Proceedings of the 9th International Workshop on Ship and Marine Hydrodynamics, 26 – 28 August 2015, Glasgow, UK

GPU Based Acceleration of MPS for 3D Free Surface Flows

Haizhou Li[1,2], Youlin Zhang[1,2] and Decheng Wan[1,2*]

1. State Key Laboratory of Ocean Engineering, School of Naval Architecture, Ocean and Civil Engineering, Shanghai Jiao Tong University,
2. Collaborative Innovation Center for Advanced Ship and Deep-Sea Exploration, Shanghai 200240, China

Abstract: Moving particle semi-implicit (MPS) method is a Lagrangian particle method, first proposed by Koshizuka and Oka for incompressible flows. Due to its Lagrangian nature, MPS is capable of computing large deformed free surface flow, such as violent liquid sloshing, green water and dam breaking. However, MPS suffers from high computational cost. This significantly limits its applications in 3D flows which have to involve a large number of particles. In the present work, a parallel strategy for GPU (Graphics Processor Units) based acceleration is developed, aiming to extend MPS to compute practical 3D flows. Simulation of 3D violent sloshing and 3D dam breaking is performed. Speed-up obtained by using GPU is satisfied. The solution of PPE and search of neighboring particle are accelerated up to 10 times and 6 times faster respectively based on the present parallel strategy, than serial computation. However, performance on different GPU indicates that neighbour list is a memory bound problem, while PPE solving is a compute bound problem.
Keywords: GPU; CUDA; MPS; sloshing; free surface flow

1 Introduction

In recent years, meshless methods have become a research focus as their Lagrangian nature allows them tracing flows with free surfaces largely deformed, which is a challenging task for grid based method. MPS (Moving Particle Semi-implicit) is one such method, which is first developed by S.Koshizuka, Y.Oka (1998). Similar to SPH (Smooth Particle Hydrodynamics), the computational domain is represented by particles, and there is no constant topology between these particles. Therefore, it is quite suitable to deal with largely deformed free surface flow, such as liquid sloshing (Zhang and Wan, 2012a; Zhang and Wan, 2012b), dam breaking (Khayyer and Gotoh, 2012, Shakibaeinia and Jin, 2011), wave breaking (Gotoh and Sakai, 2006, Khayyer and Gotoh, 2008, Tang et *al*, 2014, Zhang et *al*. 2014) , and ship-wave interaction (Shibata et *al*,2012a, Jen Shiang Kouh,2007, Zhang, 2013).

Despite being an excellent method for violent free surface flow, the MPS method still suffer from high computational cost due to its semi-implicit algorithm. Especially when it is applied into 3D free surface flows, a large number of particles are necessary and this can lead to the computational time increasing sharply. During the past decades, parallel computation, on multi-processor workstations or HPC, has been developed. Among the public literatures, most are based on CPU which needs large investments in equipment purchase and installation and a designated space for installing the computers and the related capacity of the cooling system used in the space. However, recently GPU becomes a preferable alternative approach

The GPUs (Graphics Processing Units) are multi-processors, which designed to optimize for the execution of massive number of threads. Until now, many particle researchers attempt to accelerate the computation by GPU based on their meshfree solvers. Harada et *al*. (2007) applied the GPU on SPH, and this is maybe the first implementation on SPH with GPU. Dom ńguez et *al*. (2014) developed an open-source solver on SPH with GPU. However, the literatues on GPU based on MPS is few. Chiemi Hori et *al*. (2011) performed a 2D calculations of elliptical drop evolution and dam break flow with GPUs. Unlike the SPH method, the semi-implicit algorithm is adopted to obtain the pressure field and its consuming time is much more than that in the SPH using the explicit algorithm.

In this paper, Graphics Processor Units (GPUs) is developed in the frame of MPS (Moving Particle Semi-implicit) based on in-house particle solver MLParticle-SJTU. Firstly, MPS and GPU are introduced briefly. Then the process of searching for neighbor particles and the implementation of solving the PPE (Poisson Pressure Equation) are discussed. Finally, the performance between GPU and CPU are analysized. In addition, the numerical results will be validated with experimental data.

