2nd International Conference on the Material Point Method for Modelling Soil-Water-Structure Interaction

Anura3D
MPM Research Community

Paper number: mpm2019.81
www.anura3d.com
www.mpm2019.eu

A large-scale parallelised material point method system

Youkou, Dong[1,2,*], Lin, Mu[1,2], Dong, Wang[3]
[1]College of Marine Science and Technology, China University of Geosciences, 388 Lumo Road, Wuhan 430074, China
[2]Shenzhen Research Institute, China University of Geosciences, Shenzhen 518057, China
[3]Shandong Provincial Key Laboratory of Marine Environment and Geological Engineering, Ocean University of China, 238 Songling Street, Qingdao 266100, China

[*] E-mail: dongyk@cug.edu.cn

ABSTRACT

The material point method (MPM) suffers from high computational intensity in simulating large deformation problems. This paper presents a parallel computing strategy for the MPM with multiple Graphics Processing Units (GPUs) to boost the method's computational efficiency. Domain decomposition method is used to split the workload over subdomains onto a number of GPUs. On each GPU the MPM algorithm is parallelised over nodes or particles. Benchmark problem of deep penetration of T-bar is analysed to quantify the speedup of the 2-GPU parallel simulations over the sequential counterparts on the central processing unit. The maximum speedup with 2 GPUs is 120.

KEY WORDS: material point method, parallel computing, large deformation, T-bar

INTRODUCTION

The numerical simulation of large deformation problems, such as the impact of submarine landslides on pipelines (Dong et al., 2017) and the free-falling of torpedo anchors in water and soil (Kim et al., 2015), is technically challenging in mesh entanglement for the traditional mesh-based methods. The material point method (MPM) developed recently can be regarded as a combination of the finite element and meshfree methods, remaining acceptable balance between the computational cost and accuracy for large deformation analyses. However, large-scale problems, such as cone penetration test with penetration depth of more than 20 times the diameter of the cone (Fan et al., 2018), challenge the processing capabilities and memory of existing computing systems. It is well-recognized that an efficient parallel system is necessary to extend the applications of the MPM (Dong et al., 2015; Dong et al., 2017), which requires many processors with dedicated memories working together on a specific problem. In this paper, an efficient cluster architecture featuring multiple-GPU parallelisation for the MPM is preliminarily developed using Message Passing Interface (MPI) and Compute Unified Device Architecture (CUDA). Domain decomposition is performed with the MPI to distribute the workload over GPUs in terms of subdomains. Intercommunication between the GPUs is implemented for the neighbouring nodes and the particles moving out of the original subdomains. A benchmark case, deep penetration of T-bar, is used to evaluate the performance of the multiple-GPU parallel strategy. Acceleration of the multiple-GPU parallel simulation over the CPU sequential counterpart is quantified in terms of speedup.

PARALLELISATION OF MPM

MPM-GeoFlow, an in-house MPM program, was developed on the basis of the open-source package Uintah (Guilkey et al., 2012) with enhancement of a contact algorithm 'Geo-contact' (Ma et al., 2014) and GPU parallel computing strategies (Dong et al., 2015; Dong et al., 2017; Dong and Grabe, 2018). The explicit updated Lagrangian calculation is based on the generalised interpolation material point method presented by Bardenhagen and Kober (2004). The definitions of the stresses and strains follow finite strain theory taking account of incremental rotation of the configurations between time steps for objectivity: the stresses are measured with the Cauchy stress and updated with the Jaumann rate, and the strains are calculated with the logarithmic strain and updated with the deformation rate. Each incremental step is divided into functions: initialisation of nodal variables, interpolation from particles to nodes, calculate nodal velocities and accelerations, and update state of particles.

