GPU-Accelerated Parton Cascade in Heavy-Ion Collisions

Qingjun Liu, Weiqin Zhao, Fang Liu, Ningming Nie, and Chunbao Zhou

***Abstract*—A widely used Monte Carlo event generator is A Multi-Phase Transport model (AMPT) for relativistic heavy-ion collisions. It depends on Zhang's Parton Cascade (ZPC) package to simulate initial stage parton cascade. Based on ZPC, we have developed a code for the simulation of the parton cascade to exploit the powerful parallel processing capability of GPU. The goal is to accelerate the simulation of the parton cascade in a system of partons that is formed in ultrarelativistic heavy-ion collisions. Named PCG (Parton Cascade on GPU), the code makes real time collision detection among N interacting partons formed in a heavy-ion collision parallelized. The parallelization was implemented by using CUDA C. With simulating Pb-Pb collisions at sqrt(sNN)=2.76 TeV as a use case, we first verified the correctness of PCG through comparison of the output of PCG with those of ZPC, then we estimated the computational efficiency of PCG to be 2x to 3x relative to ZPC, which is a serial code and only runs on CPU. Therefore PCG is viable for being integrating into AMPT for simulating heavy-ion collisions and can save large amount of computing resources for large scale AMPT-based event generation in ultrarelativistic heavy-ion collisions at sqrt(sNN)=2.76 TeV.**

***Index Terms*—GPU, CUDA C, simulation of parton cascade, ultrarelativistic heavy-ion collision.**

I. Introduction

Monte-Carlo event generators [1]-[6] are essential in high energy physics. Through analysis of large number of events generated by event generators, scientists not only get insight into physical mechanisms on the theory side [7]-[9], but also evaluate detector performance on the experiment side [10]-[12]. However in order to generate statistically significant number of events, huge amount of resources, in terms of computing time as well as electrical power, need to be consumed. Therefore it is of great interest to accelerate event generation by utilizing emerging high performance computing technologies, such as those involving graphical processing units (GPUs) and CUDA [13], [14], which have been enabling variety of applications [15]-[18] to gain phenomenal speedups.

Nowadays, in the field of ultrarelativistic heavy-ion collisions, one of the widely used Monte Carlo event generators is A Multi-Phase Transport (AMPT) model [1]. It simulates high energy heavy-ion collisions and is of great help for physicists to study, on the surface of the earth, a new form of matter called quark-gluon-plasma [19], which exists immediately after the birth of the cosmos according to the Big Bang theory [20]. Among the three main modules of AMPT [1], HIJING [2], ZPC [3] and ART [4], which run on CPU serially and are programmed in FORTRAN, ZPC mainly simulates parton cascade process that dominates the initial stage of ultrarelativistic heavy-ion collisions. Because ZPC utilizes most of the time for simulating the initial stage of a heavy-ion collision when QGP is formed through the parton cascade, this work focuses on GPU-based acceleration of the simulation of the cascade process.

The rest of this paper is mainly organized as follows. In Section II, we present brief introduction of the simulation of the parton cascade process in ultrarelativistic heavy-ion collisions. Then an algorithm for developing the parton cascade on GPU (PCG) to accelerate the cascade is introduced in Section III. In Section IV, we present results from PCG compared with those from ZPC. We give summary and conclusion in Section V.

II. Outline of the Simulation of Parton Cascade

Taking place in a system of interacting N-partons, parton cascade can simply be described as successive two-parton collisions, which follow the laws of the perturbative quantum chromodynamics [21] and satisfy certain geometrical criteria for the collision to happen. According to AMPT, the system of partons may be formed through a melting mechanism due to high temperature in the early stage of heavy-ion collisions [1]. As in ZPC, the first step in the simulation of parton cascade in a heavy-ion collision is to detect the earliest two-parton collision among a large number of possible two-parton collisions that satisfy the collision criteria, then simulates the collision thereby the momentum-energy and space-time information for the involved two partons are updated. According to ZPC, the rest of the simulation of the parton cascade is to repeat the above two steps until no two-parton collision is detected to happen earlier than a preset physical time threshold. At this stage the parton cascade terminates. It is based this account of the parton cascade that we designed in this work an algorithm for the GPU-accelerated simulation of the parton cascade in ultrarelativistic heavy-ion collisions. For more details about the simulation of the parton cascade within ZPC and its optimization, interested readers are referred to [3], [22].

