2} = 4Nt_{\rm comm} + \frac{11}{b}Nt_{\rm comm}, \tag{3}$$

$$T_{\rm force} = \frac{N^2 t_{\rm pipe}}{bp}, \tag{4}$$

$$T_{\rm misc} = N t_{\rm misc}. \tag{5}$$

where b is the number of boards, N is the number of particles, $t_{\rm comm}$ is the time to transfer one word between the host and GRAPE, $t_{\rm pipe}$ is the clock period of the force calculation pipeline, p is the number of pipelines per board, and $t_{\rm misc}$ is the time needed to handle one particle on the host computer. For a workstation with modest speed in the present day standard, $t_{\rm comm}$ is 2×10^{-6} second, and $t_{\rm misc}$ is 2×10^{-5} second. For a GRAPE-3A with 25 MHz clock, $t_{\rm pipe}$ is 4×10^{-8} second. If we increase the number of boards, we reduce $T_{\rm force}$ and $T_{\rm comm2}$, while keeping $T_{\rm comm1} + T_{\rm comm2}$ constant. Therefore, if $T_{\rm comm1}$ is larger than $T_{\rm force}$ for a given number of boards, additional boards have no effect on the performance. The number of boards for which $T_{\rm comm1} = T_{\rm force}$ is

$$b_{\rm eq} = 1 + \frac{N t_{\rm pipe}}{11 p t_{\rm comm}} \tag{6}$$

$$\simeq 1 + \frac{N}{4400}. \tag{7}$$

Here, we used $p = 8$. This relation indicate that it would be meaningless to use more than 5 boards if the total number of particles is less than 20,000. In practice, this degradation of the performance is not very severe, since it is unlikely to have more than 10 boards and we routinely use 64k particles or more.

However, if we are going to build a much larger system, this degradation in the efficiency would severely limit the maximum size of the machine. For example, a one-teraflops machine would need 170 boards. Even if there is any way to hook 170 boards to the present host computer, we need 7.5×10^5 particles to get a reasonable performance. To make the matter worse, this number of particles is actually not enough, since the present GRAPE-3A board can handle only up to 128k particles. Larger number of particles can be handled, but with an increased cost of communication (Okumura, et al. 1993).

If we use the modified tree algorithm, the amount of computation on the host computer increases significantly while the amount of calculation on GRAPE decreases. Thus, with the present host computer and the performance of the communication, the performance of the tree algorithm is limited solely by the performance of the host computer.

The time per one timestep for the modified tree algorithm (Makino 1991a) on GRAPE-3A system is expressed as

$$T_{\rm tree} = T_{\rm const} + T_{\rm list} + T_{\rm comm} + T_{\rm force}, \tag{8}$$

where $T_{\rm const}$ is the time to construct the tree structure, $T_{\rm list}$ is the time to create interaction lists. In the modified tree algorithm, we create the list of particles that exert the force on a group of particles, and let GRAPE calculate the force from the particles in this interaction list on the group of particles in parallel. The third term, $T_{\rm comm}$, is the time to transfer data between the host and GRAPE, $T_{\rm force}$ is the time that GRAPE calculates. They are expressed as

$$T_{\rm const} = a_1 N \log_8 N, \tag{9}$$

$$T_{\rm list} = a_2 \frac{N_l}{N_{\rm g}} N, \tag{10}$$

$$T_{\rm comm} = (4\frac{N_{\rm l}}{N_{\rm g}} + 11) N t_{\rm comm}, \tag{11}$$

$$T_{\rm force} = \frac{N_{\rm l}}{bp} N t_{\rm pipe}, \tag{12}$$

where a_1 and a_2 are time coefficients which depend on the speed of the host computer, N is the number of particles in the system, $N_{\rm l}$ is the average length of the interaction list, $N_{\rm g}$ is the average number of the particles that can share one interaction list, $t_{\rm comm}$ is the time to transfer one word between the host and GRAPE-3, p is the total number of pipelines and $t_{\rm pipe}$ is the cycle time of the pipeline. The length of the interaction list, $N_{\rm l}$, depends on $N_{\rm g}$, the opening angle θ, and the distribution of particles in the system. The following gives a rough approximation of $N_{\rm l}$, neglecting the dependence on θ and the particle distribution.

