Next Generation GRAPE systems

Junichiro Makino
and
Makoto Taiji*
Department of Information Science and Graphics,
*Department of Earth Science and Astronomy,
College of Arts and Sciences, University of Tokyo,
Tokyo 153, Japan

Abstract

In this note, we overview the possible future development in GRAPE systems. In particular, we give the formula for the size of the simulation one can handle at given time for given amount of money, for globular cluster simulation, collisionless N-body simulation with treecode and P^3M, and SPH simulations. In appendix, we summarize the systems that are currently available/will be available soon.

Contents

1 Introduction.

In this note, we will try to describe what kind of machines we will be able to use for the next 5-10 years for astrophysical N-body (and SPH) simulations. In the first two sections, we will cover general technological trends that determine the progress in the speed of the general-purpose computers and special-purpose computers such as GRAPE. The bottom line of these two sections is that with the present (and predicted future) architecture of general-purpose computers, it becomes more and more difficult to take advantage of the advance in the semiconductor technology. On the other hand, special-purpose computers have no such difficulty. Thus, to be able to develop specialized machines for one's own problem is likely to become an advantage that is even more important than it is now.

The next three sections will overview more technical aspects of the GRAPE systems specific to collisional and collisionless (or SPH) problems. The main difference between the collisional and collisionless problems is the required accuracy of the force. At present, there is another difference that the collisional problem requires the individual timestep. SPH simulations, however, also require the individual timesteps. In addition, collisionless N-body simulation will soon require individual timesteps, since as we increase the number of particles, we study systems with higher density contrast. Thus the individual timestep will be necessary for both the collisional and collisionless simulation.

The difference in the requires accuracy leads to differences in both the software and hardware. On the side of the software, one can rely on approximate schemes such as P^{3}M or treecode for collisionless calculations, but it is much more difficult to use these or other fast algorithms for collisional simulations. On the side of the hardware, the difference in the required accuracy has a direct effect on the hardware cost, because the number of transistors is at least proportional to the number of bits in the number format used. Moreover, the size of the multipliers is proportional to the square of the word length. Thus one pipeline for the force calculation for collisional simulations requires 5-10 times as many transistors as that for collisionless simulations required.

The difference in the algorithm used also have some impacts on the hardware. In the case of the collisional simulation, we do use the direct summation (maybe the neighbor scheme, but the difference in speed is rather small). Thus, there is $O(N)$ difference between the amount of the work that the host computer does and the amount of the work that GRAPE does. In practice, this means that GRAPE can be $\sim N/10$ times faster than the host computer.

In the case of the collisionless simulation, the situation is not that simple. The use of the sophisticated algorithms such as P^{3}M and treecode reduces the amount of work on GRAPE, but increases the amount of work on the host computer. In the case of the treecode, the calculation cost of the treecode on a general-purpose computer is $O(N \log N)$, and what we can do with GRAPE is, at the best, to reduce the calculation cost of the host to $O(N)$. For practical values of N, this means that the speed of a GRAPE system cannot be more than a few hundred times the speed of the host computer, no matter how fast is the GRAPE hardware.

2 Trends in computer industry.

A zeroth order approximation of the speed of a computer would be:

$$P \propto n_{tr} s, \tag{1}$$

where P is the processing speed in some unit, n_{tr} is the number of transistors used in the processor, and s is the speed of the transistor. The number of transistors one can use depends on the amount of money and the time. The number of transistor one can buy is proportional to the money and for the constant amount of money it quadruples in every three years (Moore's Law). In other words, it is expressed as

$$n_{tr} \propto B2^{(Y-1994)/1.5}, \tag{2}$$

where B is the amount of money in some unit and Y is the year. To put some real numbers,

$$n_{tr} \simeq 10^4 B2^{(Y-1994)/1.5}. \tag{3}$$

Here, B has the unit of U.S. dollar.

The speed of the transistor doubles in every three years, or

$$s \propto 2^{(Y-1994)/3}. \tag{4}$$

Combining equations 2 and 4, we have

$$P \propto B2^{(Y-1994)}. \tag{5}$$

In other word, the performance of the computer of the same price should double every year.