2 Methods

2.1 Governing equations

The governing equations are the continuum equation and the Navier-Stokes equations. These equations for incompressible viscous fluid are represented as：

$$\frac{1}{\rho}\frac{D\rho}{Dt} = -\nabla\cdot\boldsymbol{V} = 0 \tag{1}$$

Foundation item: Supported by National Natural Science Foundation of China (Grant Nos. 51379125, 51490675, 11432009, 51411130131)
***Corresponding author Email:** dcwan@sjtu.edu.cn

$$\frac{D\boldsymbol{V}}{Dt} = -\frac{1}{\rho}\nabla P + \nu\nabla^2\boldsymbol{V} + \boldsymbol{g} \tag{2}$$

Where D/D*t* denotes substantial derivative, *t* = time, ***V*** = velocity vector, $\rho = density$, *p* = pressure, ν = kinematic viscosity, and ***g*** = gravity acceleration. Thanks to Lagrangian nature, convection terms do not turn up in the momentum equations, which can avoid numerical diffusion due to the discretization of convection terms.

2.2 Particle Interaction Models

2.2.1 Kernel Function

In particle method, governing equations are transformed to the equations of particle interactions. The particle interactions are based on the kernel function. In traditional MPS method, the kernel function is as following (Koshizuka, 1996):

$$W(r) = \begin{cases} \frac{r_e}{r} - 1 & 0 \le r < r_e \\ 0 & r_e \le r \end{cases} \tag{3}$$

A drawback of the above kernel function is that it becomes singular at r=0. This may cause unreal pressure between two neighboring particles with a small distance, then affect the computational stability. To overcome this, an improved kernel function is employed in this paper (Zhang and Wan, 2011b):

$$W(r) = \begin{cases} \frac{r_e}{0.85r + 0.15r_e} - 1 & 0 \le r < r_e \\ 0 & r_e \le r \end{cases} \tag{4}$$

The above kernel function has a similar form with the original kernel function (Eq.3) except its nonsingularity at *r* =0.

2.2.2 Gradient Model

Gradient operator is modeled as a local weighted average of the gradient vectors between particle *i* and its neighboring particles *j*:

$$<\nabla\Phi>_i = \frac{D}{n^0}\sum_{j\ne i}\frac{\Phi_j + \Phi_i}{|\boldsymbol{r}_j - \boldsymbol{r}_i|^2}(\boldsymbol{r}_j - \boldsymbol{r}_i)\cdot W(|\boldsymbol{r}_j - \boldsymbol{r}_i|) \tag{5}$$

Where: ϕ is an arbitrary scalar function, D is the number of space dimensions, n^0 is the initial particle number density for incompressible flow. The particle number density in MPS method is defined as:

$$<n>_i = \sum_{j\ne i} W(|\boldsymbol{r}_j - \boldsymbol{r}_i|) \tag{6}$$

2.2.3 Laplacian Model

Laplacian operator is derived by Koshizuka et *al*.(1998) from the physical concept of diffusion as：

$$<\nabla^2\phi>_i = \frac{2D}{n^0\lambda}\sum_{j\ne i}(\phi_j - \phi_i)\cdot W(|\boldsymbol{r}_j - \boldsymbol{r}_i|) \tag{7}$$

$$\lambda = \frac{\sum_{j\ne i} W(|\boldsymbol{r}_j - \boldsymbol{r}_i|)\cdot|\boldsymbol{r}_j - \boldsymbol{r}_i|^2}{\sum_{j\ne i} W(|\boldsymbol{r}_j - \boldsymbol{r}_i|)} \tag{8}$$

Where: λ is a parameter, introduced to keep the variance increase equal to that of the analytical solution.

2.2.4 Model of incompressibility

The incompressible condition in traditional MPS method is represented by keeping the particle number density constant. In each time step, there are two stages: first, temporal velocity of particles is calculated based on viscous and gravitational forces, and particles are moved according to the temporal velocity; second, pressure is implicitly calculated by solving a Poisson equation, and the velocity and position of particles are updated according to the obtained pressure.

The pressure Poisson equation in traditional MPS method is defined as (Koshizuka et al., 1998):

$$\frac{2D}{n^0\lambda}\sum_{j\ne i}(P_j - P_i)\cdot W(|\boldsymbol{r}_j - \boldsymbol{r}_i|) = -\frac{\rho}{\Delta t^2}\frac{<n^*>_i - n^0}{n^0} \tag{9}$$

where：n* is the particle number density in temporal field.

