SMOOTHED PARTICLE HYDRODYNAMICS WITH GRAPE AND PARALLEL VIRTUAL MACHINE

NAOHITO NAKASATO,1 MASAO MORI,2 AND KEN'ICHI NOMOTO1,3

Received 1996 May 20; accepted 1997 March 6

ABSTRACT

We have developed Remote-GRAPE (for “GRAvity PipE”), a subroutine library to be used with the special purpose computer GRAPE-3A. GRAPE-3A can efficiently calculate gravitational forces between particles and construct neighbor lists. All other calculations are performed on the host workstation (WS), which is directly connected to GRAPE. The use of GRAPE for smoothed particle hydrodynamics (GRAPE-SPH) can in principle greatly speed up the calculations on the WS. However, the current bottleneck of GRAPE-SPH is that its performance is limited by the speed of the host WS. To solve this problem, we implement Remote-GRAPE; it allows us to run applications that use GRAPE-3A hardware on significantly faster computers than the physical host WS. Thus, we can take advantage of fast computers even though they cannot physically be connected to GRAPE. The Remote-GRAPE system is implemented on Parallel Virtual Machine. The details of implementation are described.

We analyze the performance of Remote-GRAPE and obtain the following results. (1) When Remote-GRAPE is used to calculate gravitational forces only, the overhead due to the network decreases according to the number of particles (20%–40% of total time). (2) When Remote-GRAPE is used to calculate gravity and neighbor lists, the overhead due to the network does not occupy a large fraction of total time but only ~20%–30%, because the computation required on the slave machine is very large because of the properties of the GRAPE system. (3) We also compare the performance of Remote-GRAPE with the tree method. The tree method requires more time for a higher degree of clustering, while the required time for Remote-GRAPE does not much depend on it.

We then analyze the performance of GRAPE-SPH with Remote-GRAPE. The performance of Remote-GRAPE is about 4 times faster than GRAPE-SPH with usual GRAPE for our configuration. Using Remote-GRAPE, we can calculate the GRAPE part and other parts in parallel. This parallel method leads to a further speedup of our SPH code. We estimate how the performance of Remote-GRAPE depends on the configuration. We also show that the performance of Remote-GRAPE can be further improved.

Subject headings: galaxies: formation — galaxies: star clusters — hydrodynamics — methods: numerical

1. INTRODUCTION

The smoothed particle hydrodynamics method (SPH) is widely used to calculate three-dimensional hydrodynamics with a Lagrange scheme (Lucy 1977; Gingold & Monaghan 1977). It has been applied to many astrophysical problems. Because of its Lagrangian nature, it is suitable for problems that have large density contrasts, e.g., the formation of galaxies (Evrard 1988; Hernquist & Katz 1989; Umemura 1993; Steinmetz & Mueller 1994) or a cloud-cloud collision (Lattanzio et al. 1985; Habe & Ohta 1992). In these calculations, in order to treat star particles and/or dark matter with gas particles, we have to solve SPH with collisionless particles (an \(N\)-body system). Various codes have been developed to combine SPH and an \(N\)-body system. In these codes, gravitational forces are calculated in many different ways, such as direct summations, particle-particle/particle-mesh methods (Evrard 1988), tree methods (Hernquist & Katz 1989; Benz et al. 1990), and the method using the special purpose computer GRAPE (Umemura et al. 1993; Steinmetz 1996).

GRAPE (GRAvity PipE) is a special purpose computer for efficiently calculating gravitational force and potential (Sugimoto et al. 1990). We need a host computer, which is connected to GRAPE board, to control it and conduct other calculations (e.g., time integration). The use of GRAPE for SPH simulations (hereafter GRAPE-SPH) has many advantages, since GRAPE can not only calculate gravitational force and potential but also construct lists of neighbor particles in a short time. In SPH simulations we need the lists of neighbor particles to calculate hydrodynamic quantities. Searching neighbor particles with GRAPE is much more efficient than a direct search on the host computer. With GRAPE-SPH, therefore, we need to calculate only the pure hydrodynamic part of SPH on the host computer.

Although the cost of the \(N\)-body part of SPH can be significantly reduced by using GRAPE, the speed of the hydrodynamic part of the code is limited by the speed of the host computer. The speed of workstations (WSs) is being rapidly improved. To take full advantage of GRAPE for SPH calculations, we have to use a state-of-the-art WS as a host. If the host WS of GRAPE is much slower than a fast WS now available, the total performance of GRAPE-SPH...
is lower than that of the SPH simulation on the fast WS without GRAPE. Of course, by developing new interfaces to new WSs, we could solve this problem. This approach, however, requires much human time. Moreover, during the development of the new interface, an ever-newer WS with a different interface might become available.

