Particle Simulator Research Team

1. Team members

Junichiro Makino (Team Leader)
Keigo Nitadori (Research Scientist)
Masaki Iwasawa (Postdoctoral Researcher)
Ataru Tanikawa (Postdoctoral Researcher)
Miyuki Tsubouchi (Technical Staff)

2. Research Activities

We are developing particle-based simulation software that can be used to solve problems of vastly different scales.

Simulation schemes for hydrodynamics and structural analysis can bedivided into grid-based and particle-based methods (see Figure 1). In grid-based methods, the computational region is mapped to regular or irregular grids. Continuous distributions of physical values are represented by discrete values at grid points, and the governing partial differential equation is approximated to a set of finite difference equations.

In the case of the particle-based methods, physical values are assigned to particles, while the partial differential equation is approximated by the interactions between particles.

Both methods are widely used, and they have their advantages and disadvantages. The computational cost of grid-based schemes is generally lower than that of particle-based methods with similar number of freedoms. Thus, if an near-uniform grid structure is appropriate for the problem to be solved, grid-based methods perform better.

The advantage of the particle-based methods comes from the fact that they use "Lagrangian" schemes, in which the particles move following the motion of the fluid in the case of the CFD calculation. In the case of grid-based methods, we generally use "Eulerian" schemes, in which the grid points do not move.

There are three points in which the Lagrangian schemes are better than Eulerian schemes.

One is that the Lagrangian schemes are, to some extent, adaptive to the requirement of the accuracy, since when a low-density region is compressed to become high density.

Second one is that the timestep criteria are quite different. In the case of the Lagrangian schemes, the timestep is determined basically by local sound velocity, while in the Eulerian scheme by global velocity. Thus, if a relatively cold fluid is moving very fast, the timestep for the Eulerian schemes can be many orders of magnitude shorter than that for Lagrangian schemes.

Finally, in the case of fast-moving low-temperature fluid, the required accuracy would be very high for Eulerian scheme, since the error comes from the high velocity, while that error would be transferred to internal energy of the fluid element which is much smaller than that of the kinetic motion.

Of course, there are disadvantages of Lagrangian schemes. The primary one is the difficulty of construction of such schemes in two or higher dimensions. In the case of one-dimensional calculation, it is easy to move grid points following the motion of the fluid, but in two or higher dimensions, the grid structure would severely deform if we let the grid points follow the flow. Thus, we have to reconstruct the grid structure every so often. This requirement causes the program to become complex. Moreover, reconstruction of the grid structure (so called remeshing) means we lose numerical accuracy.

Particle-based methods "solve" this difficulty by not requiring any mesh. In particle-based methods, particles interact with its neighboring particles, not through some connection through grid, but through distance-dependent kernel functions. Thus, there is no need of remeshing. As a result, particle-based schemes are simple to implement, and can give reasonable results even when the deformation is very large. Another important advantage is that it is relatively easy to achieve high efficiency with large-scale particle-based simulation.

In the case of grid-based schemes, in order achieve some adaptivity to the solution, we have to use either irregular grid or regular grid with adaptive mesh refinment. In both cases, adaptivity breaks the regularity of the mesh structure, resulting in non-contiguous access to the main memory. In the case of the particle-based schemes, it does require some irregular memory access, but it is relatively straightforward to make good use of spacial locality, and thereby achieving high efficiency. Similarly, very high parallel performance can be achieved.

However, it has its own problems. In the case of the SPH method, it has been known that the standard scheme cannot handle the contact discontinuity well. It also require rather strong artificial viscosity, which results in very low effective Reynolds number.

Thus, in many fields of computational sciences, many groups are working on implementation of high-performance particle-based simulation codes for their specific problem.

One serious problem here is that, high-performance, highly-parallel simulation codes for particle-based simulations are becoming more and more complex, in order to make full use of modern supercomputers. We need to distribute particles to many computing nodes in an appropriate way, so that the communication between nodes is minimized and at the same time near-optimal load balance is achieved. Within each nodes, we need to write an efficient code to find neighbor particles, rearrange data structure so that we can make good use of the locality, make good use of multiple cores and SIMD units within each core.

