Hierarchical tree algorithm for collisional $N$-body simulations on GRAPE

Toshiyuki Fukushige* and Atsushi Kawai*

K&F Computing Research Co., 1-21-6 Kojima-cho, Chofu, Tokyo 182-0026, Japan

*E-mail: fukushig@kfcr.jp, kawai@kfcr.jp

Received 2016 January 18; Accepted 2016 February 6

Abstract

We present an implementation of the hierarchical tree algorithm on the individual timestep algorithm (the Hermite scheme) for collisional $N$-body simulations, running on the GRAPE-9 system, a special-purpose hardware accelerator for gravitational many-body simulations. Such a combination of the tree algorithm and the individual timestep algorithm was not easy on the previous GRAPE system mainly because its memory addressing scheme was limited only to sequential access to a full set of particle data. The present GRAPE-9 system has an indirect memory addressing unit and a particle memory large enough to store all the particle data and also the tree node data. The indirect memory addressing unit stores interaction lists for the tree algorithm, which is constructed on the host computer, and, according to the interaction lists, force pipelines calculate only the interactions necessary. In our implementation, the interaction calculations are significantly reduced compared to direct $N^2$ summation in the original Hermite scheme. For example, we can achieve about a factor 30 of speedup (equivalent to about 17 teraflops) against the Hermite scheme for a simulation of an $N = 10^6$ system, using hardware of a peak speed of 0.6 teraflops for the Hermite scheme.

Key words: galaxies: star clusters — methods: $n$-body simulations — stellar dynamics

1 Introduction

Collisional $N$-body simulations, in which the equations of motion of $N$ particles are integrated numerically, have been extensively used in studies of dense star clusters, such as globular clusters, open clusters, and clusters with black holes, and also in studies of planetary formation. One feature of collisional $N$-body simulations is the need for relatively high accuracy in the force calculations, because the total number of timesteps is very large to simulate a relatively long simulation span, such as the relaxation timescale. Another feature is a wide difference in orbital timescales of particles, since two particles can approach arbitrarily close. The individual timestep algorithm, first developed by Aarseth (Aarseth 1963), has been a powerful tool that handles the collisional $N$-body system; its basic idea is to assign different times and timesteps to particles in the system.

GRAPE (GRAvity PipE: Sugimoto et al. 1991) is special purpose hardware that can accelerate the individual timestep algorithm. The GRAPE hardware has specialized pipelines for the gravitational force calculation, which is the most expensive part of collisional $N$-body simulations. Among the individual timestep algorithms, the Hermite scheme (Makino & Aarseth 1992) can efficiently use the GRAPE hardware, in which the block individual timestep algorithm (McMillan 1986) and fourth-order Hermite integration are used. GRAPE-6 (Makino et al. 2003) is massively parallel hardware for collisional $N$-body simulations using the Hermite scheme. It consists of 1728 pipeline chips and has a peak speed of around 64 teraflops.
Although a direct summation algorithm was used for the force calculations on the GRAPE-6 system, whether it is really the best solution or not remains in question. The Barnes–Hut tree algorithm (Barnes & Hut 1986) is one of the algorithms that reduce the calculation cost by replacing forces from distant particles by those from a virtual particle at their center of mass. McMillan and Aarseth (1993) have demonstrated that it is possible to implement a combination of the Barnes–Hut tree algorithm and the individual timestep algorithm that runs efficiently on single-processor computers. However, on the GRAPE-6 system, the combination of the tree algorithm and the individual timestep algorithm was not possible, because its memory addressing scheme was limited only to sequential access to a full set of particle data, and there is not enough memory for the particle data.

We successfully implemented the combination of the tree algorithm and the individual timestep algorithm on the GRAPE-9 system. GRAPE-9 is a newly developed system that uses FPGA (Field Programmable Gate Array) devices, and the same force and predictor pipelines as in the GRAPE-6 chip are integrated in the device. The GRAPE-9 system also has an indirect memory addressing unit and a relatively large particle memory, implemented by a widely used DRAM device. The interaction lists for the tree algorithm can be stored in the GRAPE-9 system, and the force pipelines can calculate only the interactions necessary. In our implementation, the interaction calculations are significantly reduced from the direct summation in the Hermite scheme.