Here we take a different, novel approach to solve this problem. We use Parallel Virtual Machine (PVM; Geist et al. 1994), which is one of the most popular message-passing systems in parallel computing, to connect the WS that is directly connected to a GRAPE board to another fast machine. The SPH part is performed on the fast machine. In fact, all simulation code is run on the fast machine, and the WS connected to the GRAPE board serves essentially as the intelligent communication interface. Figures 1 and 2 show the present GRAPE system and our new approach, respectively. In our new approach, the combination of the WS and GRAPE (Fig. 1) behave as the “remote GRAPE” system, which is connected directly to the local area network (LAN). Thus, any computer on the LAN can be used as the host (remote host) of GRAPE. We construct the library named “Remote-GRAPE.” In this paper we present the implementation of Remote-GRAPE and discuss its performance. In §2 we summarize the GRAPE system and the bottleneck of GRAPE-SPH. In §3 we describe how Remote-GRAPE is implemented, analyze its performance, and compare it with the original GRAPE system and other schemes. In §4 we show the performance of our GRAPE-SPH code and discuss possible improvement of our code. In §5 conclusions are summarized.

2. GRAPE AND GRAPE-SPH

The GRAPE-3AF system (Okumura et al. 1993) has eight processors on one board, and the GRAPE board is connected to a host computer via a VME (Versa Module Europe) bus. The processor chip is designed to calculate gravitational force with a Plummer softening, i.e.,

$$f(r_i) = -G \sum_j m_j \frac{r_i - r_j}{(r_i^2 + \epsilon^2)^{1.5}}.$$  \hfill (1)

During the force calculation, GRAPE-3AF can construct neighbor lists simultaneously (Fukushige et al. 1991; Okumura et al. 1993). The size of the buffer for neighbor lists is $1024 \times 4$ bytes, which limits the maximum number of neighbors per board. The schematic diagram of the GRAPE system is shown in Figure 1.

We present the general performance of the GRAPE system in Figure 3. These figures show the relation between the CPU time and the number of particles, $N$, for different numbers of GRAPE boards. In our site, the GRAPE boards are connected to a Sun SPARCclassic (hereafter Classic) via an Aval Data SVA-100 VME interface. Figure 3 (left) shows the case that calculates gravity only (case G), while Figure 3

![Diagram](image-url)

Fig. 1.—Schematic diagram of the GRAPE system

![Diagram](image-url)

Fig. 2.—The concept of PVM and our library

![Graph](image-url)

Fig. 3.—Left: CPU time as a function of the number of particles $N$ in calculating gravitational force only (case G). The five solid lines correspond to the number of GRAPE boards from 1 to 5 (top to bottom). Right: Same as Fig. 3 (left), but for calculating both gravitational force and neighbor lists (case GN). In both cases, the GRAPE boards are connected to a SPARCclassic.
(right) shows the case that calculates both gravity and neighbor lists (case GN). The particles used in these experiments are randomly distributed within the calculation region (see Fig. 7 [left]), and the size of the region changes with \( N \) to make the number density almost constant. The average number of neighbor particles is almost constant (\( \sim 40 \)) for any \( N \).

In case G, the performance is dramatically improved by increasing the number of boards. In case GN, however, the use of a larger number of boards does not significantly improve the performance. This is because the speed of the VME bus is too slow to read neighbor lists for large \( N \) with many boards, and the calculations required to construct neighbor lists on the host computer is rather costly work for our machine. We compare the performances of the GRAPE system with different host computers. We use a Classic (SPECint92/SPECfp92: 26.4/21.0, which represent the integer and floating performance of the machine) and a Sun SPARCstation 10/40 (hereafter SS-10; SPECint92/SPECfp92: 50.2/60.2) with the same interface and three GRAPE boards. Figure 4 shows cases G and GN, respectively, where the solid and dotted lines represent the cases of the Classic and the SS-10, respectively. For case GN, the computation on the host computer (in short, construction of neighbor lists) is longer than for case G. Thus, the speed of the host computer apparently affects the performance.

In SPH, physical quantities at one position are calculated by smoothly averaging over neighbor particles. Thus, SPH simulations are essentially equivalent to an \( N \)-body simulation with complex, short-range “force.” In GRAPE-SPH we use GRAPE for calculating gravitational force and searching neighbor particles. Steinmetz (1996) summarized the performance of GRAPE-SPH in his Table 1 for one GRAPE-3AF board and a Sun SPARCstation 10 as a host computer. He showed that about 80% (\( N \sim 60,000 \)) of computing time is spent on the hydrodynamic and miscellaneous calculations and only \( \sim 20 \% \) is spent on the GRAPE part. This implies that higher performance is obtained if we can use a faster WS to calculate the SPH part.

