GRAPE Project

Junichiro Makino

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

Abstract

We overview our GRAPE (GRAvity PipE) project to develop special-purpose computers for astrophysical N-body simulations. The basic idea of GRAPE is to attach a custom-build computer dedicated to the calculation of gravitational interaction between particles to a general-purpose programmable computer. By this hybrid architecture, we can achieve both a wide range of applications and very high peak performance. Our newest machine, GRAPE-6, achieved the peak speed of 32 Tflops, and sustained performance of 11.55 Tflops, for the total budget of about 4 million USD.

We also discuss relative advantages of special-purpose and general-purpose computers and the future of high performance computing for science and technology.

Key words: Computational Science, Special-purpose computer, Numerical Algorithms
PACS:

1 Introduction

In this paper, we discuss a relatively unexploited approach to the computational science, namely to design and build the computer hardwares specialized and optimized for relatively narrow range of problems.

The development of the computers itself has been regarded as not really being a part of the computational science. It has been considered as what industries and/or computer scientists would do for us. This view was okay when a relatively simple computers were still very expensive and had to be shared by a number of researchers from wide variety of fields. For example, Cray-1, which was completed in 1976, was really a simple and small computer by today's standard. It has just 8 MB of SRAM memory in 16 banks, one multiplier unit and one adder unit. The estimated total gate count was just 300K. In comparison, today's single microprocessor integrates 40M transistors, which is equivalent to around 10 M gates. Thus, the world's largest supercomputer 25 years ago is less than one tenth of a today's desktop (or even notebook) PC in gate count.

With such a small gate count, it was unpractical to design a machine specialized to one application. Since it was difficult to make just one floating point calculation unit, the natural way to make a computer is to add control logics to that floating point unit so that it can perform operations specified by the software.

The available number of transistors, however, has been increasing exponentially in the last half century, in the rate of a factor of 100 per every decade. Thus, present-day microprocessors integrate the number of transistors sufficient to implement several hundreds of floating point units.

Figure 1 shows this situation rather clearly, for the case of single-chip microprocessors. When it was impossible to implement full-decode multiplier unit into a single chip, we had to use smaller multipliers which needed multiple cycles to perform one operation (before 1989). During this period, we could use the increase in the available number of transistors just to make bigger and bigger multipliers. Thus, in 1980s a factor of 100 improvement in the cycle count, which is consistent with the factor of 100 increase in the number of transistors, was achieved.

In 1990s, however, the decrease in the operation count has practically halted, as clearly seen from figure 1. In 1988, Intel shipped the first single-chip microprocessor which can perform more than one floating point operations per clock cycle, the i860. If the exponential increase in 1980s had continued in 1990s, we would see microprocessors with several hundreds of floating point units in 2001. Instead, processors like Intel Pentium 4 still perform only a few operations per cycle, even though the number of transistors has increased by a factor of 50 or so.

In other words, practically *all* the increase in the available number of transistors in 1990s was spent for things other than the arithmetic units. These "other" things include large L1 and sometimes L2 caches, multiple execution units for integer operations, datapath and control units for out-of-order execution of multiple instructions, additional logics to implement deeper pipelines and so on.

If we write the same figure for high-end supercomputers, we see essentially the same trend, but shifted by 15 years. With Cray-1 (1976), supercomputers had reached the regime where they can perform one floating-point operations per cycle. Until early 1990s, the evolution of the number of floating point units on

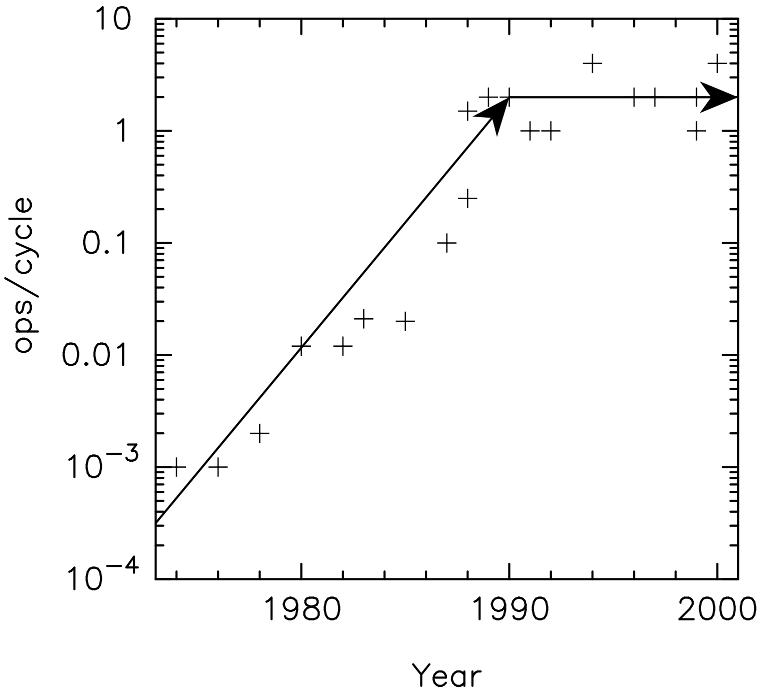

Fig. 1. Number of floating point operations performed per single machine cycle of representative high-performance microprocessors.

a single supercomputer had been very slow. Machines of early 1990s such as Hitachi S-3800, NEC SX-4 and Cray T-90 had only a few tens of arithmetic units, instead of a few thousands which was theoretically possible.

