Mon. Not. R. Astron. Soc. **000**, 000–000 (1994) Printed 16 July 1997

A parallel implementation of an Aarseth N-body integrator on the CRAY T3D

R. Spurzem[1]
[1] *Astronomisches Rechen-Institut, Mönchhofstraße 12-14, D-69120 Heidelberg, Germany*

Accepted, Received; in original form

ABSTRACT
The implementation of the most recent collisional Aarseth N-body integrator with Ahmad-Cohen scheme NBODY6++ on a general purpose massively parallel computer (CRAY T3D) is described. The performance is compared to standard computers and the results are compared to previous performance analyses of direct N-body integrators. A critical revisit of the optimal neighbour number in Ahmad-Cohen schemes shows, that for some special purpose and parallel computers it is much smaller than expected.

Key words: stellar dynamics – star clusters – numerical methods

1 INTRODUCTION

It is one of the grand challenges of theoretical astrophysics to understand the dynamics of globular star clusters. In contrast to most galaxies two-body relaxation times in such clusters are shorter than their age, but long compared to the dynamical time (to readjust virial equilibrium). Therefore globular clusters are always to a very good approximation in dynamical (virial) equilibrium, and evolve through heat conduction by the cumulative effect of distant gravitative encounters on a characteristic relaxation time scale at the half-mass radius of about 10^8 to 10^{10} years, depending on total mass and concentration (Spitzer 1987). Many astrophysical N-body systems are in contrast to globular clusters collisionless systems (like e.g. most galaxies), which means that relaxation by two-body or higher correlations does not play any role for the equilibrium dynamics and kinematics of the object. For such collisionless stellar systems the collisionless Boltzmann or Vlasov equation (or their moment equations) are the appropriate fundamental kinetic equations. One way to solve them numerically is a low-precision direct N-body simulation (orbit integrations of a large number of particles in a smooth or softened gravitational potential). Softening means that the singularity of the gravitational potential $\Phi(r) \propto r^{-1}$ for two interacting particles at a distance r from each other is bridged by an artificially modified potential of the form $\Phi \propto (r^2+\varepsilon^2)^{1/2}$ with finite ε (Aarseth 1963, Barnes & Hut 1986); thus strong interactions and, in case of a sufficiently large ε, also undesired two-body relaxation is inhibited in the simulations. Since the real system, with much more particles than in the simulation, has a larger relaxation time $t_{\rm rx}$ ($t_{\rm rx}/t_{\rm dyn} = N/(\Gamma \log N)$, cf. Spitzer 1987, with $\Gamma = 0.11$, see Giersz & Heggie 1994a) than the simulated system this effect is quite welcome. However, there is no practical way yet to increase the relaxation time by softening locally by the same factor for all portions of a spatially inhomogeneous system. The increase in relaxation time by some softening in a homogeneous infinite system could be measured by the ratio $x = \varepsilon/p_0$, where p_0 is the 90^o deflection angle impact parameter. In a system with core-halo structure p_0 is variable as well, and so only a consistently varying ε, such that x is kept constant throughout the system would mimic the relaxation time of a larger N-body system more exactly. Variable softening, however, causes trouble with energy conservation, because the binding energy between two particles remaining in constant distance from each other could vary in time and because the force would in such case not be symmetric between two particles anymore. In total this means that much care has to be taken for any low N simulation with softening, that two-body relaxation is not causing unrealistic spurious effects (see for some illustrations Spurzem & Theis 1996). Again in contrast to that, collisional simulations usually do not include any softening, aiming at a simulation where one particle in the simulated system represents a single particle in the real system as well. Close encounters, in order to avoid truncation errors and downgrading of the overall performance, are treated in relative and specially regularized coordinates (Kustaanheimo & Stiefel 1965), implementation see e.g. Aarseth 1985).

For the reader unfamiliar with stellar dynamical notions it is noted here that collisional or collisionless always refers to whether the correlations by two-body elastic encounters are important or not, i.e. whether the collisionless Boltzmann or Vlasov equation could be used. In the case of astrophysical N-body simulations of molecular cloud systems (particles with large radii) it occurs the seemingly paradox situation that the system is collisionless in this stellar dynamical sense, although direct collisions and mergers of the particles are frequent (Theis & Hensler 1993). An appropriate kinetic equation for the collisional case is the Fokker-Planck equation (Cohn 1980), its moment equations (Bettwieser 1983, Spurzem 1994, 1996), or, a high-precision direct N-body simulation, which should be able to follow all two-body encounters with arbitrarily small impact parameters efficiently and accurately.

Apart from the intrinsic interest of understanding the behaviour of large collisional N-body systems, the importance of the problem stems from its relation with current observational research, and from the aim to study the behaviour of even larger systems such as galactic nuclei. Without understanding globular clusters by means of special tools for investigating their evolution, the possibility of unravelling the mysterious behaviour of galactic nuclei seems very remote. Unfortunately, the direct simulation of such rich stellar systems as globular clusters with star-by-star modelling is not yet possible. The gap between the largest useful N-body models with N being of the order of 10^4 particles and the median globular star cluster ($N \sim 5 \times 10^5$) can only be bridged at present by use of theory. There are two main classes of theory: (i) Fokker-Planck models (henceforth FP model), which are based on the direct numerical solution of the orbit-averaged Fokker-Planck equation (Cohn 1980, Cohn, Hut & Wise 1989, Murphy, Cohn & Hut 1990), and (ii) isotropic (Lynden-Bell & Eggleton 1980, Heggie 1984, Bettwieser & Sugimoto 1984) and anisotropic gaseous models (Louis & Spurzem 1991, Spurzem 1994, 1996), which can be thought of as a set of moment equations of the Fokker-Planck equation.

These simplified models are the only detailed models which are directly applicable to large systems such as globular clusters. But their simplicity stems from many approximations and assumptions which are required in their formulation. One example is the assumption of spherical symmetry, which is inconsistent with the asymmetry of the galactic tidal field; another one the use of statistical estimates of cross sections for the formation of close binaries by three-body or dissipative (tidal) two-body encounters, and for their subsequent hardening by gravitational interactions with field stars. Such processes play a dominant role in reversing core collapse of globular clusters, which otherwise would inevitably lead to a singular density profile with infinite density at the centre (Lynden-Bell & Wood 1968, Bettwieser & Sugimoto 1984, Elson, Hut & Inagaki 1987, Hut 1993). Standard models predict core collapse (shrinking of core radius and core particle number, with increasing central density and a typical self-similar power-law profile in the transition zone between core and outer halo, Lynden-Bell & Eggleton 1980). At high central densities very close binaries with binding energies large compared to the r.m.s kinetic energy of the core stars form by three-body encounters. Subsequent superelastic scatterings with other field stars heat the core and finally reverse core collapse. The process repeats itself in a quasi-periodic manner with very large amplitude oscillattions of the central star density, provided the particle number is large enough ("gravothermal oscillations", Bettwieser & Sugimoto 1984, Goodman 1987, Spurzem 1994, Breeden Cohn & Hut, 1994). Unfortunately it is extremely difficult to find such phenomena in direct N-body simulations, because on one hand side the required high accuracy simulations are only very gradually approaching real particle numbers of globular clusters (Spurzem & Aarseth 1996 for $N = 10\,000$, Makino 1996 for $N = 32\,000$) by exploiting the most advanced supercomputers (as described in the present paper for a general purpose machine, see also Sugimoto et al. 1990, Taiji 1996, for special-purpose computers). At least for particle numbers of the order of 10^4 fluctuations caused by strong encounters disturb steady thermodynamic effects like gravothermal oscillations, and there is still not yet any proof, that the steady, high amplitude gravothermal oscillations at huge particle numbers like 10^5 will appear in the real N-body system in the same way (but see recent work by Makino 1996, which gives some hints in this direction).

