Comput Sci Res Dev (2011) 26: 145–151
DOI 10.1007/s00450-011-0173-1

SPECIAL ISSUE PAPER

Astrophysical particle simulations with large custom GPU clusters on three continents

R. Spurzem · P. Berczik · I. Berentzen · K. Nitadori · T. Hamada · G. Marcus · A. Kugel · R. Männer · J. Fiestas · R. Banerjee · R. Klessen

Published online: 29 April 2011

Abstract We present direct astrophysical N-body simulations with up to six million bodies using our parallel MPI-CUDA code on large GPU clusters in Beijing, Berkeley, and Heidelberg, with different kinds of GPU hardware. The clusters are linked in the cooperation of ICCS (International Center for Computational Science). We reach about one third of the peak performance for this code, in a real application scenario with hierarchically block time-steps and a core-halo density structure of the stellar system. The code and hardware is used to simulate dense star clusters with many binaries and galactic nuclei with supermassive black holes, in which correlations between distant particles cannot be neglected.

Keywords N-Body simulations · Computational astrophysics · GPU clusters

R. Spurzem (✉) · P. Berczik · J. Fiestas
National Astronomical Observatories of China,
Chinese Academy of Sciences, 20A Datun Rd.,
Chaoyang District, Beijing 100012, China
e-mail: spurzem@bao.ac.cn

P. Berczik
e-mail: berczik@bao.ac.cn

R. Spurzem · P. Berczik · I. Berentzen · J. Fiestas
Astronomisches Rechen-Institut (ZAH),
University of Heidelberg, Mönchhofstr. 12–14, 69120 Heidelberg,
Germany

K. Nitadori · T. Hamada
Nagasaki Advanced Computing Center, University of Nagasaki,
RIKEN Institute, Tokyo, Japan

G. Marcus · A. Kugel · R. Männer
Department of Computer Science V, Central Inst. of Computer
Engineering, University of Heidelberg, Heidelberg, Germany

I. Berentzen · R. Banerjee · R. Klessen
Institut für Theor. Astrophysik (ZAH), University of Heidelberg,
Heidelberg, Germany

P. Berczik
Main Astronomical Observatory, National Academy of Sciences
of Ukraine, MAO/NASU, 27 Akademika Zabolotnoho St.,
03680 Kyiv, Ukraine

1 Introduction

Competitive astronomical and astrophysical research requires access to competitive computing facilities. Theoretical numerical modeling of astrophysical objects, their composition, radiation, and dynamical evolution has become a third basic method of astrophysical research, besides observation and pure theory. Numerical modeling allows one to compare theory with observational data in unprecedented detail, and it also provides theoretical insight into physical processes at work in complex systems. Similarly, data processing of astrophysical observations comprises the use of complex software pipeline 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.

We are undergoing a new revolution on parallel processor technologies, especially with regard to the Graphic Processing Units. 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 [5, 19, 28, 50, 51]. Graphics processing units (GPU's)

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 8 cores on a standard 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 Gflop/s; some GPU clusters in production still use the older Tesla C1060 card, which only has intrinsic single precision support. It can be circumvented by emulation of double precision operations (e.g. [35] for an example). In this paper we also present first benchmarks of our applications on a Fermi based GPU cluster, kindly provided by LBNL/NERSC Berkeley (dirac cluster).

Scientists are using GPU since about five years already for scientific simulations, but only the invention of CUDA (Compute Unified Device Architecture [5]) as a high-level programming language for GPU's made their computing power available to any student or researcher with normal scientific programming skills. The number of scientific papers in the Harvard Astrophysics Database with GPU in title or abstract is about 250 between 2005 and 2009, and since 2010 alone again some 250 entries, which illustrates the strong increase in last years. CUDA is limited to GPU devices of NVIDIA, but the new open source language OpenCL will provide access to any type of many-core accelerator through an abstract programming language. Computational 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 ([21, 33], and earlier references therein). Clusters using GRAPE and GPU are used e.g. for direct N-body codes to model the dynamics of galaxies and galactic nuclei with supermassive black holes in galactic nuclei [10, 11, 13, 30], the dynamics of dense star clusters [8, 25, 37], in gravitational lensing ray shooting problems [46], in numerical hydrodynamics with adaptive mesh refinement [38, 47, 48] and magnetohydrodynamics [48], or Fast Fourier transformation [14, 16]. 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 [7]. 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 [12, 44] or for FFT [14, 16]. Recently new GPU implementations of Fast-Multipole Methods (FMM) have been presented and compared with Tree Codes [52, 53]. FMM codes have first been presented by [22]. 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.