The most obvious way to use multiple floating point units in a single computer is to develop a parallel computer. Thus, in 1980s, parallel computers based on one-chip microprocessors and those based on supercomputers were built. Parallel computers based on microprocessors were mostly distributed-memory processors, which were essentially a cluster of simple computers connected by relatively slow and inexpensive network. Thus large machines with more than 1000 processors were possible. Parallel computers based on supercomputers were more complicated shared-memory machines. The shared memory architecture limited the number of processors to 32 or less.

Though microprocessor-based machines used much higher number of processors, they were still slow in absolute performance simply because each microprocessor was slow. Parallel supercomputers offered higher peak performance.

However, in 1980s, the performance of microprocessors had improved drastically, because they could fully utilize the increase in the available number of transistors. Thus, by early 1990s, microprocessor-based machines started to offer better price-performance ratio than parallel supercomputers.

In 1993, Fujitsu announced VPP-500, the first parallel supercomputer with distributed memory architecture. With this architecture, available number of processors was boosted to more than 100, resulting in a very large improvement in the price performance of supercomputers.

However, this move was counteracted by the birth of "Beowulf-class" machines[1,2], which are really simple cluster of commodity PCs with Intel x86 CPUs connected by standard Ethernet. This architecture offers typically one order of magnitude better price-performance than other computer architecture, simply because the production cost is low due to mass production. "High-end" Intel x86 CPUs are made and sold in units of hundreds of millions per year, while supercomputers were sold in units of hundreds. Thus, the sheer number of chips reduces the production cost per chip, and yet huge development cost can be spent for Intel x86 chip since the total revenue would be tens of billion USDs.

Thus, at present the computer architecture which offer both the best price performance and best absolute throughput is a large cluster of PCs with microprocessors of high-volume production. However, as we have already seen at the beginning, this architecture, at present, does not make good use of the transistors available on one chip. To make matters worse, the fraction of transistors used for floating-point operations will decrease in future. In a few years, the architecture called on-chip multiprocessor will be common. This evolutionally path is again following the path of supercomputers, but with the delay of 20 years. Thus, we can "predict" the future of on-chip multiprocessor architecture in the near future rather accurately, by just looking at the evolution of supercomputers in 1980s: The evolution will be slow.

So our question here is: Is there any better way to design a computer for scientific simulation? We believe the answer is "yes", and in the rest of this paper we describe why we believe so.

In section 2, we discuss the potential advantages and drawbacks of designing and developing special-purpose systems. The potential gain is very large, since, at least in some cases, we might be able to use a fair fraction of available transistors to perform useful arithmetic operation. On the other hand, there are many practical difficulties which would offset the potential gain. In section 3, we discuss our GRAPE project[3,4] as an example of, well, reasonably successful projects to develop special-purpose computers. In section 4, we speculate on the future of the large-scale scientific computing.

2 Special-purpose computing

2.1 Astrophysical N-body problem

The idea of building special-purpose computers for specific numerical problems is certainly not new. In fact, the very first digital electronic computers (ABC and ENIAC) were designed to solve specific problems. However, the designers of early computers found that their machines could be used for a much wider range of problems, and thus the evolution of the programmable general-purpose computer started.

At the time when even the largest supercomputers did not have a fully parallel multiplier, it did not make sense to design a special-purpose computer. Any computer, either special- or general-purpose, consists of arithmetic units (mainly multipliers), control logic and memory. A special-purpose computer might have simpler control logic or smaller memory than general-purpose computer. However, if the cost of the arithmetic unit itself is a fair fraction of the total cost of the machine, we cannot gain much by trying to reduce the cost of the rest of the system.

This situation, however, has changed completely, as we've seen in the previous section. To give a concrete example, let us discuss the astrophysical N-body problem and our GRAPE (GRAvity PipE) hardware specialized for that problem. The master equation for the gravitational N-body problem is given by:

$$\frac{d^2\boldsymbol{x}_i}{dt^2} = -\sum_{j\neq i} Gm_j \frac{\boldsymbol{x}_i - \boldsymbol{x}_j}{|\boldsymbol{x}_i - \boldsymbol{x}_j|^3}, \tag{1}$$

where $\boldsymbol{x}_i$ and m_i are the position and mass of the particle with index i and G is the gravitational constant. The summation is taken over all particles (typically stars) in the system under study.

The number of particles N varies widely depending on what kind of systems are studied. A small star cluster like Hyades might consist of several thousand stars. Globular clusters typically contain one million stars, and galaxies hundreds of billions of stars. In the case of galaxies, it is impossible to model them on a star-by-star basis, and we model the distribution of stars in the six-dimensional phase space with much smaller number of particles.

If N is larger than several tens, the calculation of the right-hand side of equation (1) dominates the total calculation cost. If N is very large, we could use sophisticated algorithms such as the Barnes-Hut treecode[5], FMM[6], or the particle-mesh or particle-particle particle-mesh method[7]. However, with treecode or FMM, the direct evaluation of gravitational interaction between near-neighbor particles still dominates the total cost. Even in the case of the particle mesh method, if we want high accuracy, near-neighbor interaction must be calculated directly.

The calculation itself is rather simple. All we have to do is to calculate and accumulate the gravitational force between two particles. Also, there is very large degree of potential parallelism, since in principle we can evaluate all N^2 interactions in parallel. Thus, N-body simulations are quite well suited for execution on massively parallel processors or PC clusters.

