Progress report on ”Formation of Galactic Halos by Compact Discrete Objects”

Yasushi Suto (University of Tokyo)
Naoki Yoshida (Harvard University)
Grape Category B g02b18

March 27, 2003

1 Overview

During the 2002 Kouki term, we have been working on the development of a fast and accurate N-body solver which runs on parallel grape6 systems. We plan to use the code to simulate the formation of galactic structure. Unlike conventinal cosmological simulations that assume that simulation particles are a discrete representation of a smooth density field, our simulations are aimed at revealing how the formation and the structure of dark halos are affected if mass elements —graviational sources— are massive compact objects such as primordial black holes. For this purpose we need a fast N-body code which can accurately follow close encounters of particles.

2 Parallel-Grape-Gadget

Our code is based on parallel N-body code GADGET (Springel, Yoshida & White 2001). We replace its gravity calculation part with GRAPE-6 modules. Specifically, the code is designed to utilize the Mitaka parallel GRAPE6 cluster. We describe the implementation in this section.

2.1 GRAPE-6

GRAPE-6 is the successor of GRAPE-4, designed for high-accuracy integration of gravitational N-body system using the individual timestep scheme. Since many technical papers on grape architectures are available (e.g. Fukushige et al. 1995), I will not describe the system details in this report. The biggest advantage of GRAPE-6 over previous versions of GRAPE is that the prediction stage typically employed in leap-frog schemes can be done *on board.* To be specific, GRAPE-6 calculates the following quantities:

$$\mathbf{a}_i = \sum_j Gm_j \frac{\mathbf{r}_{ij}}{(r_{ij}^2+\epsilon^2)^{3/2}}, \tag{1}$$

$$\dot{\mathbf{a}}_i = \sum_j Gm_j \left[\frac{\mathbf{v}_{ij}}{(r_{ij}^2+\epsilon^2)^{3/2}} - \frac{3(\mathbf{v}_{ij}\cdot\mathbf{r}_{ij})\mathbf{r}_{ij}}{(r_{ij}^2+\epsilon^2)^{5/2}}\right], \tag{2}$$

and

$$\phi_i = \sum_j G m_j \frac{1}{(r_{ij}^2 + \epsilon^2)^{1/2}}, \tag{3}$$

where

$$\mathbf{r}_{ij} = \mathbf{x}_{j,\mathrm{pred}} - \mathbf{x}_i, \tag{4}$$

$$\mathbf{v}_{ij} = \mathbf{v}_{j,\mathrm{pred}} - \mathbf{v}_i. \tag{5}$$

Here, $\mathbf{x}_i, \mathbf{v}_i, \mathbf{a}_i, \dot{\mathbf{a}}_i$ are the position, velocity, acceleration, the first time derivative of acceleration of particle i, G is the gravitational constant which is fixed to be unity in GRAPE-6, and ϵ is the usual plummer softening parameter.

The position and velocity of particle j in equations (4), (5) have an additional suffix "pred" to denote that they are predicted values at a system time t using the following formulae:

$$\Delta t = t - t_j, \tag{6}$$

$$\mathbf{x}_{j,\mathrm{pred}} = \frac{\Delta t^4}{24}\mathbf{a}_0^{(2)} + \frac{\Delta t^3}{6}\dot{\mathbf{a}}_0 + \frac{\Delta t^2}{2}\mathbf{a}_0 + \Delta t \mathbf{v}_0 + \mathbf{x}_0, \tag{7}$$

$$\mathbf{v}_{j,\mathrm{pred}} = \frac{\Delta t^3}{6}\dot{\mathbf{a}}_0^{(2)} + \frac{\Delta t^2}{2}\dot{\mathbf{a}}_0 + \mathbf{v}_0. \tag{8}$$

We have dropped the subscript j in the right-hand side for clarity, and $\mathbf{x}_0, \mathbf{v}_0$, etc. are true values at time t_j. For usual cosmological simulations that employ only the lowest-order integration schemes, one simply uses

$$\mathbf{x}_{j,\mathrm{pred}} = \Delta t \mathbf{v}_0 + \mathbf{x}_0, \tag{9}$$

for force computation. The first derivative of acceleration (equation (2)) is not used in such cases, although predicted velocities may be used in the hydro part.

2.2 On-demand particle update

In past simulation codes using GRAPE such as Steinmetz's GRAPE-SPH (1996), the positions of *all* the simulation particles are predicted to be at system times on the host computer and the predicted values are sent to GRAPE at every time step. This has been a source of significant overhead, even though the force computation itself is done very fast on GRAPE. We completely eliminate this overhead of host $\rightleftharpoons$ grape communication by sending only updated particles. The implementation looks like this:

1. `determine-timesteps-and-mark-forceupdate-particles();`
2. `compute-acceleration-and-move();`
3. `update-posvel-and-send-to-grape();`
4. go back to 1.

Other non-update particles will be kept in the memory on GRAPE until their next update time. Of course, all the particles must be sent in the beginning of simulations and after domain decomposition is done.