Even for the case of very simple particle-particle interaction such as the Lenard-Jones potential or Coulomb potential, the calculation code tends to be very large, and since the large fraction of the code is written to achieve a high efficiency on a specific architecture, it becomes very hard to port a code which is highly optimized to one architecture to another architecture.

Our goal is to develop a "universal" software that can be applied to a variety of problems whose scales are vastly different. In designing such universal software, it is important to ensure that it runs efficiently on highly parallel computers such as the K computer. Achieving a good load balance with particle-based simulation is a difficult task, since using a regular spatial decomposition method causes severe load imbalance, though this works well for grid-based software. Consequently, we have developed an adaptive decomposition method that is designed to work in a way that the calculation time on each node is almost the same, resulting in near-optimal load balance. The strategy to develop such a universal software is as follows.

We first construct a highly parallel and very efficient implementation of the TreePM algorithm for gravitational N-body problem. This is actually not a completely new implementation, but the GreeM code developed by researchers of the Strategic Program for Innovative Research (SPIRE) Field 5 “The origin of matter and the universe. In collaboration with the Field 5 researchers, we improve the efficiency of the code and study the issues of the data structure, domain decomposition, load balance strategy etc.

In the second stage, we will develop a prototype of the parallel particle simulation platform. We will design the platform so that it can be used for multiple physical systems. In practice, we consider the following three applications as the initial targets.

1. Gravitational N-body simulation
2. Smoothed Particle Hydrodynamics
3. Molecular Dynamics

In the meantime, we will also investigate the way to improve the performance and accuracy of the current particle-based algorithms for hydrodynamics.

3. Research Results and Achievements

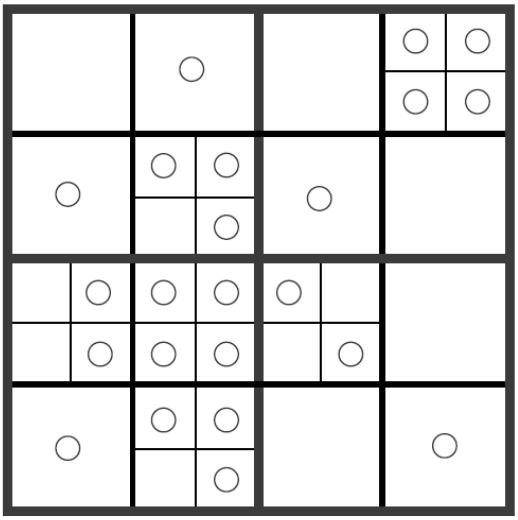
As we stated in section 2, we are working on the three major subtopics, in order to develop the universal platform for particle simulations.

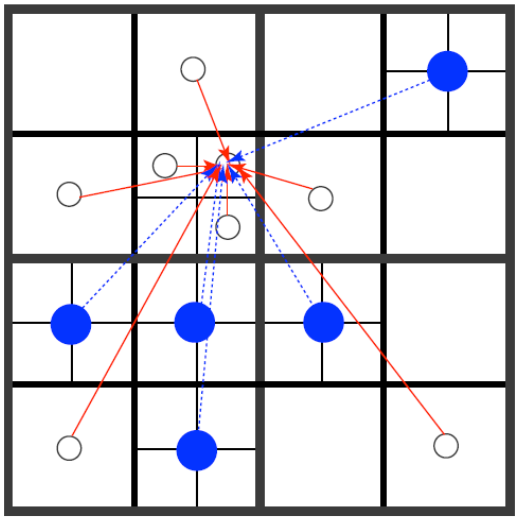
In the following, we briefly describe the status of our research in each subtopic.