III. Algorithm for PCG

The main input to PCG includes the space-time and

Manuscript received November 30, 2014; revised August 15, 2015. This work was supported by Beijing Natural Science Foundation under Project No. 1132017, National Natural Science Foundation of China, and the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry.

Qingjun Liu is with the Beijing Institute of Petro-chemical Technology, Beijing 102617 China (e-mail: liuqingjun@bipt.edu.cn).

Weiqin Zhao is with the Institute of High Energy Physics, Chinese Academy of Sciences, Beijing 100049 China (e-mail: zhaowq@ihep.ac.cn).

Fang Liu, Ningming Nie, and Chunbao Zhou are with the Supercomputer Center, Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190 China (e-mail: liuf@sccas.cn, nienm@sccas.cn, zhoucb@sccas.cn).

momentum-energy information for each of the partons in an N-parton system that is formed through string-melting when running AMPT to generate an event for an ultrarelativistic heavy-ion collision [1], [3]. This information will be updated during the parton cascade through two-parton collisions. In our previous publication [23], we reported a speedup of around 60 for the detection of the first two-parton collision in Pb-Pb collisions at sqrt(sNN)=2.76 TeV, where we used single precision in our calculations. In this work, all of our calculations are carried out by using double precision. The algorithm for simulating parton cascade in the N-parton system is formulated in the following three sub-sections and is implemented in a CUDA C code named *cudarun.cu*. The output of PCG are the space-time and momentum-energy information for all of the N partons at their last collisions and are stored in the data structures that store the main input information. These data structures are eight arrays of double precision. Due to space limitations, we intend to describe more details of all of the data structures and pseudo-code, in a separate paper. However in APPENDIX A, interested readers may have a brief sketch about the infrastructure of our algorithm for PCG.

A. Detection of the First Two-Parton Collision

First of all, let program loop over all of the 0.5(N-1) N parton pairs. In the loop, for each pair of partons, based on calculations using the space-time and momentum-energy information of the two partons, program can tell if the pair collides or not according to the physical [21] and geometrical collision criteria as defined in ZPC [3]. If a pair collides then the collision time for that pair is saved in global memory, together with their indices. What follows is the calculation of the smallest collision time among all of the collision time for all of the colliding pairs. For the calculation, we used a reduction procedure [23] where shared memory on GPU is utilized. The first collision is detected to happen between the pair of partons that collide at the smallest collision time. Using CUDA C, we implemented the aforementioned algorithm for GPU in a CUDA kernel named *datTime*() and it is launched with the block size set to be 128, according to the calculation using CUDA Occupancy Calculator [13]. The algorithm for the detection of the first two-parton collision is pretty much as previously reported in [23], however is modified in this work to use double precision. Furthermore, in this work during the detection of the first two-parton collision, we have the collision time and the indices of the colliding two partons for all of the colliding pairs saved in global memory. These saved data may be updated during the simulation of the whole parton cascade process, and are reused for the detection of the next earliest two-parton collision, as one may see in the following two sub-sections.

B. Simulation of Two-Parton Collision

Once the indices of the colliding two partons and their collision time have been determined for the first or the next earliest two-parton collision, the immediate task is to simulate the two-parton collision. Following the laws of the perturbative quantum chromodynamics [21], simulation of two-parton collision has been implemented in several FORTRAN subroutines in ZPC [3]. In this work, the simulation of two-parton collision is realized through invoking of these subroutines within the context of our CUDA C code named cudarun.cu. These subroutines were not rewritten in CUDA C because we found that they are not suitable to be parallelized. Executing these subroutines moves the involving two partons to the new location where collision happens at the collision time. In addition, through calling those subroutines, PCG updates their momenta and energies, and thus their speeds. Therefore PCG actually simulates two-parton collision on CPU, exactly as ZPC does.