However, this doubling in every year did not take place. If we plot the speed of the fastest computer for the last 40 years, it is very well approximated as a factor of 100 increase in every decade, or doubling in every 1.5 year. Well, where this discrepancy came from? If we look at the evolution of microprocessors (both CISC and RISC), the number of transistors used is well represented by equation 2, and the speed of the transistor used is also well represented by 4. Thus, the assumption that the speed of the computer is proportional to the number of the transistor used must be wrong.

As the computer architecture evolves, it becomes more complicated. In other words, the fraction of the circuit that actually do the arithmetic operation becomes smaller. This evolution is necessary to keep up with the evolution in the semiconductor technology, or to utilize the increasing number of transistors available. If we show just a few examples from recent development in computer architecture, it includes branch prediction, superscalar ALU, register scoreboarding, register renaming, out-of-order execution, on-chip secondary cache, speculative execution and so on. Whatever the meaning of them, they do require a lot of transistors to achieve rather marginal gain in speed. A microprocessor of 1995 has 100 time many transistors and roughly 10 times faster clock speed compared to a microprocessor of 1985. However, the performance increase is more like a factor of 100 than a factor of 1000, because with 100 times as many transistors, what can be done in a single clock was increased only by a factor of 10. In the near future, to utilize the available transistors in general-purpose microprocessors will be even more difficult than it is now, since present-day microprocessors are already as complecated as one can imagine for single processor. The only practical way to utilize large number of transistors will be to design multiprocessor on a single chip, which will be quite difficult.

3 Potential of GRAPE systems.

In the case of GRAPE systems, the situation is totally different from what we described in the previous section. We started the GRAPE project in 1988 (See Ebisuzaki et al. 1993 for review). The first machine we built, the GRAPE-1, had 4 MHz clock speed and the number of transistors of the order of 10^7 for the pipeline. This number is however not a very meaningful one, since in GRAPE-1 most of arithmetic operations were implemented in big lookup tables which consumes large number of transistors. In the case of the GRAPE-2, the number of the transistors for the pipeline is of the order of 10^6 and the clock speed is again 4 MHz, for the speed of 40 Mflops. For GRAPE-4, the number of transistors for the pipelines is 10^9 and the clock speed is 30 MHz, for the peak speed of 1 Tflops. Thus, we got a factor of 10,000 speedup for a 10 times faster clock and 1,000 times more transistors. In other words, we did get what we would naively expect. This is because there is no circuit used for complicated control units. In the case of a general-purpose computer, as the number of available transistor increases, some of them are used to increase the degree of parallelism in the arithmetic unit, but larger and larger fraction of the transistors are used to support this increased level of parallelism. This is most clearly demonstrated in superscalar CPUs, which typically need 10x or so many transistors to achieve a factor of 4 theoretical maximum parallelism, which in practice gives a factor of 2 or less speedup.

In practice, however, this excellent performance of GRAPE systems can only be achieved by constantly upgrading the hardware in every 3-5 years or so. The upgrading includes designing the hardware, constructing and debugging the prototype, and manufacturing the production

Table 1: Future GRAPE cost and performance

System	Year	Budget (M$)	Type	speed (Tflops)
L1	1996	0.4	collisionless	1.6
L2	1996	1	collisionless	10
H1	1997	3	collisional	16
H2	2000	10	collisional	500

Table 2: Future GRAPE: number of chips and power consumption

System	Year	speed (Tflops)	pipelines	chips	boards	power (kW)
L1	1996	1.6	800	25	1-2	0.2-0.4
L2	1996	10	5000	150	10	1-2
H1	1997	16	3000	1000	30	10-20
H2	2000	500	50000	5000	150	100-200

machine, and also includes the development of necessary softwares. These consumes fairly large amount of money and human resources.

Very roughly speaking, the speed of GRAPE system for given amount of money and the date of completion would be expressed as

$$P_{grape} = \begin{cases} 2^{Y-1992}(B-0.3) & \text{(collisionless)}, \\ 2^{Y-1994}(B-1) & \text{(collisional)}. \end{cases} \tag{6}$$

Here, P is the speed in Teraflops and B is the amount of money in 10^6 dollars. To give some ideas, Table 1 gives a few examples of the speed one can get. First column gives the identification of the system which will be used in the later discussion.

