SPECIAL-PURPOSE COMPUTER FOR GRAVITATIONAL N-BODY SYSTEM: GRAPE

JUNICHIRO MAKINO
Department of System Sciences, University of Tokyo
3-8-1 Komaba, Meguro-ku, Tokyo 153-8902, Japan

Abstract. In this paper I'll briefly overview the GRAPE (GRAvity PipE) project, in which we develop a series of special-purpose computers for stellar dynamics. First, I overview the evolution of high-performance general-purpose computers. It is shown that the "hardware efficiency" of general-purpose computers has been going down exponentially, and will continue to do so. Then, I'll describe the approach of building the special-purpose computers as an alternative. I also briefly describe the past of GRAPE project, and the development plan for GRAPE-6, which will provide around 200 Tflops by the year 2000.

1. Introduction

The evolution of the general-purpose computer in the last half century has been truly amazing. The peak speed of the fastest computer of the time has been improved by about 10 orders of magnitude in 50 years.

This unusually fast and long-lasting exponential growth has been the driving force for the advance of the entire field of numerical astrophysics. In many fields of numerical astrophysics, the improvement in the speed or memory capacity of computers has been essential. Of course, the improvement in numerical method and the progress in the understanding of the physical processes have been equally important, but without the exponential growth of the computer capability, the last half century of computational astrophysics would have been completely different.

This extremely rapid increase is expected to continue for the next 10–20 years [12]. Thus, in principle, we could be quite optimistic on the future of our field. In practice, however, there are several reasons to be skeptical about the future. In this paper, I'd like to describe the problem, and what I believe is part of the solution.

2. Past and future of general-purpose computing

Very roughly speaking, the last 10 years of the high-performance computer can be summarized as the slow decline of the vector processors and comparably slow rise of distributed memory parallel processing. This trend is driven mostly by the constraint in the silicon technology [10].

As Hillis [7] stressed, there used to be two very different ways to construct a fast computer. One is to connect the fastest possible processors (vector-parallel approach). Since first processors tend to be expensive, the number of processors you can buy used to be small. The other is to connect many relatively slow processors (MPPs). Cray and Japanese manufacturers had been pursuing the first approach, while a number of companies tried the second, with limited success.

In late 1980s, however, two ways started to converge (or, to collide). The reason is that the microprocessors, which were the building blocks of MPPs, caught up with the vector processors in the processing speed. In 1970s, there were the difference of nearly two orders of magnitude between the clock speed of vector processors and that of CMOS microprocessors. In addition, microprocessors typically required tens or even hundreds of cycles for a floating-point operation. Thus, the speed difference was at least a factor of 10^2. This factor could be as large as 10^4.

Today, the difference in the speed has almost vanished. The clock speed of today's microprocessors is considerably faster than that of vector-parallel processors, and microprocessors can perform one or more floating point operations per cycle. Thus, programs that are not very well vectorized or parallelized are actually faster on microprocessors.

As far as the performance characteristics are concerned, today's high-performance computers are not much different from a pile of personal computers connected by a fast network. We need very high degree of parallelism to make effective use of these machines. In addition, practically all machines has non-uniform-access memory architecture (NUMA), which means that we need to design the program so that data are distributed over processors in an appropriate manner. This is true for both the vector-parallel machines such as NEC SX-4 and MPPs like Hitachi SR-2201 or SGI-Cray T3E.

This characteristic of modern high-performance computers has rather strong negative impact on the future of the computational astrophysics. Firstly, to use parallel computer is not easy, even in the case of ideal parallel machine with physically shared memory. Secondly, NUMA machines are far more difficult to program compared to machines with physically shared memory. Thirdly, to develop message-passing style program is even more difficult.

In the case of N-body simulations, just to run a simple, direct-summation

scheme with constant stepsize on a massively parallel computer is already a difficult task[6]. Implementing a particle-mesh or P^3M scheme or treecode with shared timestep is not impossible, but proven to be a task which can easily eat up several years of time of a very good researcher, which could otherwise be used for doing science (see, *e.g.*, [14] or [3] for examples of parallel tree codes). Implementing an individual timestep algorithm has been demonstrated to be difficult [13].

The essential reason for this difficulty is that parallel computers are designed without any consideration for astrophysical applications, not to say N-body simulations.

In the following, we describe an alternative. Instead of adopting the algorithm to available computer, we design the most cost-effective combination of the algorithm and computer.