2 Astrophysical application

Dynamical modeling 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. Typically 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 dynamics of dense stellar systems so far has been treated as a classical Newtonian problem until recently for only a few 10^5 particles, much less than necessary. Only with the advent of accelerated hardware (GRAPE and GPU) realistic particle numbers can be approached.

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

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

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

blocked time-steps for particles in a direct N-body simulation provides the best compromise between accuracy, efficiency and scalability [1–3, 26, 31, 40]. With GPU hardware up to a few million bodies could be reached for such models [10, 11, 24]. Note that while [22] already mention that their algorithm can be used to compute gravitational forces between particles to high accuracy, Makino and Hut [31] find that the self-adaptive hierarchical time-step structure inherited from Aarseth's codes improves the performance for spatially structured systems by $\mathcal{O}(N)$—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 neighbor scheme codes, [1, 4, 32]), or in space (tree codes, [6, 34, 39]). Another possibility is the use of fast-multipole algorithms [17, 18, 23, 52, 53] or particle-mesh schemes (PM, [20, 27]) 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, [15, 36]). 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 [29, 39, 49, 54]).

3 Hardware

In this article we report on new results obtained from our recently installed GPU clusters (see Fig. 1 using NVIDIA Tesla C1060 cards in Beijing, China (laohu cluster with 85 Dual Intel Xeon nodes and 170 GPU's) and Heidelberg, Germany (kolob cluster with 40 nodes, both featuring Dual Intel Xeon nodes and Tesla GPU's of the pre-Fermi single precision only generation), and on a recent cluster with Fermi C2050 cards in Berkeley at NERSC/LBNL (dirac cluster). In Germany, at Heidelberg University, our teams have operated many-core accelerated clusters using GRAPE hardware for many years [26, 41–44]. 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

Fig. 1 *Left*: Frontier kolob cluster at ZITI Mannheim, 40 nodes with 40 NVIDIA Tesla C870 GPU accelerators, 17 Tflop/s hardware peak speed; installed 2008. *Right*: NAOC GPU cluster in Beijing, 85 nodes with 170 NVIDIA Tesla C1060 GPU's, 170 Tflop/s hardware peak speed, installed 2010

Process Engineering (IPE) of CAS in Beijing (2 Pflop/s single precision peak from order 2000 Fermi C2050 devices). Note that in the current paper all results from Chinese GPU clusters quoted were obtained with the NAOC system laohu, not yet from currently ongoing runs with the Mole-8.5 cluster. The entire CAS GPU cluster network has a total capacity of nearly 5 Pflop/s single precision peak. At NAOC one if its members is operated, the laohu GPU cluster, which features 170 NVIDIA Tesla C1060 GPU's running on 85 nodes, and is mainly used for astrophysical applications. 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. 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.

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 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, [26]). It is parallelized using MPI, and on each node using many cores of the special hardware. The code was mainly developed and tested by three of us (Peter Berczik, Tsuyoshi Hamada, Keigo Nitadori, see also [25]) and is based on an earlier version for GRAPE clusters [26]. The code is written in C++ and based on Nitadori and Makino [35] earlier CPU serial code (yebisu). The present version of φGPU code we used and tested only with the recent GNU compilers (ver. 4.1 and 4.2). More details will be published in an upcoming publication [9].

The MPI parallelization was done in the same "j" particle parallelization mode as in the earlier φGRAPE code [26]. 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 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.

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

We use native GPU support and direct code access to the GPU with only CUDA. Recently we use CUDA 3.2. Multi GPU support is achieved through MPI parallelization; each MPI process uses only a single GPU, but we can start two MPI processes per node (to use effectively the dual CPU's and 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.