$$N_{\rm l} = N_{\rm g} + 14{N_{\rm g}}^{2/3} + 84{N_{\rm g}}^{1/3}. \tag{13}$$

From equations (8) to (13), the optimal value of $N_{\rm g}$ for which the total computing time becomes the minimum is expressed as

$$N_{\rm g,opt} \simeq \left(\frac{14c_1}{3c_2}\right)^{3/4}, \tag{14}$$

where c_1 and c_2 are defined as

$$c_1 = 4t_{\rm comm} + a_2, \tag{15}$$

$$c_2 = \frac{t_{\rm pipe}}{bp}, \tag{16}$$

in the limit of $c_1 >> c_2$. The total computing time for this optimal $N_{\rm g}$ is expressed as

$$T = T_{\rm const} + \frac{17c_1N}{3}\left[1 + 14\left(\frac{3c_2}{14c_1}\right)^{1/4}\right]. \tag{17}$$

From equation (17), we can see that in the limit of $c_2 << c_1$, the computing time approaches to an asymptotic value of $T_{\rm const} + \frac{17c_1N}{3}$. If we neglect $T_{\rm const}$, half of this asymptotic speed is achieved for $c_1/c_2 = 8 \times 10^3$. The value of c_1/c_2 is around 5×10^3, for the combination of a SUN-compatible computer (PWS-1500 with 36MHz SuperSPARC CPU) and one GRAPE-3A board. Thus, for this host computer additional GRAPE-3A boards provide very small performance increase.

2.3 Communication Bottleneck - Tradeoff between the portability and the performance

Both in the case of the direct summation and the tree algorithm, the performance of the host computer would limit the total performance of the system if we build a larger machine without significantly enhancing the performance of the host computer. The computational speed of the host computer is, however, relatively easy to improve, simply because the host computers we are now using are not very fast machines in today's standard. High-end desktop workstations from companies such as DEC, IBM, HP and SGI offers 3-5 times faster speed than that of our PWS-1500.

The speed of the data transfer between GRAPE and the host computer is more difficult to improve. If we adhere to use the VME standard and use the bus-adapters commercially available, the data transfer speed would be limited to several MB/s, for any of high-speed workstations.

Total performance of the GRAPE-3A system is limited essentially by the communication bottleneck, much in the same way as the classical von-Neumann bottleneck.

There are several ways to improve the speed of the communication. First of all, we have not yet fully exploited what we can get from the VME bus. The speed we currently get from the VME bus, a few MB/s, is much lower than the theoretical peak transfer speed of the VME bus (around 60 MB/s) or what is available as the data transfer speed for DMA transfer on VME bus (typically 20 MB/s). This large discrepancy is simply due to the fact that GRAPE-3A is designed as a VME slave board without the capability of becoming a VME master.

Most of the I/O bus of present workstations are designed in such a way that it offers the good transfer rate only when it is used in the DMA (direct memory access) mode, in which the peripheral devices such as the disk drive unit accesses the main memory. When the CPU access the address space for the peripheral (so called program I/O), the data transfer rate is much lower. This is simply because present workstations are designed in that way. Typical usage of the access of the I/O bus by CPU is to set the command registers of the peripherals, and the DMA is used to actually transfer data between the main memory and the peripheral. Therefore, the data transfer speed required for the program I/O is several orders of magnitude lower than that for the DMA access.

The obvious way to improve the data transfer is to use the DMA mode. For GRAPE-3, we did not adopt it because of difficulties both in the hardware and the software.