3. Special-purpose computing

The idea of designing special-purpose computer for scientific simulation is not new. Until recently, however, it has not become widely accepted (maybe still not). This is primarily because the advantage over general-purpose computers had been limited. First of all, one had to develop hardware, which was time-consuming and costly venture. It was also necessary to develop softwares (system programs and application programs).

To make the matter worse, in a few years hardware would become obsolete and the investment on hardware and software would be lost. Of course, this is also true for general-purpose hardware. The software for general-purpose computers was, however, likely to be reused, unless there was drastic change in the architecture.

However, this situation has changed considerably in the last decade. As I described in the previous section, the development of the software for general-purpose computers has become much more difficult and costly. Moreover, we cannot expect that programs developed for the current generation of machines can be used in the future machines without extensive rewriting. Vector processors required very different way of programming than was used on scalar processors. Parallel computers require yet another way. The current transition from vector processors to microprocessors or MPPs means most of the programs tuned for vector processors are now becoming useless.

On the other hand, the cost advantage of special-purpose computers, which determines the lifetime of the machine, has been greatly improved. This is not because of the efforts at the side of the special-purpose computing, but because of the decline of the hardware efficiency of general-purpose computers. Though the peak speed of computers has been increasing expo-

nentially, the fraction of the hardware used to do arithmetic operations has been falling exponentially. In present microprocessors, only a few percents of the transistors on a chip are used to implement arithmetic units. Other 90+ % are used for cache memory and control logic. Moreover, it is quite difficult to achieve high efficiency with current microprocessors, because of the limitation in the memory bandwidth. Thus, if averaged over time, the typical efficiency of present microprocessors is around 0.1%.

If we can construct a machine with efficiency higher than, say, 10%, therefore, we can achieve the cost advantage of a factor of 100 or larger. Of course, the production cost of the machine would be higher because of the small quantity, and there would be losses due to inefficiencies in the design. However, these two combined is not likely to be as large as a factor of 100. Thus, the special-purpose architecture is becoming a quite attractive solution. Note that 10 years ago the efficiency of general-purpose computers was higher, and therefore it was more difficult to develop special-purpose computers.

There had been two very different approaches to build special-purpose computers for scientific simulations. One is to design a programmable parallel computer with various optimizations. For example, if we do not need much memory, we could replace the DRAM main memory with a few fast SRAM chips, thus reducing the total cost by a large factor. If we do not need fast communication, we can use rather simple network architecture.

In 1980s, CMOS VLSI floating-point chipsets such as Weitek 1064/65 [5] offered the theoretical speed of around 1/10 of the speed of vector processor for the cost around 1,000 USD. Thus, if one could use several of them in parallel, one could construct a vastly cost-effective computer. This led to numerous projects, some of them were highly successful. In particular, PAX[9] and Caltech Hypercubes[6] had been so successful that many companies started to sell similar machines as general-purpose parallel computers. As a result of their success, developing a parallel computer as special-purpose system has become unpractical. You can buy a better one from computer companies.

The other approach is to develop "algorithm oriented" processors. Our GRAPE (GRAvity piPE) machines is an extreme in this direction. As the name suggests, GRAPE is a device which evaluates the gravitational force between particles.

In direct N-body simulation, more than 99% of CPU time is spent to calculate the gravitational force between particles. Even in the case of more sophisticated algorithms such as the treecode, FMM and P^3M, large fraction of time is spent for pairwise force calculation. In usual implementation, the direct force calculation consumes typically about half of the total CPU time. However, it is possible to modify these algorithms to reduce the cost

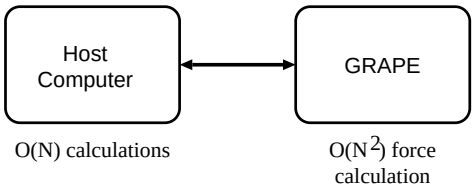

Figure 1. The basic architecture of GRAPE

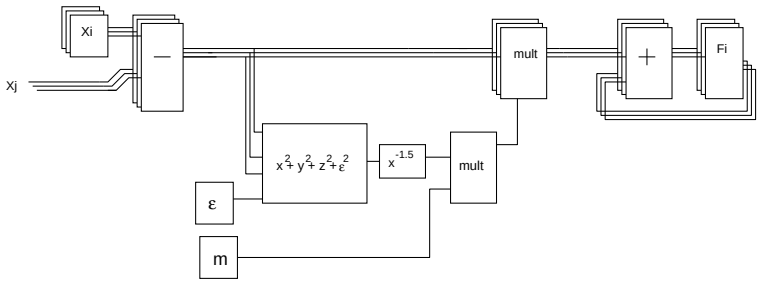

Figure 2. The force calculation pipeline

of calculations other than pairwise force calculation, by *increasing* the calculation cost of pairwise force calculation. Thus, actual gain one can achieve for these algorithms is much larger than a factor of two [1].