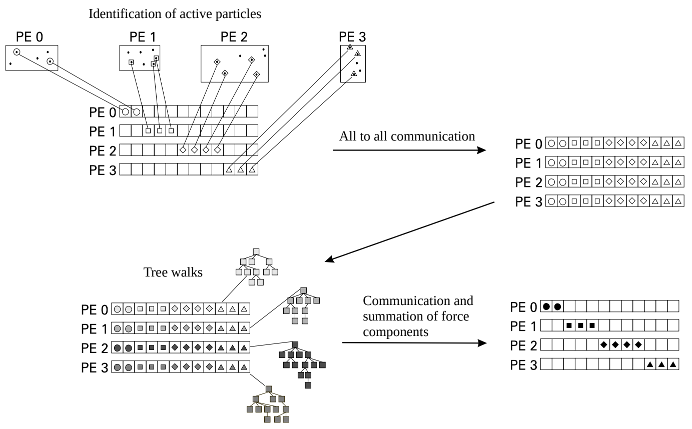

Identification of active particles
PE 0
PE 1
PE 2
PE 3
All to all communication
Tree walks
Communication and summation of force components

Figure 1: Schematic illustration of the parallel force computation scheme employed in PG6-Gadget. The local force computation done by treewalk is replaced with the direct force summation by GRAPE6.

2.3 Force computation

Force computations for update particles are done in essentially the same way as the tree-force computation in Parallel-Gadget. First, a chunk of update particles are created and shared by all the processors. Each processor computes the forces from its local particles exerting on the update particles and the contributions are summed up in the end stage. Figure 1 illustrates how this is done. Since GRAPE6 has 48 gravity pipelines per board, the force computation for the update particles is done repeatedly, i.e., 48 particles per one round. This is the only one minor complication in the force computation. The computed forces are exact down to the softening scales (which can be set very small) and the calculation speed does *not* depend on spatial clustering. Hence this code opens a new window for simulating a single halo for studies of the density profile, star cluster simulations, simulations of planet formation etc. Note that, if periodic boundaries are used, either the Ewald correction must be added or a Fourier technique must be used to add forces from periodic images. This can be done in a neat way in some cases (see the last section).

2.4 Timestepping in GADGET

GRAPE6 boards handle the time for predictor in 64 bit fixed point format. The binary point can be arbitrarily set by `g6-set-tunit(tunit)` function. The system time and timesteps must be compatible with the blockstep algorithm, which means that the timesteps must be powers of two (negative values allowed, i.e. 2^{-n} are accepted if tunit is set accordingly). The new version GADGET2 (Springel 2003) employs exactly what we need to incorporate the predictor on GRAPE6. GADGET2 maps the timespan of a simulation, say from $t_{\rm begin}$ to $t_{\rm max}$, onto discrete time bins. Then the individual timestep is set to be the largest powers of two which is smaller than the required timestep width. Thus one can simply call the exsiting `g6` functions without any modification. Nevertheless, one minor complication arises when the system is integrated in comoving coordinate, which we describe next.

2.5 Predictor for comoving integration

Gadget2 employs a particular set of variables when the comoving integration is employed. Specifically, it uses comoving position and velocity variables

$$\mathbf{x} = \mathbf{r}/a, \tag{10}$$

$$\mathbf{u} = a^2\dot{\mathbf{x}} = a\mathbf{v}, \tag{11}$$

with the cosmic expansion parameter a being as time variable. The equation of motion then becomes

$$\frac{d\mathbf{u}}{da} = \frac{1}{a^2H(a)} \times \left[-G\sum_j \frac{m_j\mathbf{x}_{ij}}{|\mathbf{x}_{ij}|^3} + \frac{1}{2}\Omega H_0^2\mathbf{x}_i \right], \tag{12}$$

and the particle position is updated viz.

$$\frac{d\mathbf{x}}{da} = \frac{\mathbf{u}}{a^3H(a)}. \tag{13}$$

Since GRAPE-6 predicts a particle's position based on the assigned velocity exactly as expressed in equation (4), we need to re-scale the velocity before passing to GRAPE-6. (Although one can change the time unit by `g6-tunit` function, it will make things just complicated.) Hereafter we use the system time and timestep precisely as defined in GADGET2. Let us re-write equation (13) in a more direct manner as implemented in the program,

$$\mathbf{x}_{\text{pred}} = \frac{\Delta a}{a^3H(a)}\mathbf{u}_0 + \mathbf{x}_0, \tag{14}$$

where $\mathbf{x}_0$ and $\mathbf{u}_0$ are the particle's position and velocity, respectively, at the particle's time. Whereas Δa in Gadget-1.1 used to be simply `All.Time - P[i].CurrentTime`, one aditional mapping is necessary in Gadget2. Comparing equation (14) with equation (9), and also by noting that we pass integer powers-of-two values to GRAPE6 as the time variable, we find that the following formula should be used:

$$\mathbf{x}_{\text{pred}} = (t - t_j)\left[\frac{\Delta a}{(t - t_j)a^3H(a)}\mathbf{u}_0\right] + \mathbf{x}_0. \tag{15}$$

Thus we need to scale the particle velocity by $\frac{\Delta a}{\Delta t_{\text{int}}a^3H(a)}$ when sending to GRAPE6.
Although this scaling is correct to the first order, one can make it a little more consistent with other time integration part in Gadget2. Namely, we compute

$$\frac{\texttt{dtdrift}(t, t_j)}{(t - t_j)} \tag{16}$$

where $\texttt{dtdrift}(t, t_j)$ is the same fucntion as that used in the drift phase.

