The Astrophysical Journal Supplement Series, 219:31 (12pp), 2015 August doi:10.1088/0067-0049/219/2/31

BLOCK TIME STEP STORAGE SCHEME FOR ASTROPHYSICAL *N*-BODY SIMULATIONS

Maxwell Xu Cai (蔡栩)[1,2], Yohai Meiron (林友海)[2,1], M. B. N. Kouwenhoven (柯文采)[2], Paulina Assmann[3,1,4], and Rainer Spurzem[1,2,5]

[1] National Astronomical Observatories, Chinese Academy of Sciences, 20A Datun Road, Chaoyang District, Beijing 100012, China; maxwell@nao.cas.cn
[2] Kavli Institute for Astronomy and Astrophysics, Peking University, 5 Yiheyuan Road, Haidian District, Beijing 100871, China
[3] Departamento de Astronomía, Universidad de Chile, Camino El observatorio 1515, Las Condes, Santiago, Chile
[4] Departamento de Astronomía, Universidad de Concépcion, Casilla 160-C, Concépcion, Chile
[5] Astronomisches Rechen-Institut, Zentrum für Astronomie, University of Heidelberg, Mönchhofstrasse 12-14, D-69120 Heidelberg, Germany

Received 2014 December 9; accepted 2015 June 23; published 2015 August 18

ABSTRACT

Astrophysical research in recent decades has made significant progress thanks to the availability of various *N*-body simulation techniques. With the rapid development of high-performance computing technologies, modern simulations have been able to use the computing power of massively parallel clusters with more than 10^5 GPU cores. While unprecedented accuracy and dynamical scales have been achieved, the enormous amount of data being generated continuously poses great challenges for the subsequent procedures of data analysis and archiving. In this paper, we propose an adaptive storage scheme for simulation data, inspired by the block time step (BTS) integration scheme found in a number of direct *N*-body integrators available nowadays, as an urgent response to these challenges. The proposed scheme, namely, the BTS storage scheme, works by minimizing the data redundancy by assigning individual output frequencies to the data as required by the researcher. As demonstrated by benchmarks, the proposed scheme is applicable to a wide variety of simulations. Despite the main focus of developing a solution for direct *N*-body simulation data, the methodology is transferable for grid-based or tree-based simulations where hierarchical time stepping is used.

Key words: globular clusters: general – methods: data analysis – methods: numerical – virtual observatory tools

1. INTRODUCTION

The gravitational *N*-body problem has posed a challenge ever since it was mathematically formulated in the 17th century by Isaac Newton. This problem of using initial conditions to determine the future motion of *N*-bodies interacting gravitationally among themselves continues to be relevant in modern day astronomy and is investigated in the context of planetary systems, star clusters, galaxies, and the universe. Mathematically, it is posed as $3N$ coupled nonlinear second-order ordinary differential equations. The solution consists of the phase-space paths of all particles as functions of time, which generally cannot be expressed by algebraic expressions or integrals.

Gravitational *N*-body simulations are currently the preferred approach for finding these solutions. They use particles to represent gravitating objects and propagate the initial conditions in time by calculating the force acting on each particle, and advancing it in time in small steps.

With advances in technology, simulations have become elaborate enough to take full advantage of computing capabilities. In particular, the availability of highly parallelized computing facilities, some using hardware accelerators (such as GRAPE, Makino & Taiji 1998[6]; FPGA[7] boards, Berczik et al. 2007; and more recently GPUs[8]) have contributed to recent progress in the field. Simulations are carried out with more particles than ever before, and for longer integration times. Modern simulations are also characterized by the inclusion of more detailed physical processes and requirements for higher numerical accuracy.

While powerful modern hardware has brought astrophysical simulations to unprecedented accuracy, complexities arise with the problem of storing the results, which is done by writing to the hard disk some or all of the properties of some or all of the particles at some pre-specified times; this is often called "taking a snapshot." The snapshot files can later be processed to learn about the evolution of the system. The storage requirement is determined by four factors: (1) the number of particles, (2) the size of the data record per particle, (3) the output frequency, and (4) the total integration time. In order to capture the detailed physical processes, high time-resolution of output is often necessary. Hence, there is a tradeoff between time-resolution and output size (or the availability of storage space and post-processing capabilities).

To illustrate this, consider some large cosmological simulations from the previous decade. The Millennium Simulation (Springel et al. 2005) followed about 10^{10} particles for nearly a Hubble time and produced only 64 snapshots of about 300 GB each. Similarly, the MareNostrum simulation (Gottloeber et al. 2006) used 2×10^9 particles and saved 135 snapshots of 64 GB each (running for a similar physical time). Even almost a decade later, this data volume still poses a challenge for storage, and more crucially for transport over a network and for analysis. Thus, there is a gap between computing power and data processing and management capabilities.

Direct summation techniques are often preferred when studying a system in which accurate orbital integration is needed and encounters are important, and/or physical assumptions have to be minimized, such as globular clusters and planetary systems. In cosmological simulations where the primary interest is to study the evolution of the large-scale structure, one often uses Tree methods (Barnes & Hut 1986) or the Fast Multipole Method (Greengard & Rokhlin 1987), which are generally much faster (for large *N*) but introduce an

[6] GRAPE: GRAvity piPEline.
[7] FPGA: Field Programmable Gate Array.
[8] GPU: Graphics Processing Unit.

approximation to the force contributions from very distant particles. Direct N-body simulations thus generally use a much smaller number of particles, today rarely more than $N = 10^6$.

Big data management has so far been in the domain of cosmological collisionless simulations due to the large number of particles. Despite the relatively small number of particles in direct N-body simulations, data output can be a challenge for this kind of simulations as well. The key challenge is that they attempt to follow accurately phenomena which happen on vastly different timescales: from white dwarf binaries, which have orbital periods of less than one hour (Brown et al. 2011), to stars orbiting in the outskirts of the cluster, which can take millions of years to complete one orbit. Calculating the evolution of the entire cluster based on the smallest time step or timescale is completely impractical (see an estimation in Section 3), and so most productive codes employ individual or hierarchical time step schemes like the Hermite scheme (Aarseth 2003). Saving the output, however, is usually done using snapshots in much the same way as for the large cosmological simulations.

The main problem with the snapshot approach is that no information is stored about what happens between snapshots. Interpolating will not always yield useful information if the process of interest occurs on a much shorter timescale than the snapshot interval. Examples of this are close encounters that may create hypervelocity stars (e.g., Yu & Tremaine 2003), resonances such as Kozai oscillations (e.g., Katz et al. 2011), the evolution of planetary systems (e.g., Hao et al. 2013), or supernova explosions. Those phenomena may be captured by the program and recorded separately, but there is no standardized way of doing so. For the same reason, it is difficult to make a smooth visualization of an energetically active subsystem. On the other hand, there might be redundant data for the dynamically inactive particles. For this reason, Farr et al. (2012) proposed an adaptive approach to data output in which only data that had recently changed during the last output interval will be written to files. They also proposed the Particle Stream Data Format (PSDF), yet another Markup Language (YAML)[9]-based structured text format to ensure machine-independence and flexibility for most simulation data.

Traditionally, snapshot files are simple ASCII files containing a table in which rows represent the particles and columns represent their properties; a header may have some additional information such as the snapshot time. This format is used, for example, by the `phiGRAPE` code (Harfst et al. 2007). While this scheme has some advantages, i.e., it is easy to process, human readable, and machine-independent, it is not native to the machine representation of data and usually requires auxiliary information to build the structure, thus resulting in much less efficient storage and longer parsing time compared to binary formats. In contrast, binary formats store the same information in a more compact way using some common representation of numerical data (such as the `IEEE754` floating-point specification), and are preferred when large volumes of data are expected (e.g., the `OUT3` file of `NBODY6`, see Aarseth 1999a). There are, however, many binary formats for particle data, differing in how the data are arranged in the file and how they are described by the metadata (see Section 5.1). Different binary formats generally produce files of similar sizes (especially when the data volume is large) and with little statistical redundancy, and therefore cannot be further reduced in size by data compression algorithms.

Big data must be written efficiently to the storage medium without interrupting or significantly slowing down the simulation process itself, thus dedicated nodes or processors are often used just to write the data to the disk, while the others continue the integration (asynchronous output). Some high-performance I/O libraries, such as `MPI-IO`, allow multiple nodes or processors to write to the same file in parallel. Beside the writing, some ways to deal with big data in this context are utilized as needed. If the data are sorted in a certain way, then this could make a snapshot file smaller by not saving the particle ID. Alternatively, if the output of a tree code makes use of the space-filling (Hilbert) curve, it is easier to rapidly access spatial sub-volumes of the data (Springel et al. 2005). Another way is to more frequently output a subset of particles of interest (POIs), such as black holes (Berczik et al. 2005, 2006). Most importantly, to reduce the amount of data that needs to be saved, at least part of the analysis is carried out "on the fly." Despite efforts toward designing highly efficient data structures, some data processing on the fly may in fact be necessary. An example of such analysis is the calculation of Lagrange radii in `NBODY6` (Aarseth 1999a) and the saving of image files of the system in `PKDGRAV` (Jetley et al. 2008).

In this paper, we propose a scalable storage scheme for N-body simulation data using the `HDF5` high-performance data format because of its hierarchical nature which allows us to store time-evolving hierarchical systems such as globular clusters. This paper is organized as follows. Section 2 describes the mode of operation of a direct N-body code. An adaptive storage scheme inspired by Farr et al. (2012) for direct N-body simulation data and an analysis of the data rate is presented in Section 3. Other possible approaches for data scaling are presented are presented in Section 4. Technical concerns and benchmarks of the proposed scheme are presented in Section 5. Finally, applications of the proposed storage scheme are presented in Section 6.