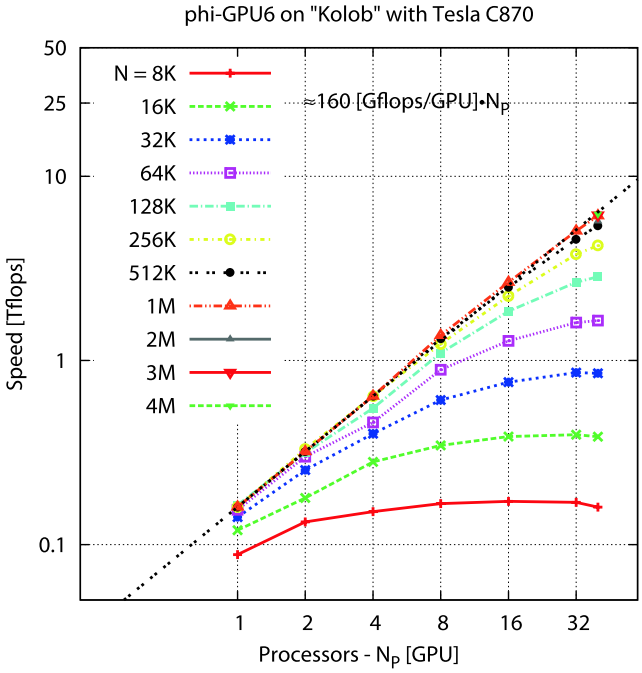
5 Results of benchmarks

The figures (Fig. 2) show results of our benchmarks, with a maximum of 164 GPU cards used (3 nodes i.e. 6 cards were down during the test period). The largest performance was reached for 6 million particles, with 51.2 Tflop/s 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 $\approx$360 Gflop/s (i.e. about one third of the theoretical hardware peak speed of 1 Tflop/s). Equivalently, for the smaller kolob cluster with 40 NVIDIA Tesla C870 GPU's in Germany, we obtain 6.5 Tflop/s with 4 million particles. This is $\approx$160 Gflop/s per card.

Finally an interesting result can be seen on the new dirac cluster at NERSC/LBNL Berkeley, where we see in Fig. 3 the performance on Fermi C2050 GPU accelerators, using the emulated double precision for some operations as we do on C1060. As expected the gain is of the order of $\approx$45% on the part of the GPU computation, which can only be seen for large particle numbers and smaller number of GPU's.

6 Conclusions

We have presented implementations of force computations between particles for astrophysical simulations, using our GPU clusters with MPI parallel codes in China and Germany. The overall parallelization efficiency of our codes is very good as one can see from the near ideal speedup in Fig. 2, and in accord with our earlier results on GRAPE clusters [26]. The larger 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.



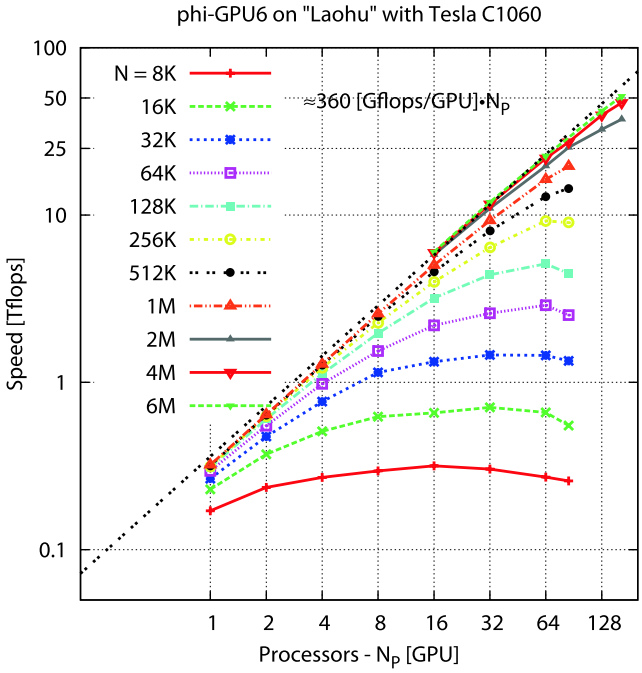


Fig. 2 Strong scaling for different problem sizes; *Left*: Same benchmark simulations for the Frontier kolob cluster at ZITI Mannheim, 6.5 Tflops/s reached for four million particles on 40 GPU's. Each line corresponds to a different problem size (particle number), which is given in the key. Note that the linear curve corresponds to ideal scaling. *Right*: NAOC GPU cluster in Beijing, speed in Teraflop/s reached as a function of number of processes, each process with one GPU, 51.2 Tflop/s sustained were reached with 164 GPU's (3 nodes with 6 GPU's were down at the time of testing)