4. The GRAPE project

Figures 1 and 2 show the basic idea of GRAPE systems. The special-purpose device performs only the force calculation, while a general-purpose host computer takes care of everything else. In the special-purpose device, a pipeline which embodies the data flow for the force calculation calculates interactions at the rate of one interactions per cycle (around 30 operations/cycle). Multiple pipelines can be attached to one memory unit so that they calculate the force on different particles. With multiple memory unit, it is also possible for multiple pipelines to calculate partial forces on one particle.

Table 1 summarizes the history of GRAPE project. New machine has become larger and more complex, but at the same time more cost-effective, even when the improvement due to the advance in the device technology is discounted.

For N-body simulations of planetary formation and globular clusters, GRAPE approach proved itself to be extremely effective. In fact, significant fraction of N-body studies in these fields are now performed on GRAPEs,

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(32bit/64bit)
HARP-1	(92/7 — 93/3)	180 Mflops, high accuracy(32bit/64bit)
		Hermite scheme
GRAPE-3A	(92/1 — 93/7)	6 Gflops/board
		some 50 copies are used all over the world
GRAPE-4	(92/7 — 95/7)	1 Tflops, high accuracy (32bit/64bit)
		Some 10 copies of small machines
MD-GRAPE	(94/7 — 95/4)	1Gflops/chip, high accuracy(32bit/64bit)
		programmable interaction

and many new and important results are published. Many copies of GRAPE hardwares are now used for SPH and N-body simulations in cosmology, galactic dynamics and galaxy formation.

5. GRAPE-6

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

Figure 3 shows the basic structure of GRAPE-6. The gravitational pipeline is essentially a scaled-up version of GRAPE-4, with the peak speed of around 200 Tflops. This part will consist of around 4000 pipeline chips, each with the peal speed of 50 Gflops. In comparison, GRAPE-6 consists of 1700 pipeline chips, each with 600 Mflops. The increase of a factor of 100 in speed is achieved by integrating six pipelines into one chip (GRAPE-4 has one pipeline which needs three cycles to calculat the force from one particle) and using 3–4 times higher clock frequency. The advance of the device technology (from 1μm to 0.25μm) these practical.

The multipurpose pipeline part is a new feature, whose goal is to widen the application range. The original GRAPE architecture consists of only two parts: GRAPE and the host (see figure 1). GRAPE calculates only gravity and everything else is done on the host. This architecture is ideal for pure N-body simulation, but not quite so if we want to deal with, for example, self-gravitating fluid using SPH.

The most costly part of SPH calculation, aside from the gravity, is the evaluation of the hydrodynamical interaction between particles. Thus, a specialized pipeline quite similar to that of GRAPE [15] could improve the

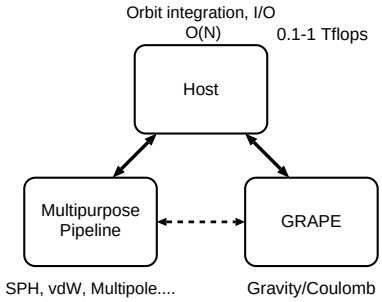

Figure 3. GRAPE-6

speed quite a lot. However, there are two reasons to believe it is difficult. The first one is that the gain one can achieve is limited. Since the interaction calculation accounts for only around 90% of the total CPU time, even if the SPH pipeline is infinitely fast the gain we can achieve does not exceed a factor of 10. The other reason is that there are many SPH algorithms. Newton's law of the gravity has not changed in the last two century, and the algorithm to calculate it is well established. However, SPH is still rather new method. One day somebody might come up with a novel method, which is much better than traditional one but cannot be implemented on a specialized hardware. Thus, it looks rather risky to develop an SPH hardware.