3.1. High-performance gravitational N-body solver

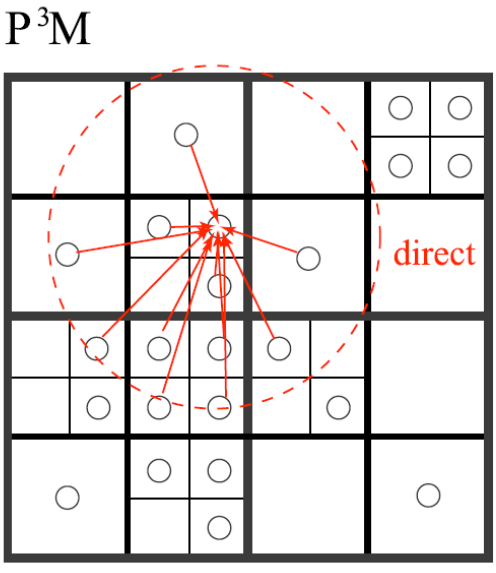
In collaboration with the researchers of researchers of the Strategic Program for Innovative Research (SPIRE) Field 5 "The origin of matter and the universe, we have developed an extremely high performance gravitational N-body solver, GreeM, for the K computer. It achieved, as of November 2012, the sustained performance of 5.67 petaflops (55% of the theoretical peak performance of the K computer). Even more important is its performance measured in the unit of the number of particles updated per second. GreeM on K integrates 4 x 10^{11} particles per second. Researchers in the US developed a similar calculation code on the BG/Q, and its measured speed was 1.6 x 10^{11} particles per second, on the BG/Q machine with the peak speed of 20Pflops. Thus GreeM on K is about 2.4 times faster than the best competing code on a machine nearly two times faster than the K computer. In other words, GreeM on K is about five times more efficient than the best competing code. The numerical accuracy was similar. In the following, we briefly describe the method used and the possible reason for the performance difference between GreeM on K and the calculation on BG/Q. The full detail of the GreeM code is discussed in Ishiyama et al (2012).

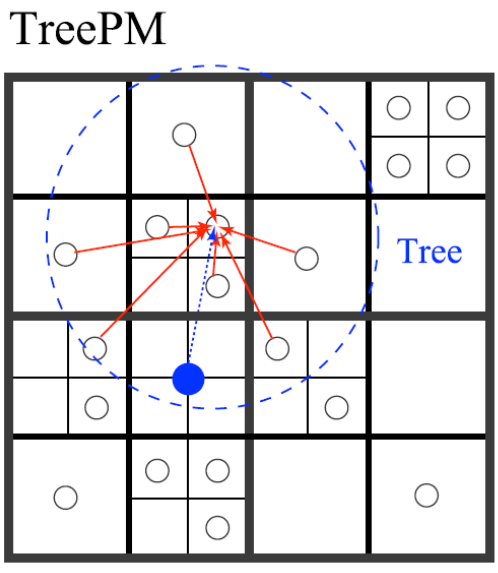
We use the TreePM algorithm as the basic method for the evaluation of gravitational interaction between particles. TreePM is a combination of the tree method and the P^3M (particle-particle particle-mesh) scheme. Figure 1 shows the basic idea of the tree algorithm. The space is divided into a hierarchical octree structure (quadtree in the figure). Division is stopped when a cell contains one or no particle. When we calculate the force on a particle, we evaluate the force from a group of particles, with size larger for more distant particles. In this way, we can reduce the calculation cost from $O(N^2)$ to $O(N \log N)$.





The tree algorithm is widely used, but when the periodic boundary condition is applied, we can actually use a more efficient scheme, since we can calculate the long-range, periodic term using FFT. The P^3M scheme has been used for such problem, but it has the serious problem that when the density contrast becomes high, the calculation cost increases very quickly. The TreePM scheme solves this difficulty by using the tree algorithm to evaluate the forces from nearby particles. Even when there are very large number of neighbor particles, the calculation cost does not increase much, since the calculation cost of the neighbor force is proportional to the logarithm of the number of neighbors.

