Bull. Nov. Comp. Center, Comp. Science, 45 (2021), 59 - 68

Distributed calculations on multiple independent devices for PCISPH method

Sergey Khayrulin

Abstract. One of the most significant drawbacks of the PCISPH algorithm (predictive corrective SPH), which is often used for the simulation of incompressible viscous liquid dynamics, is the complexity of the organization of distributed calculations on a computational cluster. At the same time, the algorithm supports the data parallelism relative to a single device. In this paper, new methods and algorithms for overcoming this drawback are suggested, based on an effective model of distributing the data between independent computing units. The resulting algorithms have been realized and tested on Novosibirsk State University's cluster.

Keyword : PCISPH, SPH, parallel computing

Introduction

In the classical N-body problem for calculating forces affecting any object, it is necessary to consider the influence of all bodies in the system. Thus, it is reasonable to assume that numerical algorithms for simulating such a system have quadratic asymptotic complexity $O(N^2)$. One of the critical problems of any numerical model of particles is scaling and involving a greater number of particles to get more detailed results. To calculate the consequent state of the system for each particle, it is essential to know the physical parameters of any other particle.

The combined PIC method (particle in cell) [1] [2] and its modifications [2] assume that the continuum is described both by particles and a fixed grid. In this case, particles carry information only about the mass, while information on fluctuations of physical properties in the system is distributed through fixed grid nodes. This means that for synchronization of parallel calculations on various cluster nodes or single node devices, only timely updates of values in the grid nodes and of information on the number of particles in the cell are sufficient. At the same time, in completely Lagrangian algorithms such as SPH [3] [4] and its modification PCISPH [5] [6], the coordinate system is associated with the movement of fluid in space (the grid moves together with the fluid) and corresponds to fixed points in the medium. Consequently, the continuum is represented by a discrete multitude of particles. Information in the medium is transmitted exclusively through particles.

Although each method is well parallelized over data, for numerical models based on a system of particles it is quite difficult to distribute calculations in systems with independent RAMs, such as cluster nodes, or on various computing devices within one computer and yet with independent RAM modules, such as GPUs, as this requires a data synchronization mechanism. This particular problem arises because particles are dynamic and not tied to specific positions, like nodes of computational grids. However, in the works [7] [8] [9], attempts are made to overcome this shortcoming by logically dividing the simulated space into non-intersecting static or dynamic subspaces — domains. Therefore, particles can be clustered according to a spatial feature, depending on the current position. This implies that each domain is processed by a separate solver. In our interpretation of the PCISPH method, we also relied on this approach. Below in Table 1 is a comparison between our and similar approaches.

Table 1. Comparison of distributed realizations for SPH/PCISPH algorithms

Source	GPGPU technology	Algorithm	Open-source
[7]	CUDA + MPI	SPH	-
[8]	CUDA	SPH	-
[9]	CUDA	PCISPH	-
Sibernetic	OpenCL	PCISPH	+

1.1. PCISPH

Formally the PCISPH algorithm represents a continuous medium as a multitude of separate particles, each interacting with others in accordance with specified laws. Each particle contains information about the local environment state: mass, velocity, density, pressure, viscosity and position in space. It is noteworthy that properties of a particle depend on the surrounding particles, specifically, on properties of particles located in close vicinity

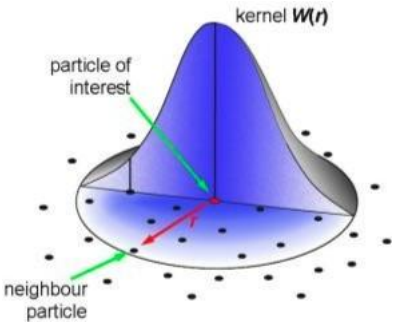

Figure 1. Smoothing of physical properties for particles relative to neighbors. The influence of particles beyond the boundary of the sphere is ignored