3. IMPLEMENTATION OF REMOTE-GRAPE AND ITS PERFORMANCE

3.1. The Motivation to Develop Remote-GRAPE

The current bottleneck of the performance of GRAPE-SPH is the slow host computer, while the choice of the host computer is limited by the interface of GRAPE-3AF. Our approach to solving this problem is to develop software that allows us to use any computer connected to the network as a host of GRAPE. This makes it possible to use a host computer of any vendor and of any performance. For this purpose, we adopt Parallel Virtual Machine as a basis of the code.

Parallel Virtual Machine is software that enables us to develop a highly portable parallel program with the message-passing style (Geist et al. 1994). We can solve large computational problems by using a heterogeneous collection of computers as a single parallel computer. In PVM, subroutines to send and receive messages can be called from C or FORTRAN. Here the message is the data that are needed for solving the problem. We name the library “Remote-GRAPE.”

3.2. Implementation of Remote-GRAPE

We call the machine on which the main program runs a master machine or master side and the machine that is connected physically to the GRAPE board a slave machine or slave side. Also, we call the program that runs on the master machine a master task and the program that runs on the slave machine a slave task. Figure 2 shows the concept of our method.

In the library of the GRAPE system, a simple example code looks like the following (see Makino & Funato 1993 for details):

```c
real scale_leng, scale_mass, eps2
real real x(3,nall), mass(nall), acc(3,nall), pot(nall)
...
call g3setscales(scale_leng, scale_mass)
call g3seteps2(eps2)
call g3setn(nall)
do j = 1, nall
  call g3setxj(j-1,x(1,j))
call g3setm(j-1,mass(j))
endo
call g3frc(x(1,i), acc(1,i), pot(i), 8)
endo
```

In the first three lines we set the scales of length and mass, the softening parameter, and the number of particles.
respectively. Then we set the positions and mass of the particles in the first do loop and call g3frc() to get gravitational force and potential in the second do loop.

When we are sending a message in PVM, we may add an identifier to the message. This makes it possible for the program, which is receiving the message, to distinguish one message from another. We adopt this mechanism at the slave task to select the routine that the master task wants to call. For example, if the master task wants to call g3setni(), it has to send the message, which contains the number of particles, to the slave task with previously the defined tag for g3setni(). The slave task will receive the message from the master task and examine the tag, and then it will call a suitable routine, g3setni(), in this case.

With this method we can call essentially any routine in the original GRAPE library from any machine. If small messages are sent many times, however, it requires a large computational and network cost. For example, it is more efficient to make a new routine that gathers many calls than to call g3setxj() separately for each particle. Thus, in our library, we send and receive the data for many particles in one message. Moreover, we compile the calling of many routines like the second do loop into one routine. A simple example of the code using Remote-GRAPE is as follows:

```c
real scale_leng, scale_mass, eps2
real x(3,nall), mass(nall), h(nall), acc(3,nall), pot(nall)
integer*2 list(nall+100)
integer begin(nall)
...
call g3setscales(scale_leng, scale_mass)
call g3setsteps2(eps2)
call g3setmass(nall, mass)
call g3setdata(nall,x(1,1), x(2,1), x(3,1))
call g3seth2all(nall, h)
call g3docalculation(1,nall,1, nall, 2)
call g3getgravity(nall, acc(1,1), acc(2,1), acc(3,1), pot)
call g3getneighbor(nall, list, begin)
```

The first two routines are the same as the original GRAPE library. The next three routines, g3setmass(), g3setdata(), and g3seth2all set, respectively, the mass, position, and neighbor searching radius of all the particles. These routines send only the data, and the slave task stores the data in an internal working space. Then calling g3docalculation() starts calculations. At this time, the slave task performs the codes that are similar to the original example. Finally, we get gravitational force, potential energy, and neighbor lists.

The arrays list and begin are neighbor lists. The array list is the whole list of neighbor particles, and array begin is a list of the index that represents the first member of a neighbor list of a particle. For example, if the first part of begin looks like (1, 24, 47, 58, ...), elements 1–23 of the array list are the neighbor particles of particle 1, elements 24–46 are the neighbors of particle 2, and so on. When an overflow of neighbor lists occurs in the slave task routine of constructing neighbor lists, we use the same method as adopted by Steinmetz (1996).