C. Collision Detection for the Next Two-Parton Collision

After simulation of the just happened two-parton collision, the space-time as well as momentum-energy information for the two partons have been renewed. Because it is based on calculations using the space-time and momentum-energy information that PCG determines if a pair of partons will collide or not, and when to collide if the pair collides, it is necessary that before detecting the next earliest two-parton collision, the collision time for each of the just collided partons with other partons need to be updated if they will collide. The counting of the number of colliding pairs as well as the calculation of the collision time for each of those pairs that contain one of the just collided two partons are realized through invoking a CUDA kernel named ijUpdatTime(…,*i*,…), where *i* stands for the index of one of those two just collided two partons. Hence for the detection of the next earliest two-parton collision, this kernel is invoked two times, once for each of the just collided two partons. It can be inferred that the update by invoking CUDA kernel ijUpdatTime(…,*i*,…) concerns with the parallelized calculation of collision time for 2(N-2) pairs. It is worth-mentioning that all of the collision time for the rest parton pairs are left unchanged in the global memory at this stage and can be accessed for the detection of the next earliest two-parton collision. Given all the collision time and indices for all the colliding pairs of partons, the rest task for detecting the next earliest two-parton collision is partly done through a reduction procedure [23] that has been used for the detection of the first two-parton collision illustrated in sub-section A. For this job we implemented a CUDA kernel named updatTime(), which also updates the number of colliding pairs by ignoring those pairs that will not collide according to the result from ijUpdatTime().

IV. Results

A. Correctness Check and Speedups for One Event

PCG is developed to take advantages of the high performance that a multi-core GPU provides for general purpose computing. It has to be correct in order to be integrated into AMPT or similar transport code to accelerate Monte Carlo event generation for ultrarelativistic heavy-ion collisions. We validate the correctness of PCG by checking the collision history from PCG with that from ZPC during the simulation of the parton cascade process in an N-parton system. The information contained in the collision history includes space-time and momentum-energy information for both of the two partons that are involved in a two-parton collision, for all the two-parton collisions in the cascade process, not only right before the two-parton collision but also right after the two-parton collision. We have examined the collision history for various types of heavy-ion collisions of Pb – Pb at sqrt(sNN)=2.76 TeV, which is the top LHC(Large Hadron Collider) energy. The result of the examination, together with the computing time for the simulation of the parton cascade process is tabulated in Table I, where t_ZPC and t_PCG stand for the computing time used for simulating the parton cascade by ZPC and PCG, respectively. As a result of the examination by using UNIX command diff, we have found that ZPC and PCG record the same two-parton collision history for each of the heavy-ion collisions with the impact parameter b ranging from 0 to 10 fm. It means PCG passed the correctness check. Therefore PCG can correctly simulate parton cascade as ZPC can, however with the exploitation of the powerful parallel processing capability GPUs have to offer for general purpose computing.

TABLE I: Results on Correctness Check and Computing Time for Simulating the Parton Cascade in an N-Parton System Generated in One Monte Carlo Event for Pb –Pb Collisions at Sqrt(sNN)=2.76 TeV with Various Impact Parameters

b(fm)	check	N	t_ZPC(s)	t_PCG(s)	speedup
0	passed	43725	419.05	133.80	3.1x
1	passed	45145	494.31	159.38	3.1x
2	passed	40858	331.08	107.76	3.1x
3	passed	37230	259.24	85.37	3.0x
4	passed	30939	175.47	57.86	3.0x
5	passed	28908	144.41	48.79	3.0x
6	passed	22324	75.20	28.08	2.7x
7	passed	21138	74.26	29.22	2.5x
8	passed	17222	46.42	19.79	2.3x
9	passed	13781	28.16	14.08	2.0x
10	passed	10159	12.88	7.77	1.7x

TABLE II: Computing Time and Speedup for Simulating the Parton Cascade in a Collision of Pb-Pb at Sqrt(sNN)=2.76 TeV with Various Impact Parameters by Running ZPC Compared with That by Running PCG, without the Overhead of Writing onto Disk the Data about Parton Collision History

b(fm)	N	t_ZPC(s)	t_PCG(s)	speedup
0	43725	410.04	120.45	3.4x
1	45145	465.30	144.65	3.2x
2	40858	321.26	95.40	3.4x
3	37230	251.25	75.56	3.3x
4	30939	160.41	50.54	3.2x
5	28908	138.43	41.89	3.3x
6	22324	71.34	23.81	3.0x
7	21138	70.40	24.62	2.9x
8	17222	43.45	16.83	2.6x
9	13781	25.97	11.60	2.2x
10	10159	11.65	6.33	1.8x