In the hardware level, the design of the board becomes quite a bit more complicated when we add the DMA capability. When we use program I/O, the board has essentially only two access modes: read and write. When the host executes a read access, the board receives the address, decode it and give the appropriate data. When the host executes a write access, the board receives the address and data, and store the received data to an appropriate place.

The DMA access is far more complicated. First, the host computer sends necessary parameters (start address, the length of the data block etc) by program I/O. Then the board initiate the DMA transfer. First, the board request the grant to use the bus, which is given by the bus arbiter (in most cases, the host computer plays the role of the arbiter). After the board gets the bus grant, it performs the actual data transfer in which the board controls the address lines and other control signals of the VME bus. When the data transfer is finished, the board activates the interrupt to tell the host that the data transfer has ended. To design the hardware which takes care of all of these things takes lots of time.

In the level of the software, again, the DMA transfer is quite a bit more complicated. The program I/O is quite easy to use on most of UNIX-based workstations, since most of them offer the memory-mapping capability. With memory mapping, the operating system maps the physical address space of the I/O bus into the virtual address space of the user program. As a result, the user program can access the board with simple assignment statement. Thus, all software for GRAPE runs in the user mode, and we do not have to modify the operating system. This also means the source program for GRAPE is portable to any UNIX-based workstations.

With the DMA access, things are not that simple. On the UNIX operating system, the DMA operation is usually handled by the operating system (or the device drivers of the operating system). A DMA read operation would proceeds roughly like the following. First, the user program setup the buffer to receive the data and parameter structure to be passed to the operating system. Then, the user program performs the system call. After the system call, the control is transferred to the operating system. The operating system receives the parameter structure. Then, it usually allocates its own buffer space, and send the command to the peripheral device

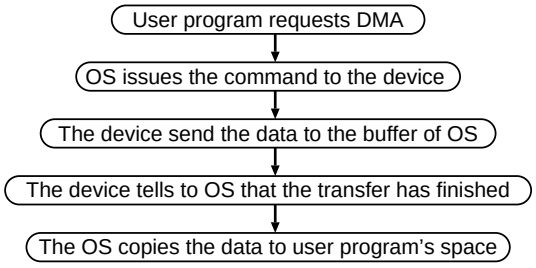

Figure 2: A DMA read operation

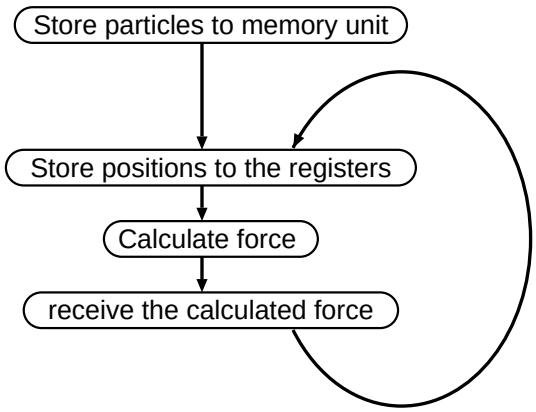

Figure 3: Calculation flow on GRAPE-3

so that the device will store the data to its buffer. After that, the process will sleep so that other process can continue while the present process is waiting for the completion of the DMA operation. When the device initiate the hardware interrupt to tell the completion of the DMA operation, the process wakes up and store the data in the buffer of the operating system's space to the buffer in the user space. Figure 2 shows the flow of the control for a DMA read operation.

The DMA operation is rather hard to use because of the following two reasons. The first one is related to the performance. Since the structure of the software involved to a single DMA operation is very complicated, the overhead time for one DMA operation is very large. Even with present fast RISC workstations, it is not unusual that one DMA operation take several hundreds microseconds. Let us consider one example in which one DMA operation requires minimum time of 200 μs and the transfer rate of DMA operation is 20 MB/s. In order to obtain half of the maximum transfer rate, we must transfer a block with the size of 4 KB.