If we can "program" the pipeline unit, we can eliminate most of the risks. If someone comes up with a new and improved SPH scheme, a programmable pipeline could still be used for that. Moreover, such a programmable pipeline might be used for many other problems.

One might wander whether a programmable pipeline is a practical concept or not. Didn't the author argued against the programmability in section 3? Well, the advance in the FPGA (field-programmable gate array) technology has made the new approach viable [2].

An FPGA can be programmed to realize different functions by loading the configuration data. An FPGA consists of many logic blocks and a switching matrix. A logic block is typically a small lookup table. A SRAM block is used to implement this lookup table so that its function can be changed. The switching matrix can also be programmed to make connections in different ways.

This programmability incurs quite large inefficiency. The circuit size which can be implemented in the current largest FPGAs is equivalent to 10^5 transistors, while largest LSIs contain more than 10^7 transistors. In addition, there is also speed difference of factor 3–5.

Even with these large overheads, however, FPGAs are now becoming more efficient than general-purpose microprocessors. The reason is quite

simple. The efficiency of FPGAs has not been falling too rapidly, since the relative overhead is roughly independent of the technology. In fact, the speed penalty is decreasing, since the signal propagation delay is becoming more important. This delay is not much different for FPGA and usual LSIs.

We have developed a small experimental machine, the PROGRAPE-1 [8]. It has two large FPGAs. The FPGA chips in PROGRAPE-1 can house, for example, one pipeline of GRAPE-3 [11] or WINE-1 [4]. GRAPE-6 will include a massively-parallel version of this PROGRAPE system, which can be used for various applications like SPH, Ewald method, and van-der-Waals force calculation in molecular Dynamics.

The GRAPE-6 will be completed by the year 2000. We plan to make small version of GRAPE-6 (peak speed of "only" a few teraflops) commercially available by that time. We've found that the commercial availability of small machines is essential to maximize the scientific outcome from GRAPE hardwares.

This work is supported in part by the Research for the Future Program of Japan Society for the Promotion of Science (JSPS-RFTP 97P01102).

References

1. Athanassoula E., Bosma A., Lambert J. C., and Makino J. (1998) *MNRAS***293**, pp. 369–380.
2. Buell D. and Arnold J.M.and Kleinfelder W. (1996) *Splash 2: FPGAs in a Custom Computing Machine*. IEEE Comp. Soc. Press, Los Alamitos, CA.
3. Dubinski J. (1996) *New Astronomy*, **1**, pp. 133–147.
4. Fukushige T., Makino J., Ito T., Okumura S. K., Ebisuzaki T., and Sugimoto D. (1993), *PASJ*, **45**, pp. 361–375.
5. Fandrianto J. and Woo B. Y. (1985), In *Proceedings of Seventh Symposium on Computer Arithmetic*, pages 93–100. IEEE.
6. Fox G. C., Williams R. D., and Messina P. C. (1994) *Parallel Computing Works!*, Morgan Kaufmann, San Francisco, California.
7. Hillis W. D. (1985) *The Connection Machine*. MIT Press, Cambridge, Massachusetts.
8. Hamada, T., Fukushige, T., Kawai, A, and Makino, J. (1998), In *this proceedings*.
9. Hoshino T. (1992), In Mendez R. (ed) *High Performance Computing: research and practice in Japan*, pages 239–256. John Wiley and Sons, Baffins Lane.
10. Makino J. and Taiji M. (1998) *Special Purpose Computers for Scientific Simulations – The GRAPE systems*. John Wiley and Sons, Chichester.
11. Okumura S. K., Makino J., Ebisuzaki T., Fukushige T., Ito T., Sugimoto D., Hashimoto E., Tomida K., and Miyakawa N. (1993), *PASJ*, **45**, pp. 329–338.
12. Sterling T., Messina P., and Smith P. H. (1995) *Enabling Technologies for Petaflops Computing*. MIT Press, Cambridge, MA.
13. Spurzem R. (1996), submitted to *MNRAS*.
14. Warren M. S. and Salmon J. K. (1992), In *Supercomputing '92*, pages 570–576. IEEE Comp. Soc., Los Alamitos.
15. Yokono Y., Ogasawara R., Takeuchi T., Inutsuka S., Miyama S. M., and Chikada Y. (1996), In Tomisaka K. (ed) *Numerical Astrophysics Using Supercomputers*. National Astronomical Observatory, Japan.