The plan of this paper is as follows. In section 2 we describe the implementation of the tree algorithm on the Hermite scheme using GRAPE-9. In section 3, we present the performance and accuracy of our implementation. Section 4 is for discussion.

2 Implementation

In this section, we describe how the interactions are calculated using the tree algorithm in our implementation. Ideally, the interaction list should be created at every block timestep using predicted particle data, but this is not practical. In our implementation, the tree structure and the interaction lists are created only at intervals of Δt_{tree} and the same interaction lists are used during Δt_{tree}. Therefore, the interval Δt_{tree} becomes a cause of error in the interaction calculations since the tree structure is deformed as time advances. The interval Δt_{tree} has to be small enough not to affect the simulation results and its performance. In our implementation, the maximum size of timestep is set to be Δt_{tree} for simplicity.

With the original Hermite scheme, the previous GRAPE system (GRAPE-6) performs the integration of one step in the following way:

1. As the initialization procedure, the host computer sends all data of all particles to the memory on GRAPE.
2. The host computer selects the particles to be integrated at the present system time.
3. Repeat 4–6 for all particles selected:
4. The host computer predicts the position and velocity of the particle, and sends them to GRAPE.
5. GRAPE calculates the force from all other particles, and then returns the results to the host computer.
6. The host computer integrates the orbits of the particles and determines the new timestep. The updated particle data are sent to the memory on GRAPE.
7. The host computer updates the present system time and goes back to step 2.

In our new implementation with the tree algorithm, the GRAPE-9 system performs the integration of one step in the following way (the bold item numbers show the steps that have changed from the original algorithm):

1. At intervals of Δt_{tree} (and initially), the host computer makes the tree data. The procedure includes construction of a tree structure, identification of groups of particles for which the same interaction list is used by traversing the tree structure, and creation of interaction lists for the groups. The host computer sends all particle data, the interaction lists for all groups, and the tree node data listed in the interaction lists to the memory on GRAPE.
2. The tree node data are stored in the memory as (pseudo-) particles that have positions, velocities, accelerations, and their time derivatives.
3. The host computer selects the particles to be integrated at the present system time.
4. Repeat 4–6 for all particles selected:
4a. The host computer sends the index number of the interaction list for the particle to GRAPE.
5. GRAPE calculates the force from the particles in the interaction list, and then returns the results to the host computer.
6. The host computer integrates the orbits of the particles and determines the new timestep. The updated particle data are sent to the memory on GRAPE.
7. The host computer updates the present system time and goes back to step 2.

The differences from the original algorithm are in three steps: in step 1, at intervals of Δt_{tree}, the tree structure and the interaction list are created and sent to GRAPE. In step 4a, the index number of the interaction list is sent to
GRAPE. In step 5, the force are calculated from the particles in the interaction list, instead of from all particles.

In order to efficiently use the GRAPE hardware, we use the modified tree algorithm (as already described in step 1), which was developed by Barnes (1990) and implemented on the GRAPE hardware by Makino (1991). With this algorithm, tree traversal is performed for a group of neighboring particles and an interaction list is created for the group. The maximum number of particles in the group, \( n_{\text{crit}} \), is set to be optimal to keep the total computing time at a minimum. As we increase \( n_{\text{crit}} \), the interaction calculation on GRAPE increases since interactions between particles in a group are calculated directly and the interaction list becomes longer. On the other hand, as we decrease \( n_{\text{crit}} \), the efficiency of usage of the GRAPE hardware becomes lower, since the number of particles in the same group at each block step becomes smaller on average. For the present system, \( n_{\text{crit}} = 2000–4000 \) is close to optimal. Note that, with such \( n_{\text{crit}} \), interactions with a rather large number of neighboring particles, about \( 10^4 \) (for \( \theta = 0.5 \)), are directly calculated.

The part of the tree algorithm in our code is almost the same as that used in the previous studies (Fukushige et al. 2005; Yoshikawa & Fukushige 2005). The simulation program is written using the GRAPE-6 compatible API library with two additional functions for step 1 and steps 4a–5, respectively.

