Proceedings of the Seventh International Conference on
Parallel, Distributed, GPU and Cloud Computing for Engineering
Edited by: P. Iványi, F. Magoulès and B.H.V. Topping
Civil-Comp Conferences, Volume 4, Paper 4.1
Civil-Comp Press, Edinburgh, United Kingdom, 2023
doi: 10.4203/ccc.4.4.1

Hybrid Parallelization of Discrete Element Software for Heterogeneous Resources

O. Bystrov, R. Pacevič and A. Kačeniauskas

**Department of Graphical Systems,
Vilnius Gediminas Technical University,
Lithuania**

Abstract

This paper presents the hybrid parallelization of DEM software developed by using OpenCL for shared-memory architectures and MPI for distributed-memory heterogeneous resources. Resource-aware partitioning based on the weighted RCB method adapts computational workload to heterogeneous Docker containers of the OpenStack cloud. The execution time of benchmark on 7 heterogeneous containers, including the container equipped by GPU, is reduced up to 32.6% of the execution time obtained by using unweighted repartitioning. The speedup of parallel computations up to 6.0 is measured on 8 heterogeneous containers. The replacement of 3 faster containers by 3 slower ones slightly decreases the speedup up to 7.4% of the speedup measured on 5 homogeneous containers.

Keywords: hybrid parallelization, discrete element method, resource-aware partitioning, heterogeneous cloud resources, OpenCL, MPI.

1 Introduction

The discrete element method (DEM) [1] is widely used when materials require to be simulated at the level of individual particles. However, the simulation of media at the particle level of detail has the disadvantage of making DEM computationally very expensive. Naturally, to solve the industrial-scale problems, parallel computing on heterogeneous cloud resources is an obvious choice to increase computational capabilities. The intentions to reduce memory consumption, improve load balancing and better utilize available multicore resources motivated researchers to develop

hybrid parallelization of DEM software. Henty [2] implemented a hybrid parallelization of the message-passing and shared-memory models for spherical particles, but the pure MPI code was always more efficient than a hybrid scheme. The LIGGGHTS DEM software [3] employed uses a Cartesian grid of subdomains with a recursive multi-sectioning algorithm for global domain decomposition and RCB method [4] for defining subsets of particles assigned to threads. Incardona et al. [5] demonstrated the scalable framework OpenFPM for shared-memory and distributed-memory implementations of particle and particle-mesh codes. Yan and Regueiro [6] examined the hybrid MPI/OpenMP mapping schemes and influences of memory/cache hierarchy for 3D DEM simulations. However, pure MPI implementation achieved higher efficiently than hybrid MPI/OpenMP software. In the discussed research, only OpenMP was used for shared-memory programming.

GPU has a higher parallel structure, which makes them very efficient for particle-based computations. A few efforts have been made to use the combined GPU and MPI technology [7]. However, the communication overhead among GPUs significantly reduces the parallel performance because of the costly data transfer to the CPU memory and the MPI communication among different nodes. It is worth noting that only CUDA was employed for shared-memory programming on GPU together with MPI technology for distributed-memory communications in the case of DEM software. Very few attempts to exploit low-cost cloud resources for computationally demanding DEM software have been reported in the academic literature [8]. However, heterogeneity of cloud resources was not considered. This paper presents the hybrid MPI/OpenCL parallelization of DEM software, employing resource-aware partitioning for heterogeneous cloud resources.

2 Hybrid parallelization of DEM software

Hybrid parallelization of DEM software is developed to exploit the potential of different types of memory and to simplify mapping of subsets of particles to heterogeneous multi-core cloud resources. In the present research, the discrete element model for granular flows of the non-cohesive frictional visco-elastic spherical particles is considered. The dynamic behaviour of a discrete system is described by the motion and deformation of the interacting individual particles within the framework of Newtonian mechanics. An arbitrary particle undergoes motion characterized by three translational and three rotational degrees of freedom. The details of the governing equations, implemented model and conventional DEM procedures can be found in [9].