As discussed earlier, however, our aim here is to achieve a better use of transistors than what is realized with present microprocessors. Since N-body simulations have a large degree of parallelism, we first concentrate on how to make use of large number of transistors available in one chip. We return to the problem of parallelization later.

The problem with present microprocessors is that very small fraction of the available transistors is used to implement floating point arithmetic operations. The number of floating point units in a single one-chip microprocessor has been almost constant for the last 10 years, though the number of transistors has increased by almost two orders of magnitude.

Why the number of floating point units has been just a few, even though there are enough transistors to integrate hundreds of them? There are essentially two reasons. The first reason is that we do not know how to use such a large number of arithmetic units. To be precise, it is of course possible to use multiple arithmetic unit in parallel for applications with high degree of parallelism such as N-body simulations. Most of scientific applications do have high degree of parallelism, and would run with high efficiency on processors with large number of arithmetic units. However, at present microprocessors are designed to achieve the following two goals. The first is to run the most widely used applications as fast as possible. That is, to run Microsoft Windows operating system and Microsoft Office application package. The other is to run standard benchmarks such as SPEC CPU2000 suites as fast as possible. Of course, these are most important applications and benchmarks from the viewpoint of commercial success. However, it goes without saying that the performance of a microprocessor for scientific applications has rather little to do with its performance in processing MS-Word documents. Clearly, multiple floating point units would not help much improving the performance of MS-Word.

The second reason is that it is difficult to provide sufficient memory bandwidth to keep processors busy. It is not impossible to integrate 100 floating point arithmetic unit on a chip. However, if they operates independently, or even if

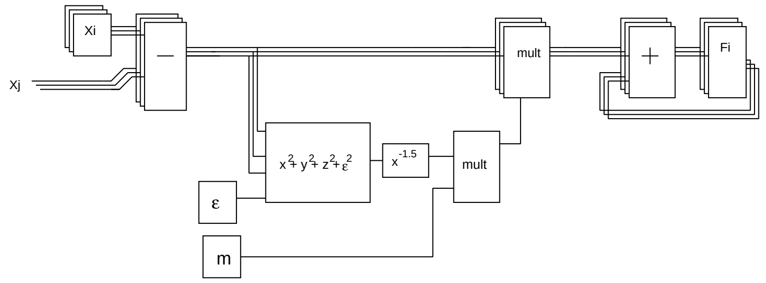

Fig. 2. The force calculation pipeline

they operate in the SIMD fashion, each unit need to get two words of data from memory and to store one word of data, to perform one floating-point operation. If these 100 arithmetic units operate on 500 MHz clock, which is rather slow by today's standard, the necessary memory bandwidth would be 150 Gwords/s or 1.2 Tbytes/s, which is about 500 times higher than the speed of the off-chip transfer bandwidth of today's microprocessors. A 2GHz Intel Pentium 4 processor has the theoretical peak transfer bandwidth of just 3.2 GB/s.

Of course, the necessary bandwidth is smaller for operations like summation. The difference, however, is small compared to the factor of 500 discrepancy shown above. This huge discrepancy means it is not easy to keep just one processor busy. With present clock speed of around 2GHz, one processor needs the bandwidth of 48 GB/s, or 15 times more than what is actually provided. It is quite understandable that architects of microprocessors are not too interested in adding more floating point units. External memory cannot feed a single unit. Unless there is a way to vastly increase the bandwidth, little or no gain is achieved by increasing the number of floating point units, even for scientific applications. It makes a lot of sense that they used a large fraction of the transistors on chip to cache memory, which can somewhat reduce the necessary off-chip memory bandwidth.

To summarize, though the current chip design with just one or two arithmetic units looks like a waste of the transistors, processor architects have good reasons to choose such designs.

2.2 Special-purpose computer

If we are to design a computer specialized for N-body simulation, or rather, the calculation of the gravitational interaction between particles, we can work around the limitations discussed above.

Figure 2 shows the basic "processor" for our N-body computer. Calculation of the gravitational force from one particle to another requires some 40 floating point operations, depending on how you count division and square-root operations. If we assign 10 operations to each of them, we end up with about 40 operations per interaction. Instead of using one arithmetic unit to perform these 40 operations, we connect 40 floating point arithmetic units in the form of a pipeline, which calculates and accumulate the force from one particle at each clock cycle.

With this approach, we can work around the two limitations discussed above. The first one was the efficiency of the system on commercial applications. This we simply ignore. Clearly not everybody who buys a PC want a fast computer for astrophysical N-body simulation.

The second limitation is the memory bandwidth. With the simple pipeline of figure 2, 40 floating point operations are performed for the input of 4 words. Thus, there is the saving of a factor of 30, compared to a single arithmetic unit or multiple units working in an SIMD fashion.

When we integrate multiple pipelines, we can let these pipelines to calculate the forces on different particles from the same particle. Thus, we can integrate multiple pipelines into a chip without increasing the necessary memory bandwidth.

It is also possible to reduce the required bandwidth for one pipeline, by let a single pipeline calculate forces on multiple particles. This can be done by adding the register files for the input particles and calculated forces, and by switching them at each clock cycle. We call this the Virtual Multiple Pipeline (VMP) architecture[8]. The idea of VMP is quite similar to what is now called as multithredding, but the difference is that VMP reduces the required memory bandwidth, while usual multithredding only relaxes the requirement for the memory latency.