We implemented these algorithms on the GRAPE-9 (model 5000) system. The GRAPE-9 system consists of 8–16 GRAPE-9 cards, connected to the host computer via a PCI Express (PCIE) switch device (PLX PEX8696). The GRAPE-9 card is a PCIE extension card on which one FPGA device and one DDR2 SDRAM (SO-DIMM module) memory are mounted. In the FPGA device, the force and predictor pipelines, almost identical to the GRAPE-6 chip, and an indirect memory addressing unit are integrated, as illustrated in figure 1. The interaction lists for the tree algorithm are stored in the indirect memory addressing unit of the FPGA device, actually in its on-chip memory, and all the particle data and tree node data are stored in the memory unit, which consists of the DDR2 SDRAM memory. According to the interaction list, the indirect memory addressing unit outputs an address entry for the memory unit, and the force pipelines calculate only the interactions necessary. We use an Altera Cyclone V 5CGX9C9 for the FPGA device. The large size of the on-chip memory in this device is one of the reasons that enables our implementation. For the present implementation, we use a configuration in which 14 force pipelines (compatible with GRAPE-6) and one predictor pipeline are integrated, operating at 98 MHz. Other details of the GRAPE-9 system will be discussed elsewhere.

As for the indirect memory addressing unit, we use the same particle index unit as in GRAPE-5 (Kawai et al. 2000), which was designed for the cell-index method (Quentrec & Brot 1975) to handle short-range forces in a periodic boundary condition. Figure 2 shows a block diagram for the indirect memory addressing unit. It consists of the cell-index memory and two counters: the cell counter and the particle index counter. In the cell-index memory, sets of start address and count number are stored. According to the output of the cell-index memory, the particle index counter generates entries to the memory unit. The cell counter

![Fig. 1. Overall structure of the GRAPE-9 system.](https://academic.oup.com/pasj/article-abstract/68/3/30/2223122/6830223122)
indicates the address entry for the cell-index memory. Actually, in step 4a, the host computer sends the start address and count number of the cell counter for the group of the interaction list. Because the size of the on-chip memory of the FPGA device is limited (about several Mbits), we store the interaction list for the tree algorithm in this form instead of as full sets of indices. The entry size of the cell-index memory is 98304 for the present FPGA device. In order to reduce the total length of the interaction lists in the cell-index memory, we rearrange all particles in the Peano–Hilbert order and store the tree node data for each group in consecutive locations in the memory unit. Since we typically use 1 GB DDR2 SDRAM SO-DIMMs (8 GB at maximum), the memory unit can store 10 million particles at the maximum for each card.

When we perform calculations using multiple GRAPE-9 cards, we use two parallelization methods in combination: (1) multiple cards calculate the force on the same set of particles (i), but from a different set of particles (j) (called j-parallel); (2) multiple cards calculate the force on a different set of particles (i) whose interaction lists (group) are different (called i-parallel).

3 Performance and accuracy

In this section, we discuss the performance and accuracy of our implementation. For benchmark runs, we integrated the Plummer model with equal-mass particles. We use the standard units (Heggie & Mathieu 1986) in which \( M = G = -4E = 1 \). Here, \( M \) and \( E \) are the total mass and energy of the system, and \( G \) is the gravitational constant. The timestep criterion is that of Aarseth (1999) with \( \eta = 0.01 \). For the softening parameter, we used an \( N \)-dependent softening, \( \varepsilon = 1/N \). We set \( n_{\text{crit}} = 4000 \) for all runs except for runs of \( \theta = 0.3 \), \( N = 512 \) k and \( 1 M \) (\( n_{\text{crit}} = 7000 \) and 14000), and the interval \( \Delta t_{\text{tree}} = 1/64 \).

We used the GRAPE-9 system that consists of eight GRAPE-9 cards and whose peak speed is 630 Gflops. Here we count the operations for the gravitational force and its time derivatives as 57 floating-point operations. The host computer has an Intel Core i7-3820 (four cores, 3.6 GHz) CPU. Communications between the host computer and each GRAPE-9 card are generation 1 four-lane PCIE (1 GB s\(^{-1}\) peak for each direction). In order to use eight cards simultaneously, we used a parallelization method whose degree is 4 for \( j \)-parallel and 2 for \( i \)-parallel.