To give some more idea on how these machines look like, we estimated the number of pipelines, number of chips, number of boards and power consumptions, which is summarized in table 2.

If we keep using the forced air cooling, the heat generation per unit volume of the machine must be kept constant. This limits the number of chips per board to be of the order of 50. Any kind of liquid cooling system would increase the development cost and running cost quite significantly. So here we assumed that we will not use the liquid cooling system.

Another important point one has to take into account is the time to develop the hardware, that is $2+-1$ years depending on various factors. The machine for collisionless simulation would took somewhat shorter time.

In the following sections, we will discuss the size of the system we can handle and the requirement for the host computer for a given speed of GRAPE.

Table 3: Performance: Globular cluster simulation

System	Year	N
GRAPE-4(1T)	1995	$5-10\times10^4$
H1(16T)	1997	$10-20\times10^4$
H2(1P)	2000	$5-10\times10^5$

4 Collisional Systems.

4.1 Direct summation

Here, we give a rough estimate of the kind of simulation we can handle for collisional simulation. As an example, we give the simulation of isolated cluster of equal-mass particles. Table 3 gives the number of particles that can be handled with the CPU time of 1 month, for the simulation over a few collapse times. The time to perform a more realistic simulation varies depending on what effect we take into account. Inclusion of the mass spectrum, galactic tidal field, disk shock, bulge shock, and stellar evolution all accelerate the evolution of the cluster significantly, resulting in somewhat larger number of particles. The inclusion of the primordial binaries, on the other hand, have some impact in the opposite way. The calculation cost is assumed to be proportional to N^3, where N is the number of particles.

The requirement on the host computer will be rather modest. The requirement for the speed of the host computer is expressed as

$$P_{host} = 5\times10^{-5}P_{grape}^{2/3}, \tag{7}$$

where P_{host} is the effective speed of the host computer in Tflops. The power index of 2/3 comes from the fact that the calculation cost on GRAPE is $O(N^3)$ while that on host is $O(N^2)$. The coefficient is from that of present GRAPE-4. System H1 will need the host computer with the effective speed of 200 Mflops, which means the peak speed of the order of 1-2 Gflops. This is about the speed the microprocessors will reach by that year. The peak speed of microprocessors shipped in 1994 is around 300 Mflops. So we can expect the usual factor 4 increase in every three year, that results in 1.2 Gflops by 1997. System H2 will need 5 Gflops, or the peak speed of the order of 50 Gflops. We need a multiprocessor machine with around 10 processors, which will not be very expensive (less than one million dollars).

4.2 Sophisticated algorithms on general-purpose systems

At least two different algorithm to improve the speed of the collisional simulation is known. The first one is the neighbor scheme, which gives the theoretical performance gain of the order of $(\alpha N)^{1/4}$, where α is the ratio between the speed of the neighbor force calculation and that of the total force calculation (Makino and Hut 1990). Thus, if we can keep α to be larger than 0.01, we can gain a factor of 10 in the total speed.

The other possibility is to use the treecode with a high accuracy, as investigated by McMillan and Aarseth (1993). This could, in principle, provide another factor of 10 increase in speed.

Here, we have to consider two rather different questions. The first one is the following: Is GRAPE with simple direct summation really faster than the most sophisticated algorithm on a general-purpose vector-parallel computer that one can buy for the same amount of money? The other is whether it would be practical to implement more sophisticated algorithms in GRAPE hardware.

Let's deal with the first question. The gain in the performance by sophisticated algorithms is larger for larger N. So let us analyze the target system H2. The peak speed of the general-purpose computer one will be able to buy for 10 M dollars in year 2000 will be around 3 Tflops, if we make a rather optimistic prediction. In other words, GRAPE will be about 300 times faster. Thus, there is no chance for the neighbor scheme on general-purpose computer to win. In the case of the treecode, if it achieves the speed close to the peak speed, a general-purpose computer might be nearly as fast as GRAPE for 1M particles. In practice, however, it is extremely unlikely that the treecode achieves the speed more than 10% of the peak performance. The reason is that the degree of the parallelism of a 3 Tflops machine will be too large to efficiently run a treecode with individual timestep. The machine most suitable to individual timestep treecode would be shared-memory vector-parallel machines along the line of Cray C-90 or NEC SX-4, for which the speed of treecode is of the order of 10-20 % of the theoretical peak, even for the shared timestep version (see, *e.g.*, Makino and Hut 1989). It is difficult to imagine that an individual timestep treecode will get better performance for much larger number of processors. Thus, we can conclude that even with 1 M particles, GRAPE with simple direct summation will be faster than the most sophisticated algorithm on a general-purpose computer of the same price by a factor larger than 30.