Recent detailed comparisons have established a good relationship between such simplified models and direct N-body simulations, but for the even more idealized case of an equal mass system (identical individual mass in contrast to real star clusters) and rather low particle numbers (Giersz & Spurzem 1994, for $N = 250$, 500, 1000 and 2000, and also Giersz & Heggie 1994a, b). Only in one case has a two-mass system been compared with direct N-body simulations (Spurzem & Takahashi 1995). Unfortunately for much larger particle numbers post-collapse star clusters cannot be understood easily by extrapolating the results for low N. Since two-body relaxation scales with $N/\log(\Gamma N)$ (see above), or in other words, all N-body results *before* core bounce (the point where binary effects reverse core collapse) match exactly if one scales the time with this factor (Giersz & Heggie 1994a, b), but the formation of three-body and tidal two-body binaries introduces other timescales into the problem, which do not scale with the same factor of N.

Therefore it remains very important to get more data from direct N-body simulations with N as high as allowed by the present-day computational technology and software, in order to strengthen confidence in the statistical models based on the Fokker-Planck and other approximations. To simulate for example an $N = 10\,000$ star cluster to core bounce one needs to calculate some 900 initial half-mass crossing times. To do this accurately a high-order integration scheme with high intrinsic accuracy and no artificial softening of the potential is necessary. The N-body codes of Aarseth (1963, 1985) using direct force calculation have long time served as standard tools for this ambitious purpose. While the version NBODY5, described in his 1985 review remained the top-level tool for long time (see e.g. Spurzem & Aarseth 1996), recent advances in computer hardware (parallel computers, special purpose machines) made improvements of the software necessary as well. In particular a detailed and thorough examination of the scaling behaviour of the computational time for well-used algorithms as a function of particle number N and processor number n_{proc} on a parallel machine is necessary to find the optimal strategy for implementation and production runs. Before discussing the specific new measurements to be presented in this paper, a brief overview is given over the previous work in this field.

2 PERFORMANCE ANALYSIS OF DIRECT N-BODY SIMULATIONS

In their pioneering study of the performance of N-body simulations Makino & Hut (1988, henceforth MH) found that in star clusters which have only a small density contrast in the system the performance gain of an individual time step scheme (each particle has its own timestep) is of the order

$O(N^\alpha)$, where α is one or larger. Therefore a fast multipole method to compute the gravitational force (Greengard & Rokhlin 1987) is not appropriate here. A direct force integrator is asymptotically an $O(N^2)$ algorithm, whereas the fast multipole method is an $O(N)$ algorithm (Greengard 1990). But the Aarseth type schemes are very efficiently tailored for individual time steps (Aarseth 1985), and this outweighs the advantage of the fast multipole method, which requires the same timestep for all particles of the system to reach its optimal $O(N)$ scaling (Greengard 1990). The given arguments apply for the asymptotic scaling of the methods only, so a more detailed comparison of absolute timings could be useful. Unfortunately at present there is no fast multipole code available, which can compete with the other special features of the Aarseth codes, as e.g. regularization of close two-body motion, high accuracy Hermite integration scheme, higher order chain regularizations. It is worth mentioning here, however, one step in that direction, an accurate individual time step $O(N \log N)$ integrator based on a TREE algorithm (McMillan & Aarseth 1993).

Considerable space in MH is devoted to the examination of a two-timestep scheme (neighbour scheme) like NBODY5. Here each particle has two timesteps: a small one for near-neighbour interactions and a longer for distant interactions (Ahmad & Cohen 1973, henceforth AC). With the best choice of the average neighbour number the ratio of the timesteps can be of the order of ten (see Figs. 2a, b). In the short time intervals only the force exerted by the neighbours is updated, the distant force extrapolated by a low order Taylor series; only at the large time intervals the full force is calculated and the neighbour lists updated. Although such method does not change the principal $O(N^2)$ scaling of a direct brute force N-body integrator, it significantly reduces the absolute CPU times needed for a fixed particle number and physical time interval of simulation. The efficiency of the neighbour scheme when implemented on a general purpose parallel machine (CRAY T3D) is in the focus of the present study.

In order to parallelize the algorithm the individual time step scheme has to be changed to a hierarchical time step scheme, in which timesteps of all particles are commensurate, such that at any given time an entire group of particles is due for (parallel) force calculation; this is achieved by allowing only variations of timesteps with a fixed factor of two instead of arbitrary variations in the individual time step scheme (Makino & Aarseth 1992). Another improvement was introduced in the cited paper, namely a change of the integrator used to advance the particles on their orbit, from the previous four-step Adams-Bashforth-Moulton integrator to another two-step Hermite integrator. The Hermite scheme allows on the average twice as large timesteps for the same accuracy and also its storage requirements are less; this new version of the code (Makino 1991, Makino & Aarseth 1992) is denoted by NBODY6.

Simultaneously NBODY4 has been developed, which also includes the Hermite integrator and a hierarchical time step scheme, but does not include the AC neighbour scheme (Aarseth 1996). The reason to create NBODY4 is the development of special purpose computers for gravitational N-body force calculations, like the Tokyo Teraflop machine, which are not well suited to dividing the force between neighbours and non-neighbours (Sugimoto et al. 1990, Taiji 1996).

MH found in their analysis of the performance of N-body simulations that, using an optimal neighbour number of $N_n \propto N^{3/4}$, the gain in computing speed $\alpha_{\rm AC}$ compared to the standard individual time step scheme would be $\alpha_{\rm AC} \propto N^{1/4}$. This seems not to be too much, but since even with the most advanced supercomputers a complete simulation of a globular cluster ($N \approx 10^5$) is not yet possible, the machines are used at the maximum possible particle number with huge computational times of the order of weeks or months for single runs. Therefore even a moderate speedup factor is highly desirable, and it is worth while to revisit the analysis of MH by including the effects of a possible parallelization, to look again for optimal neighbour numbers, and to compute a handy formula to estimate computational times as a function of N, N_n, $n_{\rm proc}$ (the number of processors working in parallel) and other parameters of the code. The theoretical formula was checked (and slightly adapted) to represent well the timings of our actual code implementation on a CRAY T3D and we believe it can be used with confidence to extrapolate for larger N. This paper is organized as follows: first we briefly describe our code implementation on the CRAY T3D, thereafter the theoretical ansatz by MH to estimate the performance is presented and checked against the empirical timings. Finally the conclusion summarizes the results.