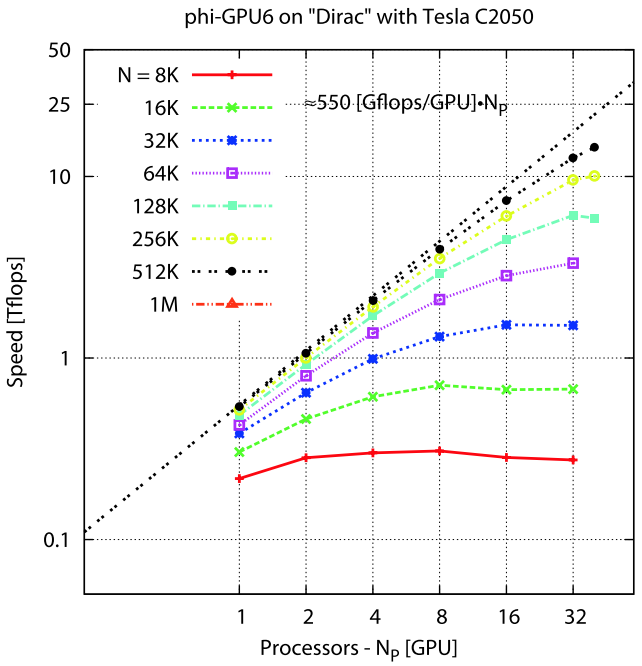


Fig. 3 The performance of φGPU on Fermi C2050 (dirac cluster in Berkeley) accelerated nodes; speed in Teraflop/s reached as a function of number of processes, each process with one GPU; limit to 40 GPU's as this is the maximum on the dirac cluster. Each line corresponds to a different problem size (particle number), which is given in the key

The wall clock time T needed for our particle based algorithm to advance the simulation by a certain physical time integration interval scales as

$$T = T_{\text{host}} + T_{\text{GPU}} + T_{\text{comm}} + T_{\text{MPI}} \quad (1)$$

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 using the blocking communication scheme, so there is no hiding of communication. This will be improved in further code versions, but for now it eases profiling. The dominant term is in the linearly rising part of the curves in Fig. 2 just T_{GPU}, while the turnover to flat is dominated by MPI communication. The interested reader may refer to the previous paper of Harfst et al. [26] for further details about our definitions and measurements (for the case of GRAPE instead of GPU, but analogous) and to our paper in preparation [9, 45] for new data with the new GPU hardware.

To our knowledge the direct N-body simulation with six million bodies in the framework of a so-called Aarseth style code (Hermite scheme, hierarchical 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 [3]—though it is already strongly used in production, useful features such as regularization of few-body encounters and an Ahmad-Cohen neighbor scheme [4] are not yet implemented. Only with those the code would be equivalent to NBODY6, which is the most efficient code for single workstations [1, 3], eventually with acceleration on a single node by one or two GPU's (work by Aarseth and Nitadori, see NBODY6[4]). NBODY6++ [40] 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 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 neighbor scheme [4], 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 favorable direct N-body application reach about one third of the theoretical peak speed sustained for a real application code with individual 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.

Acknowledgements We cordially thank Institute of Process Engineering (IPE) of Chinese Academy of Sciences (CAS), Ge Wei, Wang Xiaowei and their colleagues for continuous support and cooperation; 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.[5] Chinese Academy of Sciences has supported this work by a Visiting Professorship for Senior International Scientists, Grant Number 2009S1-5 (RS), and National Astronomical Observatory of China (NAOC) of CAS by the Silk Road Project (RS, PB, JF partly). RS and PB want to thank Xue Suijian for valuable advice and support. The special supercomputer laohu at the High Performance Computing Center at National Astronomical Observatories of China, funded by Ministry of Finance under the grant ZDYZ2008-2, has been used. We thank the computer system support team at NAOC (Gao Wei, Cui Chenzhou) for their support to run the laohu cluster. 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). P.B. acknowledges the special support by the NAS Ukraine under the Main Astronomical Observatory GRAPE–GRID computing cluster project.[6] P.B.'s studies are also partially supported by the program Cosmomicrophysics of NAS Ukraine. I.B. and the kolob cluster are 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