Figure 3 shows the calculation time, \( T \), to integrate the system for one time unit as a function of the number of particles, \( N \). The triangles, squares, and pentagons indicate the results with opening angles for the tree algorithm \( \theta = 0.3, 0.5 \) and 0.75, respectively. The stars indicate the results for the Hermite scheme.
Fig. 4. Equivalent performance, $S$, defined in the text, as a function of the number of particles, $N$. The triangles, squares, and pentagons indicate those with opening angles for the tree algorithm $\theta = 0.3, 0.5,$ and $0.75$, respectively. The stars indicate the results for the Hermite scheme. The thin dashed line indicates the peak performance for the Hermite scheme.

Figure 4 shows another plot with an equivalent performance, $S$, defined by

$$S = \frac{57Nn_{\text{step}}}{T} \frac{n_{\text{step,h}}}{n_{\text{step}}} = \frac{57Nn_{\text{step,h}}}{T},$$  \hspace{1cm} (1)$$

where $n_{\text{step}}$ is the total number of individual timesteps to integrate one time unit, and $n_{\text{step,h}}$ is that for the Hermite scheme. Equivalent performance means the performance in the case that we perform the same simulation within the same time using the Hermite scheme. For the ratio, $n_{\text{step,h}}/n_{\text{step}}$, of $N = 1 M$, we used instead those of $N = 512 k$ for each $\theta$. The ratio $n_{\text{step,h}}/n_{\text{step}}$ itself is close to unity; for example, $n_{\text{step,h}}/n_{\text{step}} = 1.042$ for $N = 512 k$, $\theta = 0.75$, which means even if we use the tree algorithm the total number of individual timesteps does not increase too much. We can see that a factor of about $N/30 k$ speedup (for $\theta = 0.5$) is achieved against the original Hermite scheme.

Figure 5 shows the errors in the total energy as a function of time up to simulation time $t = 10$ for $N = 256 k$. The calculation of the potential energy is obtained with direct summation on GRAPE, not with the tree algorithm. We can see that the errors in our implementation of the tree algorithm increase linearly as time advances. In figure 6, the errors in the total energy at time $t = 10$ are summarized. From both figures, we can see that the errors for $\theta = 0.3$ are comparable to those for the original Hermite scheme.

We now discuss the breakdown of the calculations using a simple performance model. The calculation time per one particle step is expressed as

$$T = T_{\text{grape}} + T_{\text{tree}} + T_{\text{comm}} + T_{\text{host}}.$$ \hspace{1cm} (2)$$

In table 1, the terms on the right-hand side list the times measured in actual runs. The first term on the right-hand side, $T_{\text{grape}}$, is the time to calculate the force and its time derivative for one particle on GRAPE-9, expressed as

$$T_{\text{grape}} \approx \bar{N}_{\text{int}} \frac{\bar{n}_i}{n_{\text{pipe}}}.$$ \hspace{1cm} (3)$$

where $\bar{N}_{\text{int}}$ is the average number in the interaction list for the tree algorithm. In the case of one GRAPE-9 card, $t_{\text{pipe}} = 7.6 \times 10^{-10} \text{s}$. The factor $\bar{n}_i/n_{\text{pipe}}$ expresses the decrease in performance when the number of particles that calculate interactions simultaneously (at step 5) is less than $n_{\text{pipe}}$. Here, $n_{\text{pipe}}$ and $\bar{n}_i$ are the maximum and average number of the particles that calculate interactions simultaneously, respectively. The number $\bar{n}_i$ becomes much smaller for the tree algorithm than in the original Hermite scheme, because, even in the same block step, the particles belong to several different groups. The number $\bar{n}_i \sim 30$ in actual

![Graph showing equivalent performance](image-url)
runs, which does not depend much on \( N \) or \( \theta \). For the present system, \( n_{\text{pipe}} = 56 \), since it has 14 real force pipelines and each real pipeline serves as 4 virtual multiple pipelines (Makino et al. 1997).