4.3 Sophisticated algorithms on GRAPE

Now we proceed to the second question, namely whether reasonably efficient implementation of the neighbor scheme or treecode with individual timestep on GRAPE is possible or not. For the neighbor scheme, our opinion is that it is not impossible, but it will increase the design time by a fairly large amount (up to one year), to the effect that part of the gain in the performance will be cancelled by the longer development time. Also, since we gain a factor of 10 in speed, now we need a host computer that is 4 times faster than the host computer for simple direct summation. For system H2, this means we will need to use a 40-processor machine. A distributed memory machine like IBM SP-2 with 40 processors is clearly not too well suited to run the individual timestep algorithm. On the other hand, we can expect a shared memory multiprocessor machine such as SGI PowerChallenge or, more traditional Cray-C90, with the peak speed of 200 Gflops will cost less than 1 M dollars. That kind of machines is well suited for individual timestep algorithm.

Is it practical to implement a high-accuracy treecode on GRAPE? We do not really think so. There are several possibilities in implementing a high-accuracy treecode in hardware. The most straightforward one is the extension of the present HARP architecture, in which one simply calculates the higher-order multipole moments as well as the center-of-mass contribution. The tree construction and tree traversing is performed on the host computer. In this case, the GRAPE system that is more than 1000 times faster than the host will not be very useful, and the speedup over the speed of the host computer will be limited to a factor of 100. As we demonstrated above, the treecode on the general-purpose computer with the price same as that of system H2 is about 30 times slower than the direct summation on H2. So, the treecode with that host of 3 Tflops peak speed will be factor 3 faster than system H2. This gain is too small and the neighbor scheme is more practical. In principle, we could move more work to GRAPE from the host. First, to let the GRAPE do the tree traversing is not impossible, but the gain is small, another factor three or so, since you still have to construct the tree on the host computer. To move the tree construction itself to the special-purpose computer is, in our opinion, too complicated to be considered as practical possibility.

5 Collisionless N-body Systems.

5.1 Treecode

In this section, we will give rough estimate of the speed one can get from the treecode. As discussed in Makino (1991), if the speed of GRAPE is more than 1000 times faster than the speed of the host computer, the CPU time of the treecode on GRAPE is determined solely by the speed of the host computer. Therefore, in order to have a reasonable balance between the load of the host and that of the GRAPE, the host computer should have the speed at least 1/1000 of that of GRAPE.

The speed of the host computer, as a function of the time and price, would be expressed as

$$P_{host} = 10^{-3} \times 2^{(Y-1994)/1.5} B_{host}, \tag{8}$$

where B_{host} is the price in M dollars. To have a balanced system, we pose the condition

$$P_{grape} = 1000 P_{host}. \tag{9}$$

Combining equations 6, 8 and 9, we have

$$B_{host} = \begin{cases} 4 \times 2^{(Y-1994)/3}(B_{grape} - 0.3) & \text{(L-systems)}, \\ 2^{(Y-1994)/3}(B_{grape} - 1) & \text{(H-systems)}. \end{cases} \tag{10}$$

Equation 10 implies that the cost of the host computer will become more important in the future, because of the difference in the doubling time of the performance between the host and GRAPE. In three years, the speed of GRAPE will be increased by a factor of 8 for the same cost. Since we fix the ratio between the speed of the GRAPE and host, we will need host computer which is 8 times faster than the host of three years before. Since the speed of general purpose computer increases only by a factor of four in three years, we will have to spend twice as much money for the host. In other words, GRAPE will be less expensive in the future.

Table 4 gives the cost and performance of this balanced system. The costs are in M dollars. Note that this table is based on equation 8, which might be a bit too optimistic. In particular, if we are to construct really balanced system, the investment to the main memory and disk would roughly double the cost of the host computer.