3 NBODY6++

In close cooperation with S. Aarseth the NBODY6 code was further developed to NBODY6++, which has been implemented on general purpose parallel computers. A key to the parallelization is the average size $N_{\rm gr}$ of the particle groups, the number of particles at one timestep for which a force calculation and orbit integration is simultaneously necessary (compare Figs. 1 and next section). The loop in the main integrator over all particles of a group was split into two parts, one containing the neighbour force computation, a second one over a subset of the particles of the group, which is due for a regular (non-neighbour) force calculation as well. Both loops now are executed in parallel. The parallelization strategy on the CRAY T3D, which is a MIMD computer with (in Edinburgh EPCC) 320 processing elements (PE's, max. 256 accessible for the single user), was SPMD (single program, multiple data), meaning that the same source program ran on all PE's, working -in the parallel code sections- on different data. After each parallel force calculation new positions, velocities, forces, force time derivatives, neighbour lists, time steps and some more data per particle have to be transmitted to all other processors for $N_{\rm gr}$ particles, where $N_{\rm gr}$ is the size of the particle group. The parallelization was done by using CRAFT (CRAY Fortran), which, by help of compiler directives automatically parallelizes the program and issues data send and receive instructions for the machine. Table 2 gives an overview for the total CPU times $t_{\rm tot}$ needed by NBODY6++ to integrate one reference time (half-mass crossing time of Plummer's model) as a function of particle number N and processor number $n_{\rm proc}$ (initialization time not included). The meaning of the other columns in Table 2 will be discussed in Sect. 4.

Table 1. Glossary

N	total particle number
N_n	optimal or average neighbour number
$N_{\rm gr}$	average number of particles to be advanced at the same time (i.e. being member of a group)
$N_{{\rm gr},i}$	as $N_{\rm gr}$, but for an individual particle at a particular time step.
S_i	(theoretical) speedup if the work to compute forces for $N_{{\rm gr},i}$ particles is perfectly distributed on a given number of processors on a parallel computer
S	average speedup for force calculation on a parallel computer estimated from Eq. (3).
$N_{\rm irr}$	cumulative number of irregular (small) time steps
$dt_{\rm irr}$	average irregular (small) time step
$N_{\rm reg}$	cumulative number of regular (large) time steps
$dt_{\rm reg}$	average regular (large) time step
dt_4	average time step determined from full force (as used in NBODY4)
$n_{\rm proc}$	number of processors or processing elements on a parallel computer (PE's)
γ	average ratio between large and small time steps
α	average ratio between dt_4 and small time steps
η	average ratio between large time steps and dt_4
$\alpha_{\rm AC}$	speedup gained by Ahmad-Cohen scheme relative to standard individual time step scheme
$C_{\rm tt}$	CPU time per particle per irregular time step (henceforth c.p.t.)
$t_{\rm est}$	total CPU time estimated from $t_{\rm est} = C_{\rm tt} N_{\rm irr}$
$t_{\rm tot}$	total CPU time measured; note that $\Delta t = t_{\rm tot} - t_{\rm est}$ is due to inaccuracies in all estimated scalings used for the quantities in Eq. (1), and to inter-processor communication overhead on parallel machines.
k_i	c.p.t. for certain parts of the simulation, as described below (on parallel computers k_i measure the computational time extrapolated down to one processor. Note that k_i are defined such that they depend on N, N_n, …, but not on $n_{\rm proc}$ of a parallel computer.
β_i	speed-up of times measured by k_i on a parallel computer (henceforth s.).
$\hat{k}_i$	c.p.t. on a parallel computer (depends on N, N_n, …, and $n_{\rm proc}$; $\hat{k}_i = \beta_i k_i$
β_0, k_0	s., c.p.t per pairwise (irregular) force calculation.
β_{0r}, k_{0r}	s., c.p.t per pairwise (regular) force calculation.
β_2, k_2	s., c.p.t per neighbour required for predictions.
β_4, k_4	s., c.p.t per neighbour to maintain neighbour lists. (e.g. polynomial corrections, list update).
β_6, k_6	s., c.p.t for everything not included above.

Table 2. Data for one initial half-mass crossing time simulation by NBODY6++ on the CRAY T3D, times in seconds; quantities explained in main text (Sect. 4).

$n_{\rm proc}$	N	N_n	$N_{\rm irr}$	$t_{\rm tot}$	$t_{\rm est}$	$\Delta t/N_{\rm irr}$
8	3500	176	2.80 E+06	1585	1456	4.61 E-05
64	3500	"	"	507	374	4.75 E-05
128	3500	"	"	539	271	9.57 E-05
8	6000	104	6.30 E+06	3653	2631	1.62 E-04
16	6000	"	"	2173	1580	9.41 E-05
64	6000	"	"	1106	693	6.56 E-05
128	6000	"	"	1148	518	1.00 E-04
64	8000	106	9.35 E+06	1733	1063	7.17 E-05
128	8000	"	"	1602	782	8.77 E-05
256	8000	"	"	2304	625	1.80 E-04
64	15000	107	2.24 E+07	4760	2758	8.94 E-05
128	15000	"	"	4056	1959	9.36 E-05
256	15000	"	"	5250	1519	1.67 E-04

4 ESTIMATE OF THE TIMINGS

Our basic ansatz for estimating the CPU time (guided by, but slightly changed here compared to MH) is

$$C_{\rm tt} = \frac{1}{\gamma}\Big(\underset{(A)}{\beta_{0r}k_{0r}N} + \underset{(B)}{\beta_4 k_4 N_n}\Big) + \underset{(C)}{\beta_0 k_0 N_n} + \underset{(D)}{\beta_2 k_2 N_n} + \underset{(E)}{k_6} \qquad (1)$$

All notation is kept as close as possible to MH; $C_{\rm tt}$ is the amount of computing time per small timestep. It consists of five summands on the right hand side of Eq. (1), which will be referred to in the following as parts A – E (left to right), as indicated in the line beneath the equation. Part A refers to the calculation of the full force; since this is updated in larger, regular timesteps only, a factor $1/\gamma$ is applied here, where γ is the average ratio of large to small timesteps. The factor of proportionality k_{0r} measures the CPU time necessary per force calculation (force calculation refers to the computation of the force between two single particles, so the computation of the total force on one particle consists of $N-1$ force calculations). In contrast to MH k_0, the time constant for calculation of one force during the neighbour force calculation, is distinguished here from the one (k_{0r}) for the total force, because measurements show it may differ from it, e.g. according to different vector length on vector computers. Part B is the overhead necessary for any change of the neighbour lists (occurring in the large time step intervals as part A), where k_4 measures the time for this normalized to the average neighbour number. Part C is the time for the neighbour force calculation, k_0 measuring the time per pairwise force again.

Part D is due to predictions; let us consider the members of a group of particles (called here test particles, number $N_{\rm gr}$), defined by the condition, that they are due to be advanced on their orbit at a given time. In order to determine the neighbour force exerted on the test particles, positions and velocities of all neighbour particles have to de extrapolated by a low-order Taylor series to the present time, because the neighbour particles generally will have incompatible time steps with the test particles. The standard approach is just to predict separately every neighbour for all test particles. In this case the computational effort for prediction per timestep should scale just with N_n (consider here for simplicity N_n as the average neighbour number of the test particles), hence k_2 in Eq. (1) should be independent of N and N_n. Practically it occurs, that very often the test particles have overlapping neighbour memberships, i.e. the merged set of the neighbours contains much less than $N_{\rm gr} \times N_n$ particles. If in an average sense the inequality $N_{\rm gr} \times N_n > N$ holds, it is on a sequential computer more efficient to just predict all particles to the present time. As one can see in Fig. 3b, d the measured k_2 on a sequential computer decreases with N_n, which proves the success of the above described concept.

However, the present concept is poor on a parallel machine, because it inhibits parallelization with respect to the $N_{\rm gr}$ test particles. As a result we see in Fig. 4 that k_2 is constant as a function of the processor number used. In a more recent version of the code we have resorted again to a full neighbour prediction separated for the test particles. Then the neighbour prediction can be shifted into the parallel section of the code and k_2 drops with increasing parallelization (not plotted in this paper).