Unfortunately, with present GRAPE-3 hardware it is not easy to make the block size large. Figure 3 indicates how the calculation proceeds on GRAPE-3. First the host computer stores the positions of eight particles (here, eight is the number of pipelines on one board). Then the board starts the calculation. After the calculation is finished, the host computer reads the calculated forces. The size of the data block for one transfer is of the order of 100 bytes. Thus, we cannot use a traditional device driver model for the data transfer with GRAPE.

The second problem is that the development of the softwares necessary for DMA operation is much more complicated than that for program I/O. This is essentially because we have to go into the details of the operating system itself. Moreover, the portability among different platforms

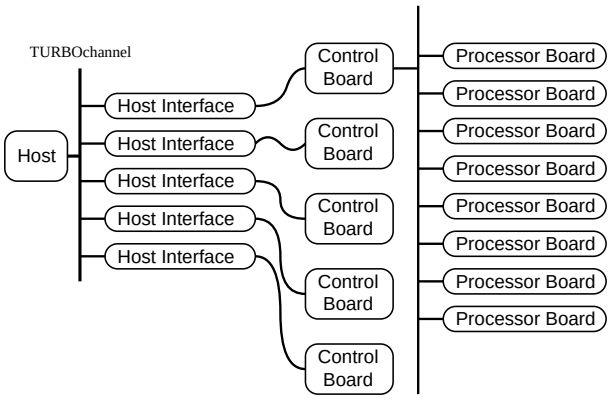

Figure 4: The GRAPE-4 system

would be almost completely lost. The UNIX operating system offers reasonable portability of source for user programs, by offering the standardized set of library functions. However, the source for the operating system is much less portable, simply because it is not standardized. We have to develop the software for each of the host computer, which is not a very pleasant thing.

We did realize these problems with the DMA data transfer, and that was the main reason why we adhered to the program I/O in GRAPE-3. As a result, GRAPE-3 is relatively each to connect to different systems. We so far used seven different host computers, with minimal change in the interface program. Moreover, the hardware design of GRAPE-3 was very simple, and it took relatively short time to develop the hardware and the software. On the other hand, its performance is limited by the speed of the communication.

3 The GRAPE-4 system.

3.1 The Architecture

The GRAPE-4 is a massively parallel GRAPE system designed to run 1920 pipelines in parallel. As such, the system architecture of GRAPE-4 has become more complicated than that of GRAPE-3. Figure 2 shows the basic structure. The interface to the host computer is the TURBOChannel bus, an I/O bus used in DEC's desktop workstations. The host interface board (HIB) is connected to the TURBOChannel. The HIB is then connected to the control board (CB) , which controls eight processor boards (PB). The structure of one processor board is similar to that of GRAPE-3, except that the number of pipelines per board is very large (48 versus 8 for GRAPE-3A) and the interface to the control board is a simple synchronous bus with very wide data width (96 bits).

The main function of the control board is to sum up the partial forces calculated on the processor boards. In GRAPE-4, all boards calculate the force on same set of particles, from different set of particles. This is different from GRAPE-3A, in which different boards calculate the force on different sets of particles, from the same set of particles. This difference comes mainly from the fact that GRAPE-4 is designed for the individual (or hierarchical) timestep algorithm (McMillan 1986, Makino 1991b). With the hierarchical timestep algorithm, the number of particles for which we can calculate the forces in parallel is rather small. Thus, if the hardware

is designed so that it calculates the forces on too many particles in parallel, large fraction of hardware might become idle.