1. Aarseth SJ (2003) Gravitational N-body simulations. Cambridge University Press, Cambridge, p 430. ISBN: 0521432723
2. Aarseth SJ (1999a) From NBODY to NBODY6: the growth of an industry. Publ Astron Soc Pac 111:1333
3. Aarseth SJ (1999b) Star cluster simulations: the state of the art. Celest Mech Dyn Astron 73:127
4. Ahmad A, Cohen L (1973) A numerical integration scheme for the N-body gravitational problem. J Comput Phys 12:389–402
5. Akeley K, Nguyen H, Nvidia X (2007) GPU gems 3, programming techniques for high-performance graphics and general-purpose computation. Addison-Wesley, Reading
6. Barnes J, Hut P (1986) A hierarchical $O(N \log N)$ force-calculation algorithm. Nature 324:446
7. Barsdell BR, Barnes DG, Fluke CJ (2009) Advanced architectures for astrophysical supercomputing. In: The proceedings of ADASS XIX, Sapporo, Japan, Oct 4–8 2009. ASP Conf. Series. arXiv:1001.2048
8. Belleman RG, Bedorf J, Portegies Zwart SF (2008) High performance direct gravitational N-body simulations on graphics processing units II. An implementation in CUDA. New Astron 13:103
9. Berczik P, Hamada T, Nitadori K, Spurzem R (2011, in preparation) The parallel GPU N-body code φGPU
10. Berczik P, Merritt D, Spurzem R, Bischof H-P (2006) Efficient merger of binary supermassive black holes in nonaxisymmetric galaxies. Astrophys J 642:L21
11. Berczik P, Merritt D, Spurzem R (2005) Long-term evolution of massive black hole binaries. II. Binary evolution in low-density galaxies. Astrophys J 633:680
12. Berczik P, Nakasato N, Berentzen I, Spurzem R, Marcus G, Lienhart G, Kugel A, Männer R, Burkert A, Wetzstein M, Naab T, Vasquez H, Vinogradov SB (2007). Special, hardware accelerated, parallel SPH code for galaxy evolution. In: SPHERIC—smoothed particle hydrodynamics European research interest community. p. 5
13. Berentzen I, Preto M, Berczik P, Merritt D, Spurzem R (2009) Astrophys J 695:455
14. Chen Y, Cui X, Mei H (2010) Large-scale FFT on GPU clusters. In: Proceedings of the 24th ACM International Conference on Supercomputing, Tsukuba, Ibaraki, Japan, ICS '10. ACM, New York, pp 315–324
15. Couchman HMP, Thomas PA, Pearce FR (1995) Hydra: an adaptive-mesh implementation of P 3M-SPH. Astrophys J 452:797
16. Cui D, Liao N, Wu W, Tan B, Lin Y (2010) Fast ARFTIS reconstruction algorithms using CUDA. In: Zhang W, Chen Z, Douglas C, Tong W (eds) High Performance Computing and Applications. Lecture Notes in Computer Science, vol 5938. Springer, Heidelberg, pp 119–126

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

[5] http://iccs.lbl.gov.

