ISSN 1392 – 124X INFORMATION TECHNOLOGY AND CONTROL, 2011, Vol. 40, No. 4

DYNAMIC DOMAIN DECOMPOSITION APPLIED TO HOPPER DISCHARGE SIMULATION BY DISCRETE ELEMENT METHOD

Darius Markauskas

Laboratory of Numerical Modelling, Vilnius Gediminas Technical University
Saulėtekio St. 11, Vilnius, LT-10223, Lithuania
e-mail: darius.markauskas@vgtu.lt

Arnas Kačeniauskas

Laboratory of Parallel Computing, Vilnius Gediminas Technical University
Saulėtekio St. 11, Vilnius, LT-10223, Lithuania

Algirdas Maknickas

Department of Information Technologies, Vilnius Gediminas Technical University
Saulėtekio St. 11, Vilnius, LT-10223, Lithuania

crossref http://dx.doi.org/10.5755/j01.itc.40.4.978

Abstract. The paper presents the development of dynamic domain decomposition applied to hopper discharge simulation by the discrete element method (DEM). Parallel neighbour search algorithm, non-blocking interprocessor communication and dynamic load balancing are implemented in the DEM code. Parallel speed-up analysis is performed solving the complex hopper discharge problems containing 100000 and 300000 particles. The influence of the granular flow character on the load balance is investigated.

1. Introduction

The discrete element method became widely recognized after the pioneering work published by Cundall and Strack [3]. The main advantage of the DEM is a possibility to model highly complex particle systems using the basic data on individual particles without making oversimplifying assumptions. The method allows simulation of motion and interaction between the particles, taking into account the microscopic geometry and various constitutive models. Over the past decade, the DEM was utilised in a variety of industrial applications [2].

The DEM has been extensively applied to examine different phenomena inside the granular materials. The granular flow from hoppers and silos has a wide range of applications in industry [29]. The conducted research is mainly focused on three aspects: wall stress/pressure, discharge rate and internal properties. The study of the bulk material pressure on the walls of a hopper is very important for hopper design [7]. The prediction of the discharge rate is of importance for the effective operation and control of a transport system, and is difficult due to inhomogeneous solid distribution, irregular velocity profile and diverse particle size [15]. It is very important to understand the microscopic structure and its relations to the mechanisms governing hopper flow [20]. DEM simulation takes into account the discrete nature of granular materials, and therefore is very effective for this purpose. The combined approach of DEM and averaging method offers a convenient way to link fundamental understanding generated from DEM-based simulations to engineering application often achieved by continuum modelling [28].

The main disadvantage of the DEM technique, in comparison with the well-known continuum methods, is impressive computational resources necessary to solve large-scale industrial problems. The complex character of hopper flow could require large number of particles and short time steps resulting in a long computing time.

Naturally, for the solution of industrial-scale problems, parallelization becomes an obvious option for significantly increasing computational capabilities. Early attempts to parallelize particle computations were based on two main ideas: force decomposition and domain decomposition. In the first class of methods, a pre-determined set of force computations is assigned to each processor. Such methods have shown good performance for shared memory computers [19, 23], but the global character of the employed algorithms produces interprocessor communication

overhead on distributed memory machines. Recent attempts to perform straightforward parallelization of DEM codes by using OpenMP did not result in very high parallel efficiency and scalability [6]. In the second class of methods, the domain decomposition [21] is employed. The basic idea of this technique is the partitioning of the computational domain into subdomains, each being assigned to a processor. The subdomains exchange data with each other through their boundaries [4]. Efficient and scalable shared and distributed memory parallelization is achieved in the area of molecular dynamics [10, 11], because the simple mechanics of interparticle contact is employed. Parallel visualization of particle systems on multi-core computer clusters is investigated in [13].

The design of parallel DEM algorithms presents a new challenge to computational scientists. Considerable efforts have been expended by scientists to design reconfigurable co-processors for DEM simulations [24] and even to optimize DEM codes for GPUs [22]. Two basic types, static or dynamic domain decomposition strategies, are extensively used in solving the time-dependent problems. Static domain decomposition works by assuming the fixed interdomain boundaries [12, 17]. The influence of material polydispersity to the performance of static domain decomposition is presented in [14]. More flexible, but more complicated dynamic decomposition algorithms [9] allow us to move boundaries during the simulation keeping load balancing of individual processors.

The parallel DEM algorithms employing the domain decomposition differ from analogous parallel processing in the continuum approach [4]. During hopper discharge moving particles dynamically change the workload configuration [16], making parallelization of DEM software much more difficult and challenging. The cubic decomposition methods and hierarchical trees employ moving boundaries to keep optimal shaped subdomains [5, 27]. Dual-level domain decomposition reduces the memory size per processor of the calculation [11, 26]. As the degree of natural algorithmic concurrency inherent in explicit time integration procedures is high, dynamic domain decomposition can yield high speed-up on different hardware configurations [25].

In spite of a considerable progress in developing parallel DEM software, its application to the solution of large-scale problems is rather limited. Only recently, some successful attempts have emerged in tackling problems of a similar nature [2, 25]. Moreover, the presented speed-up analyses are rarely performed solving complex applications. In the present research, the developed dynamic domain decomposition is applied to simulate hopper discharge problem. A particular manifest of this paper is twofold: to measure a parallel speed-up of the developed software, and to investigate the influence of the granular flow character to the workload balance.

The paper is organized as follows. In Section 2, governing relations and the methodology of the discrete element method are described. Section 3 discusses the developed algorithm of dynamic domain decomposition. In Section 4, parallel performance analysis and numerical results are presented, while the concluding remarks are given in Section 5.

2. Governing relations and DEM methodology

The dynamic behaviour of the non-cohesive frictional visco-elastic particle system governed by the Newton's second law is considered. Three translations and three independent rotations expressed in terms of the forces and torques at the centre of the *i*-th particle are as follows:

$$m_i \frac{d^2 \boldsymbol{x}_i}{dt^2} = \boldsymbol{F}_i \,, \tag{1}$$

$$I_i \frac{d^2 \boldsymbol{\theta}_i}{dt^2} = \boldsymbol{T}_i \,, \tag{2}$$

where m_i, and I_i are mass and inertia moments, while vectors $\boldsymbol{x}_i$ and $\boldsymbol{\theta}_i$ initiate the position of the particle centre and the orientation of particle *i,* respectively. Vectors $\boldsymbol{F}_i$ and $\boldsymbol{T}_i$ present the sum of contact force $\boldsymbol{F}_{i,cont}$, and gravity force $\boldsymbol{F}_{i,\,grav}$ as well as the corresponding torques:

$$\boldsymbol{F}_i = \sum_{j=1,\, j\neq i}^{N} \boldsymbol{F}_{ij} + m_i \boldsymbol{g} \,, \tag{3}$$

$$\boldsymbol{T}_i = \boldsymbol{T}_{i,contact} = \sum_{j=1,\, j\neq i}^{N} \boldsymbol{T}_{ij} = \sum_{j=1,\, j\neq i}^{N} \boldsymbol{d}_{cij} \times \boldsymbol{F}_{ij} \,, \tag{4}$$

where $\boldsymbol{d}_{cij}$ is particle geometry-dependent vector, pointing from the particle center to contact center.

The employed interparticle contact model considers a combination of elasticity, viscous damping and friction force effects. Actually, the contact between two material particles is modelled by a spring and dashpot in both the normal and tangential directions and an additional slider in tangential direction. Thus, the interparticle force vector $\boldsymbol{F}_{ij}$ describing the contact between the particles *i* and *j* may be expressed in terms of normal and tangential components $\boldsymbol{F}_{n,ij}$ and $\boldsymbol{F}_{t,ij}$, respectively. The normal component $\boldsymbol{F}_{n,ij}$ presenting a repulsion force comprises elastic and viscous ingredients. The tangential component $\boldsymbol{F}_{t,ij}$ reflects static or dynamic frictional behaviour. The static force describes friction prior to gross sliding and comprises elastic and viscous ingredients, while the dynamic force describes friction after gross sliding and is expressed by the Coulomb's law. Interparticle friction is defined by internal friction coefficient μ.

For evaluating the contact forces (3)-(4), all contacts between the particles and their neighbours must be detected. A cell-based method [8] is used for contact detection to reduce the number of all particle pair combinations. A three-dimensional domain of the granular media is divided into cubic cells of the size slightly larger than the diameter of the largest particle.

Then contact search is performed only between particles in neighbouring cells.

The dynamical state of all particles at the time t, resulting from the action of the particle forces (3)-(4), is obtained by numerical integration of the equations of motion (1)-(2). The solution of these equations is obtained by the explicit 5^{th} - order Gear's predictor-corrector scheme with a constant time increment Δt. The details of these procedures can be found in [1].

3. Parallel algorithm and load balancing

The presented research is focused to the development of DEMMMAT_PAR code [17, 18], while parallel algorithms are extended to support dynamic load balancing. Parallelization of software is based on the dynamic domain decomposition, considered to be one of the most efficient coarse grain strategies for scientific and engineering computations.

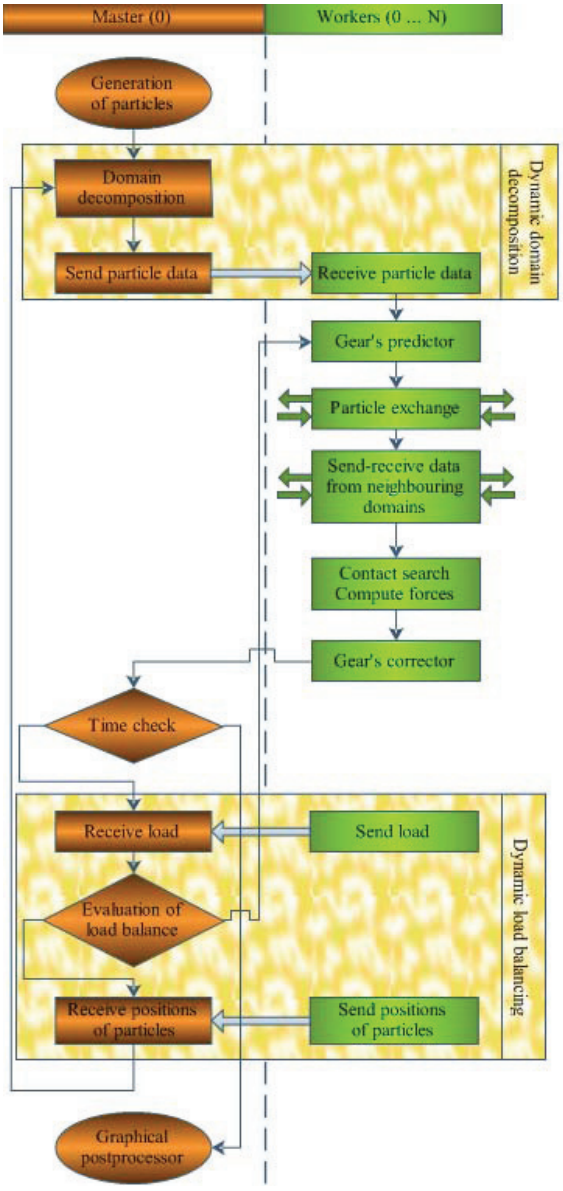

Figure 1. Parallel DEM algorithm

The parallel algorithm (Figure 1) is designed as follows. Initially, the pre-processor generates particles. The master processor divides the three-dimensional domain into the approximately equal subdomains containing a roughly equal number of particles. Parallel planes are employed to achieve this purpose. Then, the generated particles are assigned to relevant processors, while the particle data are distributed by using MPI calls. Thus, the initial domain decomposition including interprocessor communication is finished.

The main CPU time-consuming computational procedures of the DEM software are time integration and computation of contact forces, including contact detection. These intensive tasks are performed by the workers including the zero MPI process, which also serves as the master. The implemented domain decomposition perfectly parallelizes time integration performed in the time loop without any interprocessor communication. Each processor computes Gear's predictor and Gear's corrector independently by using locally stored data. Fast contact detection based on a parallel version of the cell algorithm presents more challenge because it requires interprocessor communication. Some neighbouring cells of the processed cell may belong to another processor. The processors need to exchange the information about the particles, which are near the division boundaries in the nearby subdomains.

The communication between workers is designed as follows. The initial portion of communications is performed after completing the Gear's predictor step, when the processors exchange particles as they move from one subdomain to another. This interprocessor communication is optional, requiring sending-receiving a small amount of data. However, it needs extensive manipulation on the data structures. It is worth noting that the conventional codes based on the continuum approach do not perform the described communication. The main portion of communications is performed to exchange particle data from the cells near the division boundaries. The employed communication model is created for grid networks. The received particle co-ordinates are placed as contiguous data directly into the arrays containing local particles. No time is spent for rearranging the data, except for creating the buffered messages for interprocessor communication. The inherent synchronisation of this message passing algorithm ensures good performance of parallel computation on the distributed memory PC clusters. The communications are performed by the non-blocking MPI routines MPI_ISEND and MPI_IRECV, which significantly improve parallel efficiency of the code.

However, during hopper discharge simulation particles move through the whole computational domain, dramatically change the initial workload configuration and cause significant load imbalance. Thus, dynamic load balancing algorithm becomes necessary in case of complex hopper flows. A universal algorithm

performing the dynamic load balancing presents some challenges due to a complex code structure, programming efforts and computational costs. A fast and simple iterative algorithm based on moving planes (Figure 2) is implemented to achieve satisfactory load balancing. During the analysis, the particles move from one subdomain to another. Significantly different numbers of particles in the subdomains indicate the load imbalance, but the precise load can be evaluated measuring time of computations performed by each processor. The load coefficients representing the time consumed per particle computations are calculated for each subdomain. Thus, the global load can be evaluated multiplying numbers of particles in subdomains by relevant load coefficients. The user controls the allowable load imbalance by specifying predefined percentage. New decomposition is performed, when the load change in any subdomain reaches a predefined value. The positions of planes are iteratively adjusted in such a way that the number of particles multiplied by load coefficients in each subdomain would be as close as possible to the average value. However, during the dynamic decomposition, particles and their data should be exchanged between the processors, which insignificantly increases the overall communication time.

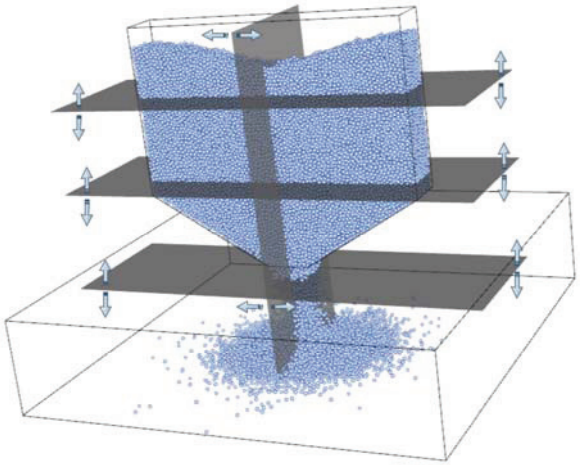

Figure 2. Dynamic domain decomposition

4. Numerical results and discussions

The hopper discharge computations and parallel speed-up measurements were performed on Vilnius Gediminas Technical University cluster VILKAS (Rocks Cluster, CentOS release 5.4, x86_64 architecture) based on multi-core architecture. The cluster consists of 23 nodes including Intel® Core2Quad Q6600 2.40GHz CPU (2x4MB L2 cache and bus frequency equal 1067 MHz), 4x1GB DDR2 800 RAM, 320GB HDD (SATA II Extensions and 16 MB cache). Nodes are connected to 1Gbps Ethernet LAN by D-Link DGS 1224T Gigabit Smart Switch (24-Ports 10/100/1000Mbps Base-T Module).

4.1. Hopper discharge simulation

The problems associated with handling flow of granular materials in hoppers are of great significance in pharmaceutical, food, cement and chemical industries. The challenges relevant to particle segregation, the effects of granular material vibration, attrition, formation of blockage or erratic flow zones, dust explosions and wall collapses are encountered during the operation period of hoppers. The discrete approach enables simulation of the dynamical behaviour of granular material by direct introspection of physical effects of individual particles on the resulting behaviour of flowing granular material.

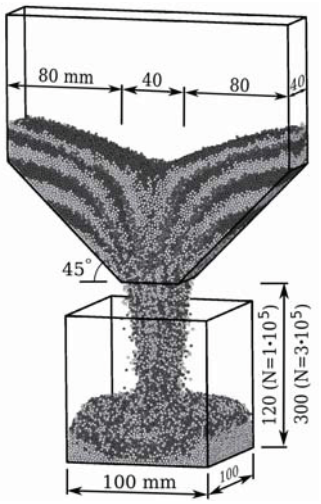

Figure 3. Geometry of the hopper discharge problem

The hopper discharge actually means the flow of the particles and their falling from the hopper due to the opening of the orifice. The configuration of hopper is presented in Figure 3. The rigid container walls are considered to be fixed frictional boundaries. The dimensions of the orifice are 40 x 40 mm.

Granular material is modelled as the assemblies of non-cohesive spherical particles $N = 100000$ and $N = 300000$. The particle radii R_i varying over the range from 1.95 to 2.35 mm are generated with uniform distribution. The total volume V of the material is equal to $V = 5.28 \cdot 10^{-4}$ m^3 for $N = 1 \cdot 10^5$ particles and $V = 1.584 \cdot 10^{-3}$ m^3 for $N = 3 \cdot 10^5$ particles. Elasticity modulus of the particle is equal to $E = 1 \cdot 10^6$ Pa, the restitution coefficient is equal to 0.5. Interparticle friction is characterized by the friction coefficient $\mu = 0.4$. Material parameters for particle-wall interactions are assumed to be the same as those used for describing the interparticle relations.

The initial state of the particulate material was generated numerically by simulating the process of filling. The state of hopper at 0.5 s after the opening of orifice is showed in Figure 3. The particles were initially coloured depending on the layer. The picture allows us to follow the entire particles' flow structure and their interlayer migration as well as detecting a zone of intensive mixing.

4.2. Parallel performance analysis

The parallel performance of computations was evaluated by measuring the speed-up S_p

$$S_p = \frac{t_1}{t_p}, \tag{5}$$

where t_1 is the program execution time for a single processor; t_p is the wall clock time for a given job to execute on p processors. The benchmark tests are repeated up to ten times and the averaged values are presented in figures. The parallel performance tests carried out on the PC cluster VILKAS are presented in Figure 4 and Figure 5. The speed-up (5) gained relative to a sequential run as a function of the number of processors is shown for the systems consisting of 100000 and 300000 particles.

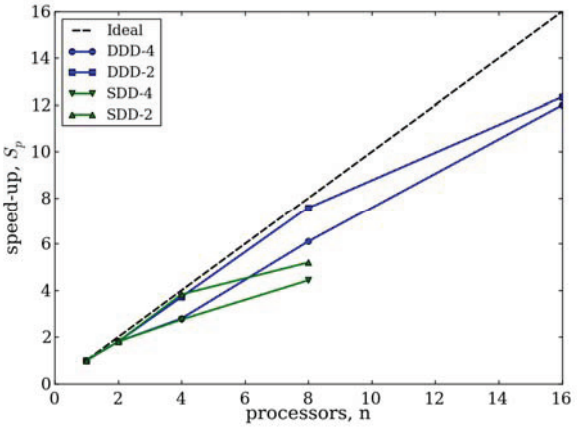

Figure 4. Speed-up measured performing static domain decomposition and dynamic domain decomposition

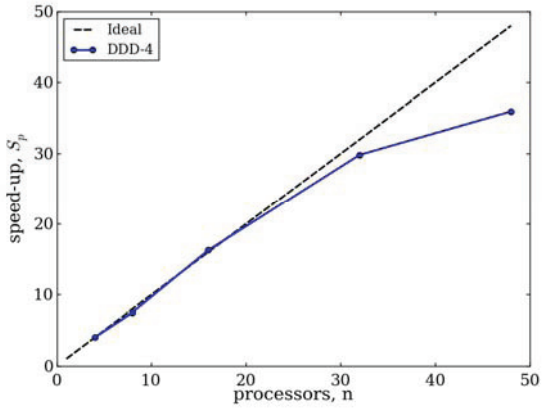

Figure 5. Parallel speed-up measured simulating the large system consisting of 300000 particles

Figure 4 shows quantitative comparison of speed-ups measured performing static domain decomposition and dynamic domain decomposition of the hopper containing N = 100000 particles. It is well known fact that Intel® Core2Quad processors lack sufficient parallel performance, when four memory intensive processes are executed per node. Thus, speed-ups (5) attained executing four processors per node are compared with those executing two processors per node in order to illustrate this phenomenon. The curves "DDD-2" and "DDD-4" represent speed-ups of dynamic domain decomposition obtained by running two processes per node and four processes per node, respectively. The curves "SDD-2" and "SDD-4" represent speed-ups of static domain decomposition measured by executing two processes per node and four processes per node, respectively. The special curve "Ideal" illustrates the ideal speed-up. The hopper solution domain can easily be divided to four subdomains neglecting the dynamic character of particle flow. Therefore, very close values of speed-up are measured for four processes. The satisfactory static subdivision of the hopper domain to eight or more subdomains is more challenging or even impossible. Thus, the dynamic domain decomposition significantly outperforms the static decomposition. When the number of processes is small and two processes run per node (the curve "DDD-2"), the measured speed-up is close to linear. The reduction of the speed-up owing to communication overhead is obtained for 16 processes. The parallel speed-up is largely determined by the ratio of local computations over interprocessor communications. As the number of processors increases, for a fixed problem size, the communication cost eventually become dominant over the local computation cost.

Figure 5 shows parallel speed-up measured simulating larger system consisting of 300000 particles. It is difficult to simulate this system by using one process, because of long computing time. A slight reduction of the speed-up is obtained by using 48 processes, because of the limited decomposition topology leading to communication overhead. The direct comparison of the measured speed-up with the results of other authors seems to be complicated, because of different hardware platforms and software implementations. Despite of that, the parallel speed-up reported in the present paper can compete with that reported in the overviewed literature. The presented results show that the implemented dynamic domain decomposition is well designed for simulation of hopper discharge on the PC clusters.