Physical properties are smoothed according to equations 1, 2

$$A(x) = \int_{\Omega} A(x')\delta(x - x')dx' \quad (1),$$

$$A(x) = \int_{\Omega} A(x')W(x - x', h)dx' \quad (2),$$

where $A(x)$ is a physical quantity, δ is Dirac's delta function replaced by so-called kernel function W. Kernel function has the following properties:

$$\int$$

$$W(x - x',h)dx' = 1 \lim ,$$

$$W(r,h) = \delta(r) \; {}_{h\to 0},$$

where h is a constant value also called the smoothing radius. As noted above, the influence of particles located beyond the sphere with a radius h and the center in a particular particle is neglected. The hydrodynamic model is described by Navier–Stokes equations for incompressible viscous liquid:

$$\rho\left(\frac{\partial}{\partial t} + u \cdot \nabla\right) = -\nabla p + \mu\nabla \cdot \nabla u + f$$

$$\nabla \cdot u = 0$$

The numerical model is represented by a limited multitude of particles enclosed in the boundary box. To optimize the neighbor search, all medium in the algorithm boundary box is divided into discrete amounts of non-overlapping spatial cells. These cells are static and do not change their volume and position throughout the simulation time. Therefore it is possible to cluster particles by spatial cell identification *cellID*. Spatial cell edge length is equal to 2h Figure 2.

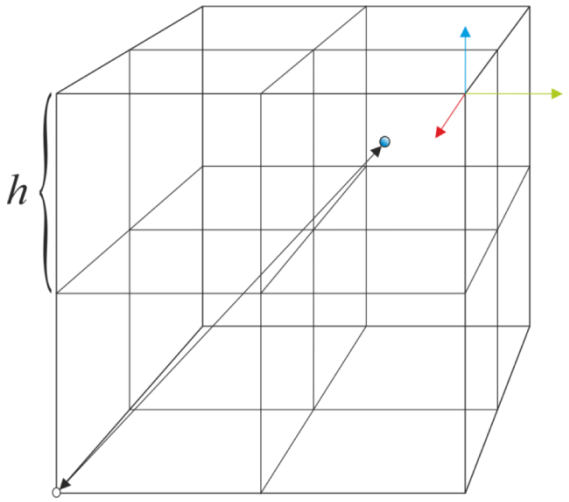

Figure 2. Spatial cell. The blue dot represents a particle

For any particle, the spatial cell index can be defined as

$$cell(x,y,z,h) = \left(\left\lfloor\frac{x - x_{min}}{2h}\right\rfloor, \left\lfloor\frac{y - y_{min}}{2h}\right\rfloor, \left\lfloor\frac{z - z_{min}}{2h}\right\rfloor\right) = (i,j,k),$$

$$cellID = j + k \cdot gridCellY + i \cdot gridCellZ \cdot gridCellY$$

where x,y,z are the coordinates, h is the smoothing radius.

1.2. Calculation distribution

The key concept of the algorithm consists in distribution of data through devices. Each particle processed by a particular device must have all necessary and sufficient data for calculating the neighbor list and physical properties on each new simulation step. The number of particles processed by one device should be correlated with its performance. Additionally, since the system is not static and particles are moving, the algorithm must dynamically distribute data among the devices to support particle distribution and relevance of the data on each device.

As stated above, particles could be clustered by *cellID* known at any particular moment *t*. At the same time, because our primary goal is to parallelize fluid modeling by the PCISPH method, we can assume that it is necessary and sufficient to divide the space along a plane collinear with the gravity vector. Obviously, for updating the properties of particles located in adjacent areas of domains, it is necessary to have information about particles located in neighboring cells from other domains and, at the same time, located in the memory of another device. Such particles can be called imaginary or motionless because the calculation of their physical properties and the determination of their new positions are not performed.

The synchronization of calculations is provided in two aspects: first, the worktime — devices must work simultaneously; second, the data should be consistent. The algorithm should satisfy the following requirements:

1. Scalability over quantity of devices.
2. Devices must work independently.
3. Devices must work synchronously:

(a) devices start working together;

(b) average worktime of a device should differ by no more than a small error.

4. For each device, the algorithm should automatically determine the number of particles.

Initial data contain information about particles: location, velocity, etc; particles are ordered by *cellID* value. It also contains information about calculation devices. Their number and weight coefficients for data distribution are calculated automatically. Figure 3 presents an example of data distribution.