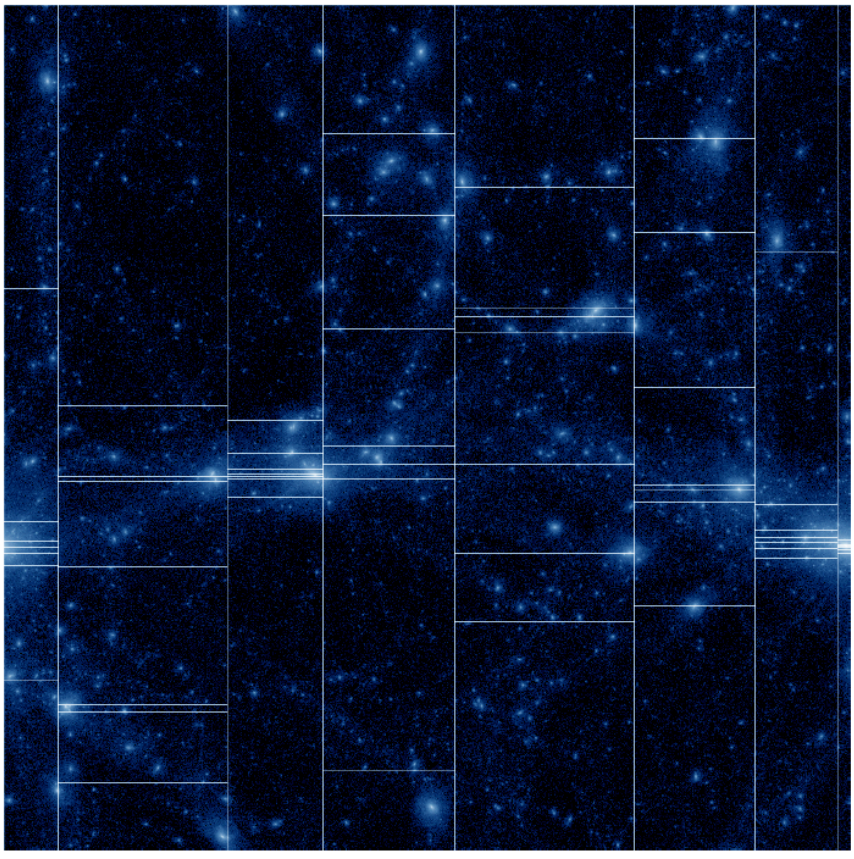




In order to map the problem to the distributed-memory parallel computer such as the K computer, we adopted the approach to divide the space into domains and assign particles in one domain to one calculation node. We used the orthogonal recursive multisection method developed by the team leader some years ago. It is the generalization of the orthogonal recursive bisection (ORB), which has been widely used in many parallel implementations of the tree algorithm.

With ORB, we recursively divide space into two halves, each with the same number of particles. An obvious disadvantage of the ORB approach is that it can utilize the computing nodes of integral powers of two. Thus, in the worst case we can use only half of the available nodes.

The difference between the multisection method and the ORB is that with the multisection method we allow the divisions to arbitrary number of domains, instead of bisection. This would allow too many possible divisions. In our current implementation, we limit the number of levels to three, and make the numbers of divisions at all levels as close as possible. Thus, our domain decomposition is topologically a simple three-dimension grid. This fact makes the multisection method well suited to the machines with the 3D torus network like the K computer.



Originally, with the ORB method the domains contain the same number of particles. It was soon realized that this division did not provide the best load balancing, and people started to use the number of interaction calculations as the measure for the calculation cost. We found that even that is not ideal, and have adopted a much better approach. We simply measure the cost in terms of the CPU seconds, and assign particles the average CPU time. Then we divide the space so that each node should require the same CPU time. This approach turned out to be able to achieve the near-ideal load balance.