2.1 Implementation of heterogeneous resource-aware partitioning

Partitioning well known as domain decomposition is implemented in the developed DEM software as efficient coarse grain parallelization strategy for distributed-memory architectures. The RCB method [4] from the Zoltan library [10] is used for partitioning particles into subsets because it is highly effective for particle simulations. Heterogeneous resources, such us containers of highly different computing performance, can cause substantial load imbalance of the parallel software.

The percentage load imbalance measure λ quantifies the uneven distribution of computational load by using the following formula:

$$\lambda = \left(\frac{L_{\max}}{L_{\text{avg}}} - 1 \right) \cdot 100\% \tag{1}$$

where L_{avg} is the averaged load over all processes and $L_{\max}$ denotes the largest load. The time consumed by computational procedures is almost the exact measure of the computational load. For resource-aware partitioning, the load is internally monitored measuring the computing time by timers implemented in DEM procedures.

Heterogeneity of resources is specified by different values of weights that result in subsets of different size after repartitioning. New weight W_i^{new} is computed according to runtime measured computational load Li of parallel process i:

$$W_i^{\text{new}} = W_i^{\text{old}} \cdot \left(2 - \frac{L_i}{L_{\text{avg}}} \right) \tag{2}$$

where W_i^{old} is previous weight of process *i*. It is worth noting that these weights also consider variations of application load and system load on virtualized hardware. Thus, the previous weights help reducing significant oscillations that often occurs in dynamic simulations. Finally, RCB-based repartitioning adapts subsets of particles to heterogeneous resources according to runtime measured load of application-specific computations.

All computations are performed in the time loop. At the beginning of the time step, the computational load is measured for resource-aware partitioning. Then, repartitioning of particles into subsets is performed if one of two conditions is satisfied. Commonly, repartitioning is performed if load imbalance exceeds the predefined value. In the present research, more sophisticated approach is developed according to specific needs of DEM computations. A small portion of communication is required when MPI processes exchange particles as the particles move from one subset to another. This communication is optional and should be performed only in the case of a nonzero number of exchanging particles. However, information exchange is necessary in each time step. Despite its local character, information exchange and internode particle data transfer reduce the parallel efficiency of computations. Therefore, this particle exchange is performed only during repartitioning procedure. Moreover, skinning technique is employed to avoid frequent repartitioning. To make contact with particle from another subset, internal particle needs time to cross the ghost layer, which can be estimated [11]. Thus, sometimes repartitioning should be performed with frequency defined by granular flow physics despite low workload imbalance. Therefore, the first condition based on load imbalance is supplemented by the second condition based on the frequency defined by application physics. The subroutines of Zoltan library perform parallel repartitioning for distributed-memory architectures. After repartitioning, migration of particles from old partition to a new one should be handled. New ghost layers are

defined as well as ghost particles are registered. At the end of repartitioning code, data of internal particles are transferred from the host memory to the OpenCL device memory for main shared-memory computations. The main communications, exchanging data of ghost particles between neighbouring partitions, are performed before the main DEM computations. The main DEM procedures are carried out by OpenCL kernels on shared-memory multicore nodes. At the end of the time step, the results can be copied from the OpenCL device memory to the host memory and stored on the hard disk drive.

2.2 OpenCL kernels

The main computational procedures of DEM are implemented by using OpenCL kernels. Main kernels are performed on the thread per particle basis, which takes advantage of the massive parallel computation capabilities of modern hardware and can be considered to be the most suitable parallelism in the case of DEM computations. The parallel algorithm for shared-memory architectures can be outlined as follows. Kernel 1 performs the contact search by using the standard uniform grid method. The output of the kernel contains the contact list. Kernel 2 handles the contact history for friction force computations. The kernel maps the contacts of the previous time step to the newly detected contacts of the current time step. Kernel 3 computes the contact forces and the moments between all overlapped particles. Kernel 3 can be treated as the main kernel because it performs the largest part of computations on a thread per particle basis. Kernel 4 is aimed at computing the boundary conditions and the external gravitational force. The last Kernel 5 performs the time integration by using the explicit velocity Verlet algorithm [11].