The source term of the Poisson equation in Eq. 9 is solely based on the deviation of the temporal particle number density from the initial value. As the particle number density field is not smooth, the pressure obtained from Eq. 9 is prone to oscillate in spatial and temporal domain. To suppress such unphysical oscillation of pressure, Tanaka and Masunaga (2010) proposed a mixed source term for PPE, which combines the velocity divergence and the particle number density. This improved PPE is rewritten by Lee et *al*.(2011) as:

$$<\nabla^2 P^{n+1}>_i = (1-\gamma)\frac{\rho}{\Delta t}\nabla\cdot V_i^* - \gamma\frac{\rho}{\Delta t^2}\frac{<n^*>_i - n^0}{n^0} \tag{10}$$

where：γ is a blending parameter with a value between 0 and 1. The range of $0.01 \le \gamma \le 0.05$ is better according to numerical experiments conducted by Lee et *al*.(2011).

2.2.5 Free Surface boundary condition

In the MPS method, the free surface dynamic condition is enforced by assigning zero pressure for surface particles. By now, some approaches have been developed to detect the free surface particles. Koshizuka and Oka (1996) recognizes the surface particles according to the particle number density. Tanaka & Masunaga (2010) and Lee et *al*. (2011) judge the surface particle using number of neighbor particles. Khayyer *et al*. (2011) proposed a new criteria based on asymmetry of neighboring particles in which particles are judged as surface particles according to the summation of x-coordinate or y-coordinate of particle distance. In the present study, we employ a detection method which is also based on the asymmetry arrangement of neighboring particles, but use different equations, aiming at describing the asymmetry more accurate, as follows(Zhang and Wan, 2011c):

$$< \mathbf{F} >_i = \frac{D}{n^0} \sum_{j \neq i} \frac{1}{|r_i - r_j|} (r_i - r_j) W(r_{ij}) \tag{11}$$

The particle satisfying:

$$< \mathbf{F} >_i > \alpha \tag{12}$$

is considered as surface particle, where α is a parameter, and has a value of 0.9 $|\,\boldsymbol{F}\,|^0$ in this paper, $|\,\boldsymbol{F}\,|^0$ is the initial value of $|\,\boldsymbol{F}\,|$ for surface particle.

3. Implementation

3.1 MPS Algorithm

MPS adopts semi-implicit scheme to achieve temporal integration. The viscous force and gravity are calculated explicitly, while pressure is implicitly obtained by solving so-called PPE (pressure Poisson equation). The algorithm in MPS is shown in Fig. 1.

According to analysis of computational time cost by each procedure, it is found that the particle interaction and PPE solving are most time consuming, which are up to 80% or even more. Therefore, parallelization analysis focus on these two parts in this paper.

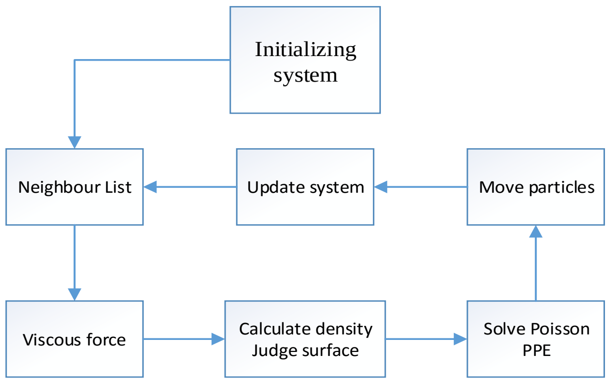


Fig.1 The flow chart of MPS Method

3.2 program on GPU

3.2.1 programming model

Graphics Processing Units (GPUs) have become important in providing processing power for high performance computing applications. Computing on the GPU, or GPGPU, is a steadily maturing technology. The main programming models you can choose for your general computation on GPUs are CUDA or OpenCL.

OpenCL (The Open Computing Language) is an open standard that can be used to program CPUs, GPUs, and other devices from different vendors. Its programming languages is like the C. Although OpenCL promises a portable language for GPU programming, its generality may entail a performance penalty.

CUDA (Compute Unified Device Architecture) is a parallel computing platform and programming model created by NVIDIA and implemented by the graphics processing units (GPUs) that they produce. CUDA includes the programming languages (such as CUDA C/C++, CUDA FORTRAN, CUDA python and other common programming languages extend),the related compilers, and a set of debug and profile tools, and produce good performance.

With a comprehensive consideration, a model of CUDA has been chosen in this paper. And the combination of CPU and GPU implementation is employed in the present work, In other words, the implementation is on partial GPU. Thus, the data exchange between GPU and CPU is required several times during each loop step.

3.2.2 Neighbor List

The calculation fluid domain in MPS method is discretized by a set of initial distribution of particles, which possess material properties. Then the evolution of fluid system is driven by the interaction between particles. During the process, only particles that within a given distance will be calculated in the interactions. Hence, an algorithm used for finding out all the neighbor of each particles should be employed.