Table 4: Performance: balanced treecode

System	Year	Total cost	Host cost	N	GRAPE speed
L1	1996	1	0.6	2×10^8	1.6Tflops
L2	1996	5	4	10^9	10 Tflops
H1	1997	7	4	10^9	10 Tflops
H1'	1997	3	1.3	3×10^8	3 Tflops
H2	2000	46	36	5×10^{10}	0.5 Pflops
H2'	2000	10	7	10^{10}	100 Tflops

5.2 P^3M calculation

So far, implementation of P^3M algorithm on GRAPE-3AF has been not overly successful. While direct summation provides the speed 10-20 times Cray YMP and treecode 3-5 times YMP, P^3M gave the result less than YMP. There are two reasons for this. The first one is that GRAPE-3AF is not efficient in calculation the PP force, since PP force requires a cutoff which is approximated by 3 or more superimposed forces on GRAPE-3AF. The second reason is that PM force is done entirely on the frontend. Since the computational cost of PM part is of the order of 1-10% on a general-purpose computer, this limits the asymptotic gain to be factor of 10-100.

The first limitation can be solved by (a) putting faster GRAPE or (b) using the GRAPE hardware that can handle the force with cutoff, which is relatively easy to implement if the required accuracy is not very high.

The second problem is more tricky. In principle, one can use coarser grid to reduce the calculation cost of PM part. For example, if one use 32^3 grid instead of 64^3 grid, the cost of FFT becomes nearly a factor of 10 less. This, however, have two limitations. The first one is that even if we reduce the grid point, the cost of PM force cannot go below $O(N)$ with fairly large coefficient. The second one is the accuracy of the force. PM force has some grid dependence, while PP force does not. Thus, it is not clear whether we can use arbitrary large grid size. In the following, we give a very rough guess for the calculation cost, ignoring the second problem. Let us assume that the calculation time of PM part is expressed by the following formula:

$$T_{PM} = C_1 N + C_2 M \log M, \qquad (11)$$

Where C_1 and C_2 are constants, N is the number of particles, and M is the number of grid points. The calculation proportional to the number particles involves the interpolation of the forces from the values in the cell, and assignments of the mass of the particles to the grid points. I guess it would be reasonable to assume these operations requires of the order of 100 floating point operations. FFT requires some 10 floating point operations in each stage. So assuming 30 (to take into account the fact we are dealing with 3D FFT), we have

$$T_{PM} = t_{host}(100N + 30M \log M), \tag{12}$$

where t_{host} is the time to perform one floating point operation on host.

The cost of the PP force is given by

$$T_{PP} = \frac{9^3 N^2}{M} t_{grape}, \tag{13}$$

where t_{grape} is the time to calculate one particle-particle force on GRAPE. Here we neglect the time of the communication and assumed that the cut-off length of the PP force is 3 times the PM grid size. Thus, to pose equal loads on GRAPE and the host, we have

$$t_{grape} = t_{host} \left(\frac{0.14M}{N} + \frac{0.04M^2 \log M}{N^2} \right). \tag{14}$$

For example, to have a good speed balance for $N/M = 10$, we need $t_{host}/t_{grape} \sim 50$. Assuming that one force calculation cost about 30 floating point operations, a GRAPE which is 1500 times faster than the host would provide good balance for $N/M = 10$. Note that for this ratio between N and M, the $O(M \log M)$ contribution to the PM force is already smaller than $O(N)$ contribution. Thus further decrease in M is not very meaningful. For the case of calculation without GRAPE, the optimum point is around $N/M = 0.1$. This is close to the typically used value(8 more cells than particles). Compared with this, calculation with GRAPE would be faster by a factor of 30.

For the above cost analysis we assumed that the distribution of particles is homogeneous, which is far from true in the late stage of the simulation. The effect of the increase in the cost depends on various things, in particular on how far one wants to follow the evolution. In practice, a factor of 30 increase in the cost of PP force is not unusual. If we adapt this factor of 30 increase, we can utilize a GRAPE with the speed 50,000 times faster than the host, using the ratio $N/M = 10$. If we optimize the grid size for the PP calculation on the general purpose computer ($t_{grape} = 30t_{host}$), we have $N/M = 0.02$. The speedup is now a factor of 100 or around. If we have the GRAPE with 1,000 times faster than the host, the optimum is now $N/M = 1$, and the speedup is of the order of 30. Note that this "optimum" value of $N/M = 0.02$ for general-purpose computer is not practical since it requires too large memory. If we use $N/M = 0.1$, the calculation cost on general purpose computer becomes 3 times more than that of the optimal one.