2.3 MPI communication

After repartitioning performed by RCB method, some particles migrate from old partition to a new one. Redistribution of particles requires internode communication handled by MPI. The RCB method is attractive as a dynamic load balancing algorithm because it implicitly produces incremental partitions and limits particle transfer among nodes caused by repartitioning. At the end of repartitioning, MPI processes exchange information of ghost particles, which is necessary for internode communications performed before the main DEM procedures in each time step.

On shared-memory multi-core node each OpenCL device performs computations only on its subset of particles. However, it needs to share information with OpenCL devices, working on other nodes, about particles that are near the division boundaries in ghost layers. This internode communication is implemented by using MPI subroutines. The main portion of communications is performed prior to the main DEM computations. Data of ghost particles are exchanged between neighbouring partitions by MPI. The exchange of positions and velocities of particles is a common strategy often used in DEM codes [3] because it allows nodes independently to perform contact search and computation of forces. It is worth noting that data of ghost particles, transferred by MPI from other nodes at each time step, also should be copied to the OpenCL device memory, which makes internode data exchange very expensive.

3 Performance Analysis

A gravity packing problem was solved on heterogeneous containers to measure the parallel performance of the DEM software based on the hybrid MPI/OpenCL implementation. The gravity packing problem was considered because it was commonly employed as a performance benchmark [3], [8]. The solution domain was assumed to be a cubic container with the 1.0m-long edges. Granular material, falling under the influence of gravity, was presented by an assembly of 1000188 and 4000000 monosized particles. Half of the container was divided into cubic cells with particles embedded into the cell centres. The initial velocities of the particles are defined randomly, having their magnitudes over the range of [0.0; 0.1] m/s. 10000 time steps equal to 1.0×10^{-8} s were performed, which resulted in the physical time interval of 0.0001 s. In the case of the considered benchmark, movement of particles with skinning technique required to repartition subsets of particles at each 1000 time steps.

3.1 Cloud resources

The private cloud is built by using OpenStack Train 2019 version [12]. Compute service Nova for virtual machines, compute service Zun for containers, image service Glance, networking service Neutron, container network plugin Kuryr, block storage service Cinder and identity service Keystone are deployed on the OpenStack cloud. The parallel DEM software as a service is deployed on top of the provided platforms, such as GNU compilers, OpenCL, Open MPI and the Zoltan library.

	Cores	Architecture	RAM, [GB]	HDD, [TB]
CN-6700-4	4	i7-6700	16	0.5
CN-4790-4	4	i7-4790	16	0.5
CN-E5-20	20	E5-2630	16	0.5
CN-GPU	1792 (CUDA)	Tesla™ P100	12	0.5

Table 1: Characteristics of containers.

The cloud infrastructure is composed of OpenStack service nodes and compute nodes connected to 1 Gbps Ethernet LAN. Hardware characteristics of faster compute nodes hosting the containers are listed below: Intel®Core i7-6700 3.40 GHz CPU, 32 GB DDR4 2133 MHz RAM, 34.13 GB/s memory bandwidth, and 1 TB HDD. Hardware characteristics of slower nodes are listed below: Intel®Core i7-4790 3.60 GHz CPU, 32 GB DDR3 1866 MHz RAM, 29.87 GB/s memory bandwidth and 1 TB HDD. The NVIDIA® Tesla™ P100 GPU (1792 FP64 CUDA Cores, 12GB HBM2, 549GB/s memory bandwidth) is installed on the node with following hardware characteristics: Intel®Xeon™ E5-2630 2.20GHz 2xCPU, 32GB DDR4 2133MHz RAM, 2x34.13 GB/s memory bandwidth and 1 TB HDD. Characteristics of containers are provided in Table 1. Docker version 20.10.7 containers managed by Zun are used in the cloud infrastructure. Ubuntu 20.04.3 LTS (Focal Fossa) is installed in the containers. The container CN-GPU provides access to NVIDIA® Tesla™ P100 GPU.