Let us define the partition as a subset of particles processed by a single device. The partition also contains information about ”ghost” particles. The size of each partition depends on the performance coefficient. For the calculation of this characteristic, a heuristic function is suggested. The function calculates the coefficient based on the possible numbers of flows that can be run simultaneously on a particular Device 3.

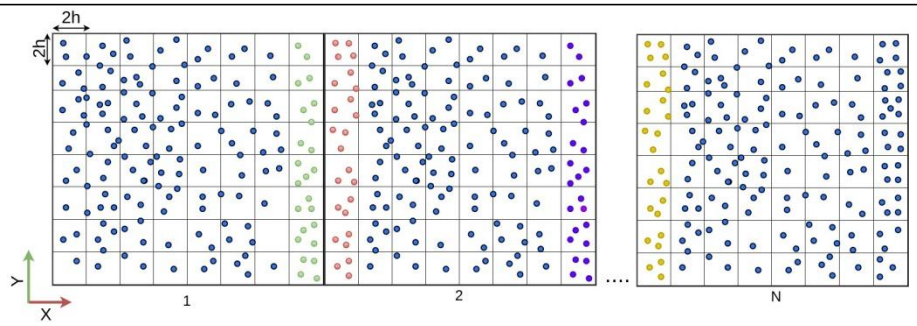

Figure 3. Data distribution among solver devices for PCISPH algorithm. Different colors on the boundary area indicate ghost particles that also must be loaded into a particular solver

$$\epsilon(d_i) = D \cdot WG\,, \tag{3}$$

where d_i is a device, D is a number of available streaming multiprocessors (CUDA cores for NVIDA devices), for CPU is a number of cores, WG is the dimension of the work group for the device. For example, a Radeon R 290X GPU has 44 streaming multiprocessors and WG = 256. In order to achieve runtime synchronization, it is necessary that before each iteration, the data are distributed among the devices proportionately to the performance of the devices. The optimal number of particles for processing by the *i*th device is determined by the following equation 4:

$$N_i' = \left\lfloor N \cdot \frac{\epsilon(d_i)}{\sum_j \epsilon(d_j)} \right\rfloor. \tag{4}$$

When defining a partition, it must also be taken into account that all particles in one cell must be processed by the same device.

1.3. Computation model flow

The computation model defines an abstract representation of how instruction threads are executed in a heterogeneous system. The host part of the program controls the calculation flow and synchronization of data between nodes. A node is an independent device GPU/CPU with isolated memory. Depending on the number of nodes, an appropriate number of parallel threads is created. Each thread runs code separately but in the same address space.

The control flow structure can be divided into several layers depending on the performed task. The main process is an abstraction over the calculation flow, controlling initialization/synchronization data and models, and controlling initialization of devices (solvers). The layer of solvers consists of a list of solver structures, describing the order of command flow for a particular device. Running instructions process a subset of particles, for which the solver is responsible. The last layer is the layer of devices that defines the structures responsible for implementing parallel computing on a particular device. The calculation model is illustrated in Figure 4.