As a kind of the simplest method, the brute-force of neighbor search is to find neighbors from all the other particles. Suppose N is the number of particles, the time complexity of brute-force search method will be O(N^2),which is too time consumes. To speed up the search procedure, a common method is to divide the domain into subdomains, which can greatly reduce the search range. The algorithm of neighbor list used in this paper is based on Dominguez et *al*. (2010) and Green (2008).The domain is divided into grid at space of 2h and 4h, where h is the is the distance between two adjacent particles in the initial configuration.

3.2.3 Solving the PPE

Describing pressure Poisson equation Eq.10 can generate a linear system AX = B. Furthermore, the matrix A is a sparse matrix, which most of the elements are zero. To minimize the storage of A matrix, the algorithm of CSR (Compressed row Storage) is applied. As the solver of this linear system, the BI-CGSATB (H.A 1922) method is employed in this paper. In addition, the GPU-accelerated linear algebra libraries of CULA-Sparse is utilized for accelerating on GPU.

4. Verification

In this section, simulations for the verification of GPU-CPU implementation has been discussed. Based on CPU particle solver, MLParticle-SJTU, the implementation optimized for both neighbor list and Poisson PPE solving of using GPU has been carried out. Although the algorithms used in CPU and GPU-CPU are the same, the results maybe not exactly the same. According to Chiemi et *al*. (2011), this mainly due to: the error accumulated by no ECC supported, the multiply-add results might be different in GPU and x86 CPU, the influence of the reliability of compiler and so on.

4.1 Experiment parameters

In this paper, the computational model is a 3-D sloshing case, which described in Chang et *al*.(2013).The experimental configuration is depicted in Fig 2. The length of the tank is L=0.79m, its height is H=0.48m, and its width is W=0.48m. The depth of water is d=0.144 m, corresponding filling level is 30%. The measured Point P is near the free surface, 0.12m from the bottom of the tank. The tank is subjected to a sinusoidal horizontal motion as below:

$$x = -A\cos(\omega t) \tag{13}$$

Where A is the amplitude with the value of 0.0575 m, $\omega = 4.49 rad/s$ is the frequency, which is equal to the first order resonant frequency of fluid motion.

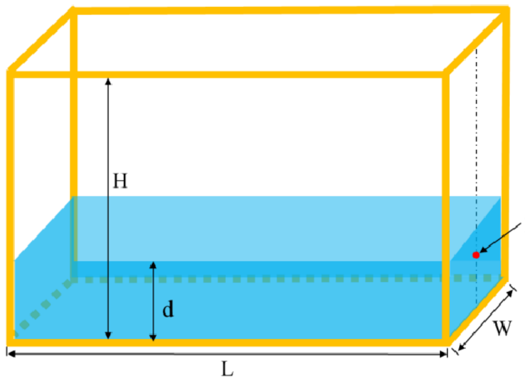


Fig.2 Configuration of sloshing tank

4.2 Simulation parameters

The simulation configurations of MPS used on CPU and GPU-CPU are the same. The total particle number used in this case has reached 660k, which includes 410k fluid particles and 250k boundary particles. The initial particle space is 0.005m, and the time step is 0.00015s. The radius of influence is chosen to be 3.5d (d: diameter of particle) for the Laplacian model and 2.1d for the gradient model. The acceleration of gravity is g=9.81m/s. The density of water is $\rho = 1000 kg/m^3$

4.3 Calculation results

The time histories of impact pressure obtained by experiment and numerical simulations on both CPU and GPU-CPU are compared in Fig. 3. It is seen that the pressure pattern looks like a typical “church roof” profile. Numerical results by both CPU and GPU-CPU are in good agreement with the experimental data, except the pressure peaks which shows discrepency due to unphysical oscillation of pressure field duiring simulation.