Since the throughput of the message transfer is rather low, we minimize the amount of data to be transferred. Though the original GRAPE library has both single-precision and double-precision routines, the accuracy of GRAPE itself is limited approximately to single precision. Thus, to get the best performance of Remote-GRAPE, we use all messages in our library in single precision. In case we need double precision, we set a double-to-single-precision conversion factor, because the data transfer is always done in single precision. The required cost to convert single to double precision is small. For the same reason, we use unsigned short (2 byte) integers for neighbor lists. When neighbor lists are used, the number of particles is limited to ~ 64,000.

### 3.3. The Performance of Remote-GRAPE

Here we analyze the performance of Remote-GRAPE. The benefit of using Remote-GRAPE over usual GRAPE is that Remote-GRAPE is almost machine independent, as far as the machine is supported by PVM. In compensation, we have to transfer the data across the LAN, which is usually slower than the bus that connects the GRAPE board and the host computer. Here we compare the performance of Remote-GRAPE and usual GRAPE.

In our site, the master machine is a DEC AlphaStation 600 5/266 (SPECint92/SPECfp92: 292.8/433.5), and the slave machine is a Sun SPARCclassic with five GRAPE-3AF boards. Hereafter, all values are obtained from timing on these machines, and the timing is elapsed time, not CPU time. The reason why we present elapsed time is that if we use usual routines or commands to get CPU time in the program using the Remote-GRAPE library, it returns CPU time on only the master side. This time does not include the time spent on the slave side and to transfer data over the LAN, so it is meaningless for our purpose. When we examine the performance, there are a few or no active processes except our interested program. Also, we use both Ethernet (10 Mbps) and Fast-Ethernet (100 Mbps) as the backbone of the LAN. The difference between the two backbones is also analyzed. We summarize test cases for GRAPE and Remote-GRAPE in Table 1.

#### 3.3.1. Case RG: Gravity Only

For _N_ particles, the time require to calculate gravitational forces by GRAPE is expressed as

\[ t_G = a_1 N^2 + (a_2 + a_3)N \]

where _a_1 is the time required to calculate the gravity of one particle, _a_2 is the time to transfer data via the VME bus, and _a_3 is the time spent for miscellaneous calculations on the slave machine. The time required to transfer messages via the LAN is expressed as

\[ t_{LAN} = a_4 N \]

where _a_4 is the time required to calculate the gravity of one particle. Using Remote-GRAPE, the total required time is

| Case | Gravity* | Neighbor Lists* | Remoteb | Sectionc |
|------|----------|-----------------|---------|----------|
| G…….. | yes      | no              | no      | 2        |
| GN…….. | yes      | yes             | no      | 2        |
| RG…….. | yes      | no              | yes     | 3.3.1    |
| RGN….. | yes      | yes             | yes     | 3.3.2    |

* This column indicates whether neighbor lists (N) are calculated in addition to gravity (G).

b This column indicates whether Remote-GRAPE is used.

c This column indicates the section in which the test case is discussed.
We estimate the values of \( a_1, a_2, \) and \( a_3 \) as

\[
a_1 = 5.0 \times 10^{-8}/N_{\text{chip}} \approx 1.3 \times 10^{-9} \text{ s},
\]

\[
a_2 = 2.0 \times 10^{-5} \text{ s},
\]

\[
a_3 = 4.0 \times 10^{-5} \text{ s},
\]

where \( N_{\text{chip}} = 40 \) is the number of chips and \( a_2 \) and \( a_3 \) depend on the speed of the slave machine (see Okumura et al. 1993; but the clock period of the GRAPE-3AF chip is 20 MHz). For calculating gravity forces only, the master task has to send positions of particles (3 \( \times \) 4\( N \)) and masses (1 \( \times \) 4\( N \) bytes), and then the slave task has to send gravity forces (3 \( \times \) 4\( N \) bytes) and potential energies (1 \( \times \) 4\( N \) bytes). Thus we estimate

\[
a_4 = 32/r_{\text{LAN}},
\]

where \( r_{\text{LAN}} \) is the effective transfer rate of the LAN, which depends on the size of the message.

The results of our experiments are given in Figure 5. Figure 5 (left) shows the relation between the elapsed time and the number of particles, \( N \), where the solid, dotted, and dashed lines represent \( r_{\text{LAN}} \) for 100 Mbps, \( t_{\text{LAN}} \) for 10 Mbps, and \( t_{\text{G}} \), respectively. We find that \( r_{\text{LAN}} \) for this figure is 500--1000 Kbytes s\(^{-1} \) for 100 Mbps and 500--700 Kbytes s\(^{-1} \) for 10 Mbps. For 100 Mbps, we cannot get as high a performance as previously expected. This is because the effective transfer rate depends on several factors, e.g., the state of the LAN, the speed of both machines and their bus, and, most important, the setting and quality of the software (operating system and device driver). Figure 5 (right) shows the experimental relation between \( t_{\text{LAN}}/t_{\text{RG}} \) and \( N \) for 100 Mbps and 10 Mbps by the solid and dotted lines, respectively. The particles used in this experiment are the same as in Figure 7. It is clearly seen that the ratio decreases as \( N \) increases. This means that calculations with RemoteGRAPE take almost the same time as with usual GRAPE when \( N \) is large. Also in this case, the performance of the slave machine does not affect the performance of RemoteGRAPE, as shown in §2.