Note that many of the above ideas can be applied to the calculation of particle-particle interaction on general-purpose programmable computer. Thus, with some careful programming (sometimes in assembly languages), it is possible to achieve the performance close to the theoretical peak performance of the processor. Here, the simple fact that there are only a few arithmetic units on a chip limits the ultimate performance.

Our GRAPE (GRAvity PipE) project is based on this idea of integrating multiple (both physical and virtual) pipelines specialized for gravitational force calculation into one chip (or one board, when the number of available transistors was not as large as what is now). To give an idea, the GRAPE-6 system which was completed in July 2001 integrates 1024 pipeline chips, each with 6 pipeline processors which calculate the gravitational force and its first time derivative. As a result, single GRAPE-6 processor chip integrate about 350

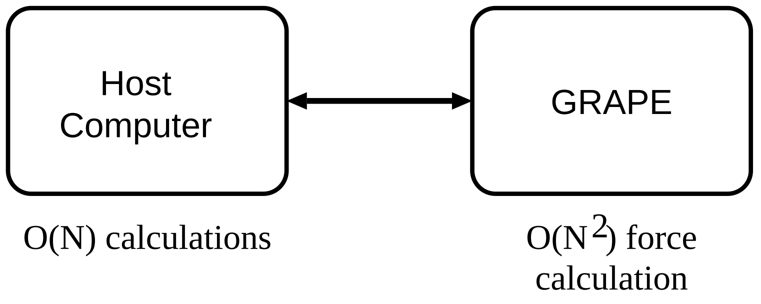

Fig. 3. The basic architecture of GRAPE

floating point arithmetic units.

The clock speed of GRAPE-6 chips is only 100 MHz, though it uses reasonably advanced 0.25μm technology. The reason for this rather low clock speed is simply that we did not have sufficient resource to fine tune the design. Even so, with 350 arithmetic units a single chip offers peak speed exceeding 30 Gflops, which is still more than a factor of 10 better than that of fastest microprocessors. In addition, because of this rather low clock speed the power consumption and therefore heat dissipation are small (around 15 W). This low power consumption is quite important in reducing the overall cost of the system and the running cost (electricity is expensive in Japan).

Since the GRAPE hardware performs only the force calculation, all other operations, such as the time integration of the orbits of stars and analysis of the calculated results must be done on somewhere else. Therefore we connect the GRAPE hardware and general-purpose computer by some communication link. This is the basic structure of our GRAPE systems, shown in figure 3.

3 GRAPE Project

We started GRAPE Project in 1988. The first machine we completed, the GRAPE-1 [9] was a single-board unit on which around 100 IC and LSI chips were mounted and wire-wrapped. We used commercially available IC and LSI chips to implement force calculation pipeline. This choice was natural consequence of the fact that we lacked both money and experience to design custom LSI chips. In fact, none of the original design and development team of GRAPE-1 had the knowledge of electronic circuit more than what was learned in basic undergraduate course for physics students.

GRAPE-2 is similar to GRAPE-1 since it was also based on commercial LSI chips. The difference is in the numerical accuracy. For GRAPE-1, we used an unusually short word format, to make the hardware as simple as possible. The input coordinates are expressed in 15 bit fixed point format. After subtraction, the result is converted to 8 bit logarithmic format, in which we use just 3 bits for "fractional" part. This format is used until we obtain $1/r^3$. The final accumulation was done in 48 bit fixed point, to avoid overflow and underflow. The advantage of the short format like 8-bit logarithmic format is that we could use ROM chips to implement complex functions that require two inputs. Any function of two 8-bit words can be implemented by one ROM chip with 16-bit address input. Thus, all operations other than the initial subtraction of the coordinates and final accumulation of the force were implemented by ROM chips.

The drawback of GRAPE-1 was its limited accuracy, which was insufficient for fairly wide range of astrophysical simulations, though for many other applications accuracy of GRAPE-1 turned out to be sufficient. With GRAPE-2, we used standard IEEE-754 format (64 bit for initial subtraction and accumulation, and 32 bit for all other operations).

GRAPE-3 was our first machine with custom LSI chip. The number format was again the combination of the fixed point and logarithmic format similar to what were used in GRAPE-1, but implementation of the arithmetic operations were quite different since we could not integrate large tables to a custom LSI chip. Conversions between fixed and logarithmic formats were implemented by shifters and small lookup table, and addition in logarithmic format is implemented by two adders and one small lookup table. Chip design was done as a joint research project between University of Tokyo and Fuji Xerox Corp. The chip was fabricated using 1μm design rule by National Semiconductor. The number of transistors on chip was 110K. Single chip operates at 20MHz clock speed, offering the speed of about 0.8 Gflops. We designed a printed-circuit board with 8 chips, for the speed of 6.4 Gflops per board. Thus, GRAPE-3 is also our first trial to integrate multiple pipelines into a system.

Also, GRAPE-3 was the first GRAPE machine to be manufactured and sold by a commercial company. Nearly 100 copies of GRAPE-3 have been sold to more than 30 institutes (more than 20 outside Japan).

With GRAPE-4, we finally integrated a high-accuracy pipeline into one chip. Also, with this chip we added the additional pipeline to calculate the first time derivative of the force, so that we can implement high-order time integration algorithms in a simple and efficient way [10]. We could not fit a full single pipeline to a chip with the technology available at that time. So we designed a "1/3" version of the pipeline which processes x, y and z components of coordinates serially in three consecutive clock cycles. The chip was fabricated using 1μm design rule by LSI Logic. Total transistor count was about 400K.