The second term, \( T_{\text{tree}} \), is the time for the tree data processing spent in the host computer (step 1), which is listed in Table 1 for \( \Delta t_{\text{tree}} = 1/64 \). The time \( T_{\text{tree}} \) is proportional to \( 1/\Delta t_{\text{tree}} \) and does not depend much on \( N \). The third term, \( T_{\text{comm}} \), expresses the time to transfer data between the host computer and GRAPE, which includes data conversion. Since about 200 bytes of data transfer are required per one particle step, the sustained transfer speed is about 200 MB s\(^{-1}\). The fourth term, \( T_{\text{host}} \), is the time for the host computer to perform the computations to integrate one particle other than \( T_{\text{tree}} \).

As for the breakdown, at first, we note that, in the first term \( T_{\text{grape}} \), the decrease in performance due to small \( \bar{n}_i \) is rather large and the sustained performance decreases to about half of its peak performance. This is partly because \( n_{\text{pipe}} = 56 \) for the present system is not small enough. A system with \( n_{\text{pipe}} \) less than 30 is undesirable for our implementation. Second, the largest term is \( T_{\text{comm}} \) among the three terms other than \( T_{\text{grape}} \), and the fraction to \( T_{\text{grape}} \) is not small compared to the original Hermite scheme because the number of interactions \( \bar{N}_{\text{int}} \) is not large, of course. Third, the time \( T_{\text{tree}} \) is small enough compared to the other terms in the case of \( \Delta t_{\text{tree}} = 1/64 \).

### 4 Discussion

We successfully implemented the hierarchical tree algorithm on the individual timestep algorithm (the Hermite scheme) for collisional \( N \)-body simulations on the GRAPE-9 system. The present GRAPE-9 system has an indirect memory addressing unit and the memory unit is large enough to store all particle data and also the tree node data. In our implementation, the interaction calculations are significantly reduced, compared to the direct \( N^2 \) summation in the original Hermite scheme.

In comparison to other methods that also successfully reduce the amount of calculation for the individual timestep algorithm, our implementation has one advantageous feature in that interactions from particles at an intermediate range are evaluated in a more accurate way. The neighbor scheme (Ahmad & Cohen 1973; Nitadori & Aarseth 2012) is an example of such methods. In this scheme, the force on a particle is divided into two components, the neighbor force and the regular force, and the amount of calculation is reduced by evaluating the regular force less frequently. \( \bar{P}^T \bar{T} \) (Particle-Particle Particle-Tree; Oshino et al. 2011; Iwasawa et al. 2015) is another example. In \( \bar{P}^T \bar{T} \), the force on a particle is split into short-range and long-range contributions. The short-range forces are evaluated with the Hermite scheme and the long-range forces are evaluated with the tree algorithm and a leapfrog integrator. It is reported that less accurate evaluation for intermediate range forces might influence the angular momentum evolution (see Iwasawa et al. 2015).

At present, the GRAPE-9 system is probably a good solution for the implementation of the hierarchical tree algorithm on the individual timestep algorithm. Further improvement with a next-generation FPGA device would provide more powerful computing systems. The shipment within one year of a new FPGA device (Altera Arria 10) that has more than four times the number of logic elements, three times the operation speed, eight times the data transfer speed (generation 3, eight-lane PCIE), four times the on-chip memory size, and ten times the memory bandwidth (DDR3/DDR4 SDRAM), compared to the current FPGA device (Altera Cyclone V), has been announced. A new system using such an FPGA device would be able to provide about ten times the performance with smaller \( n_{\text{pipe}} \) (~30), which is another required ingredient for our implementation.

The porting of our implementation to other accelerators, such as general purpose graphical processing devices (GPGPU), is presumably feasible and in preparation. Typically, a very large number of parallel operations must be executed on such an accelerator. Since, for our implementation, the number of the interaction calculations that can be executed in parallel becomes smaller, some ingenuity would be necessary for the efficient use of such an accelerator.