Finally, in order to achieve the high efficiency, the efficiency of the force calculation kernel is extremely important. In the case of the K computer, we need to make use of the two-way SIMD unit, and the fact that two units are there. In order to achieve this goal, we developed a new expression for the spline kernel for the force cutoff, which requires only one masked operation. For the force kernel, we have achieved the performance of 72.8% of the theoretical peak or actually 97% of the theoretical limit when we take into account the fact not all floating-point operations are mapped to FMA operations.

Our current implementation is fairly straightforward, and there is nothing unusual. Thus, we have some difficulty in understanding why the competing code is much slower. The most likely reason is that the competing code is the modification of the P^3M code developed by the same group for the Roadrunner supercomputer, which has the IBM Cell processor. Either their code is not yet optimized for the BG/Q, or the original structure of the P^M code resulted in some intrinsic limitation of the performance.

We have developed a "reference code" for gravitational N-body simulation on the K computer. This code is fairly well optimized for the K computer, and shows quite good scalability for even for relatively small-size problems. The asymptotic speed per timestep for large number of nodes is around 7ms. This speed is comparable to that of highly optimized molecular dynamics codes on K, even though our code is designed to handle highly inhomogeneous systems.

We will use this code as the reference implementation for more generalized particle simulation platform which will be described in the next subsection.

3.2. Particle Simulation Platform

We have made detailed specification of the particle simulation platform, which we call FDPS (Framework for Developing Particle Simulator).

The basic idea of FDPS is that the application developer (or the user) specified the way the

particles interact with each other, and the rest is taken care by FDPS. Here, "the rest" includes I/O, domain decomposition and re-distribution of particles, evaluation of interactions between particles, including those in different domains (different MPI processes, for example).

In practice, there are many additional details the user should give. Consider a relatively simple case of particles interacting with soften $1/r$ potential. There are a number of small but important points one has to decide on. For example, what algorithm should be used for the interaction calculation? Even if we limit the possibilities to reasonably adaptive schemes for open boundary problems, we have the choice between Barnes-Hut tree and FMM. For both algorithms, there are many different ways to parallelize them on distributed-memory parallel computers. Also, there are infinitely many variations for the time integration schemes.

The base layer of FDPS offers the domain decomposition based on the recursive multisection algorithm (Makino 2004), with arbitrary weighting function for the load balancing (Ishiyama et al 2009). It also offers the parallel implementation of interaction calculation between particles.

The domain decomposition part takes the array of particles on each node as the main argument. It then generates an appropriate domain for each node, redistribute particles according to their locations, and returns.

The interaction calculation part takes the array of particles, the domain decomposition structure, and the specification of the interaction between particles as main arguments. The actual implementation of this part need to take into account a number of details. For example, the interaction can be of long-range nature, such as gravity, Coulomb force, and interaction between computational elements in the boundary element method (BEM). In this case, the user should also provide the way to construct approximations such as the multiple expansion and the way to estimate error. The interaction might be of short-range nature, with either particle-dependent or independent cutoff length. In these cases, the interaction calculation part should be reasonably efficient in finding neighbor particles.

We have completed the specification document for API of these part, and currently working on the prototype (single-node) implementation of this API.

3.3. Improvements on SPH

SPH (Smoothed Particle Hydrodynamics) has been used in many field, including astrophysics, mechanical engineering and civil engineering. Recently, however, it was pointed out that the standard formulation of SPH has numerical difficulty at the contact discontinuity.

We have been working on the possible solution on this problem, and have made two significant steps in this year. The first one is the generalization of the density-independent SPH to an arbitrary equation of state, and the second one is its further generalization which requires the continuity of neither density nor pressure.

The density-independent SPH is a new formulation of SPH we proposed in 2011. It uses the pressure, instead of the density, as the basic variable using which we evaluate the gradient of other quantities. With hydrodynamics, the pressure is continuous everywhere, except at the shock front. In the case of SPH, we use the artificial viscosity so that the physical variables are all continuous and differentiable even at the shock front. Thus, by using pressure as the basic variable, we can avoid the numerical difficulty associated with the contact discontinuity.