For GRAPE-4, we got a rather large grant of about 2 million USD. So we were able to build a large system consisted of 1728 pipeline chips (36 PCB

Table 1
History of GRAPE project

GRAPE-1	(89/4 — 89/10)	120 Mflops, low accuracy
GRAPE-2	(89/8 — 90/5)	40 Mflops, high accuracy(32bit/64bit)
GRAPE-1A	(90/4 — 90/10)	240 Mflops, low accuracy
GRAPE-3	(90/9 — 91/9)	14 Gflops, high accuracy
GRAPE-2A	(91/7 — 92/5)	180 Mflops, high accuracy
HARP-1	(92/7 — 93/3)	180 Mflops, high accuracy
		Hermite scheme
GRAPE-3A	(92/1 — 93/7)	6 Gflops/board
		some 80 copies are used all over the world
GRAPE-4	(92/7 — 95/7)	1 Tflops, high accuracy
		Some 10 copies of small machines
MD-GRAPE	(94/7 — 95/4)	1Gflops/chip, high accuracy
		programmable interaction
GRAPE-5	(96/4 — 99/8)	5Gflops/chip, low accuracy
GRAPE-6	(97/8 — 01/7)	32 Tflops, high accuracy

boards each with 48 pipeline chips). GRAPE-4 system operates on 32 MHz clock, delivering the speed of 1.1 Tflops. Completed in 1995, GRAPE-4 was the first computer for scientific calculation to achieve the peak speed higher than 1 Tflops. Also, for 1995 and 1996 it was awarded the Gordon Bell Prize for peak performance, which is given to a real scientific calculation on parallel computer with the highest performance.

Technical details of machines from GRAPE-1 through GRAPE-4 can be found in our book[4] and reference therein.

GRAPE-5[11] was an improvement over GRAPE-3, with a similar, but more accurate number format than that was used for GRAPE-3. Also, it integrated two full pipelines which operate on 80 MHz clock. Thus, single GRAPE-5 chip offered the speed 8 times more than that of the GRAPE-3 chip, or the same speed as that of a 8-chip GRAPE-3 board. GRAPE-5 was awarded the 1999 Gordon Bell Prize for price-performance. The GRAPE-5 chip was fabricated with 0.35μm design rule by NEC.

GRAPE-6 is similarly the improvement over GRAPE-4. Since it is our newest hardware, we'll give a close inspection of its architecture in the next section.

Table 1 summarizes the history of GRAPE project. Figure 4 shows the evolu-

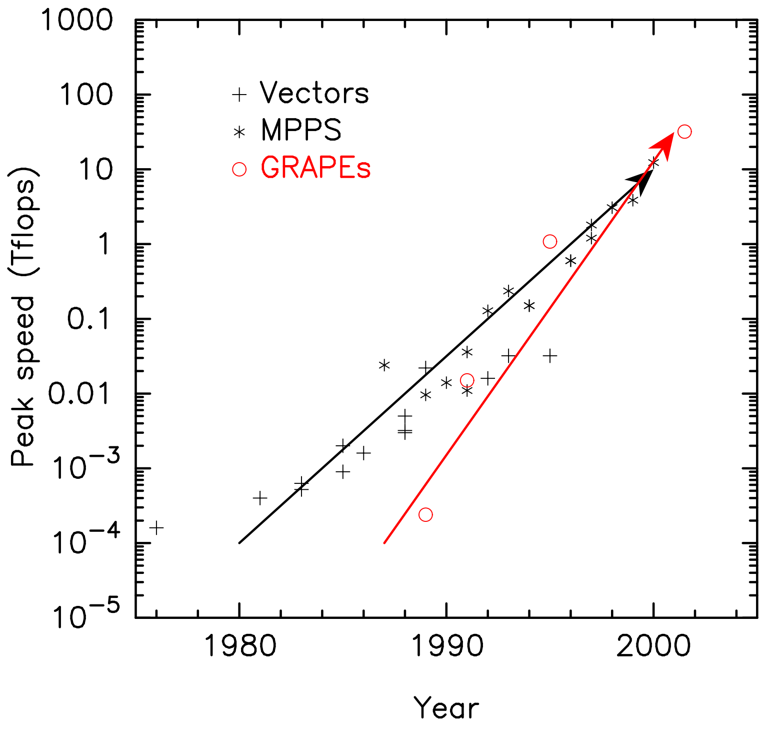

Fig. 4. The evolution of GRAPE and general-purpose parallel computers. The peak speed is plotted against the year of delivery. Open circles, crosses and stars denote GRAPEs, vector processors, and parallel processors, respectively.

tion of GRAPE systems and general-purpose parallel computers. One can see that evolution of GRAPE is faster than that of general-purpose computers.

These GRAPE hardwares, including GRAPE-6, have been applied to a number of astrophysical problems both by our group and by other researchers worldwide. Since there are too many of interesting results, we do not try to list them. Some of the resent results are summarized in the proceedings of IAU Symposium 208 "Astrophysical Supercomputing Using Particles", held in July 2001. Some more informations can be found at `http://www.astrogrape.org`.

3.1 Machines for Molecular Dynamics

Classical MD calculation is quite similar to astrophysical N-body simulations since in both cases we integrate the orbit of particles (atoms or stars) which interact with other particles with simple pairwise force. In the case of Coulomb force, the force law itself is the same as that of the gravitational force, and the calculation of Coulomb force can be accelerated by GRAPE hardware.