2. DIRECT N-BODY SIMULATIONS

In the direct N-body scheme, the equation of motion for a particle of index i in a system containing N particles takes the form (Aarseth 2003)

$$\ddot{\boldsymbol{r}}_i = -G \sum_{\substack{j=1 \\ j \neq i}}^{N} \frac{m_j (\boldsymbol{r}_i - \boldsymbol{r}_j)}{|\boldsymbol{r}_i - \boldsymbol{r}_j|^3}, \tag{1}$$

where m_j are the masses of the other particles, N is the total number of particles, $\boldsymbol{r}$ are the positions, and G is the gravity constant. Full calculation of the mutual gravitational forces for a system of N particles corresponds to $\sim N^2$ terms. The positions and velocities are subsequently updated by assuming that the evaluated force exerted on the particle is constant or can be interpolated with a polynomial during a certain time step Δt (see more information about the integrator below). When the time step Δt is shared among all of the particles and the total simulation time is $T_{\rm total}$, the number of force calculations is

$$S = \frac{1}{2} N (N-1) \frac{T_{\rm total}}{\Delta t}. \tag{2}$$

The choice of Δt varies among different integration algorithms. Employing higher-order algorithms allows faster

[9] http://www.yaml.org/

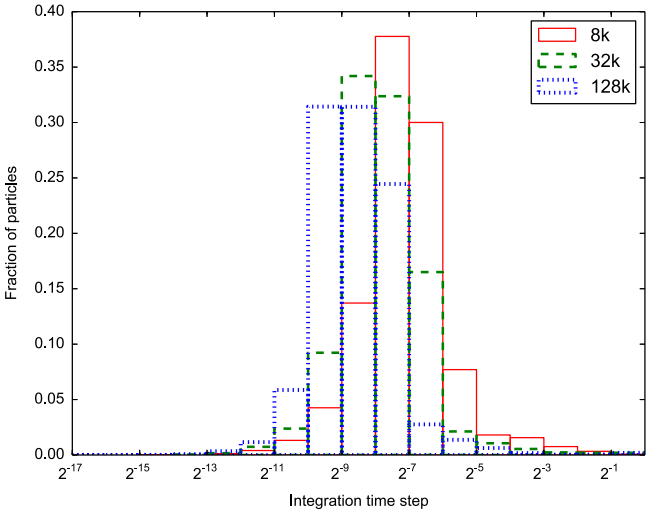

Figure 1. Time step distribution for Plummer model realizations with $N = 8$k, 32k and 128k at $t = 1$ Hénon time unit, simulated with the direct N-body code NBODY6++. The code employs the Block Time Step (BTS) integration scheme and a fourth-order Hermite integrator. For some arbitrarily defined maximum time step $\Delta t_{\max}$, all smaller time steps are given by $\Delta t_n = \Delta t_{\max}/2^{n-1}$, where n is called the "depth of integration." In this figure, the time steps are in Hénon units. The peaks of the three distributions are shifted to the left as N increases, illustrating that systems with higher number density have more close pairs, which lead to smaller time steps on average.

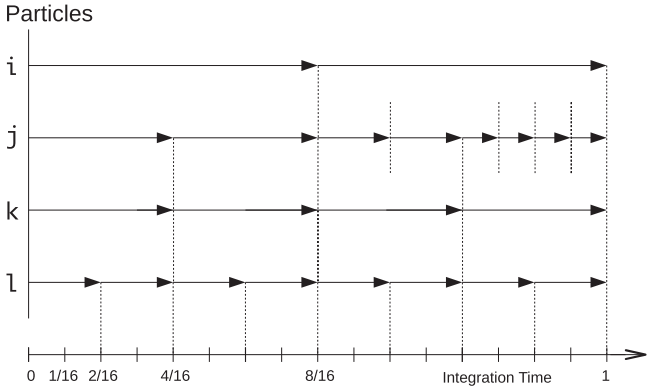

Figure 2. Schematic illustration of a four-particle system integrated with a block time step (BTS) scheme. Particles (i, j, k, l) are assigned individual time steps according to the forces exerted on them. It is assumed here that the total integration time is 1 (in arbitrary unit) and the minimum integration time is 1/16. At $t = 1/16$, no particle is scheduled to be integrated, as none have a time step smaller than $\Delta t = 2/16$. As the system proceeds to $t = 2/16$, particle l is the only particle with a time step short enough to schedule an integration. At $t = 4/16$, particles (j, k, l) are scheduled for integration while particle i is still outside the list. The full system is integrated at $t = 8/16$. Since after that particle j becomes increasingly active, it is integrated every time step starting from $t = 12/16$. The BTS scheme assigns time step in a hierarchical fashion based on Equation (4), and therefore guarantees the commensurability of the individual time step of all particles.

convergence but requires additional computational effort to calculate the high-order terms (Hut & Makino 2003). The fourth-order Hermite integrator was demonstrated to be successful in achieving an acceptable balance between accuracy and speed (Aarseth 1999b). Nevertheless, in the dense central regions of galaxies or star clusters where close encounters may occur frequently, very small integration time steps still have to be taken in order to ensure accuracy, which will dramatically slow down the simulation. Close encounters will eventually cause tight binary systems to form: such systems require permanent treatment with extremely small time steps. In this almost inevitable scenario, should all integration points be saved, the sizes of the resulting data files would be overwhelming. For instance, in a globular cluster with $N = 10^5$, if the system is to be evolved for 1000 Hénon time units[10] with $\Delta t = 10^{-4}$ (which is relatively large; see histogram in Figure 1), more than 10^{18} bytes (1 exabyte) of data would be generated in total. This would pose great challenges even for modern storage arrays and data analysis.

As direct computation of the $\mathcal{O}(N^2)$ algorithm is expensive, optimization schemes such as individual time step (ITS) and the Ahmad-Cohen neighbor scheme (ACS; Ahmad & Cohen 1973) were developed to dramatically reduce the computational costs (Aarseth 2003). Almost all modern N-body integrators now employ the ITS scheme. The basic idea is that since gravity follows an inverse-square law, particles from regions of different density experience different magnitudes of force. The density profiles of globular clusters can be roughly approximated by a power law, such as the Plummer model (Plummer 1911) or the King model (King 1966). Stars in the outskirts of star clusters usually move relatively unperturbed for timescales comparable to hundreds of times the corresponding timescales of the central particles, and hence long time steps can be used for their integration. Stars in the central regions, however, frequently experience violent interactions (close encounters) with their neighbors and therefore require much smaller integration time steps. Integration of a particle with index i is therefore carried out using a time step $\Delta t_{\rm i}$, which is often taken to be (e.g., Aarseth 2003)

$$\Delta t_i = \sqrt{\eta \frac{|\boldsymbol{a}_i||\boldsymbol{a}_i^{(2)}| + |\dot{\boldsymbol{a}}_i|^2}{|\dot{\boldsymbol{a}}_i||\boldsymbol{a}_i^{(3)}| + |\boldsymbol{a}_i^{(2)}|^2}}, \quad (3)$$

where $\boldsymbol{a}_{\rm i}$ is the acceleration of particle i (the total force acting on it divided by its mass), and $\dot{\boldsymbol{a}}_i$, $\boldsymbol{a}_i^{(2)}$, and $\boldsymbol{a}_i^{(3)}$ are the first, second, and third derivatives of the acceleration. The parameter η controls the accuracy of the integration and a commonly used value is $\eta = 0.02$ (Aarseth 2003). Depending on different density profiles, the ITS scheme reduces the computational complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(N^{4/3})$, and a larger gain can be achieved with centrally concentrated systems (Makino & Hut 1988). Here, a particle is considered as *active* if its state is changed significantly on timescales comparable to the integrator time step. Figure 1 shows a time step distribution of time steps for systems with $N = 8$k, 32k, and 128k (in this paper k $= 2^{10} = 1024$).

According to Equation (3), the time step $\Delta t_{\rm i}$ of particle i can get an arbitrary value. In practice, however, in order to divide particles into groups according to their time steps, the block time step (BTS) scheme is often employed, permitting particles in the same time step group to be advanced at the same time (Hayli 1967, 1974; McMillan 1986). Figure 2 illustrates how particles are advanced in the BTS scheme. For instance, in the hierarchical scheme used by NBODY6 and its parallel version NBODY6++[11] (Spurzem 1999; Spurzem et al. 2008) as well as

[10] The N-body unit system is referred to here as the Hénon unit system in honor of Michel Hénon.

[11] This paper makes no distinction between NBODY6 and NBODY6++.

many other Aarseth-type codes, the time steps are defined as

$$\Delta t_n = \Delta t_{\max}/2^{n-1}, \tag{4}$$

where n is the level of integration and $\Delta t_{\max}$ is a predefined maximum time step (which in practice is often taken as one Hénon time unit).