In the case of an ideal gas, we can calculate the pressure easily from the internal energy of particles, but if the equation of state is non-ideal, we cannot calculate the pressure explicitly. We can obtain the pressure by solving an implicit equation, and found that the additional cost of solving the equation is actually small. The reason is that we can also integrate the time evolution of the pressure, and therefore can obtain very good initial guess. Iteration with simple direct substitution is stable and fast enough.

We also developed a very different way to achieve the density independence. In DISPH, we used the pressure-energy pair of intensive and extensive thermodynamic variables to construct the volume estimator of a particle. This estimator works great at the contact discontinuity, at which the pressure is almost constant but the density is discontinuous. However, it behaves poorly where the pressure changes rapidly. One example is the surface of a fluid, either that of liquid or self-gravitating gas. Since the pressure at the surface is by definition zero, the volume estimator based on the pressure cannot give a valid volume element.

We constructed an SPH scheme which uses artificial density-like quantity as the base of the volume estimator. It evolves through usual continuity equation, but with additional diffusion term. Thus, we can guarantee the continuity and differentiability of this quantity, except at the initial condition or at the moment when two fluid elements contact with each other. This scheme seems to work extremely well, and we are currently working on the way to extend this scheme so that it can handle free surface accurately.

4. Schedule and Future Plan

We plan to release the first prototype of the platform by FY 2014. It will have the basic abilities to run on large-scale parallel computers with reasonable load-balancing, for multiple forms of the interparticle interaction formula. We will extend this to fully user-specifiable interface to interparticle interactions in the future release.

References

Ishiyama, T. Nitadori, K, and Makino, J., 2012, 4.45 Pflops astrophysical N-body simulation on K computer: the gravitational trillion-body problem, SC '12 Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis Article No. 5 IEEE Computer Society Press Los Alamitos, CA, USA ©2012 ISBN: 978-1-4673-0804-5

5. Publication, Presentation and Deliverables

(1) Journal Papers

[1] "Few-body modes of binary formation in core collapse", Tanikawa, Ataru; Heggie, Douglas C.; Hut, Piet; Makino, Junichiro, Astronomy and Computing, Volume 3, p. 35-49.

[2] "The Cosmogrid Simulation: Statistical Properties of Small Dark Matter Halos", Ishiyama, Tomoaki; Rieder, Steven; Makino, Junichiro; Portegies Zwart, Simon; Groen, Derek; Nitadori, Keigo; de Laat, Cees; McMillan, Stephen; Hiraki, Kei; Harfst, Stefan, The Astrophysical Journal, Volume 767, Issue 2, article id. 146, p. 14, (2013).

[3] "Phantom-GRAPE: Numerical software library to accelerate collisionless N-body simulation with SIMD instruction set on x86 architecture", Tanikawa, Ataru; Yoshikawa, Kohji; Nitadori, Keigo; Okamoto, Takashi, New Astronomy, Volume 19, pp. 74-88.

[4] "Merger criteria of multiple massive black holes and the impact on the host galaxy", Tanikawa, A.; Umemura, M. Monthly Notices of the Royal Astronomical Society, Volume 440, Issue 1, pp.652-662.

[5] "Flaring up of the compact cloud G2 during the close encounter with Sgr A*", Saitoh, Takayuki R.; Makino, Junichiro; Asaki, Yoshiharu; Baba, Junichi; Komugi, Shinya; Miyoshi, Makoto; Nagao, Tohru; Takahashi, Masaaki; Takeda, Takaaki; Tsuboi, Masato; Wakamatsu, Ken-ichi, Publications of the Astronomical Society of Japan, Volume 66, Issue 1, id.1.

[6] "Evolution of star clusters in a cosmological tidal field", Rieder, Steven; Ishiyama, Tomoaki; Langelaan, Paul; Makino, Junichiro; McMillan, Stephen L. W.; Portegies Zwart, Simon, Monthly Notices of the Royal Astronomical Society, Volume 436, Issue 4, pp.3695-3706.