The asymptotic speed of P^3M code on GRAPE is about a factor of 3-10 faster than that of the treecode on GRAPE, since the number of floating point operation on host is smaller by that factor. Thus, we have table 5 for the speed of the P^3M code.

Table 5: Performance: balanced P^3M

System	Year	Total cost	Host cost	N	GRAPE speed
L1	1996	1	0.6	10^9	1.6Tflops
L2	1996	5	4	5×10^9	10 Tflops
H1	1997	7	4	5×10^9	10 Tflops
H1'	1997	3	1.3	1.5×10^9	3 Tflops
H2	2000	46	36	3×10^{11}	0.5 Pflops
H2'	2000	10	7	5×10^{10}	100 Tflops

The actual performance observed by Brieu, et al. (1994) is consistent with the above analysis. In their case, the PP force is 20 times more expensive than the PM force on a SS-10, for the ratio of $N/M = 0.125$. This is about the same as what we expected to have. With GRAPE, they used the same $N/M = 0.125$, which is close to optimal since the GRAPE is only 20 times faster than the host. This rather slow speed is partly due to the fact that the communication speed between the GRAPE and the host is slow (about 1 M word/s), and partly due to the fact that they have to call GRAPE 3 times to obtain reasonable cut-off in force. The overall speedup factor is about 10, for this factor 20 faster GRAPE. This seemingly good result is, of course, because their grid size is suboptimal for the calculation of PP force on SS-10.

6 SPH simulations.

For the authors, the calculation cost of SPH simulation is difficult to estimate, since we have no real experience of running any SPH simulations. The number of floating point operations needed for one SPH force, the average number of neighbors, the number of timesteps, the effect of the individual timestep are all important factors at which we have no handle. Here, we give rough estimate based on early experiments by Umemura et al. (1993), for which we do want to be corrected by experts.

We consider the simulation of of the formation of the cluster of galaxies, in which the formation of the cluster and the galaxies within it are followed in one calculation. We assume that we do use the individual timestep. In the case of the individual timestep, the calculation cost to follow the one half-mass crossing time of a single galaxy does not depend strongly on the structure. We assume this is true even for SPH particles.

If we are to follow the formation of the galaxies in the cosmological simulation, it means we have to follow of the order of 100 crossing times, resulting in about 10^4 timesteps.

Table 6: Performance: SPH calculation on workstation

Year	N	Speed of GRAPE
1994	5×10^5	125 Gflops
1997	$2 - 20 \times 10^6$	2-200 Tflops
2000	$1 - 10 \times 10^7$	0.03-3 Pflops

As the algorithm to calculate the gravitational force, we consider both the direct summation and the treecode. For the speed of the host computer, we make the similar assumption as we made for the treecode. The CPU time per SPH particle per timestep is given by

$$T_{sph} = N_{nb} C_{sph} t_{host}, \tag{15}$$

where N_{nb} is the number of neighbor particles, C_{sph} is the number of floating point operations to evaluate one pairwise interaction, t_{host} is the time to perform one floating point operation on host computer. As a rough guess, we adopt $N_{nb} = 50$ and $C_{sph} = 100$. For present-day high-performance workstation with the effective speed of 40 Mflops, we have $T_{sph} = 125\mu$s. Table 6 gives the number of particles can be handled on a high-performance workstation, for CPU time of one week, assuming again the average number of timesteps is 10,000. We also listed the necessary speed of GRAPE hardware, in the case of the direct summation, to finish the force calculation in the same time.

To one's surprise, the requirement for the speed of the GRAPE hardware is *huge*, easily exceeding the speed provided by the most ambitious project we considered (System H2). The reason for this excessive requirement is of course the fact that we assumed the direct summation. Now we have to consider the use of treecode, P^3M or Ewald method.