From Table I, one may also see that PCG has a higher computing efficiency than ZPC for simulating the parton cascade in a heavy-ion collision of Pb-Pb at sqrt(sNN)=2.76 TeV. This is an indication that parallelized collision detection exploiting the GPU's multi-core parallel processing capability in simulating the parton cascade are helpful for saving computing resources. Certainly the speedup, which is the ratio of t_ZPC over t_PCG in Table I, depends on what hardware and software one uses. The GPU we used is NVIDIA C1060 card [24]. In APPENDIX B we listed in more detail the hardware and software that were used for obtaining the results we presented this paper.

One may note is that the 2*x* to 3*x* speedups shown in the Table I may be improved if ignoring the time used for writing onto disk the data, which defines the parton collision history. To make the point convincible we tabulated in Table II the time ZPC, together with PCG, used for simulating the parton cascade without the overhead of writing the onto disk the data about parton collision history. What we can tell comparing Table I with Table II is that better speedups can be expected for simulating the parton cascade in an N-parton system formed in a Pb-Pb collision during data production when it may not be necessary to record parton collision history.

B. *Speedups for Statistically Significant Number of Events*

Though we see in both Table I and Table II that PCG can gain 2 to 3-fold speedup relative to ZPC, it has to be noted that the speedup is just out of analysis of simulating the parton cascade in one event. In order to come to a solid conclusion as for what a speedup PCG can gain, we use both ZPC and PCG to simulate the parton cascade in AMPT-based generation of hundreds of events for Pb – Pb collisions at sqrt(sNN)=2.76 TeV, respectively. The event-averaged computing time for the simulation of the parton cascade in one event was tabulated in Table III, together with the average speedup that we gained by using PCG. The errors are statistical, and are calculated as the standard deviation. From Table III, one may come to the same conclusion as from Table II, i.e. relative to ZPC we may gain 2 to 3 fold speedup by using PCG in the simulation of the parton cascade process varying with the impact parameter b. Because for the time being PCG simulates two-parton collision on CPU the same way as ZPC does, the speedup of PCG relative to ZPC may be attributed to the parallelized simulation of the collision detection on the GPU.

TABLE III: Average Speedup and Computing Time for Simulating the Parton Cascade in Generating Monte Carlo Events for Pb-Pb Collisions at Sqrt(sNN)=2.76 TeV by Using ZPC Compared with That by Using PCG

Number of events	b(fm)	<t_ZPC>(s)	<t_PCG>(s)	<speedup>
760	1	518.5 ± 4.5	152.2 ± 1.3	3.42 ± 0.01
960	10	12.7 ± 0.2	5.6 ± 0.1	2.20 ± 0.01

V. Summary and Conclusion

On the basis of ZPC, we have introduced a GPU-based algorithm for accelerating the simulation of the parton cascade process in the early stage of ultrarelativistic heavy-ion collisions. Implementing the algorithm, we have developed a code called PCG (Parton Cascade on GPU) by using CUDA C. Running both ZPC and PCG, we simulated parton cascade in AMPT-generated heavy-ion collision events of Pb - Pb at sqrt(sNN)=2.76 TeV with impact parameter b ranging from 0 to 10 fm. We compared results from running ZPC on CPU with those from running PCG. The comparison demonstrates that PCG can give the same parton collision history as ZPC does thus the correctness of PCG is proved. Additionally, the comparison shows that by using PCG speedups in the range of 2*x* to 3*x* can be obtained relative to ZPC in our computing environment. Therefore PCG is ready for being integrated into AMPT and can help save huge amount of computing resources in the case of AMPT-based large scale Monte Carlo event generation for ultrarelativistic heavy-ion collisions at sqrt(sNN)=2.76 TeV.