In GRAPE-4, the force on a particle is calculated as 40 partial sums, on 40 boards. Therefore, it is necessary to sum up all these 40 forces. We could design a system so that the host computer performs this job, but it would require extremely fast data transfer between the host and GRAPE-4 boards. Our design goal for GRAPE-4 system is that it would give more than 50% of the peak speed for 64k particles. The designed clock speed of the processor chip is 30 MHz, and it calculates one interaction in every three clock cycles (Here, we do not take into account the existence of the virtual multiple pipeline, since it is irrelevant to the discussion here. Since 64k particles are distributed to 40 boards, each board has 1.6k particles, and finishes the force calculation for 48 particles in $1600 \times 3/(3 \times 10^7) = 1.6 \times 10^{-4}$sec. The amount of data to be transferred between the host and GRAPE-4 boards is of the order of 100 bytes per particle per boards. Therefore the total amount of data is of the order of 100 kbytes. In order to attain 50% of the peak speed, this amount of data must be transferred in 1.6×10^{-4}sec. In other words, the transfer speed necessary is 600 MB/s.

This speed of 600 MB/s is well above the theoretical peak transfer rate of I/O buses, or even the memory buses, of most desktop workstations. Thus, the only possibility is to reduce the amount of data by computing the partial sum in hardware.

The data transfer between the HIB and the host uses the DMA transfer. In order to reduce the overhead of the operating system, we adopted the following strategy. The command registers of the HIB is mapped into the address space of the user program, in the same way as in the GRAPE-3A board. The actual data is directly transferred between HIB and the buffer in the address space of the user program. The user program, therefore, need to know the physical address of its buffer. This is done by a special system call.

3.2 Scalability

The GRAPE-4 hardware is designed without taking into account the scalability to a larger system. As a result, it is physically difficult to build a larger machine and it will not offer much performance gain, as far as the number of particle is not excessively large.

3.3 Tradeoff in the performance and the design complexity

As we described in the introduction, the development time of the hardware is the most important factor for a special-purpose computer. The GRAPE-3A system has been very successful in this point, since the full-size machine became operational roughly one year after the start of the project. The GRAPE-4 so far has taken too much time. The full-size machine is scheduled to be ready by March 1995, two and half years from the start of the project (summer 1992).

There are many reasons for this rather long development time. The most important reason is simply that GRAPE-4 is significantly more complicated than GRAPE-3. For example, on one processor board we mount 48 pipeline chips. In order to achieve this high density, we have designed an MCM (multi-chip module) carrier, which houses 8 pipeline chips on single package. In order to reduce the communication with the host, we have to design the host interface board and the controller board, which did not exist for GRAPE-3.

These design choices were inevitable to achieve the speed of 1 Tflops with the technology available at the year 1992. In addition, we took into account the advances in the speed of the

host computer to achieve 1 Tflops. It would be simply impossible to achieve 1 Tflops for, say, $N = 65536$, with the host computer available at 1992. On the other hand, if we could build a 250 Gflops machine by 1993 or 500 Gflops machine by 1994, it would be essentially as useful as 1 Tflops at 1995, according to our discussion in the introduction. Therefore, it would be worthwhile to evaluate the possibility of such solutions.

In order to cut down the design time, we have to simplify the hardware. This basically means that we should not have designed the three boards (HIB, CB, and PB). If we simply use the same architecture as that of GRAPE-3, we do not need either HIB or CB. The obvious problem is the limitation in the data transfer speed. If we use the VME bus with program I/O, the performance of the data transfer would be similar as that of GRAPE-3, and to attain reasonable performance on a 250 Gflops machine, the number of particles would be 600k. This number is already too large. Moreover, the number of particles for which the forces are calculated in parallel would be too large. Total number of virtual pipelines would be 1000. If we are to reduce this number to 100, which is the number adopted for GRAPE-4, the host computer have to play the role of CB, increasing the number of particles to 2×10^6. If each of the processor board has the DMA capability, it is possible to increase the data transfer speed to 20 MB/s or more. With this system, half of the peak speed is attained for 200k particles. This level of speed (20MB/s) can be achieved with the bus adapters to EISA, SBUS or MicroChannel. So we could use a wide variety of the host computers.

It is true that 200k particles is still rather large. To further improve the speed, we could design the interface to SBUS to obtain a much faster speed. It would take several more months to develop such a system, but would still be significantly simpler than the present plan for GRAPE-4.