There exist several independent implementation of the individual timestep algorithm and treecode. TREESPH by Hernquist and Katz (1989) is probably the simplest approach, in which one construct the tree structure from scratch at each blockstep. This scheme is, unfortunately, cannot be used with GRAPE, since the cost of reconstruction of the tree structure at each blockstep is too high. In fact, if the ratio between the minimum stepsize and average stepsize is around 100, the cost of the tree construction becomes comparable to that of the force calculation, even if the force calculation is performed on the host computer.

The most sophisticated approach is what is described in McMillan and Aarseth (1993), in which one uses the predictor for the forces from nodes, as well as the forces from particles, and rearranges the tree structure locally, when one particle goes out of its cell. This scheme still would suffer fairly large performance loss, compared to the shared timestep code, since the present implementation of the treecode on GRAPE take advantage of the fact that we can calculate the force on many particles in parallel. To get a rough idea, let us simply assume the

Table 7: Performance: treesph

System	Year	Total cost	Host cost	N
L1	1996	1	0.6	2×10^7
L2	1996	5	4	10^8
H1	1997	7	4	10^9
H1'	1997	3	1.3	3×10^8
H2	2000	46	36	5×10^9
H2'	2000	10	7	10^9

loss of the performance in speed to be a factor of 10. Then we have table 7 for the number of particles we can handle with tree.

Here, we neglected the calculation cost of SPH interactions, since the cost of gravitational force calculation by tree will be larger than the cost of SPH force calculation. For a shared timestep treecode, we assumed $T_{tree} = 10^3 t_{host}$, where T_{tree} is the calculation time on host per particle per timestep. For individual timestep, we adopt

$$T_{tree,ind} = 10^4 t_{host}. \tag{16}$$

Thus, we have $T_{tree,ind} \sim 2T_{sph}$. We could think of a combination of SPH, individual timestep and P^3M as well. If such a algorithm can be implemented, The calculation cost of the gravitational interaction on the host computer will be reduced by a factor of 10. Since the SPH cost is unchanged, however, the gain in overall speed will be about a factor of two.

7 Summary and Discussion

The main conclusion one can derive from the above analysis is that by using GRAPE hardware, one can get factor 30-300 performance gain compared to the performance of the most optimistic prediction of the most sophisticated algorithm on the general-purpose computer for foreseeable future, for all of the collisional stellar dynamics, treecode, and P^3M code. If we include SPH, the gain will be somewhat less, but not by a large factor.

However, the hardwares optimal for these problems are rather different. In the case of the collisional stellar dynamics, We need a high accuracy machine, and most of money should go to GRAPE, since the total speed is limited by the speed of GRAPE. For treecode and P^3M, on the contrary, we do not need very high accuracy and most of money should go to the host, since the host limits the performance. If we compare the tree and P^3M, we found that P^3M requires somewhat more speed on GRAPE.

Table 8: Performance: balanced system

System	Year	Total cost	Host cost	N_{gc}	N_{tree}
H1	1997	6	3	2×10^5	7×10^8
H2	2000	20	10	10^6	10^{10}

Of course, we can interpret the above statement in a slightly different way: If we design a system optimal for collisional stellar dynamics, a slight increase in the investment of money on the host computer would vastly improve the performance of the collisionless code. The other way around is also true: If we design a system optimal for collisionless calculation, to change the GRAPE part so that it is usable for collisional code requires relatively modest increase in the total cost. Table 8 gives the performance of the system in which we spend half of the money to the host computer and half of the money to GRAPE. We believe this is the best compromise of the requirements from collisional and collisionless simulations.

In practice, we have to take into account the available human resources, grant, and the necessary time to develop the system. One might think it makes sense to develop only one hardware to minimize the necessary human resource. In practice, however, it might not be true. As we stated before, a high-accuracy GRAPE is more difficult to develop than a low-accuracy GRAPE. Thus, the development of the low-accuracy system is very useful as the training of people who will then develop a high-accuracy system.

From the point of view of the grant, the minimum grant for a low accuracy system is significantly smaller than that for the high-accuracy system. If to ask first, say, a few hundred thousand dollars and then, after some scientific result, to ask for several million dollars is easier than to ask for several million dollars right from the beginning, that would be a good reason to develop a low-accuracy machine.

Acknowledgment

This note is originated from Douglas Heggie's remark in the First GRAPE Workshop, held in MPI für Astrophysik, Garching, Germany. We are most grateful to Douglas, without his suggestion this would never be written. We also thank all attendees of GRAPE workshop, in particular, Matthias Steinmetz, Rainer Spurzem, Lia Atthanasoula and Tom Theuns. Their comments during the workshop was most valuable. This work supported in part by Joint JSPS-??? grant. JM would like to thank again to Douglas Heggie for his kind hospitality during my visit to Edinburgh, in which the main part of this note is written.

References

Brieu, P., Summers, F, and Ostriker, J.P., 1994, submitted to *Ap. J. Suppl.*

Ebisuzaki, T., Makino, J., Fukishige, T., Taiji, M., Sugimoto, D., Ito, T., and Okumura, S. K., *Publ. Astron. Soc. Japan*, 1993, **45**, 269.

Hernquist, L. and Katz, N., *Astrophys. J. Suppl.*, 1989,**70**, 419.

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

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

Makino, J. and Hut, P., *Computer Physics Reports*, 1989, **9**, 199.

Makino, J. and Hut, P., *Ap. J.*, 1990, **365**, 208.

McMillan, S. L. W. and Aarseth, S. J., *Astrophys. J.*, 1993, **414**, 200.

Umemura, M., Fukushige, T., Makino, J., Ebisuzaki, T., Sugimoto, D., Turner, E. L., and Loeb, A., *Publ. Astron. Soc. J.*, 1993, **45**, 311-320.

A Machines available now or in near future

In the main text, we described the machines that we can develop in the timescale of 3-5 years. In this appendix, we briefly summarize the machine we can manufacture the copies right now or less than a year.

A.1 GRAPE-3AF

One PCB board of GRAPE-3AF currently provide the speed of 5 Gflops for 10K dollars (excluding the VAT, import tax, handling fee of the trading company and so forth...). It can handle the collisionless simulation quite well, as far as free boundary (isolated system) is concerned. Its accuracy is around 2% for pairwise interaction. For many problems, the performance of the total system is not limited by the speed of GRAPE-3AF, but by either the speed of the host computer (SS-10) or the communication (VME through Sbus-VMEbus adapter). We are thinking of developing a modified version of GRAPE-3A with faster interface to host computer (most likely a PCI bus).

A.2 GRAPE-4

One PCB board of GRAPE-4 will provide the speed of 30 Gflops. The cost of the minimum configuration of GRAPE-4 (one PCB board) will be of the order of 50-100K dollars (again excluding the VAT, import tax, handling fee of the trading company and so forth...). It handles both the collisionless and collisional simulation quite well. Right now, the only host computer that can be connected to GRAPE-4 is DEC Alpha AXP model 3000/x00, since the interface is designed to plug in to its TURBOchannel interface.

A smaller version of GRAPE-4, sometimes called HARP-3, is also available, for the cost of 40K dollars for 4-chip model (3 Gflops peak).

A.3 MD-GRAPE

We have completed the MD-GRAPE system, which can handle arbitrary force law with very high accuracy, and therefore is ideal for P^3M or other algorithm for periodic boundary. The first generation of MD-GRAPE system will have similar performance as HARP-3 but hopefully somewhat less expensive.

Though the peak performance of an MD-GRAPE board will be lower than that of the GRAPE-3AF board, one can expect a large performance improvement when applied to P^3M. The first reason is that MD-GRAPE board has the hardware support for the linked-list. With GRAPE-3A, in order to calculate the force from the 27 neighbor cells, one has to send the positions and masses of particles in all these 27 cells. Thus the amount of the communication is proportional to

$27N$. With MD-GRAPE, one can send the particles in cells as block, and then specify which cells one wants to interact with. Thus, the communication is now proportional to $3N$ (one memory write, one register write and read), resulting in a factor of 10 improvement in the communication speed. These improvement will give about a factor of 4 in the overall speed (see section 5.2).

For the first-generation MD-GRAPE, we used the VME interface as the interface to the host. This choice naturally limits the communication speed to be rather low. We plan to develop a PCI-based system, which will have at least 10 times faster communication speed. With PCI interface, a MD-GRAPE with the speed 100 Gflops can be efficiently used. Such a system will cost about 0.2 million dollars, and will take about one year to develop.