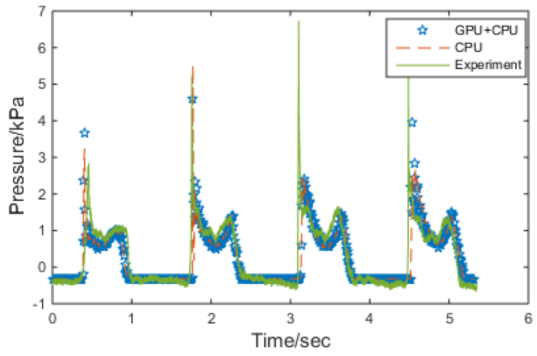


Fig.3 Impact pressure at point P

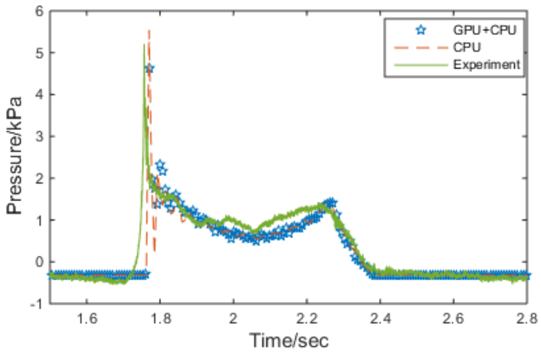
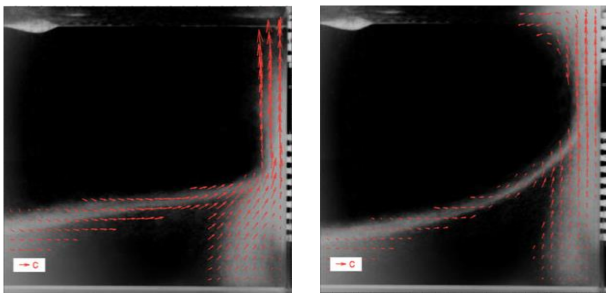


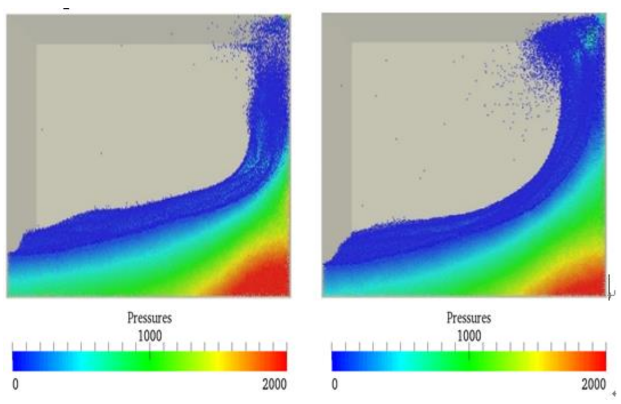
Fig.4 Impact pressure at point P during a period

Figure. 4 shows the details of impact pressure evolution at point P in one period. It can be clearly observed that the numerical results of CPU and GPU-CPU coincide very well.The numerical results are also smoother than experimental data.

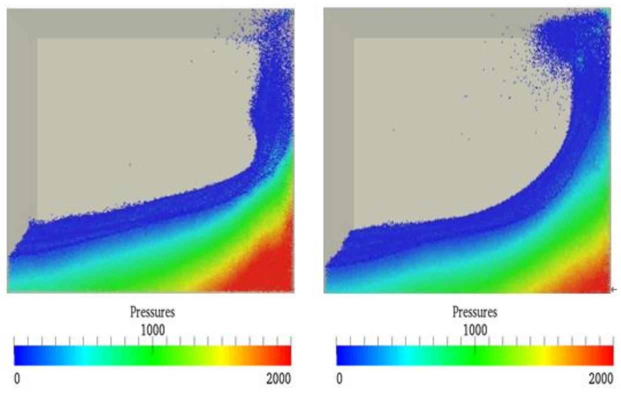
Fig.5 shows snapshots of flow fields obtained by experiment, GPU-CPU and CPU simulation. It is seen the flow is quite violent. Complicated flow phenomena, such as splashing liquid and breaking wave, are observed. However, the shape of free surface computed by GPU-CPU and CPU are in good agreement with experimental observation.



(a) experimental results



(b) results on GPU+CPU



(c) results on CPU

Fig.5 Comparison of snapshots between experiment (upper), simulation on GPU-CPU (middle) and simulation on CPU (lower)

4.4 Comparison of calculation time

In this section, the efficiency of using GPU is analyzed. A CPU of Intel(R) Core(TM) i7-4970k at 4.00 GHz and a GPU of GeForce GTX970 are used in this sloshing case. The driver version of GPU is 347.88 from NVIDIA. More details can be found in table 1.

Table 1 The hardware and software environment

Hardware	Num of cores	Memory	Programming language	Compiler
CPU	4	16GB	C++	VS2013
GPU	1664	4096MB	CUDA C	NVCC-6.5

It should be pointed out that, in this paper, the recorded time of Neighbor List is the kernel time. That is to say, only the solve time on GPU was recorded, although a data transfer between GPU and CPU will be executed later. The CPU version was running on a single core. To analysis the speed up, the executed time of single precision and double precision on CPU implementation and CPU-GPU implementation are estimated.