Domain decomposition

Domain decomposition method is used with the MPI to distribute the entire workload onto GPUs by dividing the computational domain geometrically into a number of subdomains, which includes the particles inside and the corresponding background mesh. The parallelised MPM algorithm for each subdomain is essentially calculated on a GPU with the CUDA, while some trivial operations (such as updating the particle list of the nodes) on CPU is also required. Intercommunication between the GPUs by the functions 'Interpolation from particles to nodes' and 'Update state of particles' is implemented with a MPI send/recv communication in four steps: i) read the information, i.e. mass, momentum, and internal force, of the neighbouring nodes on the global memory of the local GPU; ii) copy to the local Random Access Memory (RAM) with CUDA via PCI-Express; iii) copy to the target RAM with MPI send/recv communication via network card; iv) write to the global memory of the target GPU with CUDA via PCI-Express. To balance the workloads between the GPUs, the subdomain sizes are updated periodically at intervals of a large number of time steps, for example 50,000; accordingly a massive number of particles need to be relocated among the GPUs.

GPU parallelisation

To maximise the speedup of GPU parallelisation, all the functions are programmed to be parallelised on the GPU (Figure 1). In the function 'Interpolation from particles to nodes' the variables are interpolated from each individual particle to the related nodes. To avoid the potential data race induced by concurrently writing the interpolated variables to the common nodes, the function is parallelised across GPU cores over the nodes, which 'gathers' the interpolations from the associated particles to the node sequentially. Prior to parallel computing, a list of particles related to each node is established for the subdomain on each CPU and then saved to the global memory of the local GPU. Since the particles will flow through the fixed mesh between time steps, the particle lists of the nodes are updated periodically at intervals of a number of time steps. The particle list is expanded by covering not only the particles located inside the influence zone of the common node, but also the nearest particles. It takes a number of time steps for soil particles to flow in or out of the 'expanded' zone.

The other functions are more straightforward to be parallelised over nodes or particles. The function 'Calculate nodal velocities and accelerations' is parallelised over the nodes, i.e. the update of velocities and accelerations of each node is scheduled into a GPU core. The information required such as masses and momenta is in terms of the particular node in each core, without communication with any other cores. There are therefore no risks of data race. The workload of the functions 'Update state of particles' is decomposed over the particles, with the stresses and movements of each particle calculated in a core. In each core, the interpolated components, such as the deformation rates, velocities and accelerations, are superimposed from the related nodes one by one.

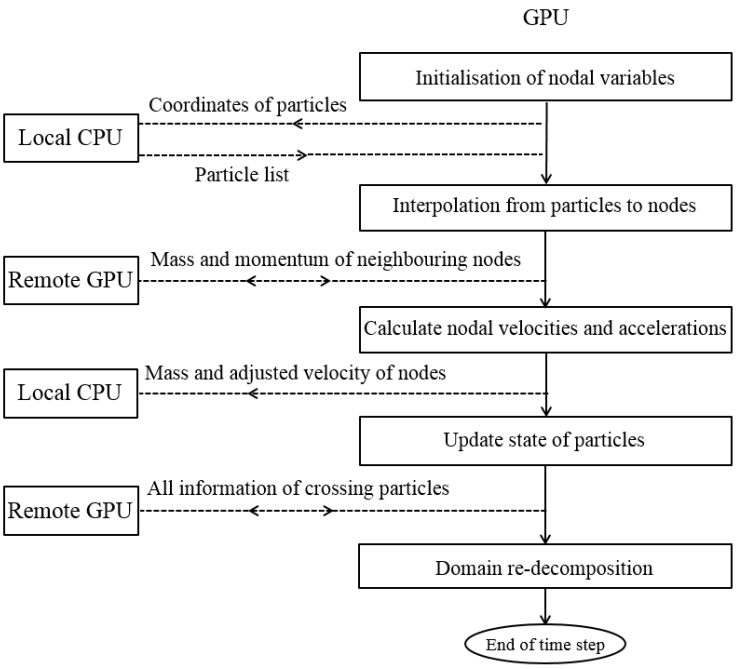

Figure 1 Operations in parallelisation of MPM

BENCHMARKS

The efficient computing cluster was composed of 2 member computers, while each was equipped with an Intel Core i7-6850K

