Table of Contents

Fast parallel particle simulations on distributed memory architectures *

M. Hipp[1], S. Kunze[2], M. Ritt[1], W. Rosenstiel[1], and H. Ruder[2]

[1] Wilhelm-Schickard-Institut für Informatik, Universität Tübingen
[2] Institut für Astronomie und Astrophysik, Universität Tübingen

Abstract. One of the major goals of the Sonderforschungsbereich (SFB) 382 is the development of parallelization strategies for physical applications. In particular, we focus on simulation methods based on particles.
In this paper, we present two different particle methods, a three-dimensional Smoothed particle hydrodynamics (SPH) code and a one-dimensional Particle-in-Cell (PiC) code with Monte-Carlo collisions (MCC). For both methods, a brief introduction to the physical model and its implementation is given. We discuss implementation and runtime aspects and detail the parallelization of the codes.
We talk about our experience porting the codes and running them on a couple of distributed memory machines, such as Cray T3E, Hitachi SR8000 and a Linux Cluster and present performance measurements of the codes. For the SPH method, some of the physical results of the simulation runs are explained.

Keywords: Object-orientation, Parallelization, Simulation.

1 Introduction

In our project the numerical method SPH is used to simulate astrophysical problems. In the last years we gained a lot of experience with several different parallel SPH codes in the programming languages Fortran, C and C++ using MPI and SHMEM on the Cray T3E [12]. We could show, that an efficient implementation of SPH is possible on machines with distributed memory [11]. In the last year the code was improved to simulate three dimensional problems and simulations with particle injection. Furthermore, the code was optimized to handle more particles with the same amount of memory allowing us to run bigger problems.

1.1 Motivation

To obtain a good spatial resolution for three dimensional problems it is necessary to increase the number of particles and the number of interactions per

* This project is funded by the DFG within SFB 382: *Verfahren und Algorithmen zur Simulation physikalischer Prozesse auf Höchstleistungsrechnern* (Methods and algorithms to simulate physical processes on supercomputers).

particle. Both result in an overall computation time and memory consumption which is too big for small desktop machines.
The overall performance for SPH simulations on vector SMP machines is not very good, because the SPH method does not provide long vectors. This results in a bad price-performance ratio for vector SMP machines like the NEC SX5 compared to machines with distributed memory (like Cray T3E, Hitachi SR8000 and becoming popular PC based Clusters) if it is possible to extract and group the communication parts of the code. So, we decided to do the parallelization using message passing with explicit communication.

2 Smoothed particle hydrodynamics

2.1 About SPH

Smoothed particle hydrodynamics (SPH) was introduced by Lucy [8] and Gingold & Monaghan [4]. It is a grid-less Lagrangian particle method for the solution of the hydrodynamic equations. Instead of solving the equations on a grid, the fluid is modelled by small interacting packets of matter that move along with the flow and carry mass and momentum. Hydrodynamic variables such as density, pressure, and temperature are assigned to each particle. The values of these quantities are determined by the interactions with the neighbour particles.
SPH is especially suited for the simulation of accretion disks because SPH can handle large density contrasts, open boundaries are easily implemented, and SPH posesses an adaptive resolution both by the variation of the interaction range of each particle and also by the particle mass. So it is possible to resolve the most interesting regions very fine. Readers interested in the basic principles of SPH find detailed reviews of the SPH method in Benz [2] or Monaghan [9].

2.2 Physical Problem

Accretion discs are very common structures. They play an important role in galaxy formation and feed the central engines in the nuclei of active galaxies. All stars form via accretion, the left-over of this process provides the material for the formation of planets.
Most binary star systems form accretion disks at some evolutionary state. Here we are concerened with cataclysmic variables, which are good laboratories to study the physics of accretion disks.
Cataclysmic variables (CVs) are close binaries with mass transfer from the secondary to the primary. The donor, a low-mass, late type main sequence star, fills its Roche lobe and loses mass to the accretor, a white dwarf (WD). In many CVs the magnetic field strength of the WD is so small that it can be neglected. In this case the overflowing matter forms an accretion disk around

the WD, and the accretion process is governed by the viscous evolution of the disk.
One aspect of the physics of accretion discs in CVs is the interaction of the in-falling gas stream with the rim of the accretion disk. As both flows, the stream and the disk, are highly supersonic, the development of shock fronts is expected at the impact zone.
This interaction region, the so-called "bright spot" or "hot spot", can be seen in many high-inclination CVs, e.g. U Gem (1965), and Z Cha (1986), as a hump in the orbital light curve shortly before the eclipse of the WD.
From the large range of different bright spot sizes, locations and intensities, and their variability, it is already clear that the underlying physics is rather complex and has to be approached by numerical simulations. Closely related to the bright spot is the question of how much of the in-falling gas stream is stopped at the edge of the disk and stored there, and how much of the stream can flow over and under the disk surface to inner parts of the disk.
By observation, there are several features seen in CVs and low-mass X-ray binaries (LMXBs) that can be explained by stream-disk overflow. In CVs the accretion stream reveals itself by its high velocity in Doppler maps and phased spectra (Lubow 1989, Shafter, Hessman & Zhang 1988, Hellier & Robinson 1994). . Furthermore, many CVs and LMXBs show so-called absorption dips in X-ray and UV around orbital phase 0.7 to 0.8. Some systems also show a shallower dip at about phase 0.1. The absorption dips at phase 0.7 can be explained if stream material overflows the disk at several disk scale heights after being deflected by shock interaction at the bright spot region.
Still missing are numerical simulations of the stream-disk impact with high spatial resolution, and simulations that take into account the further fate of the overflowing matter. For this purpose it is necessary to use the full Roche potential. Also lacking are comparable simulations for different kinds of systems with differing orbital periods, mass ratios, and mass transfer rates. Our simulations of different systems cover a wide range of these parameters, and we hope to fill this gap to some extent.

2.3 Physical Results

Simulation Setup In order to achieve a reasonable spatial resolution the disk should contain at least 50 000 particles. Simply starting the simulation with an empty disk and waiting until some form of quasi-steady state is reached is far too time consuming to allow for parameter studies. This problem can be circumvented by making use of one of the more pleasant features of SPH. Since the particles are actually to be interpreted as integration points rather than fluid particles, one is free to substitute a given particle distribution with another, equivalent distribution that represents the same physical situation, within the accuracy of the method. The trick is to take a certain number of data sets of different time steps and simply concatenate them to

get a new single data set. If we take, e.g., 10 data sets and concatenate them, the mass of each individual particle has to be divided by that factor.
This method allows for the construction of viscously evolved disks with almost arbitrary particle number in a short time.
The in-falling gas stream is set up in a way that gaussian shaped density distributions in the horizontal and vertical directions are fulfilled. The theoretical results of Lubow & Shu (1975, 1976) are used to determine the vertical and horizontal scale heights of the stream. Particles are inserted a bit downstream of the inner Lagrangian point with appropriate velocity. The density in the stream is lower than in the disk. In order to reach comparable resolution, we used more but less massive particles for the stream than for the disk.

Results The astrophysical results of these simulations are discussed in detail in Kunze, Speith & Hessman (2001). Here we can only show some exemplary results. In Figure 1 the particle distribution of the simulation of the stream disk interaction and stream overflow of the dwarf nova IP Pegasi are shown. Details of the simulations are given in the figure captions. For clarity only the particles inserted during the last orbital period are shown. The most important results of this simulations are the rather massive stream overflow over the disk, resulting in the disposition of the in-falling material not only, and not even predominantly in the outer part of the disk, but rather at a radius close to the center. The other striking feature is the elevation of stream matter high above the disk plane, neatly explaining the x-ray dips that are often observed at this orbital phase in many medium to high inclination systems.
Apart from the exemplary results given here, the simulations span a large range of mass ratios, orbital periods, mass transfer rates and thermal states of the disk. Not only could we show that stream disk overflow always plays an important role in CV disks, our results also explain the UV and X-ray dips often observed in these sysytems. Under certain circumstances, namely a small disk or an enhanced mass transfer rate, a second absorption region occurs around orbital phase0.2. This feature is observed in some systems and could not be explained before. Moreover, the re-impact of the overflowing gas onto the disk close to the white dwarf around orbital phase 0.5 can be seen as a second bright spot. Such a feature has been observed in the dwarf nova WZ Sagittae. Also this observation had no explanation yet.
To our knowledge these are the only high-resolution simulations of the stream-disk impact in CVs including the full Roche potential, which is necessary because the structure of the outer disk rim is heavily influenced by the tidal forces from the secondary star. Due to the high computational requirements these simulations profit from the use of parallel machines. Although it is in principle possible to perform such simulations on top-of-the-notch workstations, only parallel machines make it possible to cover such a large range of parameters in a reasonable time.

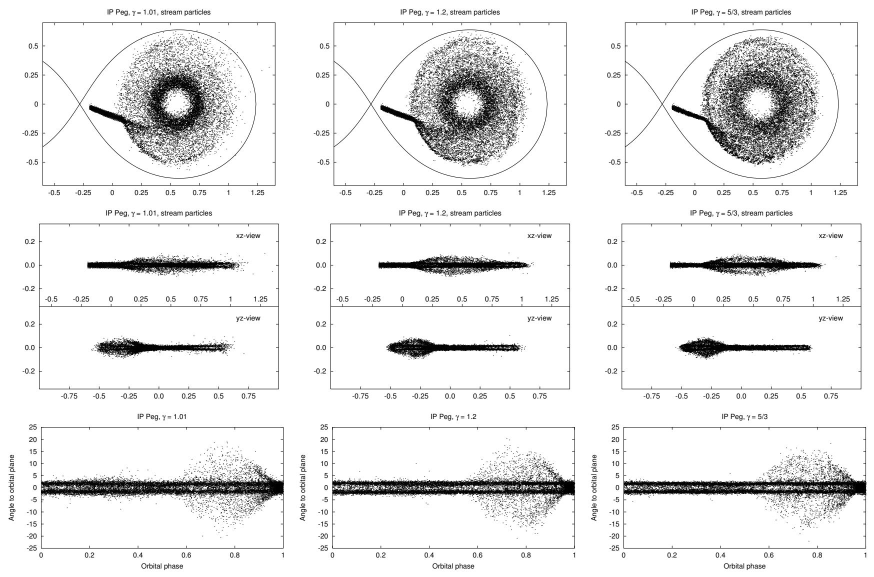

Fig. 1. Simulation of the dwarf nova IP Peg (primary mass: 1.15 $M_\odot$, secondary mass: 0.67 $M_\odot$, orbital period $3^h 48^m 20^s$, mass transfer rate: $10^{-10}\,\mathrm{M_\odot\,yr^{-1}}$) The left column shows particle distributions from a simulation with nearly isothermal equation of state, the right column is derived from a simulation with adiabatic equation of state, and in the simulation displayed in the middle column the polytropic coefficient was set to 1.2. Displayed are particles inserted during the last half orbital cycle. The upper row shows the distribution of the overflowing matter projected onto the orbital plane, the middle row shows edge-on views of the disks from two perspectives, namely perpendicular to and along the system axis. The lower row shows what the stream overflow would look like when seen from the white dwarf. Disk particles are not plotted. These simulations show that a substantial fraction of the accretion stream flows over the disk surface directly to inner parts of the disk even when the disk rim is geometrically thicker than the accretion stream at the impact region. The stream disk overflow can explain the X-ray and UV dips which are observed in many systems.

2.4 Parallel SPH Implementation

The parallel implementation of the used SPH code is written in C and parallelized using MPI. It is based on a code formerly written and optimized for the Cray T3E with the SHMEM communication library. The communication code is separated from the physical calculation in an own module allowing us to easily switch between different communication libraries such as MPI, PVM or SHMEM.
With older revisions of the MPI libraries on the Cray T3E we measured big performance differences between MPI and SHMEM for some operations, but after an update to new MPI libraries the difference between the SHMEM and MPI version was very small, with some performance advantages for the - slightly improved and optimized - MPI implementation.
The code itself is portable and runs with very minor differences on every machine, which provides the MPI communication library. We tested the code on Cray T3E, Hitachi SR8000, IBM SP and a Linux system.
SPH is a numerical method with high dynamics and therefor the code contains some complex and irregular data structures and does not profit much from vector machines or machines with a very good floating point performance.
The computation time of code is mainly consumed in two parts:

- The neighbour search, which is dominated by integer operations and
- computations with neighbour particle interactions which is mixed floating point and integer based.

Both parts have a spreaded memory access pattern and therefor cannot profit very much from processor caches.
Some smaller computations without neighbour interaction are more cache efficent but consume less than 5 percent of the overall computation time.
There is a simple API between the library containing the whole parallelization and the code containing the physics. The abstraction simplifies the extension of a simulation without knowing much about the parallelization. A user can add new physical quantities by allocating a new *parallel field.*
For the parallel computation the library provides an iterator concept to step through all particles and their neighbours and later communicates the new data. For all particle computations the user does not need to add explicit communication. But to prevent unnecessary communication the user explicitly has to register the parallel fields in the library, that are necessary for further computations. The library then ensures that for all particles returned by the iterator, that the necessary data is available on the node. If the field is no longer necessary for parallel computations, the user has to unregister the field.
On the former SHMEM based implementation there was a *load stealing* mechanism to optimize the rough static load–balancing *on-the-fly.* This part was removed in the MPI version, because it was heavily dependent on SHMEM one-sided communication operations. It showed that the MPI version of the

parallel SPH code with its static load–balancing scales very well without the load stealing.

2.5 Performance results

We measured the performance of the improved SPH 3D Code with particle injection on three different machines. Cray T3E and Hitachi SR8000 at the HLRS in Stuttgart and on the Kepler Cluster, a Linux Cluster installed in Tübingen. The Kepler Cluster has Dual SMP Pentium III nodes with 650 MHz processor speed. For the communication the cluster has a Myrinet network with a peak MPI bandwidth of about 115 MByte/s and a one-way latency of about 7μs.
We measured two different problems. A small problem with about 34 000 particles and a bigger problem with 360 000 particles.

One-node performance The small problem needs less than 100MB memory allowing us to compare the raw application performance of the three different machines, without communication. The code has fairly complex data structures and is therefor dominated by the integer and memory performance of the processors. So, using a Pentium III with its fairly bad floatingpoint performance is no disadvantage over the Alpha processors in the Cray T3E or the processors in the Hitachi SR8000. The application has a total floatingpoint performance of about 60 MFlop/s on one Pentium III processor.

Cray T3E	Hitachi SR8000	Kepler Cluster
903, 4 s	721, 3 s	467, 9 s

Parallel performance For better comparison of the three machines we decided to plot runtime information instead of speedup because the big problem fits only into at least 24 Cray T3E nodes. The big problem computes the *right hand side* 36 times and the small problem computes the *right hand side* 90 times. The three curves are all–over (wall clock) time, the time for a the nearest neighbour search and the parallel overhead including communication and additional work for load balancing (see 2, 3, 4) .

3 Particle-in-Cell with Monte-Carlo collisions

4 About PiC/MCC

The Particle-in-Cell method allows the simulation of large numbers of particles with short range interactions. In contrast to other particle methods such as SPH, the forces moving the particles are calculated using a fixed grid. Characteristic values of the particles are weighted on the grid using a

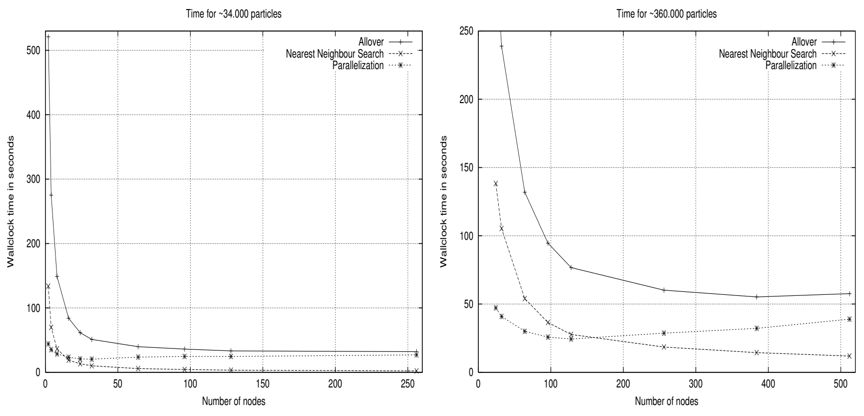

Fig. 2. The **Cray T3E** plot starts with 2 nodes for 34 000 particles and 24 nodes for 360 000 particles. The small problem does not scale very good for runs with more than 64 nodes. If we assume a speedup of 24 on 24 nodes for the big problem, we can achieve a maximum speedup of 128 for 384 processors. On 256 we have a speedup of about 124 and near 50 percent efficency. It showed, that a part with individual all–to–all communication becomes dominant for higher node numbers.

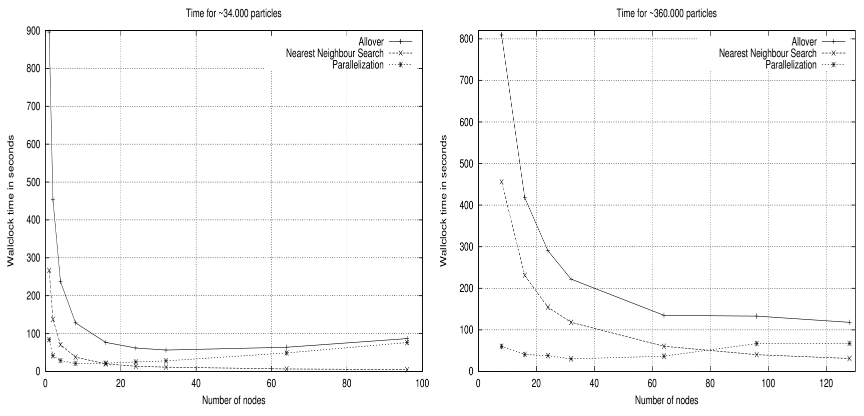

Fig. 3. We ran our tests on the **Hitachi SR8000** with pure MPI. Node numbers start at 2 nodes for the small problem and 8 nodes for the big problem. One can see a significant increase of the parallel overhead on more than 64 processors. Again the part with the individual all–to–all communication becomes very dominant. One should note, that we did not optimize the code for the Hitachi SR8000. Using threads instead of message passing for the inner node communication would probably increase the performance significantly. Therefor the maximum speedup compared with the one node run is limited to 53 on 128 nodes for the big problem.

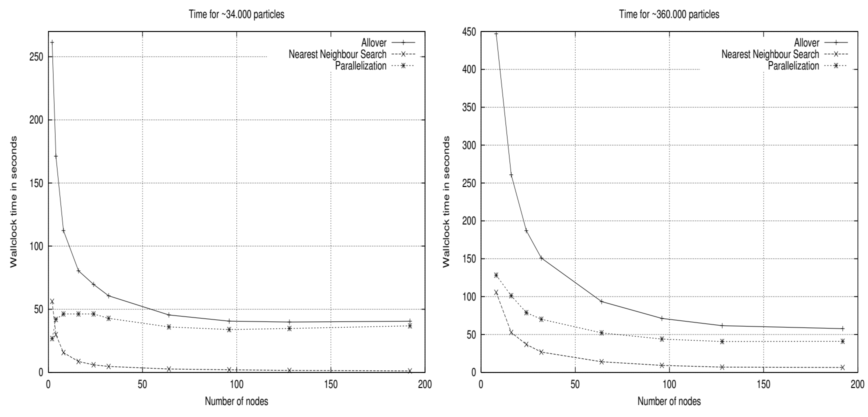

Fig. 4. On the **Kepler Linux Cluster** node numbers start at 2 nodes for the small problem and 8 nodes for the big problem. The Pentium III Processors have a fairly good integer performance and the contribution of the nearest neighbour search to the all-over time is very small. The Kepler cluster does not show the dominant all–to–all communication but has an expensive broadcast which is not this dominant on SR8000 and T3E. The big problem has a maximum speedup compared to the one node run of about 45 on 96 nodes using only one processor per node. This is worse compared to the other machines but the overall computation time using 64 nodes (128 processors) is about the same compared to 256 Cray T3E nodes.

kernel function. The momentum equations on the grid are solved using some standard method (for example finite differences). From the results the forces are calculated and interpolated back to the particles to solve the equations of motion. The basic algorithmic steps are summarized in figure 5.

The method reduces computational complexity from $O(n^2)$ to $O(n)$ for interpolating the n particles to the grid and the grid back to the particles. The complexity for solving the momentum equations on the grid is usually $O(g \log g)$ for g gridpoints. A detailed description of the method can be found in [1]

4.1 Physical problem

In our application the PiC method is used to simulate the electrostatic plasma of a direct current glow discharge in a tube. The simulation is effectively one-dimensional, since the problem has a cylindrical symmetry and we are not interested in the radius components. Each simulation particle i represents a number η_i of physical particles (electrons or ions), where $\eta \approx 2 \cdot 10^9$. The charge q_i of the particles is weighted to a regular one-dimensional grid of

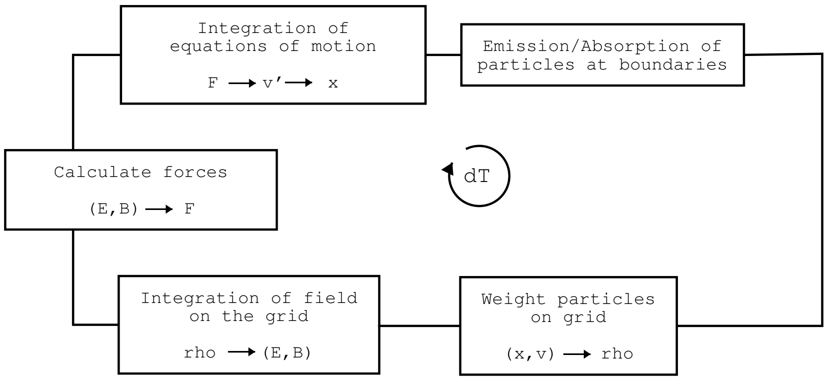
Integration of equations of motion
F → v′ → x
Emission/Absorption of particles at boundaries
Calculate forces
(E,B) → F
dT
Integration of field on the grid
rho → (E,B)
Weight particles on grid
(x,v) → rho

Fig. 5. Particle-in-Cell algorithm

width Δx using a triangular kernel function

$$W(r) = \begin{cases} 1 - \frac{|r|}{\Delta x} & \text{for } |r| \leq \Delta x \\ 0 & \text{else} \end{cases}$$

according to

$$\rho(x_g) = \frac{1}{\Delta x} \sum_{i=1}^{n} \eta_i q_i W(x_i - x_g).$$

In other words, the charge of each simulation particle is weighted according to the number of real particles represented to its two nearest grid neighbours. The normalization by Δx results in the discretized charge density ρ. Next, the electrostatic potential Φ and the electric field E are computed on the grid using discretized versions of

$$\Delta\Phi(x) = -\frac{1}{\epsilon_0}\rho(x)$$

and

$$E(x) = -\nabla\Phi(x).$$

To model the particle-particle interactions in the plasma, the PiC method has been extended by Monte-Carlo collision processes. The last step computes randomized particle-particle interactions:

- Elastic scattering
- Stimulation of a neutral gas atom by a electron
- Ionization of a neutral gas atom by an electron

Detail about the collision mechanisms can be found in [7].

4.2 Parallelization

The PiC application is implemented in C++ using object-oriented techniques. The basic structure can be found in the UML class diagram in figure 6. The application is structered in two parts: Management classes, responsible for providing a flexible and easy to configure user interface. The execution concept is centered around the CTask object, responsible for executing some part of the physical simulation. CTask and derived objects can also be made persistent in a textual representation, which serves to configure the physical environment and the execution order of calculations. Simulations with different physical behaviour can be tailored using this single configuration file [3,7].

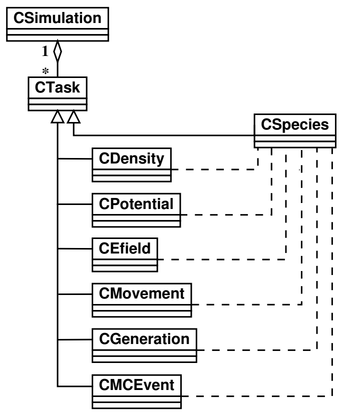

Fig. 6. Simplified class diagram of the PiC-MCC simulation

For the parallelization two aspects had to be considered: Data dependencies and load–balancing. Most of the steps in the PiC and MC collision code are independent calculations updating all particles. The only dependency lies in the global grid. Since, in our case the grid is much smaller than the number of particles ($g \ll n$), a natural approach is to parallelize over the number of particles and keep the grid redundant. After weighting the particles on the local grid, a global reduction operation provides each processor with a global grid. Next, each processor solves the grid equations in parallel. Measurements showed that this part of the calculation is lower than 1%, allowing for speedups ≥ 100.
Since the distribution of particles to processes is arbitrary, initially an optimal load-balance can be guaranteed. The particle-particle interactions calculated by Monte-Carlo collisions are divided between all processes and applied to the local subset of particles. Due to random variations of the number of particles

created or destroyed, a variation of the load-balancing over the time can be expected. Since the MC processes are equally distributed we expected the resulting load imbalance to be small.
The parallelized code is targeted for distributed memory architectures and is parallelized using TPO++, an object-oriented message-passing library [6,5].

4.3 Performance measurements

The performance of the application has been measured on the Cray T3E and the Kepler cluster. Both are distributed memory architectures with different characteristics: Single nodes of the Cray T3E are less performant (450 Mhz) and equipped with less memory (128 MB) compared to the Kepler cluster (650 Mhz dual processors and 512 MB memory per processor). On the other hand, the network of the Cray T3E reaching 300 MB/s is more performant than the 115 MB/s of Kepler.
The performance of the application is compared for three different numbers of real particles ($1.2 \cdot 10^{14}$, $1.2 \cdot 10^{15}$ and $1.2 \cdot 10^{16}$), different numbers of processors (1–96) and for different numbers of simulation steps (1000 and 5000) to analyze a possible load imbalance over the time.
The speedup and efficiency results for the Cray T3E can be found in figures 7 and 8, for the Kepler cluster in 9 and 10. Note that the speedup results on the Cray T3E are based on the 4-processor runs, since the problem is too large for 1 and 2 processors.
Both results show that the number of particles should be larger than $1.2 \cdot 10^{15}$ to obtain reasonable speedups. For $1.2 \cdot 10^{16}$ particles the application scales very good, reaching about 80% efficiency for 1000 simulation steps. The comparison of 1000 and 5000 simulation steps shows an unexpected decrease in performance. A detailed investigation reveals, that the load-imbalance due to the non-deterministic Monte-Carlo events gets worse with increasing number of steps, resulting in some processors which are responsible for 3 times of the particles compared to the optimal load-balance, effectively reducing the speedup by the same factor.
Comparing both architectures, we find for all runs the speedups on Kepler being lesser than the speedups on Cray T3E. This is due to different ratio of CPU performance to network performance of the two architectures, resulting in lesser speedup for Kepler, which has faster nodes and a slower communication network. A comparison of the total (wall clock) execution time of both runs confirms this, with Kepler being about 50% faster on single-node runs. The runtime advantage of Kepler gets smaller with increasing number of processors.

5 Conclusions and future work

The parallel 3D SPH code allows us to run bigger simulations in a reasonable time. The code is portable and all platforms are suitable for SPH production

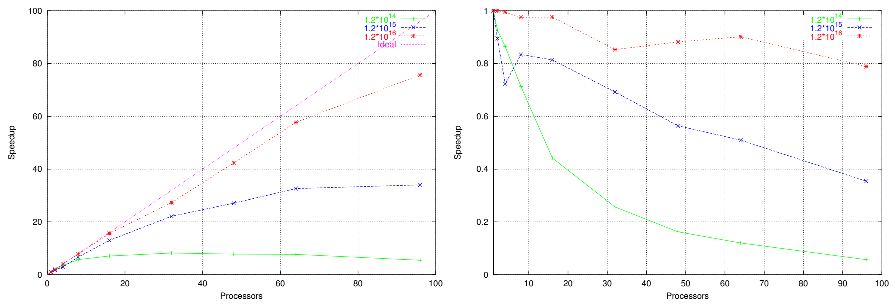

Fig. 7. Speedup and efficiency on Cray T3E for 1000 simulation steps and different number of particles.

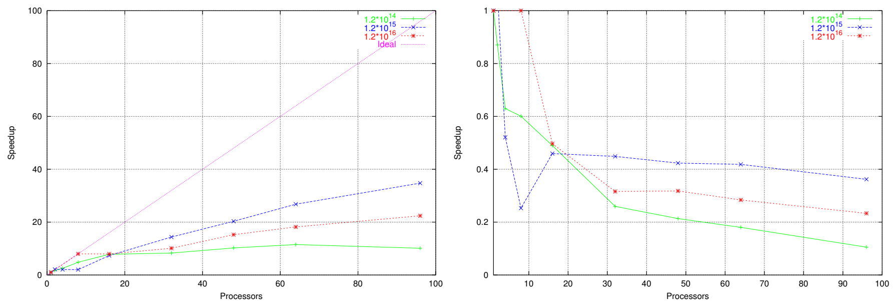

Fig. 8. Speedups and efficiency on Cray T3E for 5000 simulation steps and different number of particles.

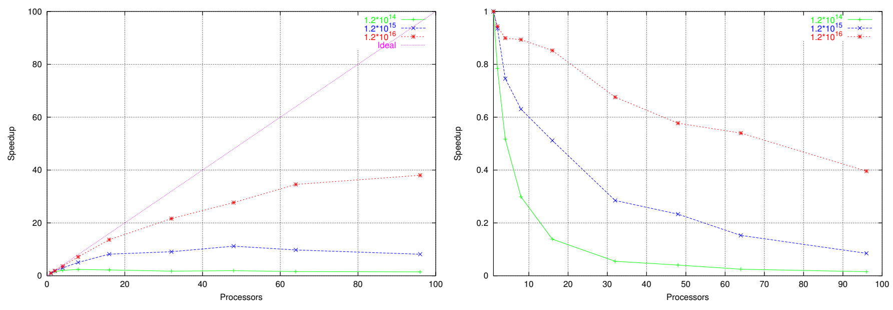

Fig. 9. Speedup and efficiency on Kepler for 1000 simulation steps and different number of particles.

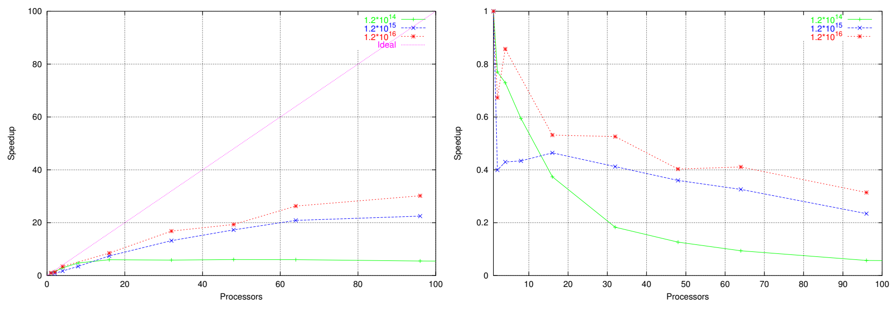

Fig. 10. Speedups and efficiency on Kepler for 5000 simulation steps and different number of particles.

runs. The platforms also allow us to compare and optimize parallel applications on machines with different parallel concepts. The main disadvantage of the Cray T3E is the amount of available memory on one node. It is often necessary to take more nodes to fit a problem into memory. This has been a problem in both applications.
On the SPH side, future works are the implementation of a fast three dimensional SPH code written in C++ and based on TPO++ together with an uniform I/O interface and data format based on XML to ease the share of simulation data with other implementations. It is also planned to implement a version with mixed MPI and thread-based parallelization. Especially machines with a mixed SMP/Message Passing architecture such as Hitachi SR8000 and the Kepler Cluster would profit from this optimization.
On the PiC side, we plan to extend the one-dimensional Particle-in-Cell code to solve two or three-dimensional problems. With respect to the parallelization, two improvements are needed: First, for large grids, the overhead solving the momentum equations on each processes must be avoided by a solver with the grid domain decomposed over all processors. Second, as the measurements showed, a significant load-imbalance can result due to the non-deterministic creation and destruction of particles on each processors. To improve the scaling behaviour, an unfrequently executed rebalancing step after reaching a user-definable threshold can improve the performance in such cases.

References

1. R. W. Hockney, J. W. Eastwood. *Computer simulation using particles.* Adam Hilger, Philadelphia, 1988.
2. W. Benz, R.L. Bowers, A.G.W. Cameron, and W.H. Press. Dynamic Mass Exchange In Doubly Degenerate Binaries. I. 0.9 adn 1.2 $M_\odot$. 348:647–667, 1990.

3. Th. Daube and H. Schmitz. Opar: Open architecture c++ plasma simulation code. Ruhr-Universitt Bonn, 1998.
4. R.A. Gingold and J.J. Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Mon. Not. R. astr. Soc.*, 181:375–389, 1977.
5. Tobias Grundmann, Marcus Ritt, and Wolfgang Rosenstiel. Object-oriented message-passing with TPO++. In Arndt Bode, Thomas Ludwig, Wolfgang Karl, and Roland Wissmüller, editors, *Lecture notes in computer science*, pages xx–yy. Springer-Verlag, 2000.
6. Tobias Grundmann, Marcus Ritt, and Wolfgang Rosenstiel. TPO++: An object-oriented message-passing library in C++. pages 43–50. IEEE Computer society, 2000.
7. A. Klaedtke. Particle-in-cell Simulationen mit Monte Carlo collisions. Master's thesis, Universität Stuttgart, Juli 1999.
8. Leon B. Lucy. A Numerical Approach to Testing the Fission Hypothesis. *Astron. J*, 82(12):1013–1924, December 1977.
9. J.J. Monaghan. Smoothed Particle Hydrodynamics. 30:543–74, 1992.
10. S.Kunze, E.Schnetter, and R.Speith. Development and Astrophysical Applications of a Parallel Smoothed Particle Hydrodynamics Code with MPI. pages 52 – 61.
11. T.Bubeck, M.Hipp, S.Hüttemann, S.Kunze, M.Ritt, W.Rosenstiel, H.Ruder, and R.Speith. SPH test simulations on a portable parallel environment. pages 139 – 155.
12. T.Bubeck, M.Hipp, S.Hüttemann, S.Kunze, M.Ritt, W.Rosenstiel, H.Ruder, and R.Speith. Parallel SPH on Cray T3E and NEC SX-4 using DTS. In W.Jäger E.Krause, editor, *High Performance Computing in Science and Engineering '98*, pages 396 – 410. Springer, 1999.