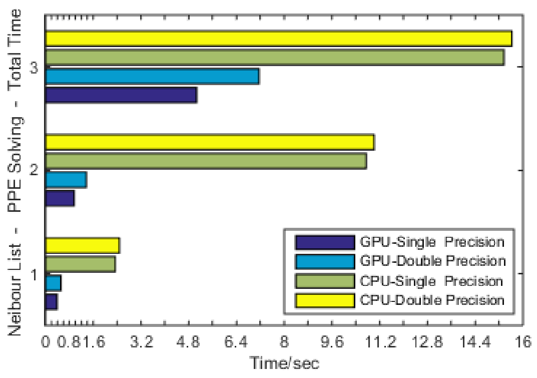


Fig.6 The execute time of different parts

The Fig.6 represents the time of main parts within a loop step. It can be observed that the PPE solving using GPU is more than 10 times faster than CPU, while the neighbor list is about 6 times faster than CPU. However, the total time using GPU+CPU is only 2~3 times faster than CPU, which mainly due to the big amount of data transfer between CPU and GPU when processing neighbor list on GPU. It can be known that an small extra time increase will also make a big influence on speed up, for the total time (except the transfer time) is small too. And the details of this influence and optimization strategy will be studied in somewhere else, but not in this paper.

5. Performance

In this section, the performance on old hardware and new hardware will be discussed. The simulation is on a case of dam breaking with 280k particles, which described in Fig. 7. Using a simpler case with less particles is to ensure that simulation can run properly in old machines. The initial particle space is 0.015m, and the time step is 0.0002s. The radius of influence is chosen to be 4.01d (d: diameter of particle) for the Laplacian model and 2.1d for the gradient model. The acceleration of gravity is g=9.81m/s. The density of water is $\rho = 1000 kg/m^3$.

The i3-2350m and GT555m come from a laptop, and the hardware maybe released about year 2012.The I7-4970k and GTX970 come from a desktop, with the hardware released in 2014. The compute capability of GTX970 is 5.2, while GT555m is 2.1.The compute capability of a device identifies the features supported by the GPU hardware and is used by applications at runtime to determine which hardware features and/or instructions are available on the present GPU. In addition, more details can be found in table 2.

5.1 Dam break results

Fig. 8 shows some snapshots on the moment of liquid impacting on right side wall. The free surfaces overturns and liquid splashes. The flow field computed by GPU are similar with that by CPU in terms of both free surface and pressure.