3.3.2. Case RGN: Gravity and Neighbor

To construct neighbor lists with Remote-GRAPE, we need additional calculations (see §2) and data transfer. In this case, the total required times are estimated as

\[
t_{\text{RGN}} = t_{\text{GN}} + t_{\text{LAN}},
\]

where

\[
t_{\text{GN}} = t_{\text{G}} + [(a_5 + a_6)N_{\text{neighbor}}]N,
\]

\[
t_{\text{LAN}} = t_{\text{LAN}} + a_7N.
\]

Here \( a_5 \) is the additional time required to read neighbor lists via the VME bus and \( a_6 \) is the time spent for construction of neighbor lists for one particle; \( a_7 \) is the time required to transfer neighbor lists via the LAN for one particle, which is expressed as

\[
a_7 = a_8 \times N_{\text{neighbor}},
\]

with \( N_{\text{neighbor}} \sim 30--100 \) being the mean number of neighbor particles for one particle.

The original GRAPE library reads the two buffers for neighbor lists twice each. Thus, it has to transfer 4 (times) \( N_{\text{neighbor}} \times 4 \) (one word) = 16\( N_{\text{neighbor}} \) bytes for one particle. If the effective transfer rate via the VME bus is 3.5 Mbytes s\(^{-1} \), \( a_5 \) is estimated as

\[
a_5 = 16/(3.5 \times 10^6) = 5.6 \times 10^{-5} \text{ s}.
\]

It is difficult to estimate \( a_6 \) because it depends highly on the integer performance of the slave machine. However, we may estimate \([(a_5 + a_6)N_{\text{neighbor}}]\) from the result of Figures 3 and 4 as \([(a_5 + a_6)N_{\text{neighbor}}] \sim 5.0 \times 10^{-4} \text{ s} \) for the Classic and \( \sim 2.4 \times 10^{-4} \text{ s} \) for the SS-10 with \( N_{\text{neighbor}} \sim 40 \). This implies that \( a_5 + a_6 \) depends linearly on the value of SPECint92, \( b_{\text{SPEC}} \), of the sample machines, as \((a_5 + a_6) \sim [(1.9 \times 10^{-5} - 2.5 \times 10^{-7})b_{\text{SPEC}}] \) s. This leads to

\[
a_6 \sim [(1.2 \times 10^{-5} - 2.5 \times 10^{-7})b_{\text{SPEC}}] \text{ s}.
\]

Since we use a 2 byte array for neighbor lists, we estimate

\[
a_8 = 2/r_{\text{LAN}} \text{ s}.
\]

This leads to \( a_7/a_8 = N_{\text{neighbor}}/16 \), so that \( t_{\text{LAN}} \) depends almost linearly on \( N_{\text{neighbor}} \).
The performance of Remote-GRAPE in making neighbor lists is shown in Figure 6. Figure 6 (left) shows the elapsed time as a function of \( N \), as in Figure 5 (left). Figure 6 (right) shows the experimental relation between \( t_{\text{LAN}}/t_{\text{RGN}} \) and \( N \). The particles used in this experiment are the same as in Figure 7. In our configurations, the fraction of time spent for communication between the master and the slave is about 30% of the total time for 100 Mbps and 40% for 10 Mbps. In other words, the overhead is acceptable. As discussed in §2, however, the time required on the slave machine (shown by the dotted line in Fig. 6 [right]) can be shortened by a factor of 2 or more by adopting a faster slave computer. Thus, whether using Remote-GRAPE is advantageous depends on the configuration and the nature of the simulation (see the next section).

### 3.3.3. Comparison with the Tree Method

Here we compare the performances of Remote-GRAPE and the tree method (Barnes & Hut 1986) on the master side. We use the tree code, which is developed by J. Barnes and freely available from on the Internet, with a little modification. The comparison on our machine between the two methods of elapsed time versus \( N \) is shown in Figure 8. The solid line shows the result of Remote-GRAPE, and other lines correspond to the tree methods. The attached number indicates the parameter \( \theta \) as defined in Barnes & Hut (1986).

The elapsed time of Remote-GRAPE is comparable to the tree method with \( \theta = 1.0 \).