3.2 Parallel performance

The parallel performance analysis is based on the speedup gained relative to a sequential run of DEM software on cloud containers CN-6700-4 and CN-4790-4. Figure 1 shows the parallel speedup as a function of the number of containers for the benchmarks of 1000188 and 4000000 particles. The curve called Ideal illustrates the ideal speedup. The curves with abbreviations HOM5, HET2+3 and HET5+3 shows the speedup obtained on 5 homogeneous containers CN-6700-4, 2 faster containers CN-6700-4 supplemented by 3 slower containers CN-4790-4 and 5 faster containers CN-6700-4 supplemented by 3 slower containers CN-4790-4, respectively. The abbreviations "1M" and "4M" represent the benchmarks of 1000188 and 4000000 particles, respectively.

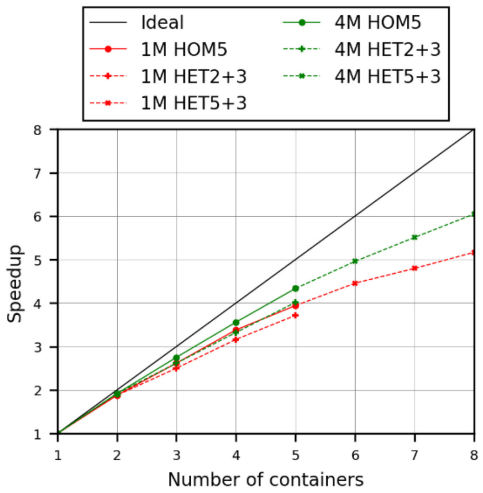

Figure 1: Parallel speedup obtained on heterogenous resources.

The parallel speedup is largely determined by the ratio of local OpenCL computations over internode MPI communications. As the number of containers increases, for a fixed problem size, the communication cost will eventually become dominant over the local computation cost after a certain stage. This high ratio of communication to computation makes the influence of a further reduction in the local computation on the overall cost of running the application very small. The relative communication cost is lower for larger benchmark of 4000000 particles than that for smaller benchmark of 1000188 particles, therefore, higher speedup is observed for larger benchmark. Speedups equal to 5.2 and 6.0 were measured, solving benchmarks of 1000188 and 4000000 particles on 8 heterogeneous containers, respectively. The obtained speedup values were close to those attained for relevant numbers of homogeneous nodes in other performance studies of DEM software based on the hybrid parallelization [3]. It is worth noting that hybrid parallelization eliminated issues related to mapping of particle subsets to multicore nodes and improved speedup obtained on uneven number of nodes [8]. The replacement of 3 faster containers by 3 slower ones slightly decreased the speedup measured on 5 containers from 3.94 to

3.72 and from 4.34 to 4.02 for benchmarks of 1000188 and 4000000 particles, respectively. The observed decrease did no exceed 7.4% of the speedup measured on 5 homogeneous containers. In the case of smaller benchmarks, execution times obtained by using weighted repartitioning were nearly equal to those attained by using unweighted repartitioning. In the case of larger benchmark containing 4000000 particles, execution times of weighted repartitioning were shorter, but the observed difference did not exceed 1.7% of the execution time of weighted repartitioning. The percentage load imbalance of computations without weighting was low and varied from 9.6% to 12.6%. Thus, low heterogeneity of resources indicated by low load imbalance percentage limited the gain of weighted repartitioning.

3.3 Heterogeneous resource-aware partitioning

The ability of the developed DEM software to repartition subsets of particles according to heterogeneous resources is presented in this subsection. Five containers CN-6700-4 were supplemented by two powerful containers CN-E5-20 and CN-GPU in the considered benchmark. Figure 2 shows time evolution of the execution time (Figure 2a) and load imbalance (Figure 2b). The dashed curves represent results of resource-aware partitioning with variable weights, while the solid lines represent that of unweighted repartitioning, which leads to partitions of nearly equal size. It can be observed that the percentage load imbalance measure of computations with unweighted repartitioning approximately equals to 20%, which is higher than in the previous case of eight heterogeneous containers. Therefore, the execution time was reduced up to 32.6% of the execution time obtained by using unweighted repartitioning. Thus, higher heterogeneity of resources indicated by higher load imbalance percentage increased the gain of runtime resource-aware partitioning. Generally, on one node the employed GPU performs DEM computations significantly faster than the CPU [13], therefore, higher load imbalance percentage as well as gain in execution time can be expected. Thus, detailed investigation of computational load, weights and communication issues is required.