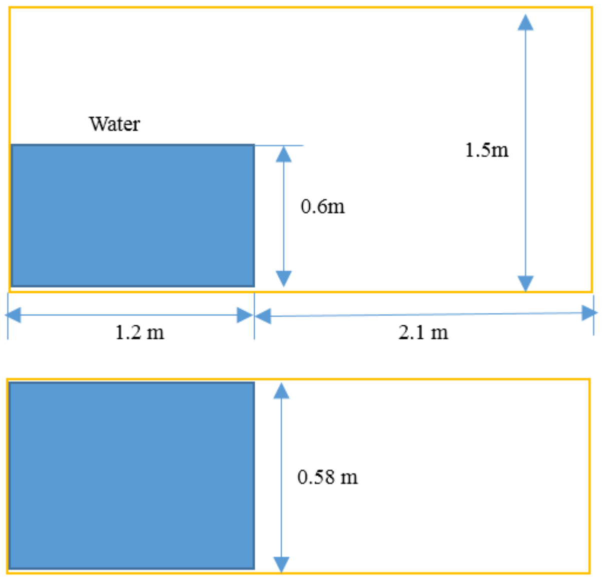


Fig.7 Calculated domain of dam break

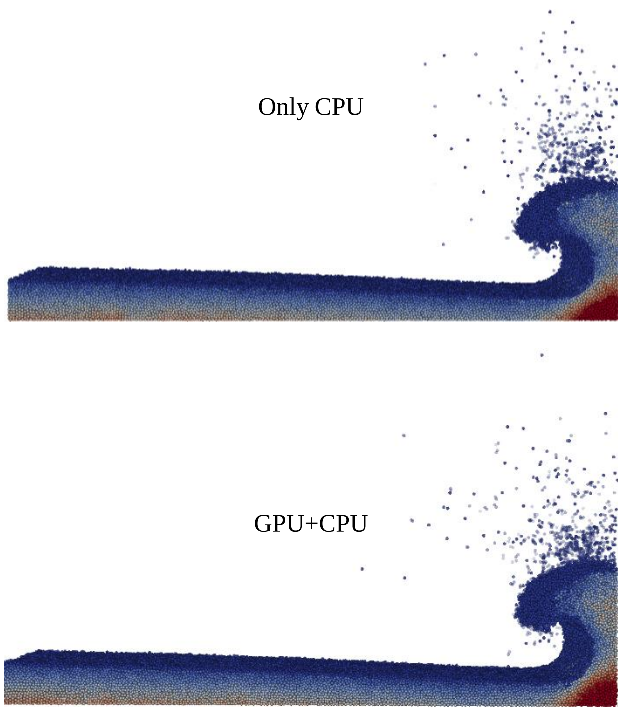


Fig.8 Snapshots of dam break simulations at t = 1.34s.

Table 2 The hardware and software environment

Hardware	Num of cores	Memory	Programming language	Compiler
I7-4970k	4	16GB	C++	VS2013
I3-2350m	2	8GB	C++	VS2013
GT555m	144	2048MB	CUDA C	NVCC-7.0
GTX970	1664	4096MB	CUDA C	NVCC-7.0

5.2 Performance Results

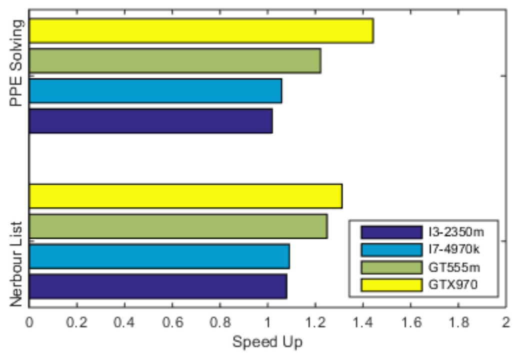


Fig.9 Single precision VS Double precision

In the previous studies of GPU, the performance is usually carried out on single precision. However, double precision might be required to obtain converged results. Thus, an analysis of discussing the double ability versus single precision has been carried out here. Fig.9 shows that the calculation speed of double precision is about the same as single precision in both new CPU and old CPU, which is slightly over 1.0. However, the single precision becomes more obvious faster than double precision on GPU, with a trend of increasing the ratio. The single precision computation ability turns more powerful might mainly due to GPUs are widely used in game industry, which usually employed single precision.

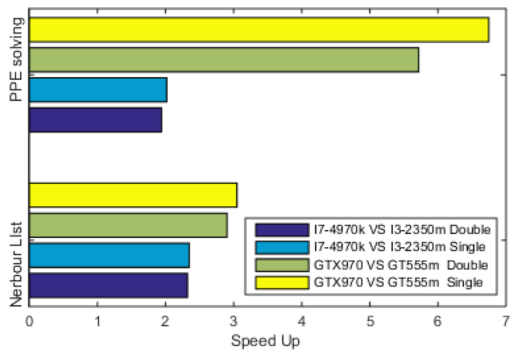


Fig.10 New hardware VS old hardware

The speedups obtain by new hardware and old hardware are discussed here. As depicted in Fig. 10, we can find that the speedup on GPU is faster than CPU, which can indicate that the compute ability of GPU has been grown faster than CPU. In addition, speedups obtained in PPE solving faster than in neighbour list on GPU-CPU. This might mainly due to the PPE solving is a computer-bound problem, while the neighbour list is a memory-bound problem (Herault *et al.*, 2010). Memory bound refers to a situation in which the time to complete a given computational problem is decided primarily by the amount of memory required to hold data. In other words, the limiting factor of solving a given problem is the memory access speed. While limiting factor of compute bound is the number of computer cores and the speed of processor.

6. Conclusion

In this paper, GPU is applied to accelerate MPS computation. Parallel strategy is achieved by using CUDA, based on the frame of MPS. Two main time consuming parts, including searching the neighbour particles and solving the Poisson pressure equation, are discussed. To analyze the parallel efficiency, a violent sloshing flow and dam breaking are carried out. Satisfied speed-up is obtained by using GPU (I3-2350m, GT555m) compared with serial computation on a personal computer with I7-4970k, GTX970, and better speedup is achieved on GPU on new hardware. Moreover, the GPU simulation with single precision produces better acceleration than that with double precision. Detailed analysis shows that neighbour list is a memory bound problem, while Poisson PPE solving is a compute bound problem. Exchanging data between GPU and CPU cost much computation time and this can be optimize in our future works.