APPENDIX A

Cudarun (*x*, *y*, *z*, *w*, *px*, *py*, *pz*, *pw*, …, *N*) may also be invoked in AMPT thus space-time and momentum energy information for N partons are ready for parton cascade. The information are contained in the following data structures: *double x*[*N*], *y*[*N*], *z*[*N*], *w*[*N*], *px*[*N*], *py*[*N*], *px*[*N*], *pw*[*N*].

The sketch for algorithm of PCG:

```
cudarun (x, y, z, w, px, py, pz, pw, …, N){
step1: after memcpy(…), on device datTime(…) calculates for each parton the number of collisions, and for each collision the collision time and indices of the two colliding partons; uses a shared-memory-based reduction procedure to get and then save in global memory the smallest collision time in each block of threads, together with the indices of the two colliding partons;
step2: after using memcpy (…), on host calculate t_collision_time for the first two-parton collision and the indices of the two colliding partons i and j;
step3:
while (t_collision time <t_preset_threshold) {
step3_1: on host, simulate two_parton collision between parton i and j and update x[i], y[i]], z[i], w[i], x[j], y[j], z[j], w[j], px[i], py[i], pz[i], pw[i], px[j], py[j], pz[j], pw[j];
step3_2: after memcpy(…), on device ijUpdatTime(…, i, …) updates collision time and indices for all of the colliding pairs that include i;
step3_3: after memcpy(…), on device ijUpdatTime(…, j, …) updates collision time and indices for all of the colliding pairs that include j;
step3_4: on device updatTime(…) updates for each parton the number of collisions, and for each collision the collision time and indices of the two colliding partons, one of which is either i or j; uses a shared-memory-based reduction procedure to get and then save in global memory the smallest collision time in each block of threads, together with the indices of the two colliding partons;
step3_5: after memcpy(…), on host update t_collision_time for the next earliest two-parton collision and the indices of the two colliding partons i and j;
        }
}
```

APPENDIX B

The version of AMPT, which generates the parton system for both PCG and ZPC to simulate parton cascade, is v1.26t4-v2.26t4. The compiler [25] we used to compile code written in FORTRAN is GNU FORTRAN (GCC) 4.1.2 20080704 (Red Hat 4.1.2-44). The compiler options for gfortran are -fdefault-real-8 -O2. For compiling our CUDA C code cudarun.cu, which contains three kernels datTime(), ijUpdatTime() and updatTime(), we in command line issued nvcc –arch=sm_13 –O2 –c cudarun.cu. The computing environment for this work mainly consists of one core of CPU that is Intel Xeon E5410 @ 2.33GHz, and one GPU that is NVIDIA C1060 Tesla T10 [24], in addition to CUDA3.2 [13] and Red Hat 4.1.2-44 Linux 2.6.18-128.el5 for x86_64. The CUDA C codes as well as the FORTRAN subroutines PCG uses are available per request.

ACKNOWLEDGMENT

Qing-Jun Liu would like to thank Professors Zi-Wei Lin and Bin Zhang for helpful discussions and allowing us to use AMPT and ZPC code, on which this work is based. Computing support is provided by the Supercomputing Center, Computer Network Information Center, Chinese Academy of Sciences.

REFERENCES

[1] Z. W. Liu, C. M. Ko, B. A. Li, and S. Pal, "A multi-phase transport model for relativistic heavy ion collisions," *Phys. Rev. C*, vol. 72, no. 064901, December 2005.

[2] M. Gyulassy and X. N. Wang, "HIJING 1.0: A monte carlo program for parton and particle production in high-energy hadronic and nuclear collisions," *Comput. Phys. Commun.*, vol. 83, pp. 307–331, December 1994.

[3] B. Zhang, "ZPC 1.0.1: A parton cascade for ultrarelativistic heavy ion collisions," *Comput. Phys. Commun.*, vol. 109, pp. 193–206, April 1998.

[4] B. A. Li and C. M. Ko, "Formation of superdense hadronic matter in high-energy heavy ion collisions," *Phys. Rev. C.*, vol. 52, pp. 2037-2063, October 1995.