2.6 Performance analysis

I used a test simulation set "ZIC-cluster" included in the publicly available Gadget package. The simulation follows the evolution of a large spherical region with comoving radius $100h^{-1}$Mpc within a ΛCDM cosmogony. The high-resolution region is represented by 140005 particles and there are 136493 boundary particles (for tidal forces), hence 276498 particles in total. There are no gas particles. I used 2 processors with a GRAPE6 board connected to each. The cpu time statistics are shown in the table below:

	Gadget-2	PGR6-Gadget	speed-up factor
Total	17825.09 sec	3493.67 sec	5.01
Gravity	16677.8 sec	2935.23 sec	5.68
Total timesteps	998	1521	–

Note that in the both cases parameters are *not* optimized. I used default values for this comparison. Thus further speed up should be expected for each run by a different factor by tuning parameters. Due to the geometry of the simulation region, PM part is not used with Gadget2. Total number of time steps of the GRAPE run is larger than Gadget, because it employs a slightly more restictive time step constraint.

2.7 Basic simulation results

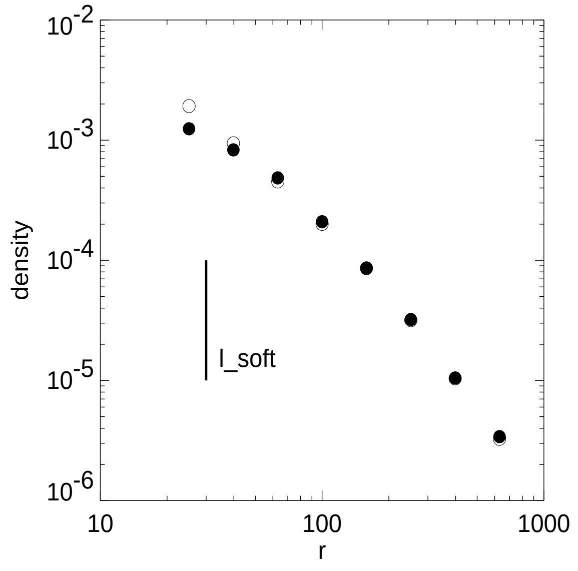
Figure 2 compares the density profile of the most massive cluster at $z = 0$ and also shows the distribution of particles in the high-resolution region. The grape result shows a shallower density profile than Gadget2, but this is likely due to the softening effect. GRAPE employs the Plummer softening whereas Gadget uses a spline function for the softening.

3 Prospects for GrapePM code

GRAPE6 has an optional function to evaluate forces with a truncated power-law (Jun Makino, private communication). It is straightforward to replace the current `gravgrape` subroutine with one that computes only short-range forces. Even nicer, the default hardwired function to be returned is Gaussian-convolved, so it should be ideal to be used together with long range forces by PM.

References

Fukushige, T., Makino, J. & Taiji, M. 1996, in *The Combination of Theory, Observations, and Simulation for the Dynamics of Stars and Star Clusters in the Galaxy*
Springel, V., Yoshida, N. & White, S. D. M., 2001, New Astronomy, 6, 79
Steinmetz, M., 1996, MNRAS, 278, 1005



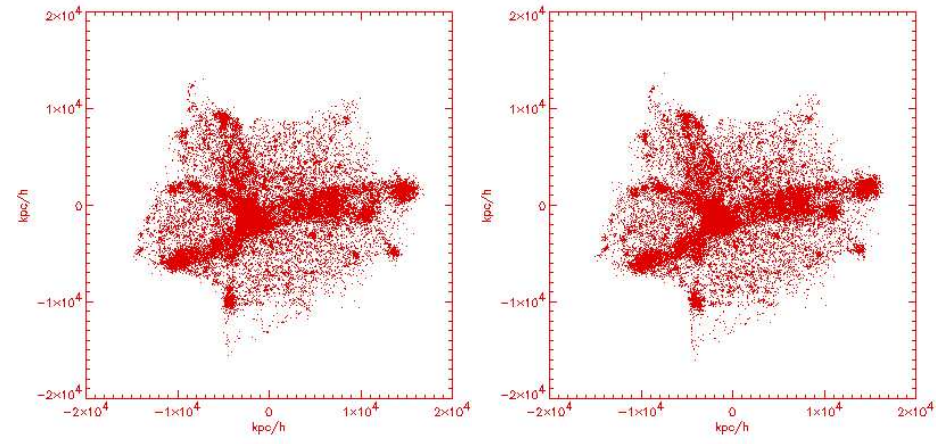


Figure 2: (TOP) Density profile of the most massive cluster at $z = 0$. Simulation by GRAPE (filled circles) and by Gadget (open circles). (BOTTOM) Distribution of the high-resolution particles at $z = 0$. Simulation by GRAPE (left) and Gadget2 (right).