Acknowledgement

This work is supported by National Natural Science Foundation of China (Grant Nos. 51379125, 51490675, 11432009, 51411130131), The National Key Basic Research Development Plan (973 Plan) Project of China (Grant No. 2013CB036103), High Technology of Marine Research Project of The Ministry of Industry and Information Technology of China, Chang Jiang Scholars Program (Grant No. T2014099) and the Program for Professor of Special Appointment (Eastern Scholar) at Shanghai Institutions of Higher Learning (Grant No. 2013022), to which the authors are most grateful.

References

A.J.C. Crespo, J.M. Domínguez, B.D. Rogers, M. Gómez-Gesteira, S. Longshaw, R. Canelas, R. Vacondio, A. Barreiro, O. García-Feal (2015), *DualSPHysics: Open-source parallel CFD solver based on Smoothed Particle Hydrodynamics (SPH)*, Computer Physics Communications, **187** , 204-216

Chiemi Hori, Hitoshi Gotoh, Hiroyuki Ikari, Abbas Khayyer (2011), *GPU-acceleration for Moving Particle Semi-Implicit method*, Computers & Fluids, **51**(1), 174-183

Monaghan JJ, Lattanzio JC (1985). *A refined particle method for astrophysical problems*. Astron Astrophys **149**, 135–43.

Green, S. (2008). *CUDA Particles*. Technical Report contained in the CUDA SDK, www.nvidia.com.

Harada T, Koshizuka S, Kawaguchi Y (2007) *Smoothed Particle Hydrodynamics on GPUs.* In Computer Graphics International. 63–70.

H.A. Van Der Vorst. Bi-CGSTAB (1992): *A fast and smoothly converging variant of Bi-CG for the solution of non-symmetric linear systems*. SIAM Journal on scientific and Statistical Computing, **13**(2), 631-644.

Hérault, A., Bilotta, G., & Dalrymple, R. A. (2010). *SPH on GPU with CUDA*. Journal of Hydraulic Research, **48**(S1), 74-79.

Hitoshi Gotoh, Abbas Khayyer, Hiroyuki Ikari, Chiemi Hori (2009). *Refined reproduction of a plunging breaking wave and resultant splash-up by 3D-CMPS method*. Proceeding of the Nineteenth International Offshore and Polar Engineering Conference. Osaka, Japan, June 21-26

Jen Shiang Kouh (2007). *Simulation of a ship with partically filled tanks rolling in waves by applying moving particle semi-implicit method*. International Conference on Engineering Education. Coimbra, Portugal

Koshizuka S, Nobe A, Oka Y (1998). *Numerical analysis of breaking waves using the moving particle semi-implicit method*. International journal for numerical methods in Fluids **26**(7), 751-769

Koshizuka, S, and Oka, Y (1996). *Moving-particle semi-implicit method for fragmentation of incompressible fluid*, Nuclear Science and Engineering, **123**(3), 421-434

Lee, BH, Park, JC, Kim, MH, Hwang, SC (2011).*Step-by-step improvement of MPS method in simulating violent free-surface motions and impact-loads*, Computer methods in applied mechanics and engineering, **200**(9), 1113-1125.

Tanaka, M, and Masunaga, T (2010). *Stabilization and smoothing of pressure in MPS method by Quasi-Compressibility*, Journal of Computational Physics, **229**(11), 4279-4290.

Zhenyuan Tang, Yuxing Zhang, Haizhou Li, Decheng Wan (2014),*Overlapping MPS Method for 2D Free Surface Flows*, The Twenty-fourth International Ocean and Polar Engineering Conference, Busan, Korea, 411-420

Yuxing Zhang, Zhenyuan Tang, Yaqiang Yang, Decheng Wan (2014),*Parallel MPS Method for Three-Dimensional Liquid Sloshing*, The Twenty-fourth International Ocean and Polar Engineering Conference, Busan, Korea, 257-265

Zhang, YX, and Wan, DC (2011b). *Apply MPS Method to Simulate Motion of Floating Body Interacting with Solitary Wave*, In Proceedings of the 7th International Workshop on Hydrodynamics, Shanghai, China, 275-279

Zhang, YX, Wan, DC (2011c). *Application of improved MPS method in sloshing problem*, In Proceedings of the 23rd Chinese Symposium on Hydrodynamics, Xi’an, China