Preface

To be completed.

Complex systems modeling and simulation approaches are being adopted in a growing number of sectors, including finance, economics, biology, astronomy, and many more. Technologies ranging from distributed computing to specialized hardware are explored and developed to address the computational requirements arising in complex systems simulations. The aim of the book is to present a representative overview of contemporary large-scale computing technologies in the context of complex systems simulation applications. The intention is to present the state of the art and to identify new research directions in this field and to provide a communication platform facilitating an exchange of concepts, ideas and needs between computer scientists and technologists and complex systems modelers. On the application side, the book focuses on modeling and simulation of natural and man-made complex systems, because high-level requirements are similar across a wide range of domains. On the computing technology side emphasis is placed on distributed computing approaches, but supercomputing and other novel technologies (e.g., specialized hardware) are also considered.

Coleraine,
October, 2010

Werner Dubitzky
Krzysztof Kurowski
Bernhard Schott

Contents

List of Contributors

Rainer Spurzem
National Astronomical Observatories, Chinese Academy of Sciences, Beijing,
Astronomisches Rechen-Institut, Zentrum für Astronomie, University of Heidelberg,
Kavli Institute for Astronomy and Astrophysics, Peking University
e-mail: spurzem@bao.ac.cn

Peter Berczik
National Astronomical Observatories, Chinese Academy of Sciences, Beijing,
Astronomisches Rechen-Institut, Zentrum für Astronomie, University of Heidelberg,
Main Astron. Observatory of National Academy of Sciences of Ukraine
e-mail: berczik@bao.ac.cn,

Ingo Berentzen
Astronomisches Rechen-Institut, Zentrum für Astronomie, University of Heidelberg, Heidelberg
e-mail: iberent@ari.uni-heidelberg.de

Wei Ge, Xiaowei Wang
Institute of Process Engineering, Chinese Academy of Sciences, Beijing
e-mail: {wge,xwwang}@home.ipe.ac.cn

Hsi-Yu Schive
Department of Physics, National Taiwan University, Taibei
e-mail: b88202011@ntu.edu.tw

Keigo Nitadori
RIKEN AICS Institute, Kobe
e-mail: keigo@riken.jp

Tsuyoshi Hamada
Nagasaki Advanced Computing Center, Nagasaki University

e-mail: hamada@nacc.nagasaki-u.ac.jp

José Fiestas
Astronomisches Rechen-Institut, Zentrum für Astronomie, University of Heidelberg
e-mail: fiestas@ari.uni-heidelberg.de

Chapter 1
Accelerated Many-Core GPU computing for physics and astrophysics on three continents

Rainer Spurzem, Peter Berczik, Ingo Berentzen, Ge Wei, Wang Xiaowei, Hsi-Yu Schive, Keigo Nitadori, Tsuyoshi Hamada, José Fiestas

Rainer Spurzem
National Astronomical Observatories of China, Chinese Academy of Sciences,
and Astronomisches Rechen-Institut, Zentrum für Astronomie, University of Heidelberg,
and Kavli Institute for Astronomy and Astrophysics, Peking University
e-mail: spurzem@bao.ac.cn

Peter Berczik
National Astronomical Observatories of China, Chinese Academy of Sciences,
and Astronomisches Rechen-Institut, Zentrum für Astronomie, University of Heidelberg
e-mail: berczik@bao.ac.cn

Ingo Berentzen
Zentrum für Astronomie, University of Heidelberg
e-mail: iberent@ari.uni-heidelberg.de

Wei Ge
Institute of Process Engineering, Chinese Academy of Sciences, Beijing
e-mail: wge@home.ipe.ac.cn

Xiaowei Wang
Institute of Process Engineering, Chinese Academy of Sciences, Beijing
e-mail: xwwang@home.ipe.ac.cn

Hsi-Yu Schive
Department of Physics, National Taiwan University, Taibei
e-mail: b88202011@ntu.edu.tw

Keigo Nitadori
RIKEN AICS Institute, Kobe
e-mail: keigo@riken.jp

Tsuyoshi Hamada
Nagasaki Advanced Computing Center, Nagasaki University
e-mail: hamada@nacc.nagasaki-u.ac.jp

José Fiestas
Astronomisches Rechen-Institut, Zentrum für Astronomie, University of Heidelberg
e-mail: fiestas@ari.uni-heidelberg.de

1.1 Introduction

Multi-scale physical and astrophysical simulations on new many-core accelerator hardware (GPU), used for practical research in our fields, are presented. We select here as examples those of our algorithms which already scale well for parallel clusters using many GPU, right into the Petaflops scale, with potential for Exaflops. These are particle based astrophysical many-body simulations with self-gravity, as well as particle and mesh-based simulations on fluid flows, from astrophysics and physics, partly also self-gravitating. Strong and soft scaling is shown, using some of the fastest GPU clusters in China, but also on other hardware resources of cooperating teams in Germany and USA, linked in the cooperation of ICCS (International Center for Computational Science). In all applications high effective performance is reached.

Theoretical numerical modelling has become a third pillar of sciences in addition to theory and experiment (in case of astrophysics the experiment is mostly substituted by observations). Numerical modelling allows one to compare theory with experimental or observational data in unprecedented detail, and it also provides theoretical insight into physical processes at work in complex systems. Similarly, data processing (e.g., of astrophysical observations) comprises the use of complex software pipelines to bring raw data into a form digestible for observational astronomers and ready for exchange and publication; these are, e.g., mathematical transformations like Fourier analyses of time series or spatial structures, complex template analyses or huge matrix-vector operations. Here, fast access to and transmission of data, too, require supercomputing capacities. However, sufficient resolution of multi-scale physical processes still poses a formidable challenge, such as in the examples of few-body correlations in large astrophysical many-body systems, or in the case of turbulence in physical and astrophysical flows.

We are undergoing a new revolution on parallel processor technologies, and a change in parallel programming paradigms, which may help to advance current software towards the Exaflops scale and help better resolving and understanding typical multi-scale problems. The current revolution in parallel programming has been mostly catalysed by the use of graphical processing units (GPU's) for general purpose computing, but it is not clear whether this will remain the case in the future. GPU's have become widely used nowadays to accelerate a broad range of applications, including computational physics and astrophysics, image/video processing, engineering simulations, quantum chemistry, just to name a few (Egri et al., 2007; Yasuda, 2007; Yang et al., 2007; Akeley et al., 2007; Hwu, 2011). Graphics processing unit are rapidly emerging as a powerful and cost-effective platform for high performance parallel computing. The GPU Technology Conference 2010 held by NVIDIA in San Jose in autumn 2010[1] gave one snapshot of the breadth and depth of present day GPU (super)computing applications. Recent GPU's, such as the NVIDIA Fermi C2050 Computing Processor, offer 448 processor cores and extremely fast on-chip-memory chip, as compared to only 4-8 cores on a standard

[1] `http://www.nvidia.com/gtc`

Intel or AMD CPU. Groups of cores have access to very fast shared memory pieces; a single Fermi C2050 device supports double precision operations fully with a peak speed of 515 Gflops; in this paper we also present results obtained from GPU clusters with previous generations of GPU accelerators, which have no (Tesla C870) or only very limited (Tesla C1060) double precision support. We circumvented this by emulation of a few critical double precision operations (e.g., Nitadori and Makino, 2008). More details can be found in the Ph.D. thesis of one of us (Keigo Nitadori), "New approaches to high-performance N-body simulations with high-order integrator, new parallel algorithm, and efficient use of SIMD hardware", Univ. of Tokyo, 2009.

Scientists are using GPU's since more than five years already for scientific simulations, but only the invention of CUDA (Compute Unified Device Architecture; Akeley et al., 2007; Hwu, 2011) as a high-level programming language for GPU's made their computing power available to any student or researcher with normal scientific programming skills. CUDA is presently limited to GPU devices of NVIDIA, but the open source language OpenCL will provide access to any type of many-core accelerator through an abstract programming language. Computational physics and astrophysics has been a pioneer to use GPU's for high performance general purpose computing (see for example the early AstroGPU workshop in Princeton 2007, through the information base[2]. Astrophysicists had an early start in the field through the GRAPE (Gravity Pipe) accelerator boards from Japan from 10 years ago (Makino et al., 2003; Fukushige et al., 2005, and earlier references therein). Clusters with accelerator hardware (GRAPE or GPU) have been used for gravitating many-body simulations to model the dynamics of galaxies and galactic nuclei with supermassive black holes in galactic nuclei (Berczik et al., 2005, 2006; Berentzen et al., 2009; Pasetto et al., 2011; Just et al., 2011) , the dynamics of dense star clusters (Belleman et al., 2008; Portegies Zwart et al., 2007; Hamada and Iitaka, 2007), in gravitational lensing ray shooting problems (Thompson et al., 2010), in numerical hydrodynamics with adaptive mesh refinement (Wang and Abel, 2009; Wang et al., 2010a; Schive et al., 2010) and magnetohydrodynamics (Wong et al., 2009), or Fast Fourier transformation (Cui et al., 2009). While it is relatively simple to obtain good performance with one or few GPU relative to CPU, a new taxonomy of parallel algorithms is needed for parallel clusters with many GPU's (Barsdell et al., 2010). Only "embarrassingly" parallel codes scale well even for large number of GPU's, while in other cases like hydrodynamics or FFT on GPU the speed-up is somewhat limited to 10-50 for the whole application, and this number needs to be carefully checked whether it compares the GPU performance with single or multi-core CPUs. A careful study of the algorithms and their data flow and data patterns, is useful and has led to significant improvements, for example for particle based simulations using smoothed particle hydrodynamics (Berczik et al., 2007; Spurzem et al., 2009) or for FFT (Cui et al., 2009). Recently new GPU implementations of Fast-Multipole Methods (FMM) have been presented and compared with Tree Codes (Yokota and Barba, 2010; Yokota et al., 2010). FMM codes have first been presented by Green-

[2] `http://www.astrogpu.org`

gard and Rokhlin (1987). It is expected that on the path to Exascale applications further - possibly dramatic - changes in algorithms are required; at present it is unclear whether the current paradigm of heterogeneous computing with a CPU and an accelerator device like GPU will remain dominant.