However, in MD calculations the calculation cost of van der Waals force is not negligible, though van der Waals force decays much faster than the Coulomb force (r^{-7} compared to r^{-2}).

It is fairly straightforward to design a hardware which can handle particle-particle force which is some arbitrary function of the distance between particles. We approximate the given function by a table of polynomials. In fact, we use this combination of table lookup and polynomial approximation for the calculation of $1/r^3$ from r^2 in GRAPE hardwares. So the actual change in the design of the hardware is rather minor.

We have designed two machines, GRAPE-2A and MD-GRAPE, following these lines of idea. GRAPE-2 was built using commercial chips and MD-GRAPE used a custom designed pipeline chip.

Another difference between astrophysical simulations and MD calculations is that in MD calculations usually the periodic boundary condition is applied. Thus, we need some way to calculate Coulomb forces from image particles. The direct Ewald method is rather well suited for implementation in hardware. In 1991 we developed WINE-1, a pipeline to calculate the wavespace part of the direct Ewald method. The real-space part can be handled by GRAPE-2A or MD-GRAPE hardware.

In 1995, a group led by Ebisuzaki in RIKEN started to develop MDM, a massively parallel machine for large-scale MD simulations. Their primary goal is the simulation of protein molecules.

MDM consists of two special-purpose hardware, massively parallel version of MD-GRAPE (MDGRAPE-2) and that of WINE (WINE-2). MDGRAPE-2 part consisted of 1536 custom chips with 4 pipelines, for the theoretical peak speed of 25 Tflops. WINE-2 part consists of 2,304 custom pipeline chips, for the peak speed of 46 Tflops.

MDM shared the 2000 Gordon-Bell performance Prize with GRAPE-6. It also was selected as the finalist for the 2001 Gordon-Bell performance Prize, again along with GRAPE-6.

4 GRAPE-6

In 1997, we started the GRAPE-6 project. It's a five-year project funded by JSPS (Japan Society for the Promotion of Science), and the planned total budget is about 500 M JYE.

The GRAPE-6 is essentially a scaled-up version of GRAPE-4[8], with the peak speed of around 100 Tflops. As of the time of writing, a 32 Tflops system with 1024 chips is in operation. The peak speed of a single pipeline chip is 31 Gflops. In comparison, GRAPE-4 consists of 1728 pipeline chips, each with

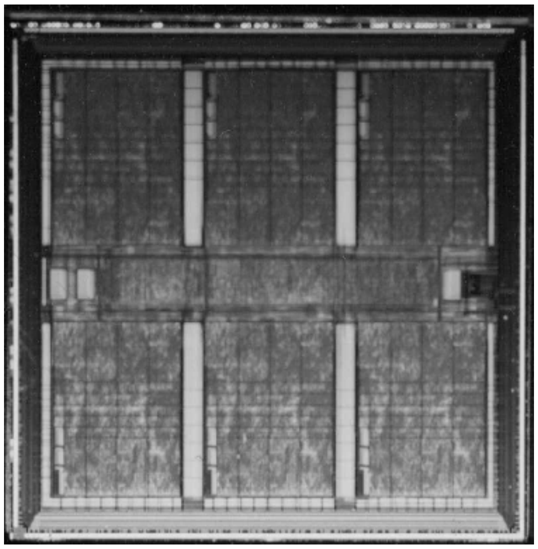

Fig. 5. The GRAPE-6 processor chip.

600 Mflops. The increase of a factor of 50 in speed is achieved by integrating six pipelines into one chip (GRAPE-4 chip has one pipeline which needs three cycles to calculate the force from one particle) and using 3 times higher clock frequency. The advance of the device technology (from 1μm to 0.25μm) made these improvements possible. Figure 5 shows the processor chip delivered in early 1999. The six pipeline units are visible.

Figures 6 and 7 shows the processor board with 32 processor chips and the 32-board system. This 32-board system has the theoretical peak speed of 32 Tflops, and has achieved the sustained speed of 11.5Tflops for the simulation of 1.4 million-body system.

We plan to extend this system to 80-board, 80-Tflops system by the end of FY 2001. Single-board systems (4-32 chips) are commercially available.

5 Discussion

5.1 Are special-purpose computers difficult to build?

We have seen that it is possible to develop special-purpose computers which offers price-performance and also absolute performance better than those of general-purpose computers by one or two orders of magnitude.

However, our GRAPE project is not the only project to develop special-purpose computers, and yet it is not too easy to name other projects which have achieved similar level of success. We briefly discuss why.