[7] "Density-Independent Smoothed Particle Hydrodynamics for a Non-Ideal Equation of State", Hosono, Natsuki; Saitoh, Takayuki R.; Makino, Junichiro, Publications of the Astronomical Society of Japan, Vol.65, No.5, Article No.108, p.11.

[8] "Dynamical evolution of stellar mass black holes in dense stellar clusters: estimate for

merger rate of binary black holes originating from globular clusters", Tanikawa, A., Monthly Notices of the Royal Astronomical Society, Volume 435, Issue 2, pp.1358-1375.

[9] "A Density-independent Formulation of Smoothed Particle Hydrodynamics", Saitoh, Takayuki R.; Makino, Junichiro, The Astrophysical Journal, Volume 768, Issue 1, article id. 44, p. 24, (2013).

(2) Invited Talks

[10] "ポスト「京」プロジェクトハードウェア概要", 牧野　淳一郎、理研和光一AICS 合同シンポジウム「京、ポスト京と基礎物理」2014/1/7。

[11] "理論家からみた統計誤差と系統誤差" 牧野　淳一郎、銀河系目標ミニワークショップ@三鷹 2013/12/27。

[12]「重力相互作用カーネルのチューニング —組み込み関数を用いた手動 SIMD 化」似鳥啓吾、「京」における高速化ワークショップ（高度情報科学技術研究機構（RIST）主催）2013/12/18。

[13] "PC クラスタの終わりの始まり？---エクサ時代のミッドレンジのあり方:" 牧野　淳一郎、PC クラスタシンポジウム 2013/12/12。

[14] "次世代高性能計算機への昨今動き(富田さんの代理的なにか)+ 今後の計算機と計算科学---なぜ我々は 7 年前の間違いを繰り返{す、した}のか" 牧野　淳一郎、計算惑星科学シンポジウム@石垣島 2013/11/23。

[15]「Xeon Phi 上での N 体計算コードの実装」 似鳥　啓吾、東京大学コンピュータ科学専攻講演会、2013/10/9。

[16] "Exascale computers? / Can we believe SPH simulations of Giant Impact?", Jun Makino, Science as Method and Methodology for Problems on the Earth and Life, Sep 15-17, 2013, Nagoya, Japan。

[17] "エクサスケールシステムに向けて --- 分散メモリ超並列アーキテクチャの復活" 牧野淳一郎、第 5 回アクセラレーション技術発表討論会(第 2 種研究会) 2013/9/6。

[18] "Exascale computers in --- 2019?", Jun Makino, Large-Scale simulation of Formation and Evolution of Planetary Systems: Kobe 2013 Summer Workshop, Aug 7-8, 2013, Kobe, Japan.

[19] "京の威力で「見えない宇宙」の正体に迫る —ダークマターの超大規模シミュレーション" 牧野　淳一郎、京コンピュータ・シンポジウム 2013/5/13。

(3) Posters and Presentations

[20] "球状星団におけるコンパクト連星形成", 谷川衝, 2013 年 12 月 25-27 日, 東京大学, 第 26 回理論懇シンポジウム。

[21] "球状星団におけるコンパクト連星形成について", 谷川　衝, 2013 年 11 月 5-6 日, 筑波大学, 第 5 回「学際計算科学による新たな知の発見・統合・創出」シンポジウム —T2K-Tsukuba、HA-PACS による計算科学の発展と、次世代コンピューティングの展望—。

[22] “二重白色矮星の合体時に発生するホットスポットの構造”, 谷川　衝, 2013 年 9 月 10-12 日, 東北大学, 日本天文学会。

[23] “Detection rate of binary black holes formed in globular clusters”, Tanikawa, A., 3-7 Jun 2013, Yukawa Institute for Theoretical Physics Kyoto University, Yukawa International Seminar 2013 Gravitational Waves Revolution in Astronomy & Astrophysics.