Finally, Part E contains all overhead other than force calculation and neighbour list changes, mainly advancing the particle, accuracy checks, and so on. However, code sections needed especially on the CRAY T3D to communicate new particle data to other processors, were *not* included in the measurements for k_6. The β-parameters are provided to include the speedup achieved as a function of processor number on parallel computers. For sequential machines $\beta_i = 1$ for all i. All scalings of the above terms (e.g. with particle number N or average neighbour number N_n) have been taken from the previous literature MH or plausible estimates; they are subject to confirmation by comparisons with our measurements.

We have performed a series of rather short runs, each extending over a maximum of two initial half-mass crossing times, starting with an initial Plummer's model. Computational times of the corresponding sections of the code were measured with system routines and the results will be discussed in the following for a normal Sun workstation, a CRAY single CPU vector computer, and a CRAY T3D massively parallel computer. We have changed the AC scheme as compared to the description in Aarseth (1985) such that the neighbour number is not adjusted self-consistently. For our measurements it is more convenient to prescribe an optimal neighbour number, such that it is always tried to keep the actual neighbour number as close as possible to it, without allowing too fast a change. This prescribed neighbour number, which for all our test runs was sufficiently close to the actually sustained average neighbour number is occurring as the abscissa N_n in some of the following data and was varied from 20 to 200. Note that in Table 2 we selected only $N_n \approx 100$ data in order not to overload it. The particle numbers used to produce the plots were chosen from the series 100, 350, 600, 1000, 3500, 6000, 10000. From the results of our measurements absolute values and scaling relations for the quantities k_i and β_i occurring in Eq. (1) are derived as will be discussed in this section below.

The data given in Table 2 are to be interpreted as follows: $N_{\rm irr}$ is the total number of irregular (small) steps counted in the run, $t_{\rm est} = C_{\rm tt} N_{\rm irr}$ is the estimated total time for the run, provided communication overheads between the processing elements are negligible. For the value given $C_{\rm tt}$ was computed according to Eq. (1) and the values for k_i and β_i derived in the following were taken. The real computational time measured $t_{\rm tot}$ (excluding initialization) is given as well; thus the quantity $\Delta t/N_{\rm irr} = (t_{\rm tot} - t_{\rm est})/N_{\rm irr}$ in the rightmost column of Table 2 measures the amount of CPU time represents all inaccuracies due to the application of simple scaling laws to determine the various quantities occurring in Eq. (1); on parallel computers this quantity is also affected by overhead for inter-processor communication as well as overhead to initialize and synchronize parallel execution in the suitable code segments. So for fixed N an increase in $\Delta t/N_{\rm irr}$ as a function of $n_{\rm proc}$ marks the point at which the speedup due to parallel execution is more and more compensated by losses due to data fragmentation on too many processors (see Table 2).

Before discussing the measured timings in more detail the inherent parallelism of the NBODY6++ algorithm shall be explored. A speedup S of a particular run is defined in the standard way as $S = t_{\rm lin}/t_{\rm par}$, i.e. the ratio of the computing times needed in linear mode (on one processor) and in the case of parallel execution. Semi-theoretical speedups for the code sections containing the parallel do loops to compute the regular or irregular forces for a group of particles are determined. For large N and computational time these two sections of the code are the dominant time consuming parts. Note further, that these parallel code sections do not only consist of the force calculation, but also contain updates of the neighbour list, advance of the particles by the corrector, determination of new timesteps and some more special procedures related e.g. to regularizations. The theoretical speedups are computed as follows: in a linear simulation run at each timestep dt_i the quantity

$$S_i = \frac{N_{{\rm gr},i}}{{\rm int}(N_{{\rm gr},i}/n_{\rm proc}) + {\rm mod}(n_{\rm proc} + N_{{\rm gr},i},\, n_{\rm proc})} \qquad (2)$$

was computed, where ${\rm int}(x)$ denotes the function giving the next integer smaller as x and ${\rm mod}(i,j)$ is the rest of the integer division i/j. The plausibility of the ansatz Eq. (2) can be checked by some simple cases as e.g. that for $N_{{\rm gr},i} = 1$ it is always $S_i = 1$ ($n_{\rm proc} \geq 1$), or that for $N_{{\rm gr},i} = n_{\rm proc}$ it follows $S_i = n_{\rm proc}$. S_i is the theoretical maximum speedup for the timestep labelled by i, computed by assuming that there are $N_{{\rm gr},i}$ particles to be advanced at that time on $n_{\rm proc}$ processing elements. Any losses by overheads due to the initialization of a parallel loop or due to data communication are neglected. Thus

$$S = \frac{N_{\rm irr}}{\sum_{dt_i} (N_{{\rm gr},i}/S_i)} \qquad (3)$$

where $N_{\rm irr}$, the total number of integration steps, is an estimate of the maximum theoretical speedup for the entire run, provided the approximations are valid. For the regular force calculation correspondingly group sizes of the regular step and the total regular step number $N_{\rm reg}$ have been taken. Figs. 1a, b plot S as a function of the processor number for different particle numbers.

One can easily see that for the regular force calculation, which is the dominant one for large N, the algorithm contains enough inherent parallelism to use even more than 1000 processors, provided N is 10^4 or larger. For the neighbour force calculation the situation is not that good, however, at 512 processors and 10^4 particles still a considerable speedup is achieved, which will become closer to the ideal speedup $S = n_{\rm proc}$ for even larger N. These theoretical conclusions, however, will only become true, if communication and synchronization losses for parallel code execution can be kept negligible. Nevertheless the result is remarkable, regarding the complicated hierarchically nested two-level time step structure of NBODY6++. It should be noted, however, that this can change drastically in systems with a different spatial structure than the here used initial Plummer's

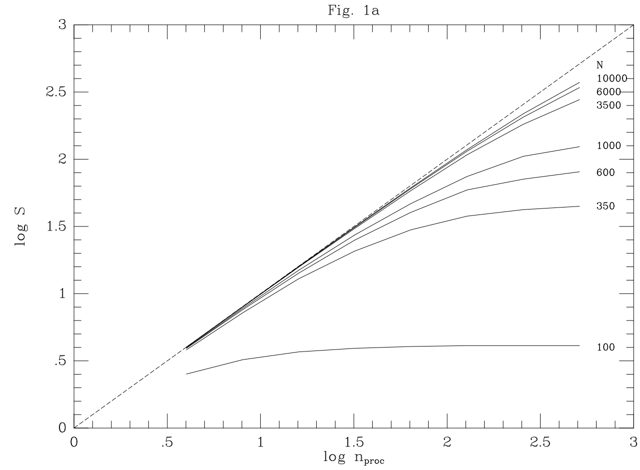

Figure 1a. Theoretical speedup (neglecting communication) of regular force calculation as a function of processor number for varying particle number N. The dashed line is the ideal maximum speed up which could be reached on a given processor number.