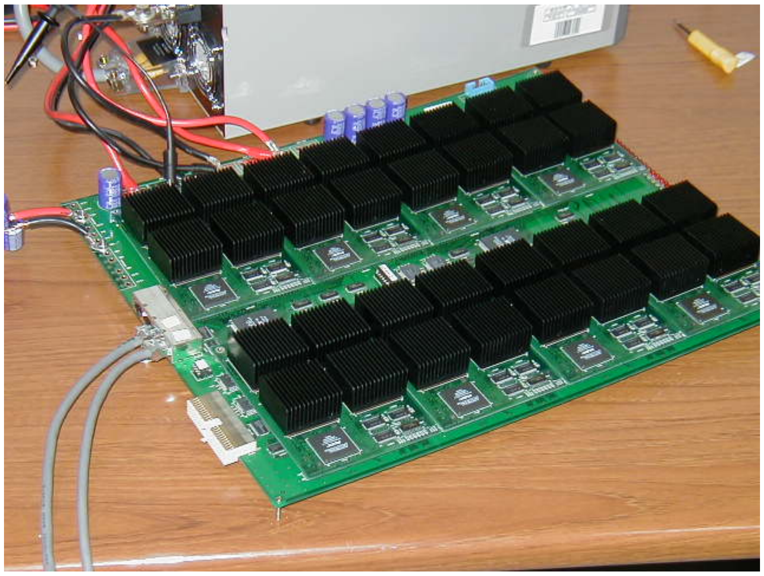

Fig. 6. The processor board of the GRAPE-6 with 32 processor chips. Four processor chips are mounted on modules, on which eight memory chips are also mounted on the bottom side. One board houses eight modules.

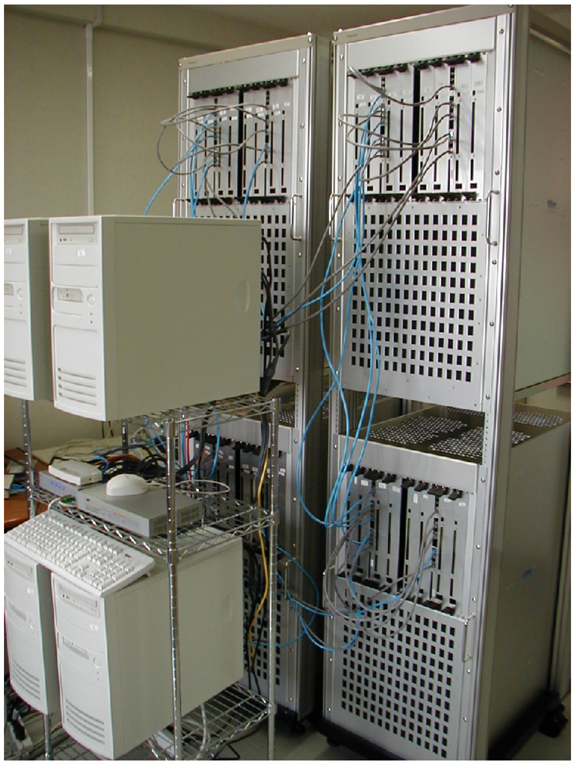

Fig. 7. The 32-board, 32-Tflops GRAPE-6 system with its host computer. The host is a cluster of 4 PCs with 1.7GHz Intel P4 processors connected by 100Mb Ethernet.

There are variety of reasons why a project ends up as a failure. These include:

(1) **Long development time**. Even though there are enormous inefficiency, the performance of general-purpose computers is still improving at the rate of a factor of 10 in every five years. Therefore, if you spend five years to develop a machine, you lose a factor of ten in relative advantage. Of course, in many cases the designers originally underestimated the development time. So this problem were very difficult to avoid.
(2) **Too small gain**. Even if your machine achieved the advantage of a factor of 10 by the time of completion, in five years it will lose all advantages. So the lifetime of your machine is rather short. If the advantage is a factor of 10 *at the time of the design*, the project is guaranteed to fail.
(3) **Too wide applications**. This is not necessarily a failure, if high performance is achieved. However, this is the most common failure which leads to both small gain and long development time.
(4) **Too narrow applications or too difficult to use**. If there is not much scientific outcome from your machine, it would not be regarded as a great success even if it achieved very good performance.
(5) **Obsolete technology/design method**. Device technology is moving very fast, and so is the design software/methodology. So, if the technology you used was three-year old, you have already lost the relative advantage of a factor of four.
(6) **Untested technology/design method**. Even though the advance of the device technology is pretty well predictable, in most cases what the manufactures claim that they can deliver would not be delivered on time. So be careful.

Of course, projects to develop general-purpose computers also suffer most of the above problems. The advantage of the special-purpose design is the possibility to make better use of the available transistors. The disadvantage is the limitations in available design resources (design experts, budget, access to the latest technology etc etc).

5.2 Future prospect

We overviewed the technological and economical trends in high-performance scientific computing. At least for certain range of problems, a special-purpose computer, or, a combination of special- and general-purpose computers such as our GRAPE systems, offer a real and proven advantage over traditional general-purpose computers. The relative advantage has been increasing, and will keep to do so for the next one or two decades, unless some radical change in the design method for general-purpose computer would take place.

Currently, it seems the direction of the evolution of general-purpose computer is a cluster of commodity PCs. Current microprocessors for commodity PCs offer very high performance in very low cost, partly because of their low production cost due to mass production, and partly because the investment for the design of the chip is actually very high. The large volume of production justifies the high development cost. However, the fact that microprocessors for commodity PCs offer performance better than any other general-purpose computers on the market does not necessarily imply their design is optimal. As we have seen, only a tiny fraction of the available transistors is used for actual arithmetic operations, and that fraction has been decreasing quite rapidly.

Special-purpose systems, at least in principle, will not suffer these problems. In practice, however, the approaches we have taken so far is becoming more and more difficult, because the initial development cost of the custom LSI goes up as the technology advances. The development cost goes up because of two reasons. The first is that the amount of work to do the logic design, test design, physical layout and design validation increases as the number of transistors in a chip increases. In the case of special-purpose systems, the amount of work for the logic and test design, which we can do in-house, does not increase too rapidly, but physical layout and design validation, which is the work of the semiconductor company, take a long time and therefore lots of money. The second reason is that the investment needed to build the semiconductor plant increases rapidly as the technology advances.