To be practical, our project proposal explicitly said that our planned goal was 1 Tflops. Therefore, even though a 250 Gflops machine would be as useful as 1 Tflops machine, we must develop a Teraflops machine. However, if we evaluate the scientific usage of the hardware, it might be better to design a simpler (and slower) hardware in a shorter time.

4 The next generation

In this section, we describe how the followup machine for GRAPE-4 would look like, assuming that the project will start by the fiscal year 1997. In the following, we call this machine as GRAPE:TNG (the next generation).

4.1 Available Technology

In 1997, the design rule of the CMOS chip would reach 0.3 μm. Finer process is possible, but here we assume 0.3 μm. This means 10 times more transistors than what was available for the present HARP chip. The clock cycle would be three times faster, resulting in the clock frequency of 100 MHz.

If we design the next hardware with the minimal change in the overall architecture, we will put 3-4 pipelines, each of which can calculate one interaction per cycle (not in three cycles as was the case with the present HARP chip). The performance of one chip would be 15-20 Gflops. With 2,000 chips, the total performance of the machine will be 30-40 Tflops.

4.2 Architecture

It would be unpractical to design the processor board so that it operates on the 100 MHz system clock. Therefore, the number of the virtual pipelines per physical pipeline will be larger than the present value. To drive the system with 25 MHz clock would not be too hard. Therefore the ratio of 4 would be reasonable, resulting in 12-16 virtual pipelines per chip.

This large number of virtual pipelines means that it would be difficult to use the present GRAPE-4 architecture, since the number of forces calculated in parallel would be too large. For a 30 Teraflops machine, we would like to try 200-300k particles for simulations of globular clusters. Therefore the number of forces calculated in parallel should not be much larger than that in the present GRAPE-4.

One alternative is to integrate the particle memory into the chip. Total number of available transistors would be around 10^7. Therefore it should be practical to integrate 10^6 or more bits of memory (1 bit of memory consumes 4 transistors) on chip. With the 4th order Hermite scheme (Makino and Aarseth 1992), one particle requires about 1k bits of memory. Therefore, we can integrate the memory for 1000 particles on a chip. With 2,000 chips, the total number of particles that can be stored is 2×10^6, which is much more than what is necessary. In this case, it is not necessary to use the virtual multiple pipeline, since the on-chip memory is fast enough.

Since the memory is local to each chip, it is natural that the chips on one board calculate the force on the same particle from different set of particles. In order to reduce the communication, it is necessary to sum the forces calculated on different chips on the board. In GRAPE-4, we used a general-purpose floating-point chip (TI 74ACT8847) to calculate the summation on the control board. In 1997, an FPGA (field programmable gate array) would be large enough to implement a 64-bit floating point unit. The other possibility is to integrate the additional adder into the pipeline chip, since it requires very small number of transistors.

With this architecture, the design of the processor board would become extremely simple, since it does not require off-chip memories and additional circuits to interface the memory and the processor chip.

If we use similar numbers of chips and boards as in GRAPE-4, we will have 40 boards, and the host computer potentially have to sum up this 40 forces. However, if we are dealing with, say, 64k particles, one or two boards is enough to keep the particles. For example, if one board has 64 processor chips, each having the memory for 1000 particles, 64k particles fit into one board. Therefore, we can use different boards to calculate the forces on different particles. It might be possible to have the top-level architecture similar to that of GRAPE-3. Of course, this simple structure is possible only because we put the function of the control board of GRAPE-4 into each processor board (or processor chips).

We believe this architecture solves many of the problems in the hardware design of GRAPE-4. The top-level architecture of GRAPE:TNG is similar to that of GRAPE-3, since each processor boards will be connected directly to the host I/O bus. The design of the processor chip would be more complicated, but the design will not be much difficult. In fact, each chip is essentially a copy of the processor board of GRAPE-4.