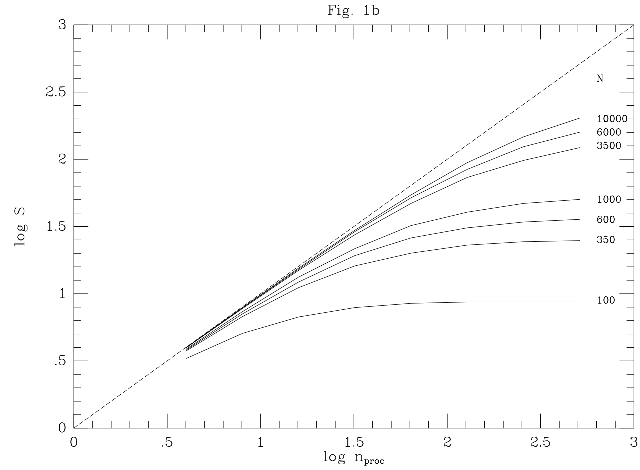

Figure 1b. As Fig. 1a, for the neigbour force calculation.

sphere. In general the speedup will be much less in strongly spatially concentrated systems.

Fig. 2a illustrates that the group size $N_{\rm gr}$ of a bunch of particles to be integrated in parallel is roughly proportional to $N^{2/3}$ (Fig. 2a), although the group size for the regular force does seem to increase even a bit more, and it shows γ, the average ratio of the large to the small timestep as a function of N. It increases with some power of N, in contrast to MH, whose Eq. (44) implies that γ does not vary with N. In Fig. 2b γ only is shown as a function of the neighbour number. Surprisingly the variation of γ is very small, an estimate of a power-law would be of the order of $\gamma \propto N_n^{1/20}$. This is again in contrast to the estimate of MH, that γ would scale as $N_n^{1/3}$. If we look at the maximum timesteps of our distribution, however, they seem to be consistent with MH, since both decrease with $N^{-1/3}$ (Fig. 2c). If the same would be true for the *average* timestep ratio, γ would be independent of N, as Eq. (44) of MH suggests. This is not the case as Fig. 2a shows, and it suggests, that the size of the neighbour sphere does not determine the average regular step, but another radius as e.g. the core radius (compare Eq. (42) of MH). Since this (in N-body units) is independent of N it could introduce a variablity of γ with N. Such difference to the analysis of MH is possibly due to the fact that we start with an already spatially structured system (Plummer's sphere) in the simulations, whereas MH assumed for their simplest case an infinite homogeneous system.

In the next series of Figs. 3a-d we display our measurements of the quantities k_i from Eq. (1) as a function of particle and neighbour number.

For all cases the contribution from part E of Eq. (1), related to the constant k_6 was negligible, as to be expected, so it was not plotted. Furthermore we see, that the distinction between k_0 and k_{0r} for the neighbour and full force calculation is justified, especially for the CRAY, where longer vector length give a speed-up even for the single pairwise force. Both of them are also constant with N and N_n, which means that parts A and C of Eq. (1) give a good representation of the scaling behaviour of the computing time. However, the situation is different for k_2 and k_4. Both exhibit a small increase with N and a fair decrease with N_n. If we fit our empirical variations with power-laws for the workstation data (in order to avoid perturbations due to vectorization effects on the CRAY) we find

$$\begin{aligned} k_2 &\propto N^{1/10} N_n^{-3/4} \ , \\ k_4 &\propto N^{1/5} N_n^{-3/4} \ . \end{aligned} \tag{4}$$

The dependence on N is so small that we do not discuss it further here. A variation of k_2 with the neighbour number comes from our special prediction scheme (see above). More difficult to understand is the variation of k_4, which tells us that the change of neighbours per particle per large timestep is not proportional to N_n, as assumed in Eq. (1) (and k_4 would be constant), but it costs something varying with a smaller power of $N_n^{1/4}$ only.

Having established a quantitative model for the timing behaviour on a single CPU now we turn to the multi-CPU CRAY T3D. Again timings of the relevant code sections as before were taken with system clock routines and the quantities $\hat{k}_i := \beta_i k_i$ derived from that and plotted in Fig. 4 as a function of the processor number. Since k_i are independent of the processor number we can deduce from scaling laws fitted to Fig. 4 the variation of β_i with the processor number, i.e. the speedup of the corresponding code sections (see Eq. 5).

From the data we conclude that the speedup of the force calculations is indeed close to the ideal one discussed earlier, even already for moderate particle numbers of a few thousand. However, the other, usually on sequential computers non-dominant parts of the total computational time (parts B and C of Eq. (1), related to neighbour list updates and prediction) become dominant at large processor numbers. The quantity $\hat{k}_2$ containing predictions of particle positions was first not parellelized at all, because as discussed above in most cases predictions over all particles were carried out in a separate loop before the force calculations. To parallelize this comparably small loop caused too much overhead time for data communication and synchronization. So for the CRAY T3D the prediction algorithm was again changed to the standard method to predict all neighbours separately for each particle, which could be done in the parallel section for the irregular force calculation. $\hat{k}_2$ plotted in Fig. 4 still

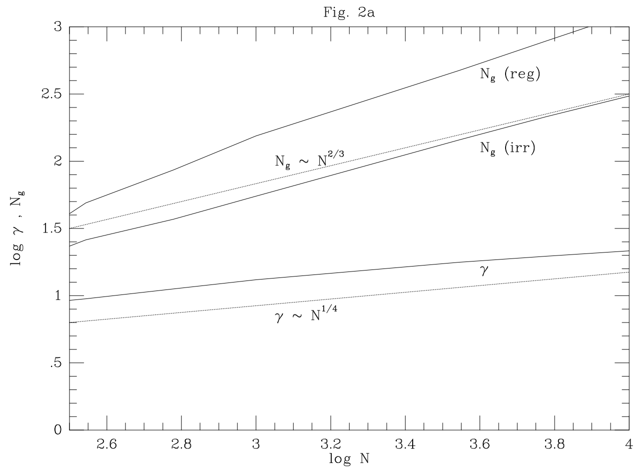

Figure 2a. Number of particles (group size $N_{\rm gr}$) whose regular or irregular (neighbour) force can be integrated in parallel, and γ, the ratio of regular to irregular timestep as a function of particle number N. The expected scaling $N_{\rm gr} \propto N^{2/3}$ for the group size and an estimated scaling of $\gamma \propto N^{1/4}$ are indicated.

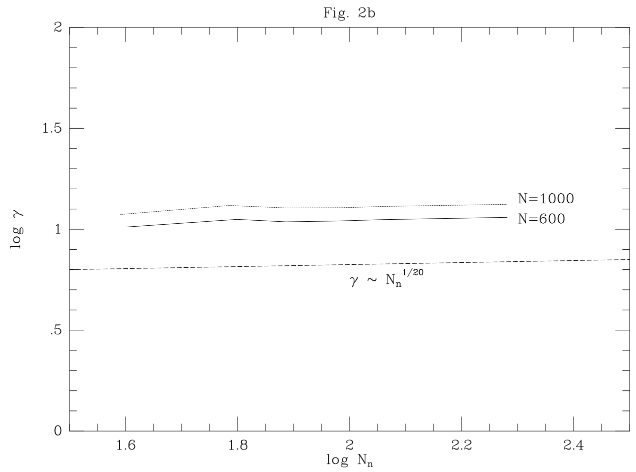

Figure 2b. γ as in Fig. 2a, but here as a function of neighbour number.