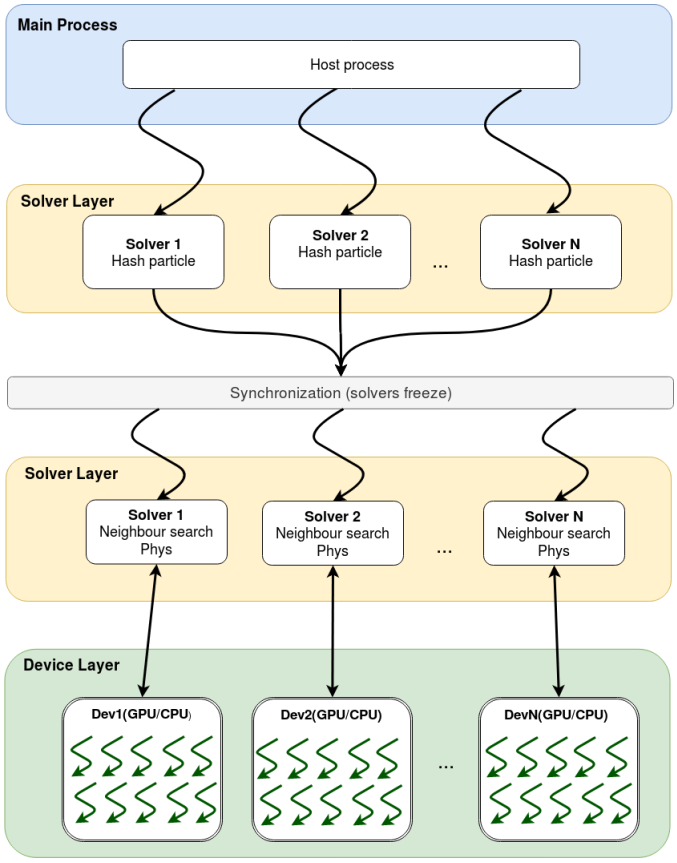

Figure 4. Computation flow model

As shown in Figure 4, at the synchronization stage, all threads are suspended, and the control thread synchronizes data between solvers. After that, all solvers reactivate their work. The mathematical model of the environment is approximated by a numerical model represented by a set of particles or an array of particles. The array is represented as a continuous section of the computer's memory. Data synchronization includes ordering/sorting the array of particles according to the corresponding value of the spatial cell index and updating them in the RAM of the devices. For synchronizing data between control threads and devices, it is proposed to use standard blocking tools, such as condition variables and [10] mutexes, which allow the elimination of the possibility of the race conditions. At the synchronization stage, all solvers are trying to lock the mutex. The thread that succeeds runs the process of updating the partition size with respect to $\epsilon(d_i)$. Next, the sorting process is launched. In the meantime, all other threads are idling and waiting for the lock to be released; after this becomes possible, all data is ordered and can be loaded into the device memory. Thus, the devices can continue their work.

1.4. Results

A number of tests were completed. Classical dam break model configuration was generated for various numbers of particles – 96368, 844203, 1183724, 3547755, 6904779, 9772237, 14204179, 20078971. The tests were run for different configurations of the computing cluster. The number of coprocessors in the system varied from 1 to 8 GPUs – NVIDIA Tesla V100 SMX2 with 80 computing units with 1024 threads on each. Tests were performed on the Novosibirsk State University computing cluster. Each simulation iteration involves several stages:

1. Generating a list of neighbors for each particle.
2. Calculation of changes of physical quantities and smoothing of densityfluctuations (PCISPH).
3. Numerical Integration (Leapfrog, Semi-implicit Euler)
4. Data synchronization

(a) Sorting, 1 thread qsort, parallel – modification of radix sort

(b) Data distribution among the solvers according to sorted array – copying new sub buffers to device memory

5. Updating data.

At each run, the iteration execution time was logged as the sum of 5:

$$T_i^{total} = T_i^{ns} + T_i^{phys} + T_i^{sync}, \quad (5)$$

where T_i^{total} is the total iteration time^i, T_i^{ns} – search for neighbors time, T_i^{phys} – time for updating physical parameters, T_i^{sync} – synchronization time. The most time-consuming operations are sorting processes – T_{sort} and the process of calculating new values of physical quantities. The search for neighbors occurs in parallel and takes from 10 to 20 % of the total computation time on the GPU. The stage of data distribution takes an insignificant amount of time relative to the sorting time, and therefore can be neglected.

At the synchronization stage, the array of particles is sorted, and new domain configurations are distributed among devices. Sorting can work in two modes: serial and parallel. For each test, a fixed number of iterations was performed, and the average time to complete one iteration was calculated as

$$t_{average} = \sum_{i=1}^{N} \frac{T_i^{total}}{N},$$

where N is the number of iterations. The results are presented in Figure 5. The obtained results allow us to conclude that we have achieved significant acceleration of calculations for discrete models described using the method of the PCISPH class.