4.3 Host computer

The requirement for the host computer would not be excessive. Both the floating point speed and the data transfer speed need to be faster than the requirement for the host of GRAPE-4 by a factor of 10. The data transfer rate would be somewhat lower, since the host computer will not

receive the partial forces for small N. The necessary speed of the floating point operation will be around 500 Mflops, which will be rather easy to obtain. The necessary speed of the data transfer would be around 500 MB/s. This speed might be difficult to obtain, since the data transfer rate of I/O bus does not increase as fast as the processing speed of CPU. At present, however, the fastest peripheral bus available on desktop workstations provides 267 MB/s. So it is not unlikely that I/O bus with the speed exceeding 500 MB/s will be around by the year 1997.

4.4 Packaging

The I/O bus with the transfer rate of 500 MB/s will not be physically large, unless it uses rather exotic technology or have very wide data width. On the other hand, the total system with some 40 processor boards cannot be physically small. Thus, the top level structure of GRAPE:TNG has to be similar to that of GRAPE-4, which is organized to 5 clusters, each are connected to the I/O bus of the host. The main difference is that the boards corresponding the host interface board and the control board would be much simpler. They are essentially a bus repeaters, with no extra intelligence.

5 Summary

In this paper, we made a rather intuitive analysis of several design issues of GRAPE-type system, using GRAPE-3 and GRAPE-4 as examples. The design of GRAPE-3 is very simple. As a result, development of the hardware and software was very easy, and the machine became operational in less than one year from the start of the project. In contrast, the design of GRAPE-4 is more complex. According to our plan, the completion of the machine will take two and half years (So far, there was no delay from the planned schedule).

This difference in the complexity comes from the fact that we tried to push the performance to the limit with GRAPE-4. We used the proprietary I/O bus in order to realize very fast communication, we used the two-level tree structure to reduce the communication, we used the MCM package to obtain higher chip density. Each of these design choices caused several more months of the design time.

The reason that we need rather complex architecture is that we chose to design GRAPE-4 so that it would achieve a reasonable efficiency for problems of the practical size, when used as a single, big machine. In retrospect, there is not much reason to design the machine in that way. We have many problems to be solved. It would make more sense to have four 250-Gflops machines than to have one one-Teraflops machine. To obtain the measured performance of one Tflops, the only thing we would have to do is to hook all the four clusters to one host, and run a large simulation. For example, if we design the hardware so that the peak speed of 1 Tflops is achieved for 1 million particles, there is no need for the control board and the development of the hardware would be much simpler.

For the next-generation machine, the choice of the right architecture would be even more important, since the amount of the circuit that can be integrated into one chip will be much larger. In the previous section, we consider an architecture in which we put almost everything into one chip, to the point one chip will have all the functionality of the processor board and the control board of GRAPE-4. With this chip, the design of the processor board will be drastically simplified and the control board will not be necessary. How long it would take to design such a chip is difficult to estimate. However, we feel the design time of the chip will not be much different, since the additional logic is still rather simple.

References

Ito, T., Makino, J., Ebisuzaki, T., and Sugimoto, D., *Computer Physics Communications*, 1990, **60**, 187.

Makino, J., *Publ. Astron. Soc. Japan*, 1991a, **43**, 621.

Makino, J., *Publ. Astron. Soc. Japan*, 1991b, **43**, 859.

Makino, J. and Aarseth, S. J., *Publ. Astron. Soc. Japan*, 1992, **44**, 141.

Makino, J., Kokubo, E. and Taiji, M., *Publications of the Astronomical Society of Japan*, 1993, **45**, 349.

McMillan, S.L.W., in *The Use of Supercomputers in Stellar Dynamics*, ed. P. Hut and S.L.W. McMillan (Springer-Verlag, Berlin), 1986, p. 156.

Okumura, S. K., Makino, J., Ebisuzaki, T., Fukushige, T., Ito, T., and Sugimoto, D., *Publ. Astron. Soc. J.*, 1993, **45**, 329.