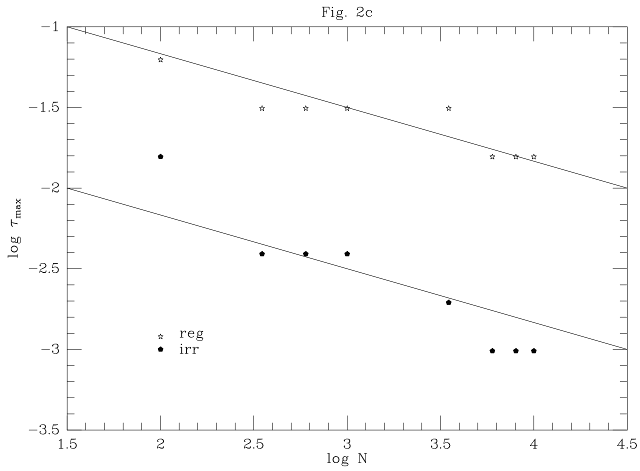

Figure 2c. Maximum timestep (regular/irregular) as a function of particle number (dots, note discrete changes due to hierarchical time step scheme, which only allows the timestep to change by a factor of two) compared with theoretical expectation (solid lines $\propto N^{-1/3}$).

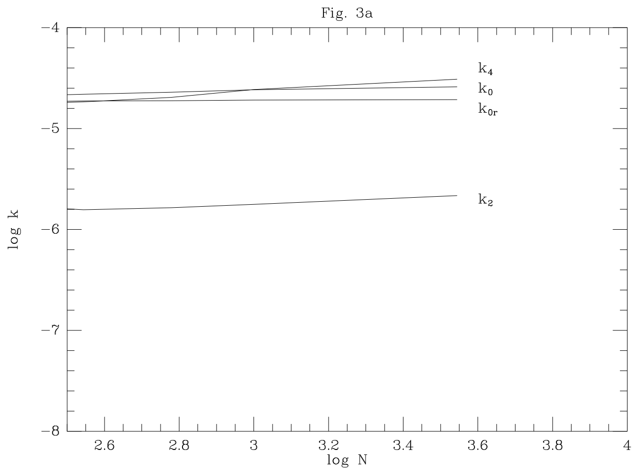

Figure 3a. Time constants k_i for $i = 0, 0r, 2, 4$ as defined in the text as a function of particle number N for the Sun workstation.

results from the non-parallel prediction and thus is a constant, With the parallelized prediction scheme a variation of $\hat{k}_2 \propto n_{\rm proc}^{-1/2}$ (not plotted) was achieved.

We have fitted the results of Fig. 4 with power laws as a function of the processor number, which resulted in

$$\begin{aligned} \beta_{0r} &\propto n_{\rm proc}^{-1} \ , \\ \beta_0 &\propto n_{\rm proc}^{-5/6} \ , \\ \beta_2 &\propto n_{\rm proc}^{-1/2} \ , \\ \beta_4 &\propto n_{\rm proc}^{-5/12} \ . \end{aligned} \tag{5}$$

With these values inserted in Eq. (1) and taking the measured times for one processor and $N = 1000$ as a normalization one can estimate computing times for any other particle, neighbour, or processor number. In this way the data in the column labelled $t_{\rm est}$ of Table 2 were computed. The separately measured code sections corresponding to parts A – E of Eq. (1) do not contain those pieces added especially on the T3D for communication of new particle data to the other processors. The difference between the time calculated according to Eq. (1) ($t_{\rm est}$) and the cpu time measured in total ($t_{\rm tot}$) contains as the dominant part the time needed for data communication on the parallel machine. Its value per irregular timestep has been computed for Table 2. The yet sparse data material allows only for a qualitative conclusion here, namely, that for small processor number the communication overhead seems to increases with N significantly, whereas for large processor numbers it is to first order independent of N, but increases with $n_{\rm proc}$. In the first case there are many particles of a given group attached to each processor, whereas in the second case there are fewer particles allocated on each processor, but a lot of messages from other processors to be handled due to the increased data fragmentation.

One can see in Table 2 that this increased communication overhead is critical for the largest N and $n_{\rm proc}$, where indeed communication already dominates the total time, and the speedup one would expect from Fig. 4 is completely compensated. Here it is not intended to derive a quantitative model for these overheads by the use of data like those in

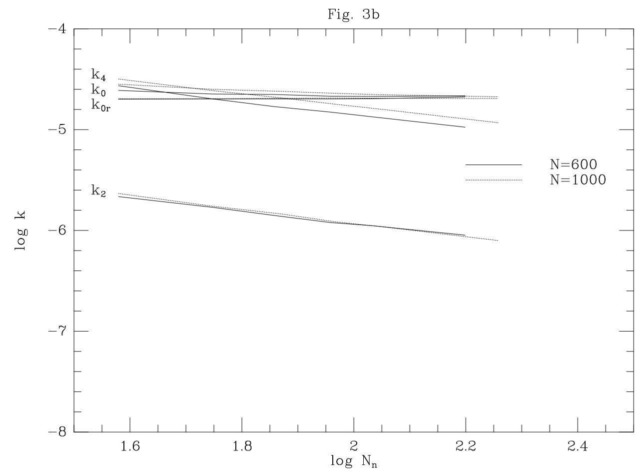

Figure 3b. As Fig. 3a, now as a function of neighbour number N_n, for two different N, as indicated.

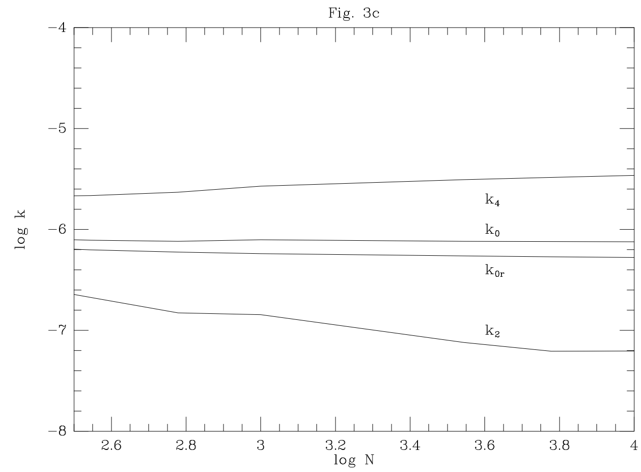

Figure 3c. As Fig. 3a, but for the single CPU CRAY vector computer.

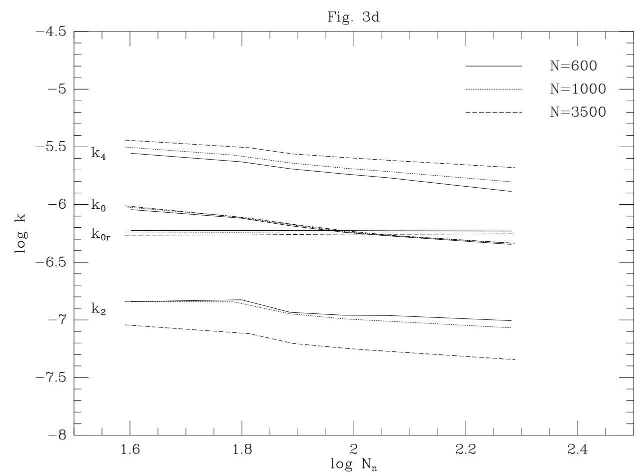

Figure 3d. As Fig. 3b, but for the single CPU CRAY vector computer.

the rightmost column of Table 2. The present implementation of data communication does not explicitly use the low-level *shmem* data communication subroutines of the CRAY T3D (instead it relys on implicit communication initiated by CRAFT, CRAY Fortran); as part of future work the communication will be optimized by hand-written calls of