### Table 1. Breakdown of calculation time.

| \( N \)      | \( \theta \) | \( \bar{N}_{\text{int}} \) | \( \bar{n}_i \) | \( T_{\text{grape}} \) (s) | \( T_{\text{tree}} \) (s) | \( T_{\text{comm}} \) (s) | \( T_{\text{host}} \) (s) |
|-------------|-------------|------------------|---------------|-----------------|-----------------|-----------------|-----------------|
| 65536       | 0.75        | 5740             | 29.3          | \( 8.5 \times 10^{-6} \) | \( 1.5 \times 10^{-7} \) | \( 9.8 \times 10^{-7} \) | \( 1.6 \times 10^{-7} \) |
| 65536       | 0.5         | 11081            | 28.8          | \( 1.6 \times 10^{-5} \) | \( 2.4 \times 10^{-7} \) | \( 9.8 \times 10^{-7} \) | \( 1.6 \times 10^{-7} \) |
| 65536       | 0.3         | 22351            | 28.6          | \( 3.3 \times 10^{-5} \) | \( 4.0 \times 10^{-7} \) | \( 9.8 \times 10^{-7} \) | \( 1.5 \times 10^{-7} \) |
| 262144      | 0.75        | 6433             | 29.8          | \( 9.2 \times 10^{-6} \) | \( 9.7 \times 10^{-8} \) | \( 1.2 \times 10^{-6} \) | \( 2.7 \times 10^{-7} \) |
| 262144      | 0.5         | 12644            | 29.5          | \( 1.8 \times 10^{-5} \) | \( 1.5 \times 10^{-7} \) | \( 1.2 \times 10^{-6} \) | \( 2.7 \times 10^{-7} \) |
| 262144      | 0.3         | 28168            | 29.4          | \( 4.0 \times 10^{-5} \) | \( 2.7 \times 10^{-7} \) | \( 1.2 \times 10^{-6} \) | \( 2.6 \times 10^{-7} \) |
Acknowledgement

We are grateful to Hiroshi Daisaka and Ataru Tanikawa for helpful discussions and various comments on this study.

References

Aarseth, S. J. 1963, MNRAS, 126, 223
Aarseth, S. J. 1999, Celest. Mech. Dyn. Astron., 73, 127
Ahmad, A., & Cohen, L. 1973, J. Comput. Phys., 12, 389
Barnes, J. E. 1990, J. Comput. Phys., 87, 161
Barnes, J. E., & Hut, P. 1986, Nature, 326, 446
Fukushige, T., Makino, J., & Kawai, A. 2005, PASJ, 57, 1009
Heggie, D. C., & Mathieu, R. D. 1986, in The Use of Supercomputers in Stellar Dynamics, ed. P. Hut & S. McMillan (New York: Springer), 233
Iwasawa, M., Portegies Zwart, S., & Makino, J. 2015, Comput. Astrophys. Cosmology, 2, 6
Kawai, A., Fukushige, T., Makino, J., & Taiji, M. 2000, PASJ, 52, 659
Makino, J. 1991, PASJ, 43, 621
Makino, J., & Aarseth, S. J. 1992, PASJ, 44, 141
Makino, J., Fukushige, T., Koga, M., & Namura, K. 2003, PASJ, 55, 1163
Makino, J., Taiji, M., Ebisuzaki, T., & Sugimoto, D. 1997, ApJ, 480, 432
McMillan, S. L. W. 1986, in The Use of Supercomputers in Stellar Dynamics, ed. P. Hut & S. McMillan (New York: Springer), 156
McMillan, S. L. W., & Aarseth, S. J. 1993, ApJ, 414, 200
Nitadori, K., & Aarseth, S. J. 2012, MNRAS, 424, 545
Oshino, S., Makino, J., & Funato, Y. 2011, PASJ, 63, 881
Quentrec, B., & Brot, C. 1975, J. Comput. Phys., 13, 430
Sugimoto, D., Chikada, Y., Makino, J., Ito, T., Ebisuzaki, T., & Umemura, M. 1990, Nature, 345, 33
Yoshikawa, K., & Fukushige, T. 2005, PASJ, 57, 849