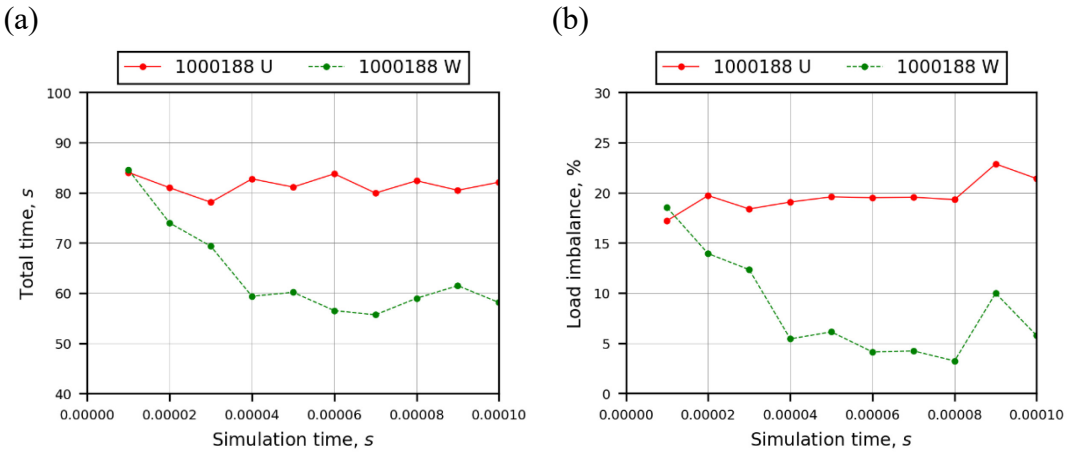

Figure 2: Time evolution of the execution time (a) and load imbalance (b).

Figure 3 presents time evolution of runtime measured computational load and weights. Initially, the load of any container CN-6700-4 was approximately 3.5 times higher than that of the container CN-GPU equipped by GPU (Figure 3a). The difference in computational loads of containers CN-E5-20 and CN-GPU was lower. The computational load of container CN-E5-20 was 2.5 times higher than that of the container CN-GPU. The differences of computational loads were relatively higher than the percentage load imbalance because the formula (1) compares the maximal load with the averaged load. The load of any slower container serves as the maximal load, while the averaged load is highly influenced by five slower containers CN-6700-4 and reduced by two powerful containers CN-E5-20 and CN-GPU. After several applications of resource-aware partitioning, all curves of computational load approached the average. Some variation of CN-GPU load around the average can be observed because of communication issues.

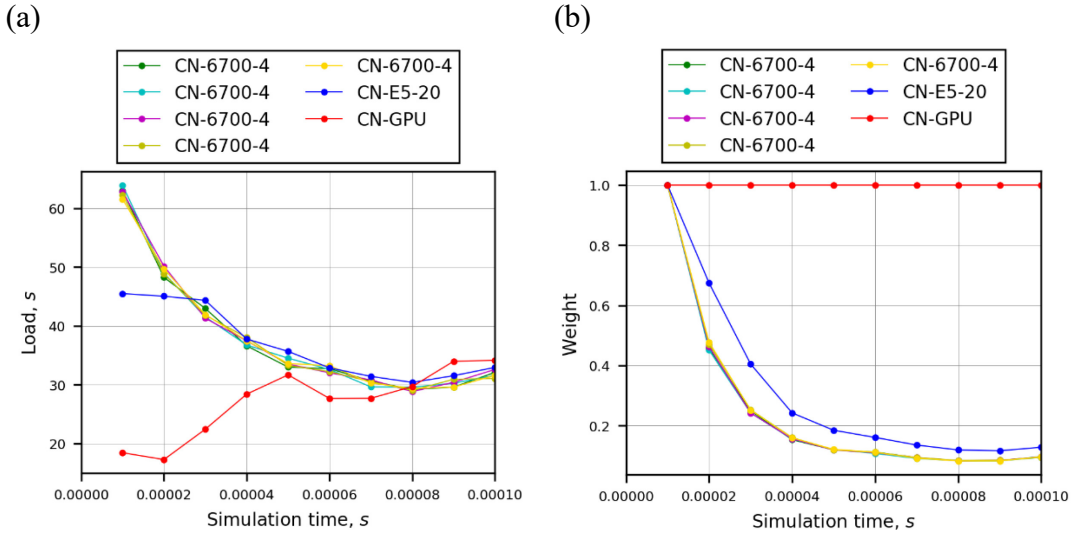

Figure 3: Time evolution of the computational load (a) and weights (b).