Roughly speaking, for same physical size of the chip, the amount of money we paid was inversely proportional to the design rule: around a quarter million USD for 1μm (GRAPE-4) and around 1 million USD for 0.25μm (GRAPE-6). In both case, the size of the chip is around 110mm^2. If this trend continues, the design cost of 0.13μm would be two million USD. The total budget must be significantly larger than the development cost of the chip, since otherwise the price of a single chip would be too high. It is not easy to get such a large fund for a project of theoretical study in pure science.

One possible compromise is the use of FPGA (Field programmable gate array) chips as the building block of the pipelined processors. This choice will reduce the initial cost from one million dollars to less than ten thousand dollars (the price of the design software), since the FPGA chip itself is mass-produced and the design is loaded to a FPGA chip by configuring the switches and lookup tables in the chip.

Roughly speaking, because of this programmability, the calculation speed achieved by one FPGA chip is 100 times lower than that can be achieved with a custom LSI of the same size and same technology. However, this difference can be offset by the possibility of using the most advanced technology, the possibility to fine-tune the design to individual problems, and most importantly, much shorter design cycle time.

To give an example, large FPGA chips at the time of writing (Summer 2001) have the nominal gate count of around 1 million, which is sufficient to implement the logic for a GRAPE-4 chip (100K gates), and the clock speed would be around 75 MHz or more. Thus, one FPGA chip can deliver 1.5 Gflops. This might not sound very impressive compared to 31 Gflops of GRAPE-6 chip or around 2 Gflops of present microprocessors. However, compared to microprocessors, to build a massively parallel machine out of FPGA is much easier, and we can expect higher execution efficiency, for the same reason as we achieved higher efficiency on GRAPE hardwares.

Here again, in the long run we might see the same problem as the general purpose computer. The on-chip wiring would ultimately limit the speed and the circuit density. Because of the requirement of the programmability, as the number of transistors on the FPGA chips increases, more and more fraction of these chips will be used for wiring. However, for the next several years, this limitation would not be too severe, and it is possible that some new design philosophy will allow us to make better use of FPGA chips.

It is at least possible to use FPGAs for "proof of the concept" studies, where we demonstrate that one particular custom design is actually usable and that it achieves better cost-performance than general-purpose solutions. If that demonstration was successful, the grant large enough to make custom LSI might be offered.

To summarize, the initial cost of the large custom LSI might become too high for the level of the amount of grants we can reasonably expect. However, machines based on FPGAs can be used for small projects. The cost advantage of FPGAs will not be as large as that of custom LSI chips, but compared to general-purpose microprocessors, they still offer large advantage. So we expect to see many successful projects to apply FPGAs for large-scale computing in the near future. The largest ones will be done on custom LSIs, but the rest will be done on FPGAs.

Acknowledgments

We would like to thank Daiichiro Sugimoto, Toshikazu Ebisuzaki, Makoto Taiji, Tomoyoshi Ito, Toshiyuki Fukushige and many others who were involved in the development of the six generations of GRAPE hardwares, and Yoko Funato, Simon Portegies Zwart, Steve McMillan, Piet Hut and again many others for discussions and collaborations in software development and scientific works on GRAPE hardwares. This work is supported by the Research for the Future Program of Japan Society for the Promotion of Science (JSPS-RFTP97P01102).

References

[1] D. J. Becker, T. Sterling, D. Savarese, D. J. E., U. A. Ranawake, C. V. Packer, Beowulf: A parallel workstation for scientific computation, in: Proceedings of the 1995 International Conference on Parallel Processing (ICPP), IEEE Comp. Soc., Los Alamitos, 1995, pp. 11–14.

[2] T. L. Sterling, J. Salmon, D. J. Becker, D. F. Savarese, How to Build a Beowulf: A Guide to Implementation and Application of PC Clusters., MIT Press, Cambridge, MA, 1999.

[3] D. Sugimoto, Y. Chikada, J. Makino, T. Ito, T. Ebisuzaki, M. Umemura, A special-purpose computer for gravitational many-body problems, Nature 345 (1990) 33–35.

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

[5] J. Barnes, P. Hut, A hierarchical o(nlogn) force calculation algorithm, Nature 324 (1986) 446–449.

[6] L. Greengard, V. Rokhlin, A fast algorithm for particle simulations, Journal of Computational Physics 73 (1987) 325–348.

[7] R. W. Hockney, J. W. Eastwood, Computer Simulation Using Particles, IOP Publishing, Ltd., Bristol, 1988.

[8] J. Makino, M. Taiji, T. Ebisuzaki, D. Sugimoto, Grape-4: A massively parallel special-purpose computer for collisional n-body simulations, The Astrophysical Journal 480 (1997) 432–446.

[9] T. Ito, J. Makino, T. Ebisuzaki, D. Sugimoto, A special-purpose n-body machine grape-1, Computer Physics Communications 60 (1990) 187–194.

[10] J. Makino, S. J. Aarseth, On a hermite integrator with ahmad-cohen scheme for gravitational many-body problems, Publications of the Astronomical Society of Japan 44 (1992) 141–151.

[11] A. Kawai, T. Fukushige, J. Makino, M. Taiji, Grape-5: A special-purpose computer for n-body simulations, Publications of the Astronomical Society of Japan 52 (2000) 659–676.