As seen in Figure 5, sort step significantly impacts the common calculation time for configuration with big numbers of particles. For example, for the configuration with 9 million particles, the average time required for single thread sorting was approximately1.6 seconds, while for parallel sorting it was 0.6 seconds

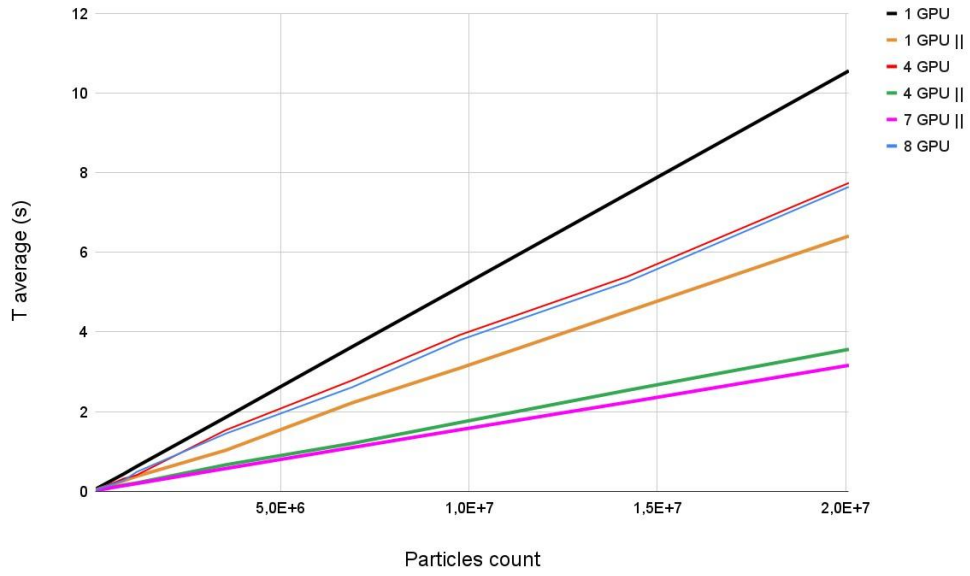

Figure 5. Test results for several GPU computing nodes from 1, 4, 7 and 8 devices. The symbol || indicates tests run with parallel sorting mode

1.5. Conclusion

Test results have shown that the algorithm for distribution of calculations and data can significantly speed up the PCISPH method. In addition, the proposed implementation of the parallel sorting algorithm will also significantly speed up the process of data ordering in comparison with a single-threaded implementation. The obtained results are supposed to be used in the Sibernetic [11], [12] project, which is based on the PCISPH method and is used to model the hydrostatic skeleton of the nematode C. elegans within the framework of the OpenWorm project [13].

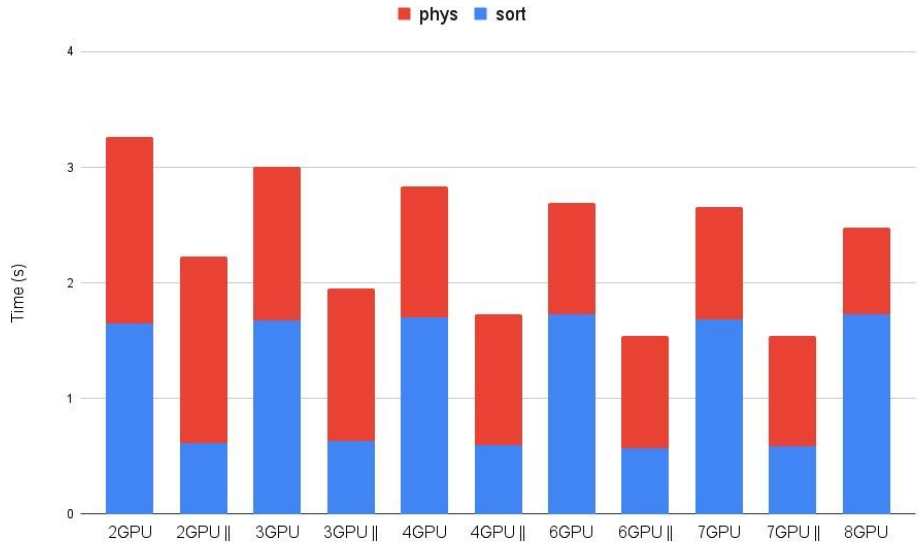

Figure 6. Average time for one iteration for configuration with 9 million particles on various cluster configurations with one thread and parallel sorting

References

[1] Harlow F. H. The particle-in-cell method for numerical solution of problems in fluid dynamics // Proc. Symp. Applied Mathematics. − 1963. − Vol. 15. − P. 269.