CPU featuring 6 cores and an NVIDIA Titan Xp GPU featuring 3840 cores along with 12 GB of global memory. The two member computers were connected with a switcher, which has a bandwidth of 1000 Mbps. The cluster would be expanded with 5 member computers with 3 NVIDIA Titan Xp GPUs on each computer in the following work. For comparison purposes, the benchmark cases were simulated with single CPU core, 12 CPU cores (OpenMP) and 2 GPUs (MPI-CUDA), respectively. The maximum number of particles is restricted by the size of the memory. Through trial calculations, the global memory of 24 GB on the two GPUs allows for the simulation of ~ 36 million particles. All the computational efforts investigated was based on double-precision to guarantee the accuracy of the simulations. A 'speedup' factor is used to quantify the effect of parallel computing, expressed as

$$\text{Speedup} = T_{\text{Sequential}} / T_{\text{Parallel}} \tag{1}$$

where $T_{\text{Sequential}}$ and T_{Parallel} represent the average runtimes of CPU sequential and CPU or GPU parallel calculations per increment.

Figure 2 Setup of computing cluster

Deep penetration of T-bar, a slender-shafted cylinder penetrometer used to measure the strength of clay (Stewart and Randolph, 1994), was simulated to assess the performance of the parallel framework. The diameter of the smooth T-bar was D = 0.04 m. The soil extensions on the horizontal and vertical directions were $L = 4D$ and $H = 8D$, respectively (Figure 3). The mesh size was d = 0.025D. The penetration speed of the T-bar was taken as 0.1D /s. A 4 × 4 particle configuration was allocated for each element fully occupied by particles prior to the calculation. The total number of the particles was 808,114. The submerged density of the soil was ρ = 650 kg/m^3. The geostatic stresses induced by the self-weight of soil were not considered. The undrained strength uniform clay s_u = 2.5 kPa. The Poisson's ratio of soil was taken as 0.49 to approximate constant volume under undrained conditions. The Young's modulus was taken as 500 times the undrained shear strength. Roller boundary conditions were assigned to the lateral boundaries and base of the soil. The time step Δt was determined with a Courant number α of 0.3

$$\Delta t = \frac{\alpha d}{\sqrt{(\lambda + 2G)/\rho}} \tag{2}$$

where G and λ are the Lamé parameters. The normalised capacity factor at the full flow of the surrounding soils is obtained as 9.58, which has a good agreement with upper bound analysis result of 9.14 (Martin and Randolph, 2006).

The horizontal size L was deliberately extended from 4D to 16D, 48D, 96D and 160D to enlarge the computational scale, with corresponding particle numbers N_p increased from 808,114 to 32,756,914. The vertical soil dimension remained unchanged. The average runtime in 50,000 incremental steps was calculated, covering all the possible costs during the entire simulation. The speedups of the 12 CPU cores and 2-GPU parallel simulations over the CPU sequential counterparts are shown in Table 1. For the CPU parallel simulations, the speedups are stable with the increase of particle numbers, which is ~ 7. This is

consistent with Dong and Grabe (2018) and slightly higher than Zhang et al. (2010). In comparison, the speedup of the GPU parallelisation increases with the increase of particle numbers, from 105 for 808,114 particles to 122 for 32,756,914 particles. And it seems the speedup of the GPU parallelisation becomes stable for particle number larger than 10 million, which means the GPU cores are fully loaded.