We examine a possible advantage of Remote-GRAPE over the tree method, when the particles are highly clustered, so that the tree is deep. To simulate such a highly clustered situation, we distribute the particles as follows: (1) half of the particles are randomly distributed in a certain region, and (2) the other half are distributed in the region whose size is larger than the former by a factor of \( \alpha \) (see Fig. 7). The results for \( N = 50,000 \) are shown in Table 2, where \( \alpha = 1 \) corresponds to the case in Figure 8. It is seen that for the tree method the required time is longer for larger \( \alpha \) of

### Table 2

| \( \alpha \) | Remote-GRAPE | Tree* (\( \theta \)) |
|-------------|--------------|---------------------|
| 1.0         | 7.64         | 12.47 7.61 2.91     |
| 5.0         | 7.63         | 19.34 12.79 6.92    |
| 10.0        | 7.64         | 20.56 14.38 8.52    |

Note.—Elapsed time (s) is shown for \( N = 50,000 \).

* For larger \( \alpha \), clustering is larger.

** Three cases of \( \theta = 0.8, 1.0, \) and 1.5, where \( \theta \) is as defined in Barnes & Hut (1986).
more strongly clustered states because it takes longer to walk for a deeper tree. In contrast, the required time for Remote-GRAPE does not depend on $a$. For large $a$, the total computational cost for Remote-GRAPE is smaller than for the tree methods. This is important because even if the initial state is not so highly clustered, the state tends to get more and more strongly clustered as the system evolves.

4. PERFORMANCE OF SPH WITH REMOTE-GRAPE

4.1. General Performance

In order to see the actual advantage of SPH with Remote-GRAPE (hereafter case S-RGN), we make a comparison with the performance of SPH that uses usual GRAPE on the slave machine (case S-GN). The detailed description of our SPH code is presented in Mori et al. (1997). In Table 3, we show the computing time spent on various parts of the calculation per step, namely, gravity and neighbor ($t_{\text{GRAPE}}$), SPH ($t_{\text{SPH}}$), and miscellaneous ($t_{\text{misc}}$), where $t_{\text{GRAPE}}$ is the time to calculate gravity and neighbor lists, and $t_{\text{SPH}}$ and $t_{\text{misc}}$ are, respectively, those for SPH calculations and miscellaneous calculations. The time spent on the host computers ($t_{\text{SPH} + \text{misc}} = t_{\text{SPH}} + t_{\text{misc}}$), which has been the bottleneck of the performance in previous GRAPE-SPH (Steinmetz 1996), differs by a factor of 20 between two cases. In case S-RGN, we compare $t_{\text{GRAPE}}$, $t_{\text{SPH}}$, and $t_{\text{misc}}$ for larger $N$ in Figure 9. For the adopted configuration, the time spent for the gravity and neighbor part occupies the largest fraction of the total time (i.e., $\sim$ 80%), and the fractions of the three parts are almost independent of $N$. The large fraction of $t_{\text{GRAPE}}$ makes the total time per step $\sim$ 4 times shorter in case S-RGN than case S-GN despite the use of a host computer that is 20 times faster. This implies that in case S-RGN, the bottleneck of the performance is now the $t_{\text{GRAPE}}$ part rather than the $t_{\text{SPH}}$ part.

We summarize the results of §3.3.1 and 3.3.2 to present the $N$ dependence of $t_{\text{GRAPE}}$ in case S-RGN as follows:

$$t_{\text{GRAPE}} = t_{\text{RGN}} + t_{\text{LAN}},$$

$$= a_1 N^2 + [a_2 + a_3 + a_4 + (a_5 + a_6 + a_8)N_{\text{neighbor}}]N$$

$$+ 5.0 \times 10^{-8}N^2/N_{\text{chip}} + [6.0 \times 10^{-5}$$

$$+ (32 + 2N_{\text{neighbor}})/r_{\text{LAN}}$$

$$+ 16/r_{\text{BUS}} + [(1.2 \times 10^{-5} - 2.5 \times 10^{-7})b_{\text{SPEC}}]$$

$$\times N_{\text{neighbor}}]N s,$$

where $r_{\text{BUS}}$ is the speed of the bus in bytes per second. For $a_3$, we use the value of Okumura et al. (1993), though its machine dependence is not known. If we adopt a typical
value for our configuration, we get
\[ t_{\text{GRAPE}} = 1.3 \times 10^{-9}N^2 + 7.5 \times 10^{-4}N \text{ s} , \]  
which is consistent with the actual result of \( t_{\text{GRAPE}} \). The additional time for the calculation of neighbor lists is
\[ t_{\text{neighbor}} = ([a_2 + a_6] + a_3)N_{\text{neighbor}} N \]
\[ = [(5.0 \times 10^{-4}) + (1.4 \times 10^{-4})]N \]
\[ = 6.4 \times 10^{-4}N \text{ s} . \]  
Thus, we can estimate \( t_{\text{neighbor}}/t_{\text{GRAPE}} \sim 0.8 \) for \( N \sim 10,000 \)–60,000. Possible ways to overcome the bottleneck would be (1) calculating neighbor lists with a faster slave machine, (2) using Remote-GRAPE from time to time to get neighbor lists by adopting a larger neighbor radius, and (3) using another method to construct neighbor lists on the master side.