The integration itself is often performed using a predictor-corrector scheme. As an example, the Hermite Scheme employed by NBODY6++ first predicts the positions $\boldsymbol{x}_{i,p}(t)$ and velocities $\boldsymbol{v}_{i,p}(t)$ at some time t:

$$\boldsymbol{x}_{i,p}(t) = \boldsymbol{x}_{i,0} + (t-t_0)\boldsymbol{v}_{i,0} + \frac{(t-t_0)^2}{2}\boldsymbol{a}_{i,0} + \frac{(t-t_0)^3}{6}\dot{\boldsymbol{a}}_{i,0} \tag{5}$$

$$\boldsymbol{v}_{i,p}(t) = \boldsymbol{v}_{i,0} + (t-t_0)\boldsymbol{a}_{i,0} + \frac{(t-t_0)^2}{2}\dot{\boldsymbol{a}}_{i,0}, \tag{6}$$

where t_0 is the starting time and the subscript 0 of the vector quantities denotes the known value of the quantity at t_0. The acceleration $\boldsymbol{a}_i$ and its first derivative $\dot{\boldsymbol{a}}_i$ are evaluated at time t at the predicted position (i.e., by direct summation), and the two higher-order derivatives of the acceleration can be evaluated at time t_0:

$$\boldsymbol{a}_{i,0}^{(2)} = -6\frac{\boldsymbol{a}_{i,0}-\boldsymbol{a}_i}{(t-t_0)^2} - 2\frac{\dot{\boldsymbol{a}}_{i,0}+\dot{\boldsymbol{a}}_i}{t-t_0} \tag{7}$$

$$\boldsymbol{a}_{i,0}^{(3)} = 12\frac{\boldsymbol{a}_{i,0}-\boldsymbol{a}_i}{(t-t_0)^3} + 6\frac{\dot{\boldsymbol{a}}_{i,0}+\dot{\boldsymbol{a}}_i}{(t-t_0)^2}. \tag{8}$$

The second and third derivatives can be used to correct the predicted values to fourth order:

$$\Delta\boldsymbol{x}_i = \frac{1}{24}\boldsymbol{a}_{i,0}^{(2)}(t-t_0)^4 + \frac{1}{120}\boldsymbol{a}_{i,0}^{(3)}(t-t_0)^5 \tag{9}$$

$$\Delta\boldsymbol{v}_i = \frac{1}{6}\boldsymbol{a}_{i,0}^{(2)}(t-t_0)^3 + \frac{1}{24}\boldsymbol{a}_{i,0}^{(3)}(t-t_0)^4. \tag{10}$$

Finally, the corrected position $\boldsymbol{x}_i(t)$ and velocity $\boldsymbol{v}_i(t)$ at the time t can be expressed as

$$\boldsymbol{x}_i(t) = \boldsymbol{x}_{i,p}(t) + \Delta\boldsymbol{x}_i \tag{11}$$

$$\boldsymbol{v}_i(t) = \boldsymbol{v}_{i,p}(t) + \Delta\boldsymbol{v}_i. \tag{12}$$

Therefore, extra terms $\boldsymbol{a}_{i,0}$ and $\dot{\boldsymbol{a}}_{i,0}$ need to be stored, and interpolation also introduces extra computational overhead. For certain applications, such as visualization, it may not be critical to compute the corrector terms, and so part of the computational and storage overhead can be further reduced.

3. BTS STORAGE SCHEME

3.1. Description

Originally inspired by the ITS scheme, Farr et al. (2012) have shown that the data can actually be significantly compressed by recording only active particles. Below, we estimate the data rate of this approach by first considering the "traditional" snapshot scheme. During one Hénon time unit, the number of data records produced by the scheme is

$$\text{Size(Snapshots)} = 2^{R_t}N, \tag{13}$$

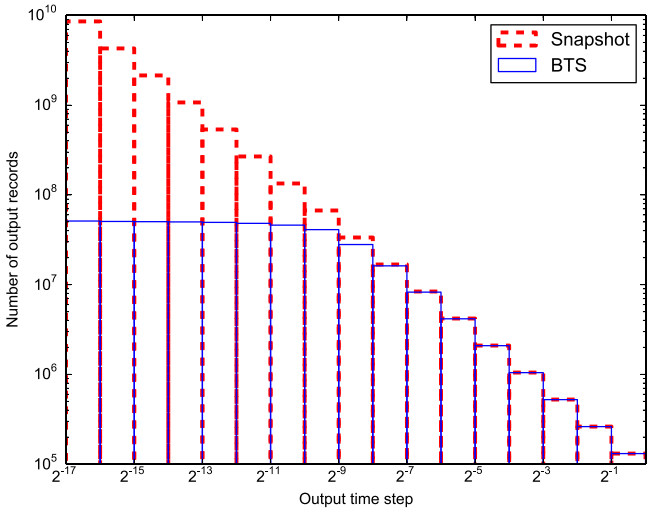

Figure 3. According to the time step distribution of the $N = 128$k simulation in Figure 1 at $t = 1$, these histograms show the number of output records as a function of output time step (in Hénon units), using the BTS scheme (solid blue histogram) and the snapshot scheme (dashed red histogram).

where R_t is the temporal resolution factor, such that in one Hénon time unit, the output operation is triggered for 2^{R_t} times. In the BTS scheme,

$$\begin{aligned}\text{Size(BTS)} &= \sum_{n=0}^{R_t-1} 2^n N_n + 2^{R_t}\sum_{n=R_t}^{\infty} N_n \\ &= \sum_{n=0}^{R_t-1} 2^n N_n + 2^{R_t}\left[N - \sum_{n=0}^{R_t-1} N_n\right],\end{aligned} \tag{14}$$

where N is the total number of particles and N_n is the number of particles with time step $\Delta t = 1/2^n$. For a given R_t, particles with integration time step $\Delta t_i \geqslant 1/2^{R_t}$ are fully resolved in the sense that output is commensurate with integration. That is, whenever these particles are integrated (or in the terminology of the Hermite scheme, corrected), their data are written to the file; moreover, this happens *only* when integration is performed. The rest of the particles, which have $\Delta t_i < 1/2^{R_t}$, are not fully resolved (for particles with time step $1/2^n$, output occurs only every 2^{n-R_t} integrations).

In the snapshot scheme, since data from all particles are written at the same time, particles with integration frequencies lower than the output frequency have to be extrapolated (or in the terminology of the Hermite scheme, predicted); this is redundant, since analysis software can perform this prediction, which is computationally very cheap. The BTS scheme compresses the data of the fully resolved particles by eliminating redundant information, which is *lossless*. The BTS scheme compresses the data of particles with integration frequency higher than the output frequency by skipping a certain number of integration points, which is *lossy*.

Figure 3 shows that the BTS file size initially grows exponentially as a function of R_t (like the snapshot scheme) but turns over at an output frequency close to the peak of the time step distribution (in Figure 1) and saturates (so that the file size remains finite even when the output frequency grows to infinity, on the left of the figure). This saturation is due to the small number of particles with very small time steps, as seen in

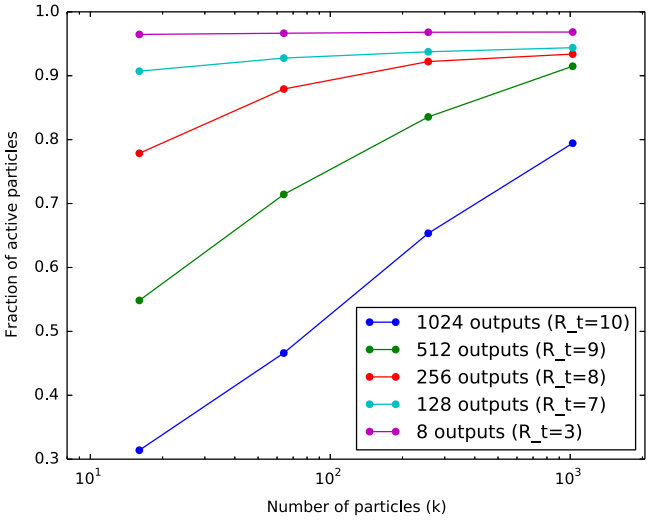

Figure 4. Fraction of active particles averaged over one Hénon time unit. Plummer systems with $N = 16$k, 64k, 256k, and 1024k are evolved, and with each N five different temporal resolutions R_t corresponding to five different output frequencies are marked with different colored lines. As the output frequency goes higher, the fraction of active particles declines, and therefore the BTS storage scheme gains significant reduction of data rates. For the same output frequency, systems with larger N have higher fractions of active particles. The two methods converge at $R_t = 0$, which yields standard snapshots.

Figure 1. Note that for very low output frequencies, the snapshot and BTS files have similar sizes, converging at $R_t = 0$ (which in this case represents the maximally allowed time step).

The snapshot scheme is prohibitively expensive if one intends to resolve the most rapidly varying particles, but this is feasible due to the convergence property of the BTS scheme. On the other hand, for low output frequencies, the methods are equivalent in terms of the number of records. The snapshot scheme might even be preferable in this case since the extra particle attributes $\boldsymbol{a}_0$ and $\dot{\boldsymbol{a}}_0$ need not be stored.

3.2. Example

Consider as an example the four-particle system illustrated in Figure 2 where the system is integrated for one Hénon time unit with a minimum integration time step of $\Delta t = 1/16$; a temporal resolution factor of $R_t = 3$ is adopted, which means that $2^{R_t} = 8$ output operations are scheduled within this one Hénon time unit. The first output (not including $t = 0$) is triggered at $t = 1/8$ when particle l is the only particle in the output list; at $t = 2/8$, however, particles (j, k, l) are active and eligible for output. At $t = 11/16$, although particle j is integrated, no output occurs as it is not a product of the output time step 2^{-R_t} and an integer, and therefore the information is lost here. It is instead included in the output list at $t = 12/16 = 6/8$ alone with particle l, and only the latest data at $t = 12/16$ will be written. At $t = 8/16$ and $t = 1/8$, the system receives a full output.

The number of particles in the output list will not exceed the total number of particles at any time. Consider again the above example when $R_t = 2$: more particles will be collected at the output points but the output interval is longer. Recall that for a full snapshot output, according to Equation (13) the size of the output is proportional to N and the output size of this scheme is a *linear* function of N. The statistics of the active particle fractions for simulations carried out with NBODY6++ are presented in Figure 4. The linear property of the BTS scheme makes the scheme suitable for very large systems, as long as the detailed evolution of the highly active particles is not important. For example, the resulting data sets can be used to generate visualization data for the overall evolution of star clusters. The data sets will have sufficient resolution to describe slow particles in the outskirts of the cluster in detail, allowing the viewers to observe the evaporation process. Since the output frequencies of highly active particles have been truncated to 2^{R_t}, the data sets will only have enough resolution for these particles if the output resolution R_t is set sufficiently high (so that the output time step is comparable to the actual integration of those particles). Even in such a case, the output size will still converge to a manageable scale, as seen from Figure 3.

3.3. Interpolation

Analysis or visualization software needs to interpolate the positions and sometimes velocities and higher derivatives of the particles between the output points. This is best done using septic splines, which are seventh degree piecewise polynomials. This method ensures that the interpolated curves exactly touch at the endpoints (or the spline's knots) and that the stored information concerning the previous and next known states is used. Lower-order interpolation can be used based on the predictor-corrector scheme (Section 2) or by applying lower-order splines that would discard some stored information (i.e., the known values of $\dot{\boldsymbol{a}}$); higher-order interpolation can be achieved if one uses more than the two nearest points. Let us define the running variable

$$\tau \equiv \frac{t - t_0}{t_1 - t_0} = \frac{t - t_0}{\Delta t} \tag{15}$$

such that $0 < \tau < 1$, where t is an arbitrary time in which we are interested in the particle's properties, t_0 is the last integration point before time t, and t_1 is the next one, and $\Delta t \equiv t_1 - t_0$ is the output time step. The interpolated position of the particle is thus

$$\begin{aligned}\boldsymbol{x}(\tau) = \boldsymbol{p}_0 &+ \boldsymbol{p}_1\tau + \boldsymbol{p}_2\tau^2 + \boldsymbol{p}_3\tau^3 + \boldsymbol{p}_4\tau^4 \\ &+ \boldsymbol{p}_5\tau^5 + \boldsymbol{p}_6\tau^6 + \boldsymbol{p}_7\tau^7,\end{aligned} \tag{16}$$

where $\boldsymbol{p}_0 \ldots \boldsymbol{p}_7$ are the spline coefficients given in Equations (17)–(24). The expressions for the velocity and higher derivatives can be easily determined from the above expression by derivation. Let the subscripts 0 and 1 represent the values of the quantities at times t_0 and t_1, respectively, so that we can write

$$\boldsymbol{p}_0 = \boldsymbol{x}_0 \tag{17}$$

$$\boldsymbol{p}_1 = \boldsymbol{v}_0\Delta t \tag{18}$$

$$\boldsymbol{p}_2 = \frac{1}{2}\boldsymbol{a}_0\Delta t^2 \tag{19}$$

$$\boldsymbol{p}_3 = \frac{1}{6}\dot{\boldsymbol{a}}_0\Delta t^3 \tag{20}$$

$$\begin{aligned}\boldsymbol{p}_4 = &-\frac{1}{6}(4\dot{\boldsymbol{a}}_0 + \dot{\boldsymbol{a}}_1)\Delta t^3 - \frac{5}{2}(2\boldsymbol{a}_0 - \boldsymbol{a}_1)\Delta t^2 \\ &- 5(4\boldsymbol{v}_0 + 3\boldsymbol{v}_1)\Delta t - 35(\boldsymbol{x}_0 - \boldsymbol{x}_1)\end{aligned} \tag{21}$$

$$\boldsymbol{p}_5 = \frac{1}{2}\left(2\dot{\boldsymbol{a}}_0 + \dot{\boldsymbol{a}}_1\right)\Delta t^3 + \left(10\boldsymbol{a}_0 - 7\boldsymbol{a}_1\right)\Delta t^2 + 3\left(15\boldsymbol{v}_0 + 13\boldsymbol{v}_1\right)\Delta t + 84(\boldsymbol{x}_0 - \boldsymbol{x}_1) \quad (22)$$

$$\boldsymbol{p}_6 = -\frac{1}{6}\left(4\dot{\boldsymbol{a}}_0 + 3\dot{\boldsymbol{a}}_1\right)\Delta t^3 - \frac{1}{2}\left(15\boldsymbol{a}_0 - 13\boldsymbol{a}_1\right)\Delta t^2 - 2\left(18\boldsymbol{v}_0 + 17\boldsymbol{v}_1\right)\Delta t - 70(\boldsymbol{x}_0 - \boldsymbol{x}_1) \quad (23)$$

$$\boldsymbol{p}_7 = \frac{1}{6}\left(\dot{\boldsymbol{a}}_0 + \dot{\boldsymbol{a}}_1\right)\Delta t^3 + 2(\boldsymbol{a}_0 - \boldsymbol{a}_1)\Delta t^2 + 10(\boldsymbol{v}_0 + \boldsymbol{v}_1)\Delta t + 20(\boldsymbol{x}_0 - \boldsymbol{x}_1). \quad (24)$$

Note that the discussion above is *per particle* and that each equation represents three vector components, which for the purpose of the interpolation are completely independent.

4. MODIFIED BTS STORAGE SCHEMES

As shown in Figure 3, the BTS file size converges with increasing output frequency, and therefore this scheme becomes mandatory for the storage of simulation data when very high temporal resolution is required. Nevertheless, BTS scheme integration data of a very large simulation can still be too large even for moderately large systems. While Equation (14) shows that that the data are scalable by specifying an output frequency R_t, this scaling technique may not provide sufficient resolution for highly active particles. An alternative scaling technique is presented in Section 4.1, allowing the user to define an individual resolution for each particle. While dynamically active particles are assumed to be interesting particles, Section 4.2 explores some possible scenarios where this may not necessarily be the case. Finally, Section 4.3 discusses an even more generic scenario in which the output may be driven by physical processes other than dynamical evolution.

4.1. Scaling with Spatial Resolution: Triggered Output for Significantly Updated Particles

In this scheme, the output is triggered *per particle*: when an individual particle has been integrated R_s times, its data become eligible for output. Since R_s defines the portion of integration for output, it defines the resolution in a spatial manner: larger values of R_s correspond to lower spatial resolution, and $R_s = 1$ corresponds to a full output of all BTS integration data. For $R_s > 1$, the scheme skips $R_s - 1$ integrations right after the current output until the next one, reducing the data rate by a factor comparable to R_s (the time steps of all particles are changing as they move in phase space and therefore it is unlikely that the reduction factor is exactly R_s). The total output rate is proportional to the total number of individual time steps, and the resulting output size grows as $N^{4/3}$ (Makino & Hut 1988).

For instance, consider again the four-particle system illustrated in Figure 2. Assuming that $R_s = 2$, the scheme skips one output right after the current output, and so the particle i will only be eligible for output at $t = 1$, particle j is eligible for output at $t = 8/16, 12/16, 14/16, 1$, and so on. With a careful choice of R_s, the scheme yields data sets with sufficient resolution for highly active particles such as hard binaries and close encounters, but also reduces the data rate of slow particles. Full output corresponds to ~1000 records per particle orbit (on average); for rendering purposes, it is still sufficient to reduce this by one order of magnitude, hence reducing the storage consumption by one order of magnitude. A comparison of the file sizes for different values of R_s is presented in Table 1. It is sensible to apply this scheme for detailed follow-ups of energetic subsystems.

Table 1
Spatial Output Resolution R_s as a Function of the Number of Records

R_s	# of Records (w/ BTS)	# of Records (w/o BTS)	Efficiency Ratio
50	112128	536870912	4788.0
40	223934	536870912	2343.0
30	272236	536870912	1972.1
20	404532	536870912	1327.1
10	766584	536870912	700.3
1	6953525	536870912	77.2

Note. The simulation was carried out with NBODY6++ for one Hénon time unit (roughly 1 Myr) and $N = 16384$ particles, where BTS is employed. $R_s = 1$ corresponds to the full output of BTS data. The smallest time step of an $N = 16384$ Plummer system is of the order of $\Delta t \sim 2^{-15}$ according to the time step distribution given by Figure 1. Hence, should there be no BTS scheme, the total number of record is proportional to $N^2\Delta t \sim 5.3 \times 10^8$ according to Equation (2) (shown in column 2). A full output of BTS data already yielded an efficiency ratio (column 2 divided by column 1) of 77.2, and together with the R_s parameter the reduction can be promising.

4.2. Dedicated Output for the POIs

Binary and triple black holes in galactic nuclei or star cluster centers, hypervelocity stars, and the host stars of planetary systems are particularly interesting objects to investigate in simulations. The output module of the integrator should therefore accommodate this need by providing high-resolution output for the POIs while suppressing the output of uninteresting particles to achieve maximum storage efficiency. POIs can be dynamically active. For example, a binary black hole system in a galactic nucleus can be so dynamically active that it will take a significant fraction of wall-clock time to resolve even with an advanced regularization technique (e.g., KS regularization). However, the bouns of these computations is that they keep the data of those active particles constantly up-to-date, allowing the output module to simply dump the data without interpolation. On the other hand, it may be interesting to follow the evolution of hypervelocity binary stars in the outskirts of the cluster (e.g., Lu et al. 2007). The forces exerted on those objects change rather slowly, despite their high velocities. Consequently, the integrator will not integrate those objects frequently; reliable dynamical data can then be achieved with interpolation, for example, with the spline method presented in Section 3.3.

4.3. Event/Attribute-driven Output

The output scenarios previously discussed are all driven by the dynamical evolution of the simulated system. Output will be triggered when the coordinates of particles change significantly. Sometimes, however, it is necessary to have the output triggered by certain events and/or attributes. For example, in starburst galaxies, the star formation process is usually the most interesting process to investigate. Critical events in stellar evolution may not necessarily correspond to critical events in the dynamical evolution. Therefore, the output strategy should instead be driven by the stellar evolution

Table 2
Astrophysical Quantities of Individual Particles in a Direct *N*-body Simulation with Stellar Evolution

Quantity	Meaning	Category
i	Unique identifier of the particle	Miscellaneous
name	User friendly label (e.g., for visualization)	Miscellaneous
t	Current time	Miscellaneous
δt	Next time step	Miscellaneous
m	Mass	St. dyn. & evo.
x	Position vector	Stellar dynamics
$\dot{x}$	Velocity vector	Stellar dynamics
a	Acceleration vector	Stellar dynamics
$\dot{a}$	Jerk vector (first derivative of a)	Stellar dynamics
ρ	Neighbor density	Stellar dynamics
ϕ	Local potential	Stellar dynamics
$t_{\rm EV}$	Stellar evolution age	Stellar evolution
$k_{\rm STAR}$	Type indicator of star	Stellar evolution
L	Luminosity	Stellar evolution
R	Radius	Stellar evolution
$T_{\rm EFF}$	Effective temperature	Stellar evolution
Z	Metallicity	Stellar evolution
δm	Mass change during $t_{\rm EV}$	Stellar evolution
$m_{\rm CORE}$	Core mass	Stellar evolution
$r_{\rm CORE}$	Core radius	Stellar evolution

process, allowing follow-up data analysis to trace the evolution of such astrophysical processes.

Gravitational dynamics codes generally provide dynamical information on the particles such as positions, velocities, and accelerations. Some codes support the simultaneous simulation of multiple astrophysical processes. For example, NBODY6++ is able to take the feedback of stellar evolution into account while handling the dynamical processes of particles. For a direct *N*-body code with stellar evolution, the possible assignment of astrophysical quantities for individual particles could be tabulated as in Table 2; binary systems are very common in such simulations, which require an auxiliary data structure to describe their properties as a whole. A possible binary data structure is presented in Table 3.

5. TECHNICAL CONCERNS AND BENCHMARKS

Even on modern supercomputers, a physically realistic simulation may take months to run. It is therefore critical to store the output such that it is accessible for further analysis. This requires that data be stored in a well-behaved, high-performance data structure. Generally, a simulation data file should meet the following requirements:

1. accuracy: correctly recording the relevant quantities;
2. time efficiency: data are written faster than they are generated, and the simulation is not slowed down significantly due to data output;
3. space efficiency: redundancy minimized;
4. interchangeability: machine/OS independent;
5. scalability: scalable to simulations big and small, simple and complicated; and
6. robustness: data loss minimized when the file is corrupted.

Data sets of *N*-body simulations are designed to describe a time-evolving system. Since inactive particles are not recorded, the interpolation of their data requires knowledge of their previously active state, making the data set itself time-dependent. Hence, ensuring data consistency would be another requirement.

Table 3
Astrophysical Quantities of Binary Systems in a Direct *N*-body Simulation

Quantity	Meaning
i_1, i_2	Unique identifiers of the two particles
P	Orbital period
A	Semi-major axis
e	Eccentricity of the binary orbit
I	Orbital inclination
I_1, I_2	Inclinations of the spins
x_c	Position vector of the center of mass
$\dot{x}_c$	Velocity vector of the center of mass

Note. Individual properties of each component can be retrieved by referring to Table 2 with i_1 and i_2.

5.1. Choosing a File Format

File formats are roughly divided into two categories: ASCII files and binary files. ASCII files are generally easier to interpret and are human readable. For example, tabular data are often stored as CSV (comma-separated values) files. Since the CSV data format simply uses one delimiter character (e.g., a comma) to separate fields and uses a line break to indicate the termination of a record, it is widely supported and can be easily imported to an analysis program. More complicated ASCII or text formats, such as XML[12] and YAML[13] also have been developed and standardized, making text files capable of describing hierarchical data structures. Text files avoid some of the problems encountered with binary files, such as endianness, padding bytes, and differences in the number of bytes in a machine word. However, they are not native to computer systems. Indeed, representation of numerical values in a text file is just literal, as these values are merely ASCII sequences and have to be converted to their intrinsic values before any computation can be performed. Standardized text formats, such as XML, structure the data with tags, which contributes to its low entropy.

In contrast, high I/O throughput are usually achieved with binary files since they are byte sequences native to the machines. High-level binary file libraries have been developed to resolve the problems of endianness, padding bytes, file headers, metadata storage, block data storage, etc. HDF5[14] (Hierarchical Data Format, version 5), for example, is an implementation of a binary file standard dedicated to handling large volumes of numerical data. It offers rich features such as compression filters, checksum filters, chunking, partial I/O, parallel I/O, and caching. It allows the data to be structured in a hierarchical fashion and accessed using POSIX-like path syntax. While the HDF5 format is designed for general purpose numerical data storage, some higher-level application programming interfaces have been developed to fit into special applications. For example, H5Part (Adelmann et al. 2007) is a portable high-performance parallel data interface for HDF5, which is dedicated to the storage of particle-based simulation

[12] http://www.w3.org/XML/
[13] http://www.yaml.org/
[14] http://www.hdfgroup.org/

data. Other widely used binary file formats in astrophysics include CDF[15] (Common Data Format), NetCDF[16] (Network Common Data Format), and FITS[17] (Flexible Image Transport System). All of these formats are self-describing and machine-independent, optimized for scientific data. The FITS data is mainly designed for image data as its name indicates, and the image metadata is stored in a human readable ASCII head, allowing an interested user to easily examine the header information with a simple text editor. CDF and NetCDF are more general data formats. Originally, they share the same conceptual model based on a multi-dimensional (array) model, but the latter has since diverged and is not compatible with the former. Some data formats are developed and optimized for more specific applications. For instance, the SDF format (Warren 2013) is used in the oct-tree-based "Dark Sky" cosmological simulations (Skillman et al. 2014).

5.2. Benchmarks

We adopt HDF5 as the native output format for the direct *N*-body code NBODY6 and NBODY6++, due to the rich features it offers and especially its interchangeability within the astronomical community. For example, the GADGET2[18] code (which was used, among others, in the Millennium Simulation mentioned above) has the option to output its snapshot data in HDF5 format; the Low Frequency Array (LOFAR) chose to use HDF5 to manage astronomical radio data (Anderson et al. 2011). The internal file layout is structured with the H5Part scheme (Adelmann et al. 2007). For the purpose of benchmarks, we also store the data as plain text CSV files, in which the particle data is described by multiple columns separated by commas, and each line describes the full data for a particle. Each floating-point number takes 8 bytes. Since the CSV format is not hierarchical, the time variable for particles in the same time group is repeated many times, as shown below:

```
t1, 1, x1, y1, z1, vx1, vy1, vz1, ...
t1, 2, x2, y2, z2, vx2, vy2, vz2, ...
......
t1, n, xi, yi, zi, vxi, vyi, vzi, ...
......
t2, 1, x1, y1, z1, vx1, vy1, vz1, ...
t2, 2, x2, y2, z2, vx2, vy2, vz2, ...
......
t2, n, xi, yi, zi, vxi, vyi, vzi, ...
......
```

The output subroutines can be easily integrated into recent versions of NBODY6 and NBODY6++,[19] in which options #46 and #47 of the input file are reserved for controlling the output file type (HDF5 or CSV) and output frequency, respectively. Detailed instructions for installation and usage can be found in Appendix A.

The output file sizes of the binary HDF5 output and text CSV output are compared in Figure 5, and the corresponding wall-clock time overheads are shown in Figure 6. It is obvious that even for very small systems, the performance difference between HDF5 files and CSV files is well pronounced: the file sizes of CSV are generally larger than the corresponding file sizes of HDF5, as more data are repeated as metadata in the CSV format. As *N* increases, they also grow faster than HDF5. The overhead of HDF5 is negligible even for high-frequency output, but the overhead of CSV is significant. Figure 7 shows the growth of the file size (a cluster simulation of one Hénon time unit) as a function of particle number; it also compares the file size dependency on different output frequencies. It shows that at lower output frequencies, the data size grows linearly as a function of *N* while at high output frequencies (corresponding to $2^{10} = 1024$ outputs per hènon time unit), the BTS scheme saves a significant fraction of the data rate.

With the scheme described in Section 3.2, Figure 7 shows that the size of the output data scales linearly with the total number of particles, thus achieving very high space efficiency, suitable for long-term simulation of very large systems. This scheme may not be able to provide sufficient resolution for

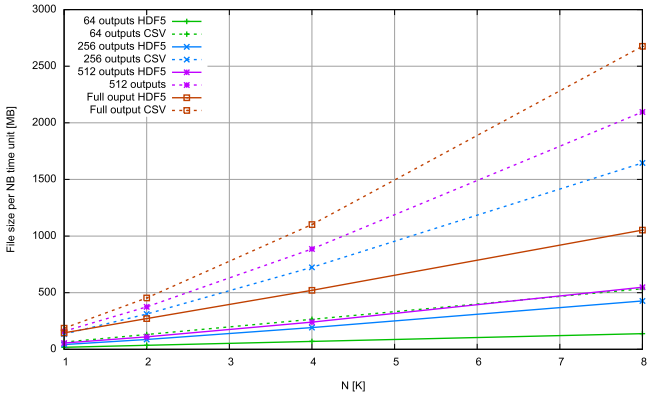

Figure 5. Size of output data file for one Hénon time unit as a function of total particle number and temporal resolution (64, 256, 512, and full output of the BTS data). The solid lines correspond to the output file size of when the output data are written in HDF5; the dashed lines correspond to the output file size when the output data are written in CSV format. With the increments of particle number and output frequency, the CSV file size grows quickly due to its large redundancy of metadata.

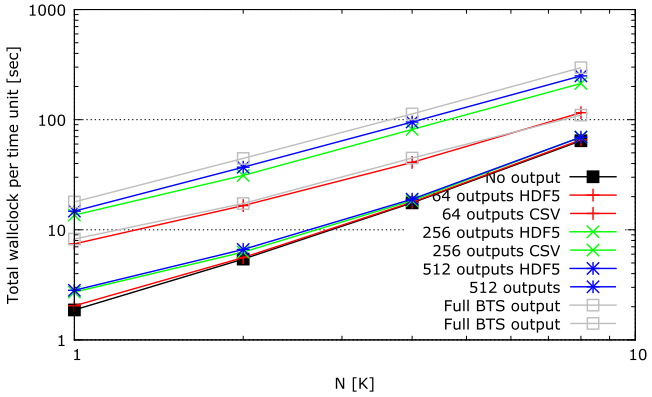

Figure 6. Total wall-clock time as a function of total particle number and temporal resolution (64, 256, 512, and full output of the BTS data) for evolving very small systems ($N = 1$k, 2k, 4k, 8k) for one Hénon time unit. The solid lines correspond to the output file size of when the output data is written in HDF5; the dashed lines correspond to the output file size when the output data are written in CSV format. This measurement is performed on four Intel(R) Core i7–3630QM CPU cores with the HDF5 library 1.8.11 (without GPU). Even for such small systems, the performance differences of HDF5 and CSV are still well pronounced. The overhead caused by the HDF5 output routine is negligible, while the corresponding overhead caused by the CSV output routine grows quickly.

[15] http://cdf.gsfc.nasa.gov/
[16] http://www.unidata.ucar.edu/software/netcdf/
[17] http://fits.gsfc.nasa.gov/
[18] http://www.mpa-garching.mpg.de/gadget/
[19] http://www.ast.cam.ac.uk/~sverre/web/pages/nbody.htm

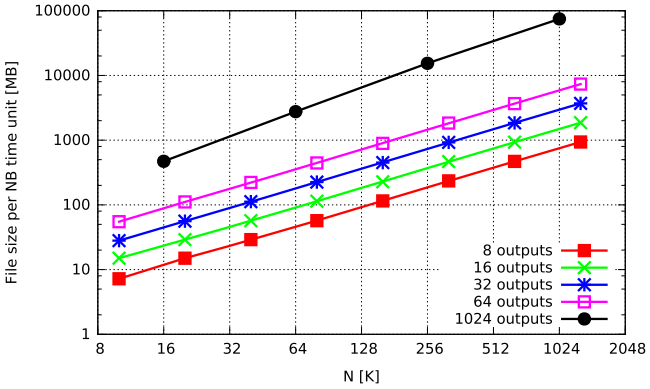

Figure 7. Size of output data file by defining temporal resolutions of 8, 16, 32, 64, and 1024 outputs per Hénon time unit. The output data are written in the HDF5 binary format. This gives a quick estimation of the simulation data size; for example, following up the evolution of a typical $N \sim 100$k system for 1000 Hénon time units with $R_t = 3$ (i.e., 8 outputs per Hénon time unit) corresponds to 100 GB of data size, which can be easily managed, even on personal computers.

highly active particles as it treats all particles equally. In fact, highly active particles can be well resolved by the scheme described in Section 4.1 where the resolution of less active particles is sacrificed.

6. APPLICATIONS

As noted in Section 4, a reasonable tradeoff between output size and loss of information can be achieved by taking the most interesting astrophysical processes into account and then adapting R_t or R_s for data scalings, which makes various applications possible. The resulting data sets can be used for post-simulation processing such as data visualization or data mining; they can also be used to store intermediate simulation data in large-scale simulations.

6.1. Simulation of Planetary Systems in Star Clusters

The BTS scheme opens a new approach for *N*-body simulations involving hierarchical architectures. The stability of planetary systems in star clusters, for example, is of fundamental importance in understand the early stage of planet formation. In fact, star clusters are the building blocks of galaxies (Lada & Lada 2003). Star formation is believed to occur in clusters as giant molecular clouds collapse. The collapse will likely result in circumstellar disks, which are the progenitors of planetary systems. If planetary systems were formed originally in the star cluster, the at least some of them would have been stable enough to survive in the star cluster environment where the densities are normally much higher than the solar neighbor and close encounters are not rare. The *Kepler* mission has been greatly successful in hunting exoplanets, yet it is worth mentioning that there are only a few exoplanets discovered in star clusters (e.g., Kepler-66, Kepler-67; see Meibom et al. 2013). This dichotomy is likely due to the post-formation disruptions of planetary systems in star clusters. This problem has been tackled in previous studies using both direct *N*-body simulations (e.g., Spurzem et al. 2009) and Monte-Carlo Simulations (e.g., Hao et al. 2013). Nevertheless, due to the huge range of both dynamical timescales and spatial scales, it is currently only feasible to investigate single planetary systems by treating the star-planet pairs as binaries and employing a regularization technique. Monte-Carlo simulations could indeed extend the study to the multiple planetary system domain, yet the results depend heavily on the quality of close encounter sampling.

With the BTS scheme, it is possible to decouple the dynamics of the whole system into the planetary and star cluster parts, and by which separating the integration of each part. To be specific, the star cluster dynamics can be integrated using a dedicated code such as `NBODY6++`. With the resulting data from that stored with the BTS scheme, one could then read the stored data, use them to calculate perturbations, and plug them into the planetary dynamics code.

As an example, we implemented the BTS storage scheme with the `HDF5`[20] file format. The time series data is stored with the `H5Part` scheme (Adelmann et al. 2007), as detailed in Table 4. The simulations of planetary systems are carried out *after* star cluster simulation is performed and the results are stored in the `HDF5` file. A certain fraction of stars with similar mass are assigned planetary systems of identical initial configurations. Each planetary system is integrated with `MERCURY6`. As the simulation progresses, the current time t is converted into the the Hénon time units, and the corresponding step in the `HDF5` file is thereby located. Accelerations at the point where each planet is located are calculated according to the loaded data, and are subsequently applied as velocity kicks. If t corresponds to the intermediate state between two adjacent time steps, then interpolation of $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ will be computed according to Equation (17)–(24), such that the acceleration at timescales comparable to the typical timescales of planets can be precisely evaluated (as demonstrated in Figure 8).

Since redundant data is minimized in the BTS storage scheme, we could adopt a very high output frequency of star cluster integration data while maintaining reasonable data file size (as shown in Figure 3). Together with the septic spline interpolation technique and making full use of all available data in the two adjacent time steps, the velocity kicks can be calculated with very high accuracy and temporal resolution. Furthermore, we parallelize the interpolation on `GPUs` with `Thrust/CUDA`.[21] In our simulations, the star cluster has 4000 stars where identical planetary systems are assigned to 1% of Solar-type stars. Each planetary system contains the four gas giants in the present-day Solar System. The BTS scheme has a temporal resolution of $R_t = 8$, corresponding to 256 outputs per Hénon time unit, or roughly 10^4 years per output. On two `Intel Xeon X5650` cores, evolving such a coupled systems for about 1 Myr takes about 12 hr. The code is not fully optimized for the purpose of benchmark, and the actual wall-clock time depends primarily on the frequency of communication between `NBODY6++` and `MERCURY6`. The communication of `NBODY6++` and `MERCURY6` is implemented within the `AMUSE` framework (Portegies Zwart et al. 2009, 2013). The scientific results of this application are presented primarily in Cai et al. (2015, and follow-up papers).

6.2. Long-term Evolution of A Massive Globular Cluster with Million Bodies

Simulations of massive globular cluster in the regime of million bodies and/or million solar masses have been made

[20] In particluar, we note that `HDF5` is chosen just as an example because it is very prevalent and flexible; other formats, such as `SDF` (Warren 2013), exist and can be used in the same way.
[21] http://docs.nvidia.com/cuda/thrust/

Table 4
Internal File Layout of the Star Cluster Time Series Data File

Step#	Attributes (scalar)	Data (vectors)
0	t_0, N_0, …	$\boldsymbol{x}_0, \boldsymbol{y}_0, \boldsymbol{z}_0, \boldsymbol{x}_0^{(1)}, \boldsymbol{y}_0^{(1)}, \boldsymbol{z}_0^{(1)}, \boldsymbol{x}_0^{(2)}, \boldsymbol{y}_0^{(2)}, \boldsymbol{z}_0^{(2)}, \boldsymbol{x}_0^{(3)}, \boldsymbol{y}_0^{(3)}, \boldsymbol{z}_0^{(3)}, \boldsymbol{m}_0, \ldots$
1	t_1, N_1, …	$\boldsymbol{x}_1, \boldsymbol{y}_1, \boldsymbol{z}_1, \boldsymbol{x}_1^{(1)}, \boldsymbol{y}_1^{(1)}, \boldsymbol{z}_1^{(1)}, \boldsymbol{x}_1^{(2)}, \boldsymbol{y}_1^{(2)}, \boldsymbol{z}_1^{(2)}, \boldsymbol{x}_1^{(3)}, \boldsymbol{y}_1^{(3)}, \boldsymbol{z}_1^{(3)}, \boldsymbol{m}_1, \ldots$
2	t_2, N_2, …	$\boldsymbol{x}_2, \boldsymbol{y}_2, \boldsymbol{z}_2, \boldsymbol{x}_2^{(1)}, \boldsymbol{y}_2^{(1)}, \boldsymbol{z}_2^{(1)}, \boldsymbol{x}_2^{(2)}, \boldsymbol{y}_2^{(2)}, \boldsymbol{z}_2^{(2)}, \boldsymbol{x}_2^{(3)}, \boldsymbol{y}_2^{(3)}, \boldsymbol{z}_2^{(3)}, \boldsymbol{m}_2, \ldots$
...	...	...
n	t_n, N_n, …	$\boldsymbol{x}_n, \boldsymbol{y}_n, \boldsymbol{z}_n, \boldsymbol{x}_n^{(1)}, \boldsymbol{y}_n^{(1)}, \boldsymbol{z}_n^{(1)}, \boldsymbol{x}_n^{(2)}, \boldsymbol{y}_n^{(2)}, \boldsymbol{z}_n^{(2)}, \boldsymbol{x}_n^{(3)}, \boldsymbol{y}_n^{(3)}, \boldsymbol{z}_n^{(3)}, \boldsymbol{m}_n, \ldots$

Note. The time series is organized as `HDF5` groups in which vector data and scalar attributes corresponding to a given time step are grouped.

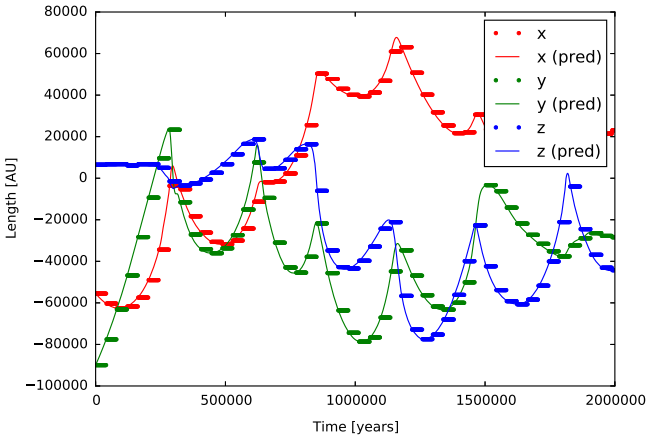

Figure 8. Interpolation of the position of a given star. Without interpolation, the position of the star is a step-like function of time (shown with dots). With interpolation, the position is smoothed, allowing velocity kicks to be preciously evaluated (shown with solid curves).

feasible only in recent years thanks to the exciting evolution of GPU-based high-performance computing technology. In a recent study, Wang et al. (2015) evolved a globular cluster with $N = 1.05$m (950k single stars and 50k binaries) for 12 Gyr. With a temporal resolution of $R_t = 3$, 8 outputs were generated for each Hénon time unit, corresponding to 475 MB of data. According to the time scaling of Hénon time units to physical time units, the total time of simulation corresponds to about 10^4–10^6 Hénon time units, depending on the total mass of the cluster. As such, the total data output of the BTS scheme is roughly 5 TB to 500 TB.

6.3. Applications for Grid-based Simulations

BTS-like storage schemes can also be very useful for grid-based simulations. Modern adaptive mesh refinement (AMR) codes, such as `Enzo` (Bryan et al. 2014) and `GAMER` (Schive et al. 2010), adopt ITS integration powered by GPU acceleration. The total number of refinement levels is typically around 10, making the evolution time steps of the root level and the highest refinement level differ by a factor of ∼1000. Hence, it is impractical to store the entire snapshot at each sub-step.

For example, in the cosmological simulations of wavelike dark matter (Schive et al. 2014), the dynamical timescale of the solitonic core in each dwarf galaxy is only about 50 Myr. Thus, it requires $\sim 1.6 \times 10^4$ data dumps from redshift one to the present day (assuming 100 dumps per dynamical timescale). Each full snapshot takes about 80 GB in a 1.5 Mpc h^{-1} comoving box with $\sim 10^{10}$ cells in total. The total amount of data in the snapshot scheme thus consumes ∼1.3 petabytes. For comparison, if we are mainly interested in the dynamical evolution of one solitonic core, then we can utilize the BTS-like storage scheme to output more frequently only the core data. For a solitonic core with a radius of 1 kpc and a simulation resolution of 60 pc, it consumes about 160 kilobyte for one data dump and 2.5 GB in total after redshift one. Accordingly, the storage requirement can be significantly reduced by a factor of $\sim 5 \times 10^5$.

6.4. Data Visualization

Astronomical data take on a multitude of forms: catalogs, data cubes, images, and simulations (Kent 2013). Because of their complexity, they are usually explored using data visualization, which is in fact reorganization of the original data by graphical means. It is particularly useful to illustrate the dynamical evolution of N-body systems. Visualization can be done in various ways, from a simple 2D plot to a realistic visual reconstruction of complicated multi-scale astrophysical processes. This simple idea can become challenging in the context of astrophysical data because of the wide dynamical range and large particle number. If the data are stored in a "compact" fashion such that only active particles are recorded, as described in Section 4, then the position needs to be interpolated using (for example) septic splines as presented in Section 3.3 prior to rendering. Since each particle is interpolated independently, this problem is "embarrassingly parallel" and very suitable for GPUs (e.g., programmed in CUDA or OpenCL). It is common that the total number of particles exceeds the total number of pixels on the viewport, and therefore the visualization program should be adjusted to the user's interests. Furthermore, because of the large dynamical ranges, data usually have to be scaled before rendering. For example, the stellar mass m can range from $\sim 0.1\ M_\odot$ to a much as $\sim 150\ M_\odot$; the power P emitted by a star is a strong function of its temperature T and radius R, as implied by the Stefan-Boltzmann law $P \propto R^2\ T^4$. If m or P are rendered directly on the screen, then massive or bright stars are easily saturated while light or faint stars are difficult to distinguish.

In practice, it is usually not enough to recreate the evolution process of an N-body system by plotting *only* the coordinates. The velocity vector, mass, size, temperature, and luminosity are then expected to be rendered as associated properties of the coordinates, such as color, symbol, or size. As an example, we adopt the astronomical plotting library `vispy` for the visualization of an `NBODY6++` simulation as Figure 9 shows; as another example, we also adopt the open-source scientific visualization package `ParaView` to visualize the mass spectrum of dense globular cluster simulations, as shown in Figure 10.

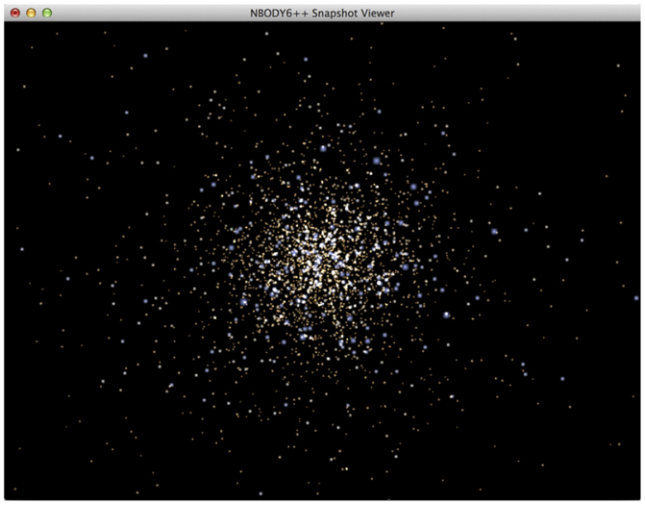

Figure 9. Visualization of NBODY6++ snapshot with the vispy library. The particles are colored according to their temperature, and are sized according to their luminosity.

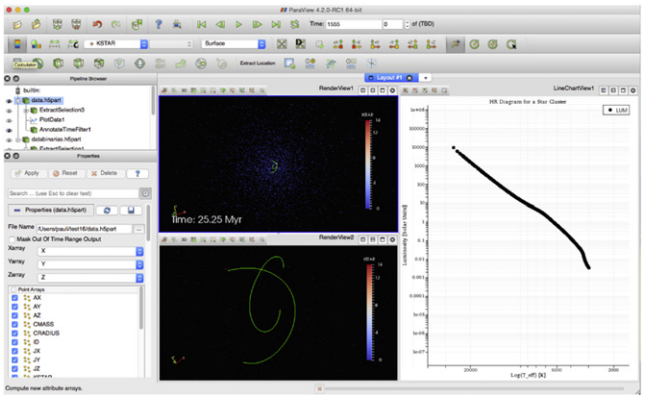

Figure 10. Visualization of NBODY6++ simulation data with ParaView. The figure shows a cluster with $N = 5000$ particles (King Model, $W_0 = 6.0$, Kroupa (2001) initial mass function). The upper left panel shows an overview of the star cluster; the botton left panel shows the trajectories of the particle of interest (POI), and in this case they are two stellar mass black holes (masses $M_1 = 10M_\oplus$ and $M_2 = 20M_\oplus$). The stars are colored with their stellar types. The corresponding H–R diagram evolving simultaneously with the cluster is shown on the right panel. The data are written in HDF5 format with the H5Part scheme, which is supported by ParaView via the built-in H5PartReader.

7. CONCLUSION

We present the BTS storage scheme for the data management of astrophysical N-body simulations, inspired by the ITS integration scheme (Makino & Hut 1988). This is an urgent response to the ever-increasing challenges posed by modern highly computationally expensive simulations. By adopting the BTS storage scheme, the growth of data can be dramatically scaled down from N^2 to $N^{4/3}$ (for the Plummer model). Depending on the uses of simulation, it is not necessary for all integration data to be recorded. Instead, a resolution parameter can be defined either in the space domain or in the time domain, which offers the flexibility of data scaling. Apart from theoretical analysis and predictions, dedicated simulations are carried out and the results are consistent with the theory. Our I/O performance benchmark of the binary HDF5 and the ASCII CSV format shows that binary formats are generally more preferable to store large, complicated data sets, yet for lightweight data sets text files exhibit their convenience for data analysis and portability. A list of astrophysical quantities for particles with potential user interests is proposed, and some of these quantities are visualized with open-source packages and libraries such as ParaView, GLnemo2, and s2plot.

The growth of numerical simulation scales and data rates implies that not only computations, but also data storage, visualization, and analysis need to be carried out distributively. Open source packages currently provide strong support for the technical implementation of these distributed systems, yet, to implement them into astrophysics-driven systems, adaptations need to be made according to the specific astrophysical context. This paper therefore addresses the concerns and possible solutions. Typical applications of the proposed scheme to astrophysical scenarios such as simulations of planetary systems in star cluster, cosmological grid-based simulations, long-term evolution of million solar mass globular clusters, and scientific visualizations are presented as well.

Our discussion is primarily focused on particle-based direct N-body simulations. The philosophy behind the proposed scheme is to "apply proper scaling to the simulation data to provide fine-grain control of the resolution of scientifically interesting data while suppressed the uninteresting ones." Despite the different algorithms used in other kinds of astrophysical simulations, such as hydrodynamics simulations, tree codes, AMR codes, Monte-Carlo simulations, and many other new algorithms under development, the methodology addressed in this paper is transferable to a wide range of scenarios.

We thank the anonymous referee for constructive comments that helped to improve the manuscript considerably. We acknowledge support by NAOC CAS through the Silk Road Project and (RS) through the Chinese Academy of Sciences Visiting Professorship for Senior International Scientists, grant Number 2009*S*1 − 5. The special GPU accelerated supercomputer laohu at the Center of Information and Computing at National Astronomical Observatories, Chinese Academy of Sciences, funded by Ministry of Finance of People's Republic of China under grant *ZDYZ*2008 − 2, has been used for some of the largest simulations. We are grateful for support by Sonderforschungsbereich SFB 881 "The Milky Way System" of the German Research Foundation (DFG), through subproject Z2 and the GPU cluster Milky Way at FZ Jülich, and for the support of the visit of M.X.C. in Heidelberg. We thank Hsi-Yu Schive for the information of cosmological grid-based simulation. We thank Peter Berczik, Long Wang, Sverre Aarseth, Marcel Zemp, and Siyi Huang for useful discussions. M.B.N.K. was supported by the Peter and Patricia Gruber Foundation through the PPGF fellowship, by the Peking University One Hundred Talent Fund (985), and by the National Natural Science Foundation of China (grants 11010237, 11050110414, 11173004). This publication was made possible through the support of a grant from the John Templeton Foundation and National Astronomical Observatories of Chinese Academy of Sciences. The opinions expressed in this publication are those of the author(s) and do not necessarily reflect the views of the John Templeton Foundation or National Astronomical Observatories of Chinese Academy of Sciences. The funds from John Templeton Foundation were awarded in a grant to The University of Chicago which also managed the program in conjunction with

Table 5
Fine-grain Control of Output Frequency and File Format for NBODY6

Option	Meaning
KZ(46)=1	Output BTS data as HDF5 (active particle only)
KZ(46)=3	Output BTS data as HDF5 (all particles)
KZ(46)=2	Output BTS data as HDF5 (active particle only)
KZ(46)=4	Output BTS data as CSV (all particles)
KZ(47)=R_t	The output frequency is 2^{R_t} times per Hénon time unit

National Astronomical Observatories, Chinese Academy of Sciences. P.A. acknowledges the financial support of FONDECYT 3130623 and C.A.S. the Visiting Fellowship for researchers from developing countries. P.A. was also funded by Chinese Academy of Sciences President's International Fellowship Initiative, grant No. 2014*FFJB*0018, and C.A.S. through a grant to the South America Center for Astronomy (CASSACA) in Santiago, Chile.

APPENDIX A
CUSTOM OUTPUT SUBROUTINES FOR NBODY6

We implemented the HDF5 and CSV custom data format output subroutines for NBODY6 and NBODY6++, both for benchmark purpose and the need of long-term data management. The subroutines can be downloaded from http://silkroad.bao.ac.cn/~maxwell/hdf5. The integration is trivial: (1) compile and install the HDF5 library from the source code, which can be obtained from http://www.hdfgroup.org/HDF5/release/obtainsrc.html; (2) copy the custom output subroutines source code custom_output.f to the Ncode directory of NBODY6; (3) modify the Makefile to add the custom_output.f file into the list of source files; (4) call the subroutine by adding one line into the intgrt.f (or intgrt.omp.f for the GPU2 version); and (5) add a common block to the "hrplot.f" so that stellar evolution data can also be dumped to the output. More detailed instruction can be found in the README file of of the downloaded package. In the NBODY6 input file, option #46 and #47 are used to control the output file type, respectively, as shown in Table 5.

APPENDIX B
VISUALIZATION TECHNIQUES OF THE HDF5-BASED BTS DATA

The HDF5 data generated by the custom output subroutines described in Appendix A can be visualized *directly* with ParaView. The H5Part reader is included in the ParaView 4.x distribution, but it is not activated by default. To enable it, users may navigate to the main menu and click "Tools | Manage Plugins," and then find the H5PartReader and select "Auto Load." After that, one will be able to load the HDF5 simulation data sets from the Open menu. After loading the file, users may select the X, Y, and Z arrays from the drop-down list for visualization.

We also implemented a vispy-based visualization script for the simulation data sets, which can be downloaded from http://silkroad.bao.ac.cn/~maxwell/hdf5.

REFERENCES

Aarseth, S. J. 1999a, PASP, 111, 1333
Aarseth, S. J. 1999b, CeMDA, 73, 127
Aarseth, S. J. (ed.) 2003, in Gravitational N-Body Simulations (Cambridge: Cambridge Univ. Press), 430
Adelmann, A., Gsell, A., Oswald, B., et al. 2007, in Particle Accelerator Conf., Progress on H5Part: a Portable High Performance Parallel Data Interface for Electromagnetics Simulations (Piscataway, NJ: IEEE), 3396
Ahmad, A., & Cohen, L. 1973, JCoPh, 12, 389
Anderson, K., Alexov, A., Bähren, L., et al. 2011, adass XX, 442, 53
Barnes, J., & Hut, P. 1986, Natur, 324, 446
Berczik, P., Merritt, D., & Spurzem, R. 2005, ApJ, 633, 680
Berczik, P., Merritt, D., Spurzem, R., & Bischof, H.-P. 2006, ApJL, 642, L21
Berczik, P., Nakasato, N., Berentzen, I., et al. 2007, in SPHERIC—Smoothed Particle Hydrodynamics European Research Interest Community, Second International Workshop, ed. A. Crespo et al. (Ourense: SPHERIC), 5
Brown, W. R., Kilic, M., Hermes, J. J., et al. 2011, ApJL, 737, L23
Bryan, G. L., Norman, M. L., O'Shea, B. W., et al. 2014, ApJS, 211, 19
Cai, M. X., Spurzem, R., & Kouwenhoven, M. B. N. 2015, arXiv:1501.01709
Farr, W. M., Ames, J., Hut, P., et al. 2012, Natur, 17, 520
Gottloeber, S., Yepes, G., Wagner, C., & Sevilla, R. 2006, arXiv:astro-ph/0608289
Greengard, L., & Rokhlin, V. 1987, JCoPh, 73, 325
Hao, W., Kouwenhoven, M. B. N., & Spurzem, R. 2013, MNRAS, 433, 867
Harfst, S., Gualandris, A., Merritt, D., et al. 2007, Natur, 12, 357
Hayli, A. 1967, Les Nouvelles Méthodes de la Dynamique Stellaire (Paris: CNRS)
Hayli, A. 1974, Numerical Solution of Ordinary Differential Equations (Berlin: Springer)
Hut, P., & Makino, J. 2003, The Art of Computational Science
Jetley, P., Gioachin, F., Mendes, C., Kale, L. V., & Quinn, T. 2008, in IEEE Int. Symp. on Parallel and Distributed Processing, IPDPS, Massively Parallel Cosmological Simulations with ChaNGa (Piscataway, NJ: IEEE), 1
Katz, B., Dong, S., & Malhotra, R. 2011, PhRvL, 107, 181101
Kent, B. R. 2013, PASP, 125, 731
King, I. R. 1966, AJ, 71, 64
Kroupa, P. 2001, MNRAS, 322, 231
Lada, C. J., & Lada, E. A. 2003, ARA&A, 41, 57
Lu, Y., Yu, Q., & Lin, D. N. C. 2007, ApJL, 666, L89
Makino, J., & Hut, P. 1988, ApJS, 68, 833
Makino, J., & Taiji, M. (ed.) 1998, in Scientific Simulations with Special-Purpose Computers-the GRAPE Systems (New York: Wiley-VCsH), 248
McMillan, S. L. W. 1986, LNP, 267, 156
Meibom, S., Torres, G., Fressin, F., et al. 2013, Natur, 499, 55
Plummer, H. C. 1911, MNRAS, 71, 460
Portegies Zwart, S., et al. 2013, CoPhC, 183, 456
Portegies Zwart, S., McMillan, S., Harfst, S., et al. 2009, NewA, 14, 369
Schive, H.-Y., Chiueh, T., & Broadhurst, T. 2014, NatPh, 10, 496
Schive, H.-Y., Tsai, Y.-C., & Chiueh, T. 2010, ApJS, 186, 457
Skillman, S. W., Warren, M. S., Turk, M. J., et al. 2014, arXiv:1407.2600
Springel, V., White, S. D. M., Jenkins, A., et al. 2005, Natur, 435, 629
Spurzem, R. 1999, JCoAM, 109, 407
Spurzem, R., Berentzen, I., Berczik, P., et al. 2008, LNP, 760, 377
Spurzem, R., Giersz, M., Heggie, D. C., & Lin, D. N. C. 2009, ApJ, 697, 458
Wang, L., Spurzem, R., Aarseth, S., et al. 2015, MNRAS, 450, 4070
Warren, M. S. 2013, arXiv:1310.4502
Yu, Q., & Tremaine, S. 2003, ApJ, 599, 1129