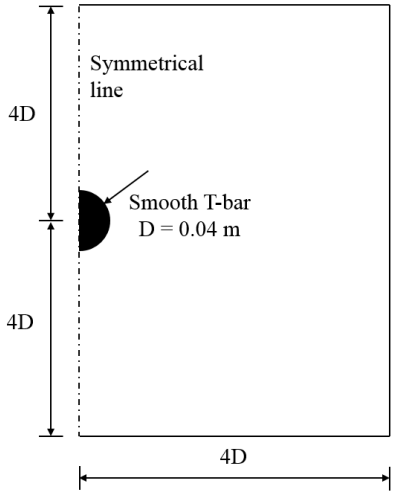

Figure 3 Schematic of T-bar penetration (not to scale)

Table 1 Speedups of CPU and GPU parallel simulations

L	Number of particles	Speedup	
		12 CPU cores	2 GPUs
4*D*	808,114	7.2	105
16*D*	3,265,714	7.1	114
48*D*	9,819,314	7.4	120
96*D*	19,649,714	6.8	123
160*D*	32,756,914	6.9	122

CONCLUSIONS

The material point method (MPM) is a computationally intensive form of finite element analysis that suffers from limitations associated with CPU and single-GPU performance. Efficient framework of multiple-GPU parallelisation offers a platform for the MPM in solving larger deformation problems. This paper presents an efficient cluster architecture featuring multiple-GPU parallelisation for the MPM using Message Passing Interface (MPI) and Compute Unified Device Architecture (CUDA). Domain decomposition is performed with the MPI to distribute the workload over GPUs in terms of subdomains. Intercommunication between the GPUs is implemented for the neighbouring nodes and the particles moving out of the original subdomains. Compared with the sequential CPU simulation, significant speedups are presented in the example application of deep penetration of T-bar. The maximum speedup with 2 GPUs is 120, which is much higher than that ~ 7 for the parallel calculations with 12 CPU cores.

ACKNOWLEDGEMENTS

This work is jointly supported by the National Key Research and Development Program of China (No. 2017YFC1404700), th e Discipline Layout Project for Basic Research of Shenzhen Science and Technology Innovation Committee (No. 20170418), the Guangdong Special Fund Program for Marine Economy Development (No. GDME-2018E001).
This work is also supported by NVIDIA Corporation with the donation of the GPU Geforce Titan Xp for this research.
The first author would also like to acknowledge the value of suggestions by Prof. Jürgen Grabe and Dr. Hans Stanford through personal communications.

REFERENCES

Bardenhagen, S. G., Kober, E. M. (2004). The generalized interpolation material point method. Computer Model Engineering and Science, 5(6), 477-96.
Dong, Y., Wang, D., and Randolph, M. F. (2015). A GPU parallel computing strategy for the material point method. Computers and Geotechnics, 66: 31-38.
Dong, Y., Wang, D., and Randolph, M. F. (2017). Investigating of impact forces on pipeline by submarine landslide using material point method. Ocean Engineering, 146: 21-28.

Dong, Y. and Grabe, J. (2018). Large scale parallelisation of the material point method with multiple GPUs. Computers and Geotechnics, 101, 149-158.
Fan, S., Bienen, B., and Randolph, M. F. (2018). Stability and efficiency considerations in the numerical simulation of cone penetration testing in sand. Géotechnique Letters, 8(1): 1-25.
Guilkey J, Harman T, Luitjens J. et al. (2012). Uintah code (Version 1.5.0). Computer program; available at <http://www.uintah.utah.edu>.
Kim, Y. H., Hossain, M. S., Wang, D. and Randolph, M. F. (2015). Numerical investigation of dynamic installation of torpedo anchors in clay. Ocean Engineering, 108: 820-832.
Ma, J., Wang, D., and Randolph, M. F. (2014). A new contact algorithm in the material point method for geotechnical simulations. International Journal for Numerical and Analytical Methods in Geomechanics, 38(11): 1197-1210.
Martin, C. M., & Randolph, M. F. (2006). Upper-bound analysis of lateral pile capacity in cohesive soil. Géotechnique, *56*(2), 141–145.
Stewart, D. P., & Randolph, M. F. (1994). T-bar penetration testing in soft clay. Journal of Geotechnical Engineering, 120(12), 2230–2235.
Zhang Y, Zhang X and Liu Y. (2010). An alternated grid updating parallel algorithm for material point method using OpenMP. Computer Modeling in Engineering & Sciences, 69(2):143–165.