4.2. Parallel Method

The performance of GRAPE-SPH can be further improved if we compute gravity and neighbor lists and other parts in parallel by using Remote-GRAPE. We have implemented a parallel version of routines, where sample code is as follows:

\[
\text{call g3setscales(scale\_leng, scale\_mass)}
\]
\[
\text{call g3seteps2(eps2)} \text{ call g3setmass(nall, mass)}
\]
\[
\text{call g3setdata(nall, x(1,1), x(2,1), x(3,1))}
\]
\[
\text{call g3seth2all(nall, h)}
\]
\[
\text{call g3docalculationpara(1, nall, 1, nall, 2)}
\]

another calculation

\[
\text{call g3getgravityasync(nall, acc(1,1), acc(2,1), acc(3,1), pot)}
\]
\[
\text{call g3getneighborasync(nall, list, begin)}
\]

Three routines are replaced with parallel routines. First, all required data on the slave side are sent by the same routines as given in §3.2. The difference between g3docalculation and g3docalculationpara is when it returns. g3docalculation returns after all the calculation on the slave side is done, while g3docalculationpara returns after the slave task starts the calculation. The master side starts other calculations after g3docalculationpara returns. The two routines, which are denoted by “async” attached to those names, receive the message asynchronously. When we call g3getgravityasync, and if the message, which contains gravity, has not arrived, it waits for the arrival of the message. g3getneighborasync works in the same way.

As mentioned in §2, we need neighbor lists to compute physical quantities in SPH. In applying the parallel routines for the SPH calculation, we compare several methods of making neighbor lists. We summarize the test cases for GRAPE-SPH in Table 4.

4.2.1. Case PS-RG: Parallel SPH using Remote-GRAPE for Gravity Only

As shown in the previous section, it is more advantageous to use Remote-GRAPE for gravity only. We first examine the method of constructing neighbor lists on the master side (case PS-RG). Results are shown in Figure 10, where the solid and dashed lines correspond to case PS-RG and case S-RGN, respectively. The other two lines show case RG (dotted line) and case RGN (short-dashed line) for comparison. The enhancement of the performance is rather dramatic. In Table 5, we present a comparison of cases S-GN, S-RGN, and PS-RG for \( N \sim 10,000 \). The performance of case PS-RG is 3 times faster than case S-RGN and 10 times faster than case S-GN. However, we use the tree structure to construct neighbor lists, so that the clustering state apparently affects the performance because of the same reason mentioned in §3.3.3. It takes 2s to construct neighbor lists on the master side if \( a \sim 1 \) (\( N \sim 10,000 \)). For the deeper tree, it takes longer (~20 s).

4.2.2. Case PS-RGN: Using Remote-GRAPE for Gravity and Neighbor Lists

Another possible method is to use the neighbor lists of the previous step, which is constructed by Remote-GRAPE,
as possible neighbor lists (case PS-RGN). In this case, we have to adopt a larger neighbor radius than case S-RGN. Figure 9 (right) indicates that the $t_{\text{GRAPE}}$ part of the calculation is dominant so that the parallel method has less advantage for our configuration. Since the $t_{\text{GRAPE}}$ part and the other parts of the calculation obey different $N$ dependence, it is worth examining if there exits an optimal number of particles that leads to the optimal performance for a given configuration in case PS-RGN. For the SPH and misc part, we estimate their $N$ dependence in our SPH code from Figure 9 (left). We get $t_{\text{SPH}+\text{misc}} = 1.8 \times 10^{-4}N$ s. Therefore, $t_{\text{GRAPE}}$ always exceeds $t_{\text{SPH}+\text{misc}}$ for any $N$.

However, Figure 9 and Table 3 are the results for pure SPH calculations. Because we are interested in the formation of galaxy and globular clusters, we include such physical processes in our SPH code as cooling, heating, and star formation (Mori et al. 1997). If we include these physical processes, $t_{\text{misc}}$ should be much longer, so that the parallel method has a greater advantage. For example, in the SPH code calculation with nonequilibrium $H_2$ cooling, it takes $\sim 2.7 \times 10^{-4}$ s to solve rate equations for 10 species on our machine. In many cases, the cooling timescale is shorter than the dynamical timescale, so that the rate equations need to be solved many times in one dynamical step, which leads to $t_{\text{misc}} > 10^{-3}N$ s on our machine. For large $N$, $t_{\text{misc}}$ is more dominant than $t_{\text{GRAPE}}$.