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

17. Dehnen W (2002) A hierarchical $O(N)$ force calculation algorithm. J Comput Phys 179:27
18. Dehnen W (2000) A very fast momentum-conserving tree code. Astrophys J 536:L39
19. Egri G et al (2007) Comput Phys Commun 177:631
20. Fellhauer M, Kroupa P, Baumgardt H, Bien R, Boily CM, Spurzem R, Wassmer N (2000) SUPERBOX—an efficient code for collisionless galactic dynamics. New Astron 5:305
21. Fukushige T, Makino J, Kawai A (2005) GRAPE-6A: a single-card GRAPE-6 for parallel PC-GRAPE cluster systems. Publ Astron Soc Jpn 57:1009
22. Greengard L, Rokhlin V (1987) A fast algorithm for particle simulations. J Comput Phys 73:325
23. Greengard L, Rokhlin V (1997) A fast algorithm for particle simulations. J Comput Phys 135:280
24. Gualandris A, Merritt D (2008) Ejection of supermassive black holes from Galaxy cores. Astrophys J 678:780–797. doi:10.1086/586877
25. Hamada T, Iitaka T The chamomile scheme: an optimized algorithm for N-body simulations on programmable graphics processing units (2007). arXiv:astro-ph/0703100
26. Harfst S, Gualandris A, Merritt D, Spurzem R, Portegies Zwart S, Berczik P (2007) Performance analysis of direct N-body algorithms on special-purpose supercomputers. New Astron 12:357
27. Hockney RW, Eastwood JW (1988) Computer simulation using particles. Hilger, Bristol
28. Hwu W-MW (2011) GPU computing gems. Kaufmann, Los Altos
29. Ishiyama T, Fukushige T, Makino J (2009) GreeM: Massively parallel TreePM code for large cosmological N-body simulations. Publ Astron Soc Jpn 61:1319
30. Just A, Khan FM, Berczik P, Ernst A, Spurzem R (2010) Dynamical friction of massive objects in galactic centres. Mon Not R Astron Soc Lett 411:653
31. Makino J, Hut P (1988) Performance analysis of direct N-body calculations. Astrophys J Suppl Ser 68:833
32. Makino J, Aarseth SJ (1992) On a Hermite integrator with Ahmad-Cohen scheme for gravitational many-body problems. Publ Astron Soc Jpn 44:141
33. Makino J, Fukushige T, Koga M, Namura K (2003) Publ Astron Soc Jpn 55:1163
34. Makino J (2004) A fast parallel treecode with GRAPE. Publ Astron Soc Jpn 56:521
35. Nitadori K, Makino J (2008) Sixth- and eighth-order Hermite integrator for N-body simulations. New Astron 13:498
36. Pearce FR, Couchman HMP (1997) Hydra: a parallel adaptive grid code. New Astron 2:411
37. Portegies Zwart SF, Belleman RG, Geldof PM (2007) High-performance direct gravitational N-body simulations on graphics processing units. New Astron 12:641
38. Schive H-Y, Tsai Y-C, Chiueh T (2010) GAMER: a graphic processing unit accelerated adaptive-mesh-refinement code for astrophysics. Astrophys J Suppl Ser 186:457
39. Springel V (2005) The cosmological simulation code GADGET-2. Mon Not R Astron Soc Lett 364:1105
40. Spurzem R (1999) Direct N-body simulations. J Comput Appl Math 109:407
41. Spurzem R, Berczik P, Hensler G, Theis C, Amaro-Seoane P, Freitag M, Just A (2004) Physical processes in star-gas systems. Publ Astron Soc Aust 21:188
42. Spurzem R, Berczik P, Berentzen I, Merritt D, Nakasato N, Adorf HM, Brüsemeister T, Schwekendiek P, Steinacker J, Wambsganß J, Martinez GM, Lienhart G, Kugel A, Männer R, Burkert A, Naab T, Vasquez H, Wetzstein M (2007) From Newton to Einstein—N-body dynamics in galactic nuclei and SPH using new special hardware and astrogrid-D. J Phys Conf Ser 78:012071
43. Spurzem R, Berentzen I, Berczik P, Merritt D, Amaro-Seoane P, Harfst S, Gualandris A (2008) Parallelization special hardware and post-Newtonian dynamics in direct N-body simulations. Lecture notes in physics, vol 760. Springer, Berlin, p 377
44. Spurzem R, Berczik P, Marcus G, Kugel A, Lienhart G, Berentzen I, Männer R, Klessen R, Banerjee R (2009) Accelerating astrophysical particle simulations with programmable hardware (FPGA and GPU). Comput Sci Res Dev 23:231–239
45. Spurzem R, Berczik P, Berentzen I, Ge W, Wang X, Schive H-Y, Nitadori K, Hamada T (2011, in press) Physics and astrophysics—multiscale simulations: accelerated many-core GPU computing on three continents. In: Dubitzky W, Kurowski K, Schott B (eds) Special volume on "Large scale computing techniques for complex systems and simulations". Wiley, New York
46. Thompson AC, Fluke CJ, Barnes DG, Barsdell BR (2010) Teraflop per second gravitational lensing ray-shooting using graphics processing units. New Astron 15:16
47. Wang P, Abel T, Kaehler R (2010) Adaptive mesh fluid simulations on GPU. New Astron 15:581
48. Wong H-C, Wong U-H, Feng X, Tang Z (2009) Efficient magnetohydrodynamic simulations on graphics processing units with CUDA. arXiv:0908.4362
49. Xu G (1995) A new parallel N-body gravity solver: TPM. Astrophys J Suppl Ser 98:355
50. Yasuda Koji (2007) J Comput Chem 29:334
51. Yang J, Wang Y, Chen Y (2007) GPU accelerated simulation. J Comput Phys 221:799
52. Yokota R, Barba L (2010) Treecode and fast multipole method for N-body simulation with CUDA. arXiv:1010.1482
53. Yokota R, Bardhan JP, Knepley MG, Barba LA, Hamada T (2010) Biomolecular electrostatics simulation with a parallel FMM-based BEM, using up to 512 GPU's. arXiv:1007.4591
54. Yoshikawa K, Fukushige T (2005) PPPM and TreePM methods on GRAPE systems for cosmological N-body simulations. Publ Astron Soc Jpn 57:849