In Figure 3b, time evolution of weights shows that more than 11.5 times lower weights were computed for partitions of the containers CN-6700-4 than those for partitions of the container CN-GPU. Weights for the container CN-E5-20 were higher than those for the container CN-6700-4, but the difference was not large, comparing to the container CN-GPU equipped by GPU. The memory bandwidth bound DEM computations did not allow exploiting computing power of 20 physical cores because of comparatively low memory bandwidth of the CN-E5-20.

Figure 4 shows time evolution of communication time (Figure 4a) and wait time (Figure 4b). At the beginning of simulation interval, the rise of communication time of the container CN-GPU was caused by the increased number of ghost particles. Communication time of the container CN-GPU remained the largest during the whole simulation time interval because of the same reason. Communication time of the other containers chaotically oscillated due to regular application of repartitioning and resulting irregular subsets of particles. At the beginning of simulation interval, the

fastest container CN-GPU quickly completed its computations and waited for the data of neighbouring partitions. The runtime resource-aware partitioning increased the computational load of the container, which gradually decreased its wait time (Figure 4b). At the beginning of simulation interval, wait time of the container CN-E5-20 was also longer than that of the containers CN-6700-4, but the difference vanished after two applications of repartitioning. At the end of simulation interval, the container CN-GPU had so much data of ghost particles to send that other containers should wait for data of neighbouring subsets even longer than CN-GPU. The RCB method does not directly optimize internode communication, therefore, communication issues can influence the load balance and overall performance of parallel software.

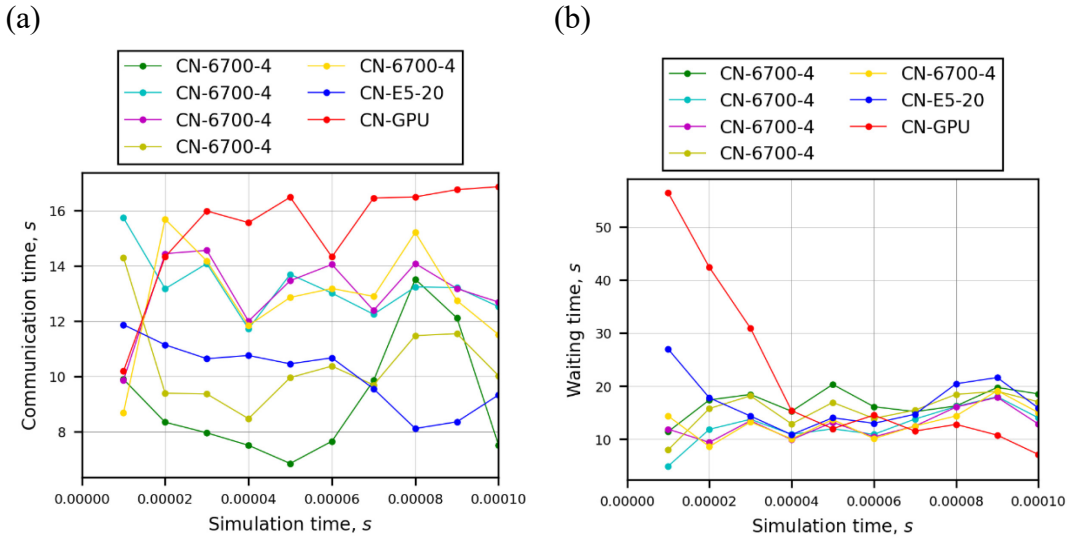

Figure 4: Time evolution of communication time (a) and wait time (b).