[5] T. Sjostrand, S. Mrenna, and P. Z. Skands, "A brief introduction to PYTHIA 8.1," *Comput. Phys. Commun.*, vol. 178, no. 1, pp. 852-867, 2008

[6] M. Bahr *et al.*, "Herwig++ Physics and Manual," *Eur. Phys. J. C*, vol. 58, no. 4, pp. 639-707, December 2008.

[7] X. N. Wang and M. Gyulassy, "Gluon shadowing and jet quenching in *A*+*A* collisions at $\sqrt{s}$ =200 *A* GeV," *Phys. Rev. Lett.*, vol. 68, pp. 1480-1483, March 1992.

[8] L. Lonnblad and T. Sjostrand, "Bose-Einstein effects and W mass determinations," *Phys. Lett.* B, vol. 351, pp. 293-301, May 1995.

[9] H. Stocker *et al.*, "Nuclear fluid dynamics versus intranuclear cascade-possible evidence for collective flow in central high-energy nuclear collisions," *Phys. Rev. Lett.*, vol. 47, pp. 1807-1810, December 1981.

[10] S. Agostinelli *et al.*, "Geant4 — a simulation toolkit," *Nucl. Instrum. Meth.* A, vol. 506, no. 3, pp. 250–303, July 2003.

[11] J. Allison *et al.*, "Geant4 developments and applications," *IEEE Trans. on Nucl. Sci.*, vol. 53, no. 1, pp. 270–278, February 2006.

[12] W. Lukas, "Fast simulation for ATLAS: Atlfast-II and ISF," *J. Phys.: Conf. Ser.*, vol. 396, no. 022031, December 2012.

[13] NVIDIA Corporation, *CUDA Toolkit*, version 3.2, November 2010.

[14] R. Farber, *CUDA Application Design and Development*, 1st ed. USA: Morgan Kaufmann, November 2011.

[15] Y. Cotronis, E. Konstantinidis, M. A. Louka, and N. M. Missirlis, "A comparison of CPU and GPU implementations for solving the convection diffusion equation using the local modified SOR method," *Parallel Computing*, vol. 40, pp. 173-185, July 2014.

[16] C. Harris, K. Haines, and L. S. Smith, "GPU accelerated radio astronomy signal convolution," *Exp. Astron*. vol. 22, pp. 129-141, October 2008.

[17] M. S. Friedrichs *et al.*, "Accelerating molecular dynamic simulation on graphics processing units," *J. Comput. Chem.*, vol. 30, pp. 864–872, April 2009.

[18] C. M. Bard and J. C. Dorelli, "A simple GPU-accelerated two-dimensional MUSCL-hancock solver for ideal magnetohydrodynamics," *J. Comput. Phys.*, vol. 259, pp. 444-460, February 2014.

[19] PHENIX Collaboration, "Formation of dense partonic matter in relativistic nucleus-nucleus collisions at RHIC: Experimental evaluation by the PHENIX collaboration," *Nucl. Phys.* A, vol. 757, pp. 184-283, August 2005.

[20] G. L. Murphy, "Big-Bang model without singularities," *Phys. Rev. D*, vol. 8, pp. 4231-4233, December 1973.

[21] B. L. Combridge, J. Kripfgang, and J. Ranft, "Hadron production at large transverse momentum and QCD," *Phys. Lett. B*, vol. 70, pp. 234-238, September 1977.

[22] R. Li, H. Jiang, H. C. Su, J. Jenness, B. Zhang, "Speculative Parallelization of Many-particle Collision Simulations," in *Proc. the 2007 International Conference on Parallel and Distributed Processing Techniques and Applications*, June 2007, pp. 128–136.

[23] Q. J. Liu, F. Liu, N. Nie, C. Zhou, and W. Q. Zhao, "GPU-based first collision detection in parton cascade in heavy-ion collisions," in *Proc. the International Conference on Electrical, Electronics, Computer Engineering and their Applications*, Kuala Lumpur, Malaysia, pp. 12–15, November 2014.

[24] NVIDIA Corporation. (2008). Tesla C1060 Computing processor board. [Online]. Available: http://www.nvidia.com/docs/IO/56483/Tesla_C1060_boardSpec_v03.pdf, September

[25] GCC Team. (February 2007). [Online]. Available: https://gcc.gnu.org/gcc-4.1/

Qingjun Liu received the BS degree in 1986, MS degree in 1989, and Ph.D. degree in 1993, respectively, in theoretical physics from JILIN University, Changchun, China.