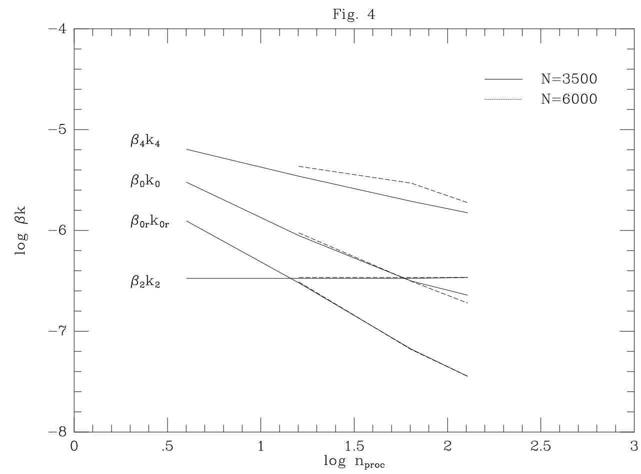

Figure 4. Quantities $\hat{k}_i = \beta_i k_i$ for $i = 0, 0r, 2, 4$ and two indicated particle numbers as a function of processor number on the CRAY T3D.

shmem-routines instead of leaving this task to the CRAFT Fortran interpreter. It is concluded that at least for the present implementation 64 or 128 processors are an optimal number of processors for the use of NBODY6++, whereas larger processor numbers do not yield any reasonable further speedup due to communication requirements.

5 THE OPTIMAL NEIGHBOUR NUMBER

MH derived an optimal neighbour number for the Ahmad-Cohen scheme proportional to $N^{3/4}$ by neglecting part B of Eq. (1) and assuming that k_2 does not vary with N_n. According to our results this is questionable, because on one hand side even on a single processor, to make part B negligible it must be $k_4 \ll k_{0r} N/N_n$. On a normal workstation this is the case, since (see Fig. 3a) it is $k_4 \sim k_{0r}$. For the vector CRAY, however, k_{0r} could be an order of magnitude smaller than k_4 (Fig. 3c), so N has to be large compared to ten times the average neighbour number. The assumption breaks down, however, if the effects of parallelization are taken into account - here the gap between $\hat{k}_{0r}$ and $\hat{k}_4$ can be as large as two orders of magnitude. Moreover, the determination of $dC_{\rm tt}/dN_n$ becomes more difficult, if the correct variations of k_2 and k_4 with N_n are taken into account. So we are not able to present a handy formula for the optimal neighbour number from our formula Eq. (1); what one can say that in most cases a rather small neighbour number of the order of 50 neighbours is sufficient even for very large N, because γ is already large there and does not vary much by increasing N_n. So other considerations can be used to select a neighbour number, which may result from the actual hardware used.

6 COMPARISON WITH NBODY4

NBODY4 is an N-body integrator with single (but individual) timestep for each particle, i.e. at each timestep the full force due to all other particles is computed. It has been implemented recently on special purpose HARP computers (Aarseth 1996). This code is less complex than NBODY6++ since it has no overhead for recomputations of the neighbour lists. Its CPU time requirements are to first order proportional to $k_{0r}N^2$ per timestep on a sequential computer. In order to compare timings of NBODY4 and NBODY6++ for the same physical time one needs to know the average ratio of their timesteps. In direct N-body algorithms of the Aarseth type timesteps are determined as a function of the force and its higher derivatives (for details see Aarseth 1985). Basically a highly fluctuating force gives small timesteps and v.v.. Let $dt_{\rm irr}$ and $dt_{\rm reg}$ be characteristic average timesteps of the AC neighbour schemes (NBODY6++); they are both computed by the standard formula, but using the forces exerted by different sets of particles (neighbours and non-neighbours). dt_4, the average timestep of NBODY4 is derived from the full set of particles; of the three different forces (neighbour force, full force, non-neighbour force) the first one is the most fluctuating, the full force intermediate, and the non-neighbour force the least fluctuating force. Therefore it is expected that on the average $dt_{\rm irr} < dt_4 < dt_{\rm reg}$, a relation which will be empirically checked now.

We have measured the ratio $\alpha = dt_4/dt_{\rm irr}$ of the timestep determined from the full force relative to the small irregular timestep in NBODY6++. It varied from $\alpha = 1.6$ for 1000 particles to $\alpha = 2.8$ for $N = 20000$, which is approximately a variation as $\alpha \propto N^{1/6}$. Asymptotically for large N we can extrapolate the CPU time ratio η of NBODY4 over NBODY6++ from our data as $\eta = \gamma/\alpha \propto N^{1/12}$ (here the approximate scaling of γ from Fig. 2a was used). This is asymptotically a small variation for large N, but if we take the actual values for $N = 10\,000$ particles of $\gamma \approx 20$ (Fig. 2a) and $\alpha = 2.46$ (measured) then $\eta \approx 8$. $\eta = dt_{\rm reg}/dt_4$ measures the possible advantage (if neighbour overhead could be neglected) of the neighbour scheme NBODY6++ as compared to the full direct force integrator NBODY4; in a naive approach one might think that this possible gain of the neighbour scheme would be measured by γ directly. We see that this is not the case, because η is some factor of two smaller than γ. But on the other hand even for 10000 particles there is a potential gain of the neighbour scheme given by η of the order of ten still remaining. It has to be proven in future work whether it will be possible in real implementations of the neighbour scheme for very large particle numbers to realize this asymptotically estimated behaviour. From our present set of data this is not yet conclusive, we have to accumulate more data for large N and bypass the present obstacle of communication between the processors. This will be subject of future work.

7 CONCLUSION

We have analyzed the performance of the direct force N-body integrator NBODY6++ for the first time on a massively parallel general purpose supercomputer (CRAY T3D) and compared the results with scalar and vector computers. NBODY6++ is an Aarseth-type N-body code (Aarseth 1985) including an Ahmad-Cohen (AC) neighbour scheme with hierarchical time steps on two levels and a Hermite integration scheme (Makino & Aarseth 1992). The algorithm turns out to be very efficiently parallelizeable, however communication and synchronization overheads inhibit further speedup of the computation if using more than 128 processors. An improvement of the performance by using specialized hardware communication routines (*shmem*) on the CRAY T3D is possible and subject of future work.

The present timings of NBODY6++ do not yet beat those of NBODY5 on single CRAY YMP (on the average 5400 seconds per crossing time, Spurzem & Aarseth 1996) by more than a moderate factor of two or three. In regard of the challenging task of N-body simulations for very large star clusters even this is a valuable improvement. But still the present implementation of the code NBODY6++ on the CRAY T3D, as it was discussed here, leaves room even for considerable improvement by using the special hardware communication routines *shmem* supplied by CRAY, work which is in progress now. It is expected that our presented runs here for the largest particle numbers and processor numbers, which were limited by communication overheads, will improve considerably by that.

Another topic of interest was the optimal neighbour number of the Ahmad-Cohen (Ahmad & Cohen 1973, AC) neighbour scheme. It turns out that even for very large N already rather small neighbour numbers of the order of 50 are sufficient to take full advantage of the neighbour scheme, this is having a regular timestep for updating the full force which is about one order of magnitude larger than the small timestep, with which only the neighbour force is updated. This is in contrast to Makino & Hut (1988, MH), who claim that the optimal neighbour number has to be of the order of $N^{3/4}$. The reasons are twofold. On one hand-side we could show, that on a massively parallel machine terms in the semi-analytic estimate for the CPU timings of direct N-body simulations (see Eq. 1), which have been explicitly neglected by MH in their analysis are not small anymore relative to the other ones. Secondly, even for a sequential vector computer or standard workstation we could not find that a neighbour number increasing by $N^{3/4}$ is optimal. Instead, the timestep ratio γ between the large, regular timestep and the small, irregular timestep of the neighbour scheme, which determines its efficiency, appears to be approximately constant as a function of neighbour number (see Fig. 2b). A possible reason for this is that we did our analyses with an already spatially structured Plummer's sphere, whereas the analysis of MH, at least in their simplest cases which we considered here, assume an infinite homogeneous system.