4 Conclusions

In the present paper, the hybrid parallelization of DEM software developed by using OpenCL for shared-memory architectures and MPI based resource-aware partitioning for distributed-memory heterogeneous resources of the OpenStack cloud. The hybrid parallelization eliminated issues related to mapping of particle subsets to multicore nodes, especially, improving speedup on uneven number of containers. Sufficient speedups equal to 5.2 and 6.0 were measured on 8 heterogeneous containers in the case of 1000188 and 4000000 particles, respectively. The replacement of 3 faster containers by 3 slower ones slightly decreased the speedup up to 7.4% of the speedup measured on 5 homogeneous containers. The developed resource-aware repartitioning handled heterogeneous cloud containers, supplemented with GPU, reducing the execution time up to 32.6% of the execution time obtained by using unweighted repartitioning.

References

[1] P.A. Cundall, O.D.L Strack, “A Discrete Numerical Model for Granular Assemblies”, Géotechnique, 29, 47–65, 1979. doi:10.1680/geot.1979.29.1.47.

[2] D. S. Henty, "Performance of Hybrid Message-Passing and Shared-Memory Parallelism for Discrete Element Modeling," in “SC '00: Proceedings of the 2000 ACM/IEEE Conference on Supercomputing”, Dallas, TX, USA, Paper 10, 2000. doi:10.1109/SC.2000.10005.

[3] R.Berger, C. Kloss, A. Kohlmeyer, S. Pirker, “Hybrid Parallelization of the LIGGGHTS Open-Source DEM Code”, Powder Technol, 278, 234–247, 2015. doi:10.1016/j.powtec.2015.03.019.

[4] M. Berger, S. Bokhari, “A Partitioning Strategy for Nonuniform Problems on Multiprocessors”, IEEE Trans. Comput, C-36, 570–580, 1987. doi:10.1109/tc.1987.1676942.

[5] P. Incardona, A. Leo, Y. Zaluzhnyi, R. Ramaswamy, I.F. Sbalzarini, “OpenFPM: A Scalable Open Framework for Particle and Particle-Mesh Codes on Parallel Computers”, Comput. Phys. Commun, 241, 155–177, 2019. doi:10.1016/j.cpc.2019.03.007.

[6] B. Yan, R.A. Regueiro, “Comparison between Pure MPI and Hybrid MPI-OpenMP Parallelism for Discrete Element Method (DEM) of Ellipsoidal and Poly-Ellipsoidal Particles”, Comput. Part. Mech, 6, 271–295, 2018. doi:10.1007/s40571-018-0213-8.

[7] J. Gan, T. Evans, A. Yu, “Application of GPU-DEM Simulation on Large-Scale Granular Handling and Processing in Ironmaking Related Industries”, Powder. Technol, 361, 258–273, 2020. doi:10.1016/j.powtec.2019.08.043.

[8] O. Bystrov, R. Pacevič, A. Kačeniauskas, “Performance of Communication- and Computation-Intensive SaaS on the OpenStack Cloud”, Appl. Sci, 11, 7379, 2021. doi:10.3390/app11167379.

[9] A. Kačeniauskas, R. Kačianauskas, A. Maknickas, D. Markauskas, “Computation and Visualization of Discrete Particle Systems on gLite-Based Grid”, Adv. Eng. Softw, 42, 237–246, 2011. doi:10.1016/j.advengsoft.2011.02.007.

[10] K. Devine, E. Boman, R. Heaphy, B. Hendrickson, C. Vaughan, “Zoltan Data Management Services for Parallel Dynamic Applications”. Comput. Sci. Eng, 4, 90–96, 2002. doi:10.1109/5992.988653.

[11] H.R. Norouzi, R. Zarghami, R. Sotudeh-Gharebagh, N. Mostoufi, “Coupled CFD-DEM Modeling: Formulation, Implementation and Application to Multiphase Flows”, Wiley, Chichester, West Sussex, United Kingdom, 2016. doi:10.1002/9781119005315.

[12] OpenStack. Available online: https://www.openstack.org/ (accessed on 29 April 2023).

[13] R. Pacevič, A. Kačeniauskas, “The Performance Analysis of the Thermal Discrete Element Method Computations on the GPU”, Comput. Inform, 41, 931–956, 2022. doi:10.31577/cai_2022_4_931.