His major research fields are high energy physics and computational physics. Since 2004, he has been a professor of physics in the Beijing Institute of Petro-chemical Technology, Beijing, China. In the period of 1994 to 2003, he worked as a Posdoc. in the Theoretical Physics Division, Institute of High Energy Physics (IHEP), Chinese Academy of Sciences, Beijing China; a research associate in the Department of Physics, University of Washington, Seattle, USA; a computing scientist in the Computing Center of IHEP, Beijing, China and a visiting computing scientist at the Computing Division of Fermilab, USA. His current research interest is GPU-accelerated Monte Carlo simulation of ultra-relativistic heavy-ion collisions. His major publications include: 1) Q. J. Liu and T. A. Trainor, "Jet quenching and event-wise mean-pt fluctuations in Au-Au collisions at √sNN = 200 GeV in Hijing-1.37," *Phys. Lett.* B, vol. 567, pp. 184-188, August 2003. 2) Q. J. Liu and W. Q. Zhao, "Elastic parton scattering and non-statistical event-by-event mean-pt fluctuations in Au - Au collisions at RHIC," *Phys. Rev.* C, vol. 77, no. 034902, March 2008. 3) Q. J. Liu and W. Q. Zhao, "Iterative Solution for Groundstate of H2+ Ion," *Commun. in Theor. Phys.*, vol. 53, no. 01, pp. 57-62, January 2010.

Fang Liu received the BS degree in 2005 in Nanjing University, the Ph.D. degree in 2010 in the Institute of Software, Chinese Academy of Sciences, where she received The Excellence Award of the Presidential Academy Awards. Since 2010, she has been working in Supercomputing Center, Computer Network Information Center, Chinese Academy of Sciences, China. She was promoted to associate professor at the beginning of this year. Her major research field is high performance computing, general purpose computing on gpu. Her major publications include: 1) F. Liu, M. Huang, X. Liu, and E. Wu, "efficient depth peeling via bucket sort," in *Proc. ACM High Performance Graphics*, 2009, New York: ACM Press, pp. 51-57. 2) F. Liu, M. Huang, X. Liu, and E. Wu, "FreePipe: A programmable, parallel rendering architecture for efficient multi-fragment effects," in Proc. *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 2010, New York: ACM Press, pp. 75-82.

Ningming Nie received the BS degree in 2005 in information and computing science in Xiangtan University, and the Ph.D. degree in 2010 in Academy of Mathematical & Systems Science, Chinese Academy of Sciences, China. Since 2010, she has being working in Supercomputing Center, Computer Network Information Center, Chinese Academy of Sciences, China. She was promoted to associate professor at the beginning of this year. Her major research field is High Performance Computing. Her major publications include: 1) N. Nie, J. Huang, W. Wang, and Y. Tang, "solving spatial-fractional partial differential diffusion equations by spectral method," *J. of Stat. Comput. and Simul.*, vol. 84, no. 6, 2014. 2) J. Huang, N. Nie, and Y. Tang, "A second order finite difference-spectral method for space fractional diffusion equation," *Science China Mathematics*, vol. 57, no. 6, pp. 1303–1317, 2014.

Chunbao Zhou received the Ph.D. degree in 2012 in the College of Computer Science and Technology in Ji-Lin University, China. Since 2012, he has been working in Supercomputing Center of CAS, Computer Network Information Center, Chinese Academy of Sciences, China.

His major research field is high performance computing and bioinformatics. his major publications include: 1) C. Zhou, X. Lang, Y. Wang, C. Zhu, Z. Lu, and X. Chi, "Parallel metropolis coupled Markov chain Monte Carlo for isolation with migration model," *Appl. Math. Inf. Sci.*, vol. 7, pp. 219-224, 2013. 2) C. Zhou, J. Wang, Y. Wang, and Y. Liang, "Identification of phage-induced genomic islands in the 13 Streptococcus pyogenes strains using genome barcodes," *Int. J. of Data Mining and Bioinformatics*, vol. 10, no. 3, pp. 269-284, 2014.