Unfortunately proper estimates of the timings of direct N-body simulations with a neighbour scheme for massively parallel machines, taking all relevant terms into account, do not allow for a straightforward determination of an optimal neighbour number as in the case of MH. This is mainly due to the fact that the efficiency factors like γ or η (see previous section) have such a surprisingly small variation with the neighbour number, even down to values of $N_n = 20$ for ten thousand or more particles. It is clear that they have to drop to one if N_n approaches unity; this result is interpreted such that most of the fluctuations of the force (which reduce the timestep) are already taken out by the first few neighbour particles. So the regular timestep is sufficiently large already for a very small neighbour number. Such result seems to be consistent with findings of Casertano & Hut (1985), that for a statistically reliable neighbour density of a particle in an N-body simulation as few as about five neighbour particles are sufficient.

These results encourage us not to neglect the possible advantage of the AC scheme against the simple full integrator NBODY4, calculating the full force at every small timestep, and to spend further effort in its software development and implementation. for general purpose massively parallel computers as well as for special purpose computers (HARP, see Aarseth 1996, Taiji 1996). On general purpose parallel machines further improvements like the parallelization of the integration of a lot of primordial binaries are quite straightforward to implement, in contrast to HARP. Present special purpose HARP computers, like the Tokyo Teraflop machine are amazingly fast in calculations of gravitational forces (see e.g. Makino & Ebisuzaki 1996). So it seemed that software developments for direct N-body simulations, which tend to increase the complexity of the remaining part of the code (not connected with force calculations) and which have to be executed on some host computer, would be obsolete. The reason is Amdahl's law, which basically says that the total speedup achieved by accelerating some fraction x of a code is asymptotically limited due to the time required to execute the not optimized fraction $1-x$ of it.

However, not everywhere such fast special purpose computers are available and there is a tradeoff between their specialization and flexibility, as in the mentioned case of primordial binaries. And even in addition to that the further development of clever algorithms of N-body simulations, their implementation and test, is considered as an interesting subject by itself. In regard of these considerations the further work with and improvement of the neighbour scheme in N-body codes is seen as a promising subject for future studies.

ACKNOWLEDGEMENTS

Without the continuous and friendly support by S. Aarseth (Cambridge) the author could not have coped with the task of developing the parallel version NBODY6++ of his code NBODY6. Thanks are also due to J. Makino, D. Sugimoto, and M. Taiji for critical discussions and for the warm hospitality by all colleagues there during my two visits at the Dept. of Arts and Sciences, Komaba Campus, Univ. of Tokyo, during which part of this work has been done. Travel support to Japan by DFG grant 446 JAP-113/18/0 is gratefully acknowledged. This work was supported by the TRACS programme of the Edinburgh Parallel Computing Centre (EPCC), by which a visit of the author in Edinburgh was financed. I wish to express my sincere gratitude to Douglas C. Heggie, who was the corresponding host there, for a lot of very helpful discussions and the warm hospitality by him and his department. The numerical calculations were performed at the CRAY T3D of EPCC and of the "Konrad-Zuse-Zentrum für Informationstechnik" (ZIB) in Berlin, the CRAY Y/MP of the HLRZ Jülich, and local Edinburgh Sun workstations. Special thanks are also due to CRAY Research and the "Rechenzentrum der Universität Stuttgart" (RUS) for providing additional CPU time for code development on their CRAY T3D.

REFERENCES

Aarseth S.J., 1963, MNRAS, 126, 223
Aarseth S.J., 1985, in Brackbill J.U., Cohen B.I., eds, Multiple time scales, Academic Press, Orlando, p. 378
Aarseth S.J., 1996, in Hut P., Makino J., eds, Proc. IAU Symp. 174, Dynamics of Star Clusters, Reidel, Dordrecht, in press
Ahmad A., Cohen L., 1973, J. Comp. Phys., 12, 389
Barnes J.E., Hut P., 1986, Nat, 324, 446
Bettwieser E., 1983, MNRAS, 203, 811
Bettwieser E., Sugimoto D., 1984, MNRAS, 208, 493
Breeden H.N., Cohn H., Hut P., 1994, MNRAS, 421, 195
Casertano S., Hut P., 1985, ApJ, 298, 80
Cohn H., 1980, ApJ, 242, 765
Cohn H., Hut P., Wise M., 1989, ApJ, 342, 814
Elson R., Hut P., Inagaki S., 1987, ARA&A, 25, 565
Giersz M., Heggie D.C., 1994a, MNRAS, 268, 257
Giersz M., Heggie D.C., 1994b, MNRAS, 270, 298
Giersz M., Spurzem R., 1994, MNRAS, 269, 241
Goodman J., 1987, ApJ, 313, 576
Greengard L., 1990, Computers in Physics, Mar/Apr 1990, 142
Greengard L., Rokhlin V., 1987, J. Comp. Phys., 73, 325
Heggie D.C., 1984, MNRAS, 206, 179
Heggie D.C., Mathieu R.M., 1986, in Hut P., McMillan S.L.W., eds, The Use of Supercomputers in Stellar Dynamics, Springer-Vlg., Berlin, p. 233
Hut P., 1993, 403, 256
Kustaanheimo P., Stiefel E.J., 1965, J. Reine Angew. Math., 218, 204
Louis P.D., Spurzem R., 1991, MNRAS, 244, 478
Lynden-Bell D., Eggleton P.P., 1980, MNRAS, 191, 483
Lynden-Bell D., Wood R., 1968, MNRAS, 138, 495
Makino J., 1991, PASJ, 43, 859
Makino J., 1996, ApJ, in press
Makino J., Aarseth S.J., 1992, PASJ, 44, 141
Makino J., Ebisuzaki T., 1996, ApJ, in press
Makino J., Hut P., 1988, ApJS, 68, 833, MH
McMillan S.L.W., Aarseth S.J., ApJ, 414, 200
Murphy B.W., Cohn H., Hut P., 1990, MNRAS, 245, 335
Spitzer L., 1987, Dynamical Evolution of Globular Clusters, Princeton Univ. Press, Princeton
Spurzem R., 1994, in Pfenniger D., Gurzadyan V.G., eds, Ergodic Concepts in Stellar Dynamics, Springer-Vlg., Berlin, p. 170
Spurzem R., 1996, in Hut P., Makino J., eds, Dynamics of Star Clusters, IAU Symp. 174, Kluwer, Dordrecht, in press
Spurzem R., Aarseth S.J., 1996, MNRAS, in press
Spurzem R., Takahashi K., 1995, MNRAS, 272, 772
Spurzem R., Theis Ch., 1996, MNRAS, to be subm.
Sugimoto D., Chikada Y., Makino J., Ito T., Ebisuzaki T., Umemura M., 1990, Nat, 345, 33
Taiji M., 1996, in Hut P., Makino J., eds, Dynamics of Star Clusters, IAU Symp. 174, Kluwer, Dordrecht, in press
Theis Ch., Hensler G., 1993, A&A, 280, 85

This paper has been produced using the Blackwell Scientific Publications TEX macros.