[2] Belotserkovskii O. M., Davydov Yu. M. Non-stationary large-particle method for gas-dynamic calculations // Zh. Vychisl. math. and mat. Fiz. − 1971. − Vol. 11. − No.1. − P. 182 – 207 (In Russian).

[3] Gingold R. A., Monaghan J. J. Smoothed particle hydrodynamics: theory and application to non-spherical stars // Monthly Notices of the Royal Astronomical Society. − 1977. − Vol. 181. − Iss. 3. − P. 375–389. − https://doi.org/10.1093/mnras/181.3.375.

[4] Monaghan J. J. An introduction to SPH // Comp. Phys. Comm. −1988. − Vol. 48. −Iss. 1. −P. 89-96. −https://doi.org/10. 1016/0010-4655(88)90026-4.

[5] Solenthaler B. Predictive-corrective incompressible SPH // ACM Transactions on Graphics. − 2009. − Vol. 28. − Iss. 3. − P. 1 – 6 . − https://doi.org/10.1145/1576246.1531346.

[6] Dominguez J.M., Crespo A.J.C. Valdez-Balderas D., Rogers B.D., Gomez-Gesteira M. New multi-GPU implementation for smoothed particle hydrodynamics on heterogeneous clusters // Comp. Phys. Comm.− 2013. − Vol. 184, Iss. 8. − P. 1848−1860. − https://doi.org/10. 1016/j.cpc.2013.03.008.

[7] Verma K., Szewc K., Wille R. Advanced load balancing for SPH simulations on multi-GPU architectures // Proc. IEEE High Performance Extreme Computing Conference (HPEC). − 2017. − P. 1 − 7. − https://doi.org/: 10.1109/HPEC.2017.8091093.

[8] Verma K., Peng C., Szewc K. and Wille R. A Multi-GPU PCISPH implementation with Efficient Memory Transfers //Proc. IEEE High Performance extreme Computing Conference (HPEC). − 2018. − P. 1−7. − https://doi.org/: 10.1109/HPEC.2018.8547542.

[9] Dijkstra E. W. Solution of a problem in concurrent programming control // Proc. Comm. ACM. − 1965. − P. 569. − https://doi.org/:1145365559.365617.

[10] Palyanov A., Khayrulin S., Larson S.D. Application of smoothed particle hydrodynamics to modeling mechanisms of biological tissue // Advances in Engineering Software. − 2016. − Vol. 98. − P. 1 − 11. − https://doi.org/10.1016/j.advengsoft.2016.03.002.

[11] Palyanov A., Khayrulin S., Larson S. D. Three-dimensional simulation of the Caenorhabditis elegans body and muscle cells in liquid and gel environments for behavioural analysis // Philosophical Transactions of the Royal Society B: Biological Sciences. − 2018. − Vol. 373. − Iss. 1758. − https://doi.org/10.1098/rstb.2017.0376.

[12] Sarma G. P., Lee C. W., Portegys T., Ghayoomie V., Jacobs T., Alicea B., Cantarelli M., Currie M., Gerkin R. C., Gingell S., Gleeson P., Gordon R., Hasani R. M., Idili G., Khayrulin S., Lung D., Palyanov A., Watts M., Larson S. D. OpenWorm: Overview and recent advances in integrative biological simulation of Caenorhabditis elegans // Philosophical Transactions of the Royal Society B: Biological Sciences. − 2018. − Vol. 373. − Iss.1758. − https://doi.org/10.1098/rstb.2017.0382