5. CONCLUSION

We describe the Remote-GRAPE library and analyze its performance. It allows us to use GRAPE-3A with a computer that is not directly connected to GRAPE. Thus, we can use the state-of-the-art computer as the host computer.

First, we analyze the performance of original GRAPE system in some detail, as summarized below:

1. If we calculate gravity forces only, the use of a larger number of boards leads to higher performance.
2. In calculating both gravity and neighbor lists, however, the computation required on the host WS is larger than the former case, so that an increase in the number of the boards is less advantageous.
3. The speed of the host WS dramatically changes the performance of GRAPE system.

Second, we analyze the performance of Remote-GRAPE for our configuration, as summarized below:

1. If we calculate gravity forces only, the time required by Remote-GRAPE is almost the same as that of original GRAPE.
2. In calculating both gravity and neighbor lists by Remote-GRAPE, the time required to transfer data does not occupy a large fraction of total time but $\sim 20\% - 30\%$ for our configuration.
3. Compared with the tree method, the advantage of Remote-GRAPE is that its performance does not depend on the state of clustering, while the required time of the tree method is longer for higher degree of clustering.

4. In the actual application, the performance of our SPH code using Remote-GRAPE is 3–4 times faster than the SPH code using original GRAPE on the slave machine.

5. We can get further high performance (10 times faster than the SPH on the slave machine) by using the parallel method with Remote-GRAPE to calculate gravity only.

We have completed the first version of Remote-GRAPE, which significantly improves the performance of GRAPE-SPH. However, there are several ways to improve further the performance of our library and GRAPE-SPH with Remote-GRAPE.

1. If we use a larger number of GRAPE boards with a fast bus and a fast slave machine, the use of GRAPE in constructing neighbor lists is more advantageous than the use of tree structure.

2. We can develop a more efficient library. In the GRAPE system, all values are converted to the fixed floating point format in the original GRAPE library and then sent to the GRAPE board. The conversion from the floating point format to the fixed format may be done by the master side. In a relatively slow slave computer, this will significantly improve the performance.

We would like to thank J. Makino and T. Suzuki for reading the manuscript and giving useful comments. We also thank T. Miwa for providing us the results with SS-10. Those who would like to use our library can contact N. N. (nakasato@astron.s.u-tokyo.ac.jp). This work has been supported in part by the Grants-in-Aid for Scientific Research (05242102, 06233101) and COE Research (07CE200) of the Ministry of Education, Science, and Culture of Japan.

REFERENCES

Barnes, J., & Hut, P. 1986, Nature, 324, 446
Benz, W., Bowers, R., Cameron, A., & Press, W. 1990, ApJ, 348, 647
Evrard, A. E. 1988, MNRAS, 235, 911
Fukushige, T., Ito, T., Makino, J., Ebisuzaki, T., Sugimoto, D., & Umemura, M. 1991, PASJ, 43, 841
Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R., & Sunderam, V. 1994, PVM: Parallel Virtual Machine (Cambridge: MIT Press)
Gingold, A., & Monaghan, J. 1977, MNRAS, 181, 374
Habe, A., & Ohta, K. 1992, PASJ, 44, 203
Hernquist, L., & Katz, N. 1989, ApJS, 70, 419
Lattanzio, C., Monaghan, J., Ponrgracic, H., & Schwarz, P. 1985, MNRAS, 215, 125

Lucy, L. 1977, AJ, 82, 1013
Makino, J., & Funato, Y. 1993, PASJ, 45, 279
Mori, M., Yoshii, Y., Tsujimoto, T., & Nomoto, K. 1997, ApJ, 478, L21
Okumura, S., Makino, J., Ebisuzaki, T., Fukushige, T., Ito, T., & Sugimoto, D. 1993, PASJ, 45, 329
Steinmetz, M. 1996, MNRAS, 278, 1005
Steinmetz, M., & Mueller, E. 1994, AA, 281, 97
Sugimoto, D., Chikada, Y., Makino, J., Ito, T., Ebisuzaki, T., & Umemura, M. 1990, Nature, 345, 33
Umemura, M. 1993, AJ, 406, 361
Umemura, M., Fukushige, T., Makino, J., Ebisuzaki, T., Sugimoto, D., Turner, E., & Loeb, A. 1993, PASJ, 45, 311