While the use of many-core accelerators is strongly growing in a large number of scientific and engineering fields, there are still only few codes able to fully harvest the computational power of parallel supercomputers with many GPU devices, as they have been recently became operational in particular (but not restricted to) in China. In China GPU computing is blooming, the top and third spot in the list of 500 fastest supercomputers in the world[3] are now occupied by Chinese GPU clusters, and one of the GPU clusters used for results in this paper is on rank number 28 (Mole-8.5 computer, see below and Wang et al. (2010b)). In this article we present in some detail an astrophysical N-body application for star clusters and galactic nuclei and star clusters, which is currently our mostly well tested and also heavily used application. Furthermore, somewhat less detailed, we present other applications scaling equally very well, such as an adaptive mesh refinement hydrodynamic code, using (among other parts) an FFT and relaxation methods to solve Poisson's equation, and give some overview on physical and process engineering simulations.

1.2 Astrophysical application for star clusters and galactic nuclei

Dynamical modelling of dense star clusters with and without massive black holes poses extraordinary physical and numerical challenges; one of them is that gravity cannot be shielded such as electromagnetic forces in plasmas, therefore long-range interactions go across the entire system and couple non-linearly with small scales; high-order integration schemes and direct force computations for large numbers of particles have to be used to properly resolve all physical processes in the system. On small scales inevitably correlations form already early during the process of star formation in a molecular cloud. Such systems are dynamically extremely rich, they exhibit a strong sensitivity to initial conditions and regions of phase space with deterministic chaos.

After merging of two galaxies in the course of cosmological structure formation we start our simulations with two supermassive black holes (SMBH) embedded in a dense star cluster, separated by some 1000 pc (1 pc, 1 parsec, about 3.26 light years, or $3.0857 \cdot 10^{18}$cm). This is a typical separation still accessible to astrophysical observations (Komossa et al., 2003). Nearly every galaxy harbours a supermassive black hole, and galaxies build up from small to large ones in a hierarchical manner through mergers following close gravitational encounters. However, the number of binary black holes observed is relatively small, so there should be some mechanism, by which they get close enough to each other to coalesce under emission of gravitational wave emission. Direct numerical simulations of Einstein's field equations

[3] `http://www.top500.org`

start usually at a black hole separation of order 10-50 Schwarzschild radii, which is, for an example of a one million solar mass black hole (similar to the one in our own galactic centre) about 10^{-5} pc. Therefore, in order to obtain a merger, about eight orders of magnitude in separation need to be bridged. In our recent models we follow in one coherent direct N-body simulation how interactions with stars of a surrounding nuclear star cluster combined with the onset of relativistic effects, lead to a black hole coalescence in galactic nuclei after an astrophysically modest time of order 10^8 years (Berentzen et al., 2009; Preto et al., 2011). Corresponding to the multi-scale nature of the problem in space we have a large range of time scales to be covered accurately and efficiently in the simulation. Orbital times of supermassive black holes in galactic nuclei after galactic mergers are of the order of several million years; in the interaction phase with single stars the orbital time of a gravitationally bound supermassive binary black hole goes down to some 100 years - at this moment there is a first chance to detect its gravitational wave emission through their influence on pulsar timing (Lee et al., 2011). Energy loss due to Newtonian interactions with field stars interplays with energy loss due to gravitational radiation emission; the latter becomes dominant in the final phase (at smaller separations), when the black hole binary enters the waveband of the planned laser interferometer space antenna (LISA[4]), where one reaches 0.01 Hz orbital frequency. Similarly in a globular star cluster time scales can vary between a million years (for an orbit time in the cluster) to hours (orbital time of the most compact binaries). The nature of gravity favours such strong structuring properties, since there is no global dynamical equilibrium. Gravitationally bound subsystems (binaries) tend to exchange energy with the surrounding stellar system in a way that increases their binding energy, thus moving further away from a global equilibrium state. This behaviour can be understood in terms of self-gravitating gas spheres undergoing gravothermal catastrophe (Lynden-Bell and Wood, 1968), but it occurs in real star clusters on all scales. Such kind of stellar systems, sometimes called dense or gravothermal stellar systems, demands special high accuracy integrators due to secular instability, deterministic chaos and strong multi-scale behaviour. Direct high-order N-body integrators for this type of astrophysical problems have been developed by Aarseth (see for references Aarseth (1999b, 2003)). They employ fourth order time integration using a Hermite scheme, hierarchically blocked individual particle time steps, an Ahmad-Cohen neighbour scheme, and regularisation of close few-body systems.

Direct N-Body Codes in astrophysical applications for galactic nuclei, galactic dynamics and star cluster dynamics usually have a kernel in which direct particle-particle forces are evaluated. Gravity as a monopole force cannot be shielded on large distances, so astrophysical structures develop high density contrasts. High-Density regions created by gravitational collapse co-exist with low-density fields, as is known from structure formation in the universe or the turbulent structure of the interstellar medium. A high-order time integrator in connection with individual, hierarchically blocked time steps for particles in a direct N-body simulation provides the best compromise between accuracy, efficiency and scalability (Makino and Hut,

[4] `http://lisa.nasa.gov/`

1988; Aarseth, 1999b,a; Spurzem, 1999; Harfst et al., 2007). With GPU hardware up to a few million bodies could be reached for our models (Berczik et al., 2005, 2006; Gualandris and Merritt, 2008). Note that while Greengard and Rokhlin (1987) already mention that their algorithm can be used to compute gravitational forces between particles to high accuracy, Makino and Hut (1988) find that the self-adaptive hierarchical time-step structure inherited from Aarseth's codes improves the performance for spatially structured systems by $\mathscr{O}(\mathscr{N})$ - it means that at least for astrophysical applications with high density contrast FMM is not a priori more efficient than direct N-body (which sometimes is called "brute force", but that should only be used if a shared time step is used, which is not the case in our codes). One could explain this result by comparing the efficient spatial decomposition of forces (in FMM, using a simple shared time step) with the equally efficient temporal decomposition (in direct N-body, using a simple spatial force calculation).

On the other hand, cosmological N-body simulations use thousand times more particles (billions, order 10^9), at the price of allowing less accuracy for the gravitational force evaluations, either through the use of a hierarchical decomposition of particle forces in time (so-called neighbour scheme codes, Ahmad and Cohen, 1973; Makino and Aarseth, 1992; Aarseth, 2003), or in space (tree codes, Barnes and Hut, 1986; Makino, 2004; Springel, 2005). Another possibility is the use of fast-multipole algorithms (Greengard and Rokhlin, 1987; Dehnen, 2000, 2002; Yokota and Barba, 2010; Yokota et al., 2010) or particle-mesh schemes (PM, Hockney and Eastwood, 1988; Fellhauer et al., 2000) which use FFT for their Poisson solver. PM schemes are the fastest for large systems, but their resolution is limited to the grid cell size. Adaptive codes use direct particle-particle forces for close interactions below grid resolution (AP3M, Couchman et al., 1995; Pearce and Couchman, 1997). But for astrophysical systems with high density contrasts tree codes are more efficient. Recent codes for massively parallel supercomputers try to provide adaptive schemes using both tree and PM, such as the well-known GADGET and treePM codes (Xu, 1995; Springel, 2005; Yoshikawa and Fukushige, 2005; Ishiyama et al., 2009).

1.3 Hardware

We present results obtained from GPU clusters using NVIDIA Tesla C1060 cards in Beijing, China (Laohu cluster with 85 Dual Intel Xeon nodes and with 170 GPU's); NVIDIA Fermi C2050 cards also in Beijing, China (Mole-8.5 cluster with 372 dual Xeon nodes, most of which have 6 GPU's, delivering in total 2000 Fermi Tesla C2050 GPU's); in Heidelberg, Germany using NVIDIA Tesla C870 (pre-Fermi single precision only generation) cards (Kolob cluster with 40 Dual Intel Xeon nodes and with 40 GPU's.); and Berkeley at NERSC/LBNL using again the NVIDIA Fermi Tesla C2050 cards (Dirac cluster with 40 GPU's).

In Germany, at Heidelberg University, our teams have operated a many-core accelerated cluster using the GRAPE hardware for many years (Harfst et al., 2007;

Spurzem et al., 2004, 2007, 2008). We have in the meantime migrated from GRAPE to GPU (and also partly FPGA) clusters (Spurzem et al., 2009, 2010, 2011), and part of our team is now based at the National Astronomical Observatories of China (NAOC) of Chinese Academy of Sciences (CAS), in Beijing. NAOC is part of a GPU cluster network covering ten institutions of CAS, aiming for high performance scientific applications in a cross-disciplinary way. The top level cluster in this network is the recently installed Mole-8.5 cluster at Institute of Process Engineering (IPE) of CAS in Beijing (2 Pflops single precision peak), from which we also show some preliminary benchmarks. The entire CAS GPU cluster network has a total capacity of nearly 5 Pflops single precision peak. In China GPU computing is blooming, the top and third spot in the list of 500 fastest supercomputers in the world[5] are now occupied by Chinese GPU clusters. The top system in the CAS GPU cluster network is currently number 28 (Mole-8.5 at IPE). Research and Teaching in CAS institutions is focused on broadening the computational science base to use the clusters for supercomputing in basic and applied sciences.

Fig. 1.1 Left: NAOC GPU cluster in Beijing; 85 nodes with 170 NVIDIA Tesla C1060 GPU's, 170 Tflops hardware peak speed, installed 2010; **Right:** Frontier Kolob cluster at ZITI Mannheim, 40 nodes with 40 NVIDIA Tesla C870 GPU accelerators, 17 Tflops hardware peak speed; installed 2008.

1.4 Software

The test code which we use for benchmarking on our clusters is a direct N-body simulation code for astrophysics, using a high order Hermite integration scheme and

[5] `http://www.top500.org`

individual block time steps (the code supports time integration of particle orbits with 4th, 6th, and 8th order schemes). The code is called φGPU, it has been developed from our earlier published versions φGRAPE (using GRAPE hardware instead of GPU, Harfst et al., 2007). It is parallelised using MPI, and on each node using many cores of the special hardware. The code was mainly developed and tested by two of us (Keigo Nitadori, Peter Berczik, see also Hamada and Iitaka (2007)) and is based on an earlier version for GRAPE clusters (Harfst et al., 2007). The code is written in C++ and based on Nitadori and Makino (2008) earlier CPU serial code (yebisu).

The present version of φGPU code we used and tested only with the recent GNU compilers (ver. 4.x). More details will be published in an upcoming publication (Berczik et al., 2011).

The MPI parallelisation was done in the same "j" particle parallelisation mode as in the earlier φGRAPE code (Harfst et al., 2007). The particles are divided equally between the working nodes and in each node we calculate only the fractional forces for the active "i" particles at the current time step. Due to the hierarchical block time step scheme the number $N_{\rm act}$ of active particles (due for a new force computation at a given time level) is usually small compared to the total particle number N, but its actual value can vary from $1 \ldots N$. The full forces from all the particles acting on the active particles we get after using the global MPI_SUM communication routines.

We use native GPU support and direct code access to the GPU with only CUDA. Recently we use the latest CUDA 3.2 (but the code was developed and working also with the "older" CUDA compilers and libraries). Multi GPU support is achieved through MPI parallelisation; each MPI process uses only a single GPU, but we can start two MPI processes per node (to use effectively for example the dual quad core CPU's and the two GPU's in the NAOC cluster) and in this case each MPI process uses its own GPU inside the node. Communication always (even for the processes inside one node) works via MPI. We do not use any of the possible OMP (multi-thread) features of recent gcc 4.x compilers inside one node.

1.5 Results of benchmarks

The figures 1.2 and 1.3 show results of our benchmarks. In the case of Laohu we use maximum 164 GPU cards (3 nodes i.e. 6 cards were down during the test period). Here the largest performance was reached for 6 million particles, with 51.2 Tflops in total sustained speed for our application code, in an astrophysical run of a Plummer star cluster model, simulating one physical time unit (about one third of the orbital time at the half-mass radius). Based on these results we see that we get a sustained speed for 1 NVIDIA Tesla C1060 GPU card of 360 Gflops (i.e. about one third of the theoretical hardware peak speed of 1 Tflops). Equivalently, for the smaller and older Kolob cluster with 40 NVIDIA Tesla C870 GPU's in Germany, we obtain 6.5 Tflops (with 4 million particles). This is 160 Gflops per card.

On the new clusters Dirac and Mole-8.5 where we use the NVIDIA Fermi Tesla C2050 cards we get the maximum performance of 550 Gflops per card. The absolute

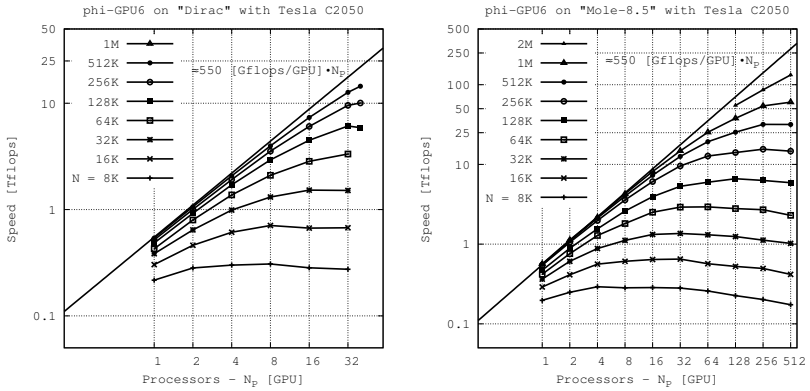

Fig. 1.2 Strong scaling for different problem sizes; **Left:** Dirac Fermi Tesla C2050 GPU system at NERSC/LBNL, almost 18 Teraflops reached for one million particles on 40 GPU's. Each line corresponds to a different problem size (particle number), which is given in the key. Note that the linear curve corresponds to ideal scaling. **Right:** Same benchmark simulations, but for the Mole-8.5 GPU cluster at IPE in Beijing, using up to 512 Fermi Tesla C2050 GPU's, reaching a sustained speed of 130 Teraflops (for two million particles). If one would use all 2000 GPU's on the system a sustained speed of more than 0.4 Petaflops is feasible. This is subject of ongoing present and future work.

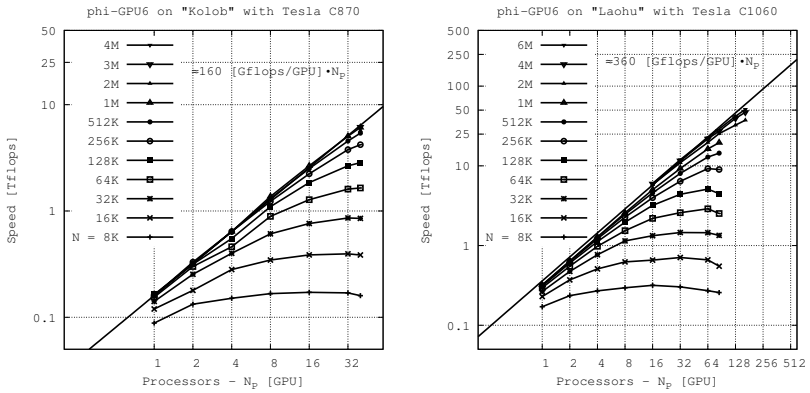

Fig. 1.3 Left: Same benchmark simulations as in Fig. 1.2, but for the Frontier Kolob cluster with Tesla C870 GPU's at University of Heidelberg, 6.5 Tflops reached for four million particles on 40 GPU's. **Right:** NAOC GPU cluster in Beijing; speed in Teraflops reached as a function of number of processes, each process with one GPU; 51.2 Tflops sustained were reached with 164 GPU's (3 nodes with 6 GPU's were down at the time of testing).

record in the performance we achieve on Mole-8.5 cluster when we run our test simulation (even for relatively "low" particle number – two million) on 512 nodes and get over the 130 Tflops total performance. In principle for larger particle number (in order of ten million) we see that the maximum performance which we can get on the whole cluster (on ≈2000 GPU's) is around 0.4 Pflops.

We have presented exemplary implementations of direct gravitating N-body simulations and adaptive mesh hydrodynamics code with self-gravity (Schive et al., 2010) using large GPU clusters in China and elsewhere. The overall parallelisation efficiency of our codes is very good. It is about 30% of the GPU peak speed in Fig. 1.2 for the embarrassingly parallel direct N-body code and still significant (order 20-40 speedup for each GPU) for adaptive mesh hydrodynamical simulations. The larger N-body simulations (several million particles) show nearly ideal strong scaling (linear relation between speed and number of GPU's) up to our present maximum number of nearly 170 GPU's - no strong sign of a turnover yet due to communication or other latencies. Therefore we are currently testing the code implementation on much larger GPU clusters, such as the Mole-8.5 of IPE/CAS.

Fig. 1.4 Left: The Mole-8.5 Cluster at Institute of Process Engineering in Beijing. It consists of 372 nodes, most with 6 Fermi Tesla C2050 GPU's. **Right:** Single node of Mole-8.5 system. (Courtesy of IPE, photos by Xianfeng He)

The wall clock time T needed for our particle based algorithm to advance the simulation by a certain physical time (usually 1 crossing time units) integration interval scales as:

$$T = T_{\rm host} + T_{\rm GPU} + T_{\rm comm} + T_{\rm MPI} \tag{1.1}$$

Data of the Mole-8.5 system	
item	quantity
Peak Performance Single Precision	2 Petaflops
Peak Performance Double Precision	1 Petaflops
Linpack Sustained Performance	207.3 Teraflops
Megaflops per Watt	431
Number of Nodes/Number of GPU's (Type)	372/2000 (Fermi Tesla C2050)
Total Memory RAM	17.8 Terabytes
Total Memory VRAM	6.5 Terabytes
Total Harddisk	720 Terabytes
Management Communication	H3C Gigabit Ethernet
Message Passing Communication	Mellanox Infiniband Quad Data Rate
Occupied area	150 sq.m.
Weight	12.6 ton
Max Power	600 kW (computing)
	200 kW (cooling)
Operating System	CentOS 5.4, PBS
Monitor	Ganglia, GPU monitoring
Languages	C, C++, CUDA

Table 1.1 Properties of the Fermi GPU cluster at the Institute of Process Engineering of Chinese Academy of Sciences (IPE/CAS); this system is the largest GPU cluster in Beijing, the third Chinese cluster, with rank 28 in the worldwide Top500 list (as of November 2010). It has been used for some of our N-body benchmarks, especially for the timing model, and by the physics simulations at IPE. Note that it has a relatively large number of GPU's per node, but our communication performance was not significantly affected (see comparison plots with Dirac cluster in Berkeley, which has only one GPU per node).

where the components of T are (from left to right) the computing time spent on the host, on the GPU, the communication time to send data between host and GPU, and the communication time for MPI data exchange between the nodes. In our present implementation all components are blocking, so there is no hiding of communication. This will be improved in further code versions, but for now it eases profiling.

In the case of φGPU code (as in the other direct NBODY codes discussed below) we use the blocked hierarchical individual timestep scheme (HITS) and a Hermite high order time integration scheme of at least 4th order for integration of the equation of motions for all particles (Makino and Aarseth, 1992). In the case of HITS in every individual timesteps we integrate the motion only for s particles, a number which is usually much less compared to the total number of particles N. Its average value $\langle s \rangle$ depends on the details of the algorithm and on the particle configuration integrated. According to a simple theoretical estimate it is $\langle s \rangle \propto N^{2/3}$ (Makino, 1991), but the real value of the exponent deviates from 2/3, depending on the initial model and details of the time step choice (Makino and Hut, 1988).

We use a detailed timing model for the determination of the wall clock time needed for different components of our code on CPU and GPU, which is then fitted to the measured timing data. Its full definition is given in Table 1.2.

Components in our timing model for direct N-body code		
task	expected scaling	timing variable
active particle determination	$\mathcal{O}(s\log(s))$	$T_{\rm host}$
all particle prediction	$\mathcal{O}(N/N_{\rm GPU})$	$T_{\rm host}$
"j" part. send. to GPU	$\mathcal{O}(N/N_{\rm GPU})$	$T_{\rm comm}$
"i" part. send. to GPU	$\mathcal{O}(s)$	$T_{\rm comm}$
force computation on GPU	$\mathcal{O}(Ns/N_{\rm GPU})$	$T_{\rm GPU}$
receive the force from GPU	$\mathcal{O}(s)$	$T_{\rm comm}$
MPI global communication	$\mathcal{O}((\tau_{\rm lat}+s)\log(N_{\rm GPU}))$	$T_{\rm MPI}$
correction/advancing "i" particle	$\mathcal{O}(s)$	$T_{\rm host}$

Table 1.2 Breaking down the computational tasks in a parallel direct N-body code with individual hierarchical block time steps; at every block time step level we denote $s \leq N$ particles, which should be advanced by the high order corrector as active or "i" particles, while the field particles, which exert forces on the "i" particles to be computed are denoted as "j" particles. Note that the number of "j" particles in our present code is always N (full particle number), but in more advanced codes like NBODY6 discussed below the Ahmad-Cohen neighbour scheme uses for the more frequent neighbour force calculation a much smaller number of "j" particles. We also have timing components for low-order prediction of all "j" particles and distinguish communication of data from host to GPU and return, and through the MPI message passing network.

In practice we see that only three terms play any relevant role to understand the strong and weak scaling behaviour of our code, these are the force computation time (on GPU) $T_{\rm GPU}$, and the message passing communication time $T_{\rm MPI}$, within which we can distinguish a bandwidth dependent part (scaling as $s\log(N_{\rm GPU})$) and a latency dependent part (scaling as $\tau_{\rm lat}\log(N_{\rm GPU})$); the latency is only relevant for a downturn of efficiency for strong scaling at relatively large numbers $N_{\rm GPU}$. Starting in the strong scaling curves from the dominant term at small $N_{\rm GPU}$ there is a linearly rising part in Fig. 1.2, just the force computation on GPU, while the turnover to a flat curve is dominated by the time of MPI communication between the computing nodes – $T_{\rm MPI}$.

To find a model for our measurements we use the ansatz

$$P = (\text{total number of floating point operations})/T \tag{1.2}$$

where T is the computational wall clock time needed. For one block step the total number of floating point operations is $\gamma\langle s\rangle N$, where γ defines how many floating point operations our particular Hermite scheme requires per particle per step, and we have

$$P_s = \frac{\gamma\, N\, \langle s\rangle}{T_s} = \frac{\gamma\, N\, \langle s\rangle}{\alpha\, N\, \langle s\rangle/N_{\rm gpu} + \beta\, (\tau_{\rm lat} + \langle s\rangle) log(N_{\rm gpu})} \tag{1.3}$$

where T_s is the computing time needed for one average block step in time (advancing $\langle s\rangle$ particles). The reader with interest in more detail how this formula can be theoretically derived for general purpose parallel computers is referred to Dorband et al. (2003). α, β and $\tau_{\rm lat}$ are hardware time constants for the floating point

calculation on GPU, for the bandwidth of the interconnect hardware used for message passing and its latency, respectively.

Our timing measurements are done for an integration over one physical time unit in normalised units ($t = 1$, which is equivalent to approximately one third of a particle's orbital crossing time at the half-mass radius), so it is more convenient to multiply the enumerator and denominator of Eq. 1.3 with the average number $\langle n \rangle$ of steps required for an integration over a physical time scale t; it is $\langle n \rangle \propto t/\langle dt \rangle$, where $\langle dt \rangle$ is the average individual time step. In a simple theoretical model our code should asymptotically scale with N^2, so we would expect $N\langle s \rangle \langle n \rangle \propto N^2$. However, our measurements deliver a slightly less favourable number $\langle s \rangle \langle n \rangle \propto N^{1+x}$, with $x = 0.31$, a value in accord with results of Makino and Hut (1988). Hence we get for the integration over one time unit:

$$P \approx \frac{\gamma\, N^{2+x}}{\alpha\, N^{2+x}/N_{\rm gpu} + \beta\, (\tau_{\rm lat} + N^{1+x}) log(N_{\rm gpu})} \tag{1.4}$$

The parameter $x = 0.31$ is a particular result for our case of the 6$^{\rm th}$ order HITS and the particular initial model used for the N-body system, Plummer's model as in Makino and Hut (1988). x is empirically determined from our timing measurements as shown in Fig. 1.5. The parameters α, β, γ and $\tau_{\rm lat}$ can as well be determined for each particular hardware used. The timing formula can then be used to approximate our code calculation "speed" for any other number of particles, GPU's, or different hardware parameters. For example, on the Mole-8.5 system we see, that for N = 10M particles if we are using 2000 GPU cards on the system we expect to get $\approx$ 390 Tflops (compare Fig. 1.5). If we use our scaling formula for the much higher node-to-node bandwidth of the Tianhe-1 system at Tianjin Supercomputing Center (this is the number one supercomputer according to the Top500 list of November 2010, with 7000 NVIDIA Fermi Tesla GPU's and 160 Gbit/s node-to-node bandwidth) we can possibly reach sustained performance of order Petaflops. This is subject of future research.

To our knowledge the direct N-body simulation with six million bodies in the framework of a so-called Aarseth style code (Hermite scheme 6th order, hierarchical block time step, integrating an astrophysically relevant Plummer model with core-halo structure in density for a certain physical time) is the largest such simulation which exists so far. However, the presently used parallel MPI-CUDA GPU code φGPU is on the algorithmic level of NBODY1 (Aarseth, 1999b) - though it is already strongly used in production, useful features such as regularisation of few-body encounters and an Ahmad-Cohen neighbour scheme (Ahmad and Cohen, 1973) are not yet implemented. Only with those the code would be equivalent to NBODY6, which is the most efficient code for single workstations (Aarseth, 1999b, 2003), eventually with acceleration on a single node by one or two GPU's (work by Aarseth & Nitadori, see NBODY6[6]). NBODY6++ (Spurzem, 1999) is a massively parallel code corresponding to NBODY6 for general purpose parallel computers. An NBODY6++ variant using many GPU's in a cluster is work in progress. Such a code

[6] `http://www.ast.cam.ac.uk/~sverre/web/pages/nbody.htm`

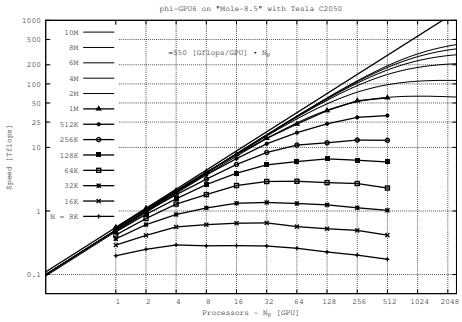

Fig. 1.5 Strong scaling for different problem sizes on Mole-8.5 cluster; each line corresponds to a different problem size (particle number), which is given in the key. The sequence of lines in the plot corresponds to the sequence of lines in the key (from top to bottom). Thicker lines with dots or symbols are obtained from our timing measurements. Thinner lines show the extrapolation for larger $N_{\rm GPU}$ and for larger N according to our timing model. As one can see we reach 550 Gigaflops per GPU card, in total on 512 GPU's about 280 Teraflops sustained code performance for our code. An extrapolation to 2000 GPU's shows we can reach 390 Teraflops on Mole-8.5 for ten million particles.

could potentially reach the same physical integration time (with same accuracy) using only one order of magnitude less floating point operations. The NBODY6 codes are algorithmically more efficient than φGPU or NBODY1, because they use an Ahmad-Cohen neighbour scheme (Ahmad and Cohen, 1973), which reduces the total number of full force calculations needed again (in addition to the individual hierarchic time step scheme), i.e. the proportionality factor in front of the asymptotic complexity N^2 is further reduced.

We have shown that our GPU clusters for the very favourable direct N-body application reach about one third of the theoretical peak speed sustained for a real application code with individual block time steps. In the future we will use larger Fermi based GPU clusters such as the Mole-8.5 cluster at the Institute of Process Engineering of Chinese Academy of Sciences in Beijing (IPE/CAS) and more efficient variants of our direct N-body algorithms; details of benchmarks and science results, and the requirements to reach Exascale performance, will be published elsewhere.

1.6 Adaptive mesh refinement hydro simulations

The team at National Taiwan University has developed an adaptive-mesh-refinement code named *GAMER* to solve astrophysical hydrodynamic problems (Schive et al., 2010). The AMR implementation is based on constructing a hierarchy of grid patches with an oct-tree data structure. The code adopts a hybrid CPU/GPU model, in which both hydrodynamic and gravity solvers are implemented into GPU and the AMR data structure is manipulated by CPU. For strong scaling, considerable speed-

up is demonstrated for up to 128 GPU's, with excellent performance shown in the figures 1.6 and 1.7.

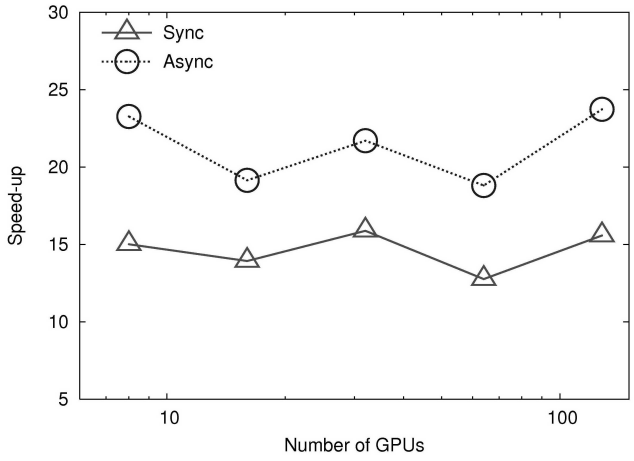

Fig. 1.6 Performance speed-up of the GAMER code as a function of the number of GPU's, measured on the Beijing Laohu cluster. The test problem is a purely baryonic cosmological simulation of ΛCDM, in which the root-level resolution is 256^3 and seven refinement levels are used, giving 32768^3 effective resolution. For each data point, we compare the performance by using the same number of GPU's and CPU cores. The blue circles and red triangles show the timing results with and without the concurrent execution between CPUs and GPU's, respectively. The maximum speed-up achieved in the 128-GPU run is about 24.

More recently, the GAMER code is further optimised for supporting several directionally unsplit hydrodynamic schemes and the OpenMP parallelisation (Schive et al., 2011, submitted). By integrating hybrid MPI/OpenMP parallelisation with GPU computing, the code can fully exploit the computing power in a heterogeneous CPU/GPU system. The figure 1.8 shows the performance benchmark on the Dirac cluster at NERSC/LBNL. The maximum speed-ups achieved in the 32-GPU run are 71.4 and 18.3 as compared with the CPU-only single-core and quad-core performances, respectively. Note that the 32-GPU speed-up drops about 12% mainly due to the MPI communication and the relatively lower spatial resolution (and hence higher surface/volume ratio) compared to that of the benchmark performed on the Beijing Laohu cluster. This issue can be alleviated by increasing the spatial resolution and also by overlapping communication with computation.

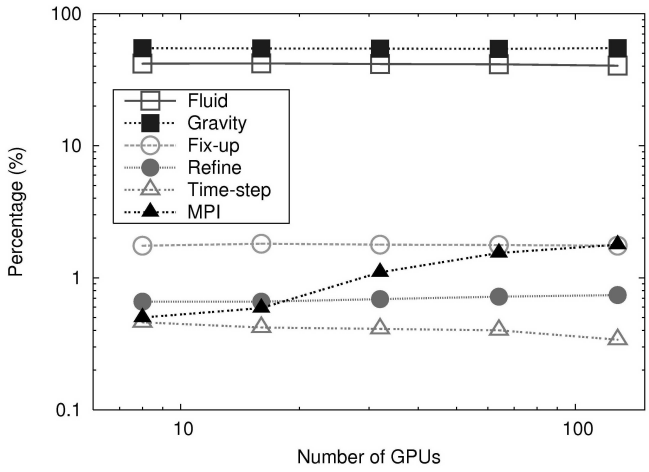

Fig. 1.7 Fractions of time consumed in different parts of the GAMER code, including the hydrodynamic solver (*open squares*), gravity solver (*filled squares*), coarse-grid data correction (*open circles*), grid refinement (*filled circles*), computing time-step (*open triangles*), and MPI communication (*filled triangles*). It shows that the MPI communication time even for large number of GPU's uses only a small percentage of time (order 1%), and hence the code will be able to scale well to even much larger GPU numbers.

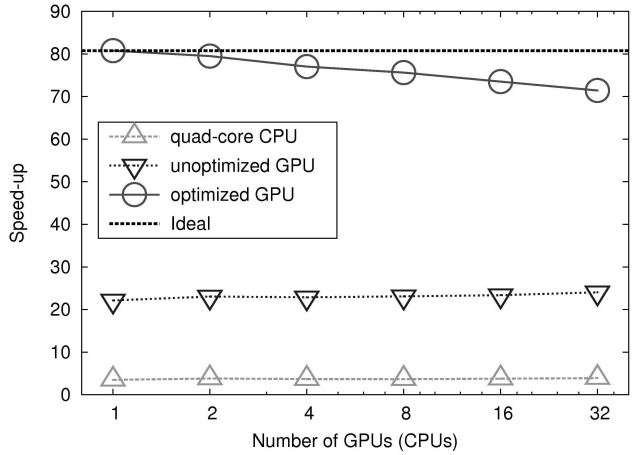

Fig. 1.8 Performance speed-up of the latest GAMER code of 2011, measured on the Dirac cluster at NERSC/LBNL. The root-level resolution is 256^3 and only four refinement levels are used. The GPU performance is compared to that of CPU runs without OpenMP and GPU acceleration. Several optimisations are implemented in the fully optimised code, including the asynchronous memory copy, the concurrent execution between CPU and GPU, and the OpenMP parallelisation. The quad-core CPU performance is also shown for comparison.

1.7 Physical multi-scale discrete simulation at IPE

Discrete simulation is, in a sense, more fundamental and straightforward as compared to other numerical methods based on continuum models, since the world is naturally composed of particles at very small and large scales, such as fundamental particles, atoms and molecules on one hand and stars and galaxies on the other hand. However, continuum methods are traditionally considered more efficient as each element in these methods presents a statistically enough number of particles. This faith has changed in recent years with the dramatic development of parallel computing. It turns out that, although the peak performance of (parallel) supercomputer is increasing at a speed higher than the Moore's law, the sustainable performance of most numerical softwares is far behind it, sometimes only several percent of it, and the percentage decreases with system scale inevitably. The complex data dependence and hence communication overheads inherent for most continuum based numerical methods presents a major cause of this inefficiency and poor scalability. In comparison, discrete simulation methods, such as molecular dynamics (MD) simulations, dissipative particle dynamics (DPD), lattice Boltzmann method (LBM), discrete particle methods (DEM) and smoothed particle hydrodynamics (SPH), etc., heavily rely on local interactions and their algorithms are inherently parallel. To final analysis, this is rooted in the physical parallelism of the physical model behind these methods. It is worthy of mention that, coarse grained particles, such DPD and PPM (pseudo-particle modelling; Ge and Li, 2003a) are now capable of simulating apparently continuous systems at a computational cost fairly comparable to continuum methods and macro-scale particle methods, such SPH and MaPPM (macro-scale pseudo-particle modeling; Ge and Li, 2001, 2003b) can also be understood as special kind of numerical discretising of continuum models.

In recent years, with the flourish of many-core computing technology, such as the use of GPU's (graphic processing unit) for scientific and engineering computing, this virtue of discrete methods is best demonstrated and further explored. A general model for many-core computing of discrete methods is "divide and conquer". A naive implementation is to decompose the computed domain into many sub-domains, which are then assigned to different processors for parallel computing of particle-particle interactions and movements. The assignment changes as the physical location of transfer from one sub-domain to another. Communications, therefore, only occur at neighbouring sub-domains. Most practical implementations, however, use more advance techniques, such as dynamic load balance, and monotonic Lagrangian grid (MLG; Lambrakos and Boris, 1987), to minimise the waiting and communication among different processors. Within each processor, each pair of particle-particle interactions and each particle-state updating are also parallel in principle, which can be carried out by each core of the processors. Currently, most many-core processes, like GPU's, are still working as an external device to the central processing unit (CPU), so data copy between the main memory and device memory is still necessary, and the communication between many-core processors across different computing nodes is routed by CPUs. Combined CPU-GPU com-

puting mode is under development, which may further reduce this communication overhead.

Some of the discrete simulation work carried out at Institute of Process Engineering (IPE), Chinese Academy of Sciences (CAS) using GPGPU's has been introduced in a Chinese monograph (Chen et al., 2009) and some recent publications, they have covered molecular dynamics simulation of multi-phase micro- and nano-flow (Chen et al., 2008), polymer crystallisation (Xu et al., 2009) and silicon crystal, CFD (computational fluid dynamics) simulation of cavity flow (Li et al., 2009) and gas-solid suspension, etc. All the simulations introduced above have been carried out on the multi-scale HPC systems established at IPE. The first system, Mole-9.7, put into use on Feb. 18, 2008, consists of 120 HP xw8600 workstations, each installed 2 Nvidia Tesla C870 GPGPU cards and 2 Intel Xeon 5430 CPUs, reached a peak performance of 120 Teraflops in single precision. The system is connected by an all-to-all switch together with a 2D torus topology of Gigabit Ethernet which speeds up adjacent communication dominated in discrete simulations. Its successor, Mole-8.7 is announced on Apr. 20, 2009 as the first supercomputer of China with 1.0 Petaflops peak performance in single precision (Chen et al., 2009). Both Nvidia and AMD GPU are integrated in this system. The designing philosophy is the consistency among hardware, software and the problems to be solved, based on the multi-scale method and discrete simulation approaches developed at IPE. The system has nearly 400 nodes connected by Gigabit Ethernet and DDR Infiniband network.

Then in 2010, IPE built the new system-Mole-8.5, which is the first GPU cluster using Fermi in the world. With the powerful computational resource of Mole-8.5 and the multi-scale software developed by IPE, several large scale applications have been successfully run on Mole-8.5:

- A MD simulation of dynamic structure of a whole H1N1 influenza virion in solution is simulated at the atomic level for the first time. The simulation system includes totally 300 million atoms in a periodic cube with edge length of 148.5nm. Using 288 nodes with 1,728 Fermi Tesla C2050, the simulation proceeds at 770ps/day with an integration time step of 1fs (Xu et al., 2010b).
- A quasi realtime DEM simulation of an industrial rotating drum, the size of which is 13.5 m long by 1.5 m in diameter, is performed. The simulation system contains about 9.6 million particles. Nearly 1/11 real speed is achieved using 270 GPU's together with online visualization (Xu et al., 2010a).
- Large scale direct numerical simulations of gas-solid fluidization have been carried out, with systems of about 1 million solid particles and 1 giga fluid particles in 2D using 576 GPU's, and of about 100 thousand solid particle and 0.4 giga fluid particles in 3D using 224 GPU's. The largest system we have run utilizing 1728 GPU's with an estimated performance of 33 Teraflops in double precision (Xiong and et al., 2010).
- A large-scale parallel molecular dynamics simulation of single-crystalline silicon nano wire containing about 1.5 billion silicon atoms with many-body potential is conducted using 1500 GPU cards with a performance of about 227 Teraflops in single precision (Hou and Ge, 2011).

1.8 Discussion and conclusions

We have presented exemplary implementations of parallel codes using many graphical processing units as accelerators, so combining message passing parallelisation with many-core parallelisation and discussed their benchmarks using up to 512 Fermi Tesla GPU's in parallel, mostly on the Mole-8.5 hardware of the Institute of Process Engineering of Chinese Academy of Sciences (IPE/CAS) in Beijing, but also on the Laohu Tesla C1070 cluster of the National Astronomical Observatories of CAS in Beijing and smaller clusters in Germany and United States. For direct high-accuracy gravitating N-body simulations we discussed how self-gravity, because it cannot be shielded, generates inevitably strong multi-scale structures in space and time, spanning many orders of magnitude. This requires special codes, which nevertheless scale with a high efficiency on GPU clusters. Also we present an adaptive mesh hydrodynamical code including a gravity solver using Fast Fourier Transformation and relaxation methods and physical algorithms used for multi-scale flows with particles. So our codes are examples that it is possible to reach the sub-Petaflops scale in sustained speed for realistic application software with large GPU clusters. Whether our programming models can be scaled up for future hardware and the Exaflops scale, however, remains yet to be studied.

1.9 Acknowledgments

Chinese Academy of Sciences (CAS) has supported this work by a Visiting Professorship for Senior International Scientists, Grant Number 2009S1-5 (RS), and National Astronomical Observatores of China (NAOC/CAS) through the Silk Road Project (RS, PB, JF partly). Institute of Process Engineering (IPE/CAS) and the High Performance Computing Center at NAOC/CAS acknowledge financial support by Ministry of Finance under the grant ZDYZ2008-2 for the supercomputers Mole-8.5 and Laohu, used for simulations of this paper. RS and PB want to thank Xue Suijian for valuable advice and support. We thank the computer system support team at NAOC (Gao Wei, Cui Chenzhou) for their support to run the Laohu cluster.

We gratefully acknowledge computing time on the Dirac cluster of NERSC/LBNL in Berkeley and thank Hemant Shukla, John Shalf, Horst Simon for providing the access to this cluster and for cooperation in the International Center of Computational Science[7], as well as the helpful cooperation of Guillermo Marcus, Andreas Kugel, Reinhard Männer, Robi Banerjee and Ralf Klessen in the GRACE and Frontier Projects at University of Heidelberg (at ZITI and ITA/ZAH).

Simulations were also performed on the GRACE supercomputer (grants I/80 041-043 and I/81 396 of the Volkswagen Foundation and 823.219-439/30 and /36 of the Ministry of Science, Research and the Arts of Baden-Württemberg) and the Kolob cluster funded by the Frontier Project at University of Heidelberg. PB ac-

[7] `http://iccs.lbl.gov`

knowledges the special support by the NAS Ukraine under the Main Astronomical Observatory GRAPE/GRID computing cluster project[8]. P.B.'s studies are also partially supported by the program Cosmomicrophysics of NAS Ukraine. The Kolob cluster and IB have been funded by the excellence funds of the University of Heidelberg in the Frontier scheme. Though our parallel GPU code has not yet reached the perfection of standard NBODY6, we want to thank Sverre Aarseth for providing his codes freely and teaching many generations of students how to use it and adapt it to new problems. This has helped and guided the authors in many respects.

References

S. J. Aarseth. Star Cluster Simulations: the State of the Art. *Celestial Mechanics and Dynamical Astronomy*, 73:127–137, January 1999a. doi: 10.1023/A:1008390828807.

S. J. Aarseth. From NBODY1 to NBODY6: The Growth of an Industry. *Publications of the Astronomical Society of the Pacific*, 111:1333–1346, November 1999b. doi: 10.1086/316455.

S. J. Aarseth. *Gravitational N-Body Simulations*. Cambridge University Press, Cambridge, UK, November 2003.

A. Ahmad and L. Cohen. A numerical integration scheme for the N-body gravitational problem. *Journal of Computational Physics*, 12:389–402, 1973. doi: 10.1016/0021-9991(73)90160-5.

K. Akeley, H. Nguyen, and Nvidia. *GPU Gems 3*. Addison-Wesley Professional, 2007.

J. Barnes and P. Hut. A hierarchical O(N log N) force-calculation algorithm. *Nature*, 324:446–449, December 1986. doi: 10.1038/324446a0.

B. R. Barsdell, D. G. Barnes, and C. J. Fluke. Advanced Architectures for Astrophysical Supercomputing. *ArXiv e-prints*, January 2010.

R. G. Belleman, J. Bédorf, and S. F. Portegies Zwart. High performance direct gravitational N-body simulations on graphics processing units II: An implementation in CUDA. *New Astronomy*, 13:103–112, February 2008. doi: 10.1016/j.newast.2007.07.004.

P. Berczik, D. Merritt, and R. Spurzem. Long-Term Evolution of Massive Black Hole Binaries. II. Binary Evolution in Low-Density Galaxies. *The Astrophysical Journal*, 633:680–687, November 2005. doi: 10.1086/491598.

P. Berczik, D. Merritt, R. Spurzem, and H.-P. Bischof. Efficient Merger of Binary Supermassive Black Holes in Nonaxisymmetric Galaxies. *The Astrophysical Journal Letters*, 642:L21–L24, May 2006. doi: 10.1086/504426.

P. Berczik, N. Nakasato, I. Berentzen, R. Spurzem, G. Marcus, G. Lienhart, A. Kugel, R. Maenner, A. Burkert, M. Wetzstein, T. Naab, H. Vasquez, and S. B. Vinogradov. Special, hardware accelerated, parallel SPH code for galaxy evo-

[8] `http://www.mao.kiev.ua/golowood/eng/`

lution. In *"SPHERIC - Smoothed Particle Hydrodynamics European Research Interest Community"*., pages 5–+, 2007.

P. Berczik, K. Nitadori, T. Hamada, and R. Spurzem. The Parallel GPU *N*-Body Code φGPU . *in preparation*, 2011.

I. Berentzen, M. Preto, P. Berczik, D. Merritt, and R. Spurzem. Binary Black Hole Merger in Galactic Nuclei: Post-Newtonian Simulations. *The Astrophysical Journal*, 695:455–468, April 2009. doi: 10.1088/0004-637X/695/1/455.

F. Chen, W. Ge, and J. Li. Molecular dynamics simulation of complex multiphase flows - Test on a GPU based cluster with customized networking. *Sci. China, Ser. B*, 38:1120–1128, 2008.

F. Chen, W. Ge, L. Guo, X. He, B. Li, J. Li, X. Li, X. Wang, and X. Yuan. Multi-scale HPC system for multi-scale discrete simulation. Development and application of a supercomputer with 1Petaflop/s peak performance in single precision. *Particuology*, 7:332–335, 2009.

H. M. P. Couchman, P. A. Thomas, and F. R. Pearce. Hydra: an Adaptive-Mesh Implementation of P 3M-SPH. *The Astrophysical Journal*, 452:797–+, October 1995. doi: 10.1086/176348.

Y. Cui, Y. Chen, and H. Mei. Improving performance of matrix multiplication and FFT on GPU. *15th International Conference on Parallel and Distributed Systems*, 729:13–+, 12 2009. doi: 10.1109/ICPADS.2009.8. URL `http://sei.pku.edu.cn/~cyf/icpads09.pdf`.

W. Dehnen. A Very Fast and Momentum-conserving Tree Code. *The Astrophysical Journal Letters*, 536:L39–L42, June 2000. doi: 10.1086/312724.

W. Dehnen. A Hierarchical O(N) Force Calculation Algorithm. *Journal of Computational Physics*, 179:27–42, June 2002. doi: 10.1006/jcph.2002.7026.

E. N. Dorband, M. Hemsendorf, and D. Merritt. Systolic and hyper-systolic algorithms for the gravitational N-body problem, with an application to Brownian motion. *Journal of Computational Physics*, 185:484–511, March 2003. doi: 10.1016/S0021-9991(02)00067-0.

G. Egri, Z. Fodor, C. Hoelbling, S. Katz, D. Nogradi, and K. Szabo. Lattice QCD as a video game. *Computer Physics Communications*, 177:631–639, October 2007. doi: 10.1016/j.cpc.2007.06.005.

M. Fellhauer, P. Kroupa, H. Baumgardt, R. Bien, C. M. Boily, R. Spurzem, and N. Wassmer. SUPERBOX - an efficient code for collisionless galactic dynamics. *New Astronomy*, 5:305–326, September 2000. doi: 10.1016/S1384-1076(00)00032-4.

T. Fukushige, J. Makino, and A. Kawai. GRAPE-6A: A Single-Card GRAPE-6 for Parallel PC-GRAPE Cluster Systems. *Publications of the Astronomical Society of Japan*, 57:1009–1021, December 2005.

W. Ge and J. Li. Macao-scale pseudo-particle modeling for particle-fluid systems. *Chin. Sci. Bull.*, 46:1503–1507, 2001.

W. Ge and J. Li. Macro-scale Phenomena Reproduced in Microscopic Systems-Pseudo-Particle Modeling of Fludization. *Chem. Eng. Sci.*, 58:1565–1585, 2003a.

W. Ge and J. Li. Simulation of particle-fluid systems with macro-scale pseudo-particle modeling. *Powder Technol.*, 137:99–108, 2003b.

L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73:325–348, December 1987. doi: 10.1016/0021-9991(87)90140-9.

A. Gualandris and D. Merritt. Ejection of Supermassive Black Holes from Galaxy Cores. *The Astrophysical Journal*, 678:780–797, May 2008. doi: 10.1086/586877.

T. Hamada and T. Iitaka. The Chamomile Scheme: An Optimized Algorithm for N-body simulations on Programmable Graphics Processing Units. *ArXiv Astrophysics e-prints*, March 2007.

S. Harfst, A. Gualandris, D. Merritt, R. Spurzem, S. Portegies Zwart, and P. Berczik. Performance analysis of direct N-body algorithms on special-purpose supercomputers. *New Astronomy*, 12:357–377, July 2007. doi: 10.1016/j.newast.2006.11.003.

R. W. Hockney and J. W. Eastwood. *Computer simulation using particles*. Bristol: Hilger, 1988.

C. Hou and W. Ge. GPU-accelerated molecular dynamics simulation of solid covalent crystals. *Molecular Simulation*, submitted, 2011.

W.-M-W. Hwu. *GPU Computing Gems*. Morgan Kaufman Publ. Inc., February 2011.

T. Ishiyama, T. Fukushige, and J. Makino. GreeM: Massively Parallel TreePM Code for Large Cosmological N -body Simulations. *Publications of the Astronomical Society of Japan*, 61:1319–, December 2009.

A. Just, F. M. Khan, P. Berczik, A. Ernst, and R. Spurzem. Dynamical friction of massive objects in galactic centres. *The Monthly Notices of the Royal Astronomical Society*, 411:653–674, February 2011. doi: 10.1111/j.1365-2966.2010.17711.x.

S. Komossa, V. Burwitz, G. Hasinger, P. Predehl, J. S. Kaastra, and Y. Ikebe. Discovery of a Binary Active Galactic Nucleus in the Ultraluminous Infrared Galaxy NGC 6240 Using Chandra. *The Astrophysical Journal Letters*, 582:L15–L19, January 2003. doi: 10.1086/346145.

S. G. Lambrakos and J. P. Boris. Geometric Properties of the Monotonic Lagrangian Grid Algorithm for Near Neighbor Calculations. *Journal of Computational Physics*, 73:183–+, November 1987. doi: 10.1016/0021-9991(87)90113-6.

K. J. Lee, N. Wex, M. Kramer, B. W. Stappers, C. G. Bassa, G. H. Janssen, R. Karuppusamy, and R. Smits. Gravitational wave astronomy of single sources with a pulsar timing array. *ArXiv e-prints*, March 2011.

B. Li, X. Li, Y. Zhang, F. Chen, J. Xu, X. Wang, X. He, J. Wang, W. Ge, and J. Li. Lattice Boltzmann simulation on Nvidia and AMD GPUs. *Chin. Sci. Bull.*, 54: 3178–3185, 2009.

D. Lynden-Bell and R. Wood. The gravo-thermal catastrophe in isothermal spheres and the onset of red-giant structure for stellar systems. *The Monthly Notices of the Royal Astronomical Society*, 138:495–+, 1968.

J. Makino. A Modified Aarseth Code for GRAPE and Vector Processors. *Proceedings of Astronomical Society of Japan*, 43:859–876, December 1991.

J. Makino. A Fast Parallel Treecode with GRAPE. *Publications of the Astronomical Society of Japan*, 56:521–531, June 2004.

J. Makino and S. J. Aarseth. On a Hermite integrator with Ahmad-Cohen scheme for gravitational many-body problems. *Publications of the Astronomical Society of Japan*, 44:141–151, April 1992.

J. Makino and P. Hut. Performance analysis of direct N-body calculations. *The Astrophysical Journal Supplement Series*, 68:833–856, December 1988. doi: 10.1086/191306.

J. Makino, T. Fukushige, M. Koga, and K. Namura. GRAPE-6: Massively-Parallel Special-Purpose Computer for Astrophysical Particle Simulations. *Publications of the Astronomical Society of Japan*, 55:1163–1187, December 2003.

K. Nitadori and J. Makino. Sixth- and eighth-order Hermite integrator for N-body simulations. *New Astronomy*, 13:498–507, October 2008. doi: 10.1016/j.newast.2008.01.010.

S. Pasetto, E. K. Grebel, P. Berczik, C. Chiosi, and R. Spurzem. Orbital evolution of the Carina dwarf galaxy and self-consistent determination of star formation history. *Astronomy & Astrophysics*, 525:A99+, January 2011. doi: 10.1051/0004-6361/200913415.

F. R. Pearce and H. M. P. Couchman. Hydra: a parallel adaptive grid code. *New Astronomy*, 2:411–427, November 1997. doi: 10.1016/S1384-1076(97)00025-0.

S. F. Portegies Zwart, R. G. Belleman, and P. M. Geldof. High-performance direct gravitational N-body simulations on graphics processing units. *New Astronomy*, 12:641–650, November 2007. doi: 10.1016/j.newast.2007.05.004.

M. Preto, I. Berentzen, P. Berczik, and R. Spurzem. Fast coalescence of massive black hole binaries from mergers of galactic nuclei: implications for low-frequency gravitational-wave astrophysics. *ArXiv e-prints*, February 2011.

H.-Y. Schive, Y.-C. Tsai, and T. Chiueh. GAMER: A Graphic Processing Unit Accelerated Adaptive-Mesh-Refinement Code for Astrophysics. *Astrophysical Journal Supplement Series*, 186:457–484, February 2010. doi: 10.1088/0067-0049/186/2/457.

H.-Y. Schive, U.-H. Zhang, and T. Chiueh. Directionally Unsplit Hydrodynamic Schemes with Hybrid MPI/OpenMP/GPU Parallelization in AMR. *The International Journal of High Performance Computing Applications*, 2011, submitted.

V. Springel. The cosmological simulation code GADGET-2. *Monthly Notices of the Royal Astronomical Society*, 364:1105–1134, December 2005. doi: 10.1111/j.1365-2966.2005.09655.x.

R. Spurzem. Direct N-body Simulations. *Journal of Computational and Applied Mathematics*, 109:407–432, September 1999.

R. Spurzem, P. Berczik, G. Hensler, C. Theis, P. Amaro-Seoane, M. Freitag, and A. Just. Physical Processes in Star-Gas Systems. *Publications of the Astronomical Society of Australia*, 21:188–191, 2004. doi: 10.1071/AS04028.

R. Spurzem, P. Berczik, I. Berentzen, D. Merritt, N. Nakasato, H. M. Adorf, T. Brüsemeister, P. Schwekendiek, J. Steinacker, J. Wambsganß, G. M. Martinez, G. Lienhart, A. Kugel, R. Männer, A. Burkert, T. Naab, H. Vasquez, and M. Wetzstein. From Newton to Einstein N-body dynamics in galactic nuclei and SPH

using new special hardware and astrogrid-D. *Journal of Physics Conference Series*, 78(1):012071–+, July 2007. doi: 10.1088/1742-6596/78/1/012071.

R. Spurzem, I. Berentzen, P. Berczik, D. Merritt, P. Amaro-Seoane, S. Harfst, and A. Gualandris. Parallelization, Special Hardware and Post-Newtonian Dynamics in Direct N - Body Simulations. In S. J. Aarseth, C. A. Tout, & R. A. Mardling, editor, *The Cambridge N-Body Lectures*, volume 760 of *Lecture Notes in Physics, Berlin Springer Verlag*, pages 377–+, 2008. doi: 10.1007/978-1-4020-8431-7_15.

R. Spurzem, P. Berczik, G. Marcus, A. Kugel, G. Lienhart, I. Berentzen, R. Männer, R. Klessen, and R. Banerjee. Accelerating Astrophysical Particle Simulations with Programmable Hardware (FPGA and GPU). *Computer Science - Research and Development (CSRD)*, 23:231–239, 2009.

R. Spurzem, P. Berczik, K. Nitadori, G. Marcus, A. Kugel, R. Männer, I. Berentzen, R. Klessen, and R. Banerjee. Astrophysical Particle Simulations with Custom GPU Clusters. *10th IEEE International Conference on Computer and Information Technology*, page 1189, 2010. doi: 10.1109/CIT.2010.215. URL `http://doi.ieeecomputersociety.org/10.1109/CIT.2010.215`.

R. Spurzem, P. Berczik, T. Hamada, K. Nitadori, G. Marcus, A. Kugel, R. Männer, I. Berentzen, J. Fiestas, R. Banerjee, and R. Klessen. Astrophysical Particle Simulations with Large Custom GPU clusters on three continents. *International Supercomputing Conference ISC 2011, Computer Science - Research and Development (CSRD)*, accepted for publication, 2011.

A. C. Thompson, C. J. Fluke, D. G. Barnes, and B. R. Barsdell. Teraflop per second gravitational lensing ray-shooting using graphics processing units. *New Astronomy*, 15:16–23, January 2010. doi: 10.1016/j.newast.2009.05.010.

P. Wang and T. Abel. Magnetohydrodynamic Simulations of Disk Galaxy Formation: The Magnetization of the Cold and Warm Medium. *The Astrophysical Journal*, 696:96–109, May 2009. doi: 10.1088/0004-637X/696/1/96.

P. Wang, T. Abel, and R. Kaehler. Adaptive mesh fluid simulations on GPU. *New Astronomy*, 15:581–589, October 2010a. doi: 10.1016/j.newast.2009.10.002.

X. Wang, W. Ge, X. He, F. Chen, Li Guo, and J. Li. Development and application of a HPC system for multi-scale discrete simulation – Mole-8.5. *International Supercomputing Conference ISC10*, June 2010b.

H.-C. Wong, U.-H. Wong, X. Feng, and Z. Tang. Efficient magnetohydrodynamic simulations on graphics processing units with CUDA. *ArXiv e-prints*, August 2009.

Q. Xiong and et al. Large-Scale DNS of Gas-Solid Flow on Mole-8.5. *Chemical Engineering Science*, submitted, 2010.

G. Xu. A New Parallel N-Body Gravity Solver: TPM. *Astrophysical Journal Supplement Series*, 98:355–+, May 1995. doi: 10.1086/192166.

J. Xu, Y. Ren, X. Yu, X. Yang, and J. Li. Molecular Dynamics Simulation of Macromolecules Using Graphics Processing Unit. *Mol. Simul., submitted*, 2009.

J. Xu, H. Qi, X. Fang, W. Ge, and et al. Quasi-realtime simulation of rotating drum using discrete element method with parallel GPU computing. *Particulogy*, in press, 2010a.

J. Xu, X. Wang, X. He, Y. Ren, W. Ge, and J. Li. Application of the Mole-8.5 supercomputer – Probing the whole influenza virion at the atomic level. *Chinese Science Bulletin*, in press, 2010b.

J. Yang, Y. Wang, and Y. Chen. . *Journal of Computational Physics*, 221:799, 2007.

K. Yasuda. . *Journal of Computational Chemistry*, 29:334, 2007.

R. Yokota and L. Barba. Treecode and fast multipole method for N-body simulation with CUDA. *ArXiv e-prints*, October 2010.

R. Yokota, J. P. Bardhan, M. G. Knepley, L. A. Barba, and T. Hamada. Biomolecular electrostatics using a fast multipole BEM on up to 512 GPUs and a billion unknowns. *ArXiv e-prints*, July 2010.

K. Yoshikawa and T. Fukushige. PPPM and TreePM Methods on GRAPE Systems for Cosmological N-Body Simulations. *Publications of the Astronomical Society of Japan*, 57:849–860, December 2005.

Glossary

Genetic regulatory network A network of genes, RNAs, proteins, metabolites, and their mutual regulatory interactions.

Genome-scale The characterization of a of biological function and components on spanning the genome of the respective organism, i.e., incorporation/consideration of all known associated components encoded in the organisms genome.

Hill function In biochemistry, the binding of a ligand to a macromolecule is referred to as *cooperative binding*. The Hill function (or Hill equation) is used to describe this effect. It is defined as $y = K[x]^h/(1+K[x]^h)$, where y, the fractional saturation, is the fraction of the total number of binding sites occupied by the ligand, $[x]$ is the free (unbound) ligand concentration, K is a constant, and h is the Hill coefficient.

Integrative spatial systems biology An emergent field in systems biology that deals with the necessary integration of spatial properties into integrative biology.

Law of mass action In chemistry, the law of mass action states that the rate of a chemical reaction is directly proportional to the molecular concentrations of the reacting substances. The law of mass action covers the equilibrium as well kinetic aspects (reaction rates) of chemical reactions.

Model reduction The approximation of a model of a complex (non-linear) dynamical systems, with the aim of obtaining a simplified model that is easier to analyze but preserves essential properties of the original model.

Ordinary differential equation In chemical kinetic theory, the interactions between species are commonly expressed using ordinary differential equations (ODEs). An ODE is a relation that contains *functions* of only one independent variable (typically t), and one or more of its derivatives with respect to that variable. The order of an ODE is determined by the highest derivative it contains (for example, a first-order ODE involves only the first derivative of the function). The equation $5x(t)+\dot{x}(t) = 17$ is an example of a first-order ODE involving the independent variable t, a function of this variable, $x(t)$, and a derivative of this function, $\dot{x}(t)$. Since a

derivative specifies a rate of change, such an equation states how a function changes but does not specify the function itself. Given sufficient initial conditions, various methods are available to determine the unknown function. The difference between ordinary differential equations and partial differential equations is that partial differential equations involve partial derivatives of several variables.

Partial differential equation Is similar to an *ordinary differential equation* except that it involves functions with more than one independent variable.

Sensitivity analysis An important tool to study the dependence of systems on their parameters. Sensitivity analysis helps to identify those parameters that have significant impact on the system output and capture the essential characteristics of the system. Sensitivity analysis is particularly useful for complex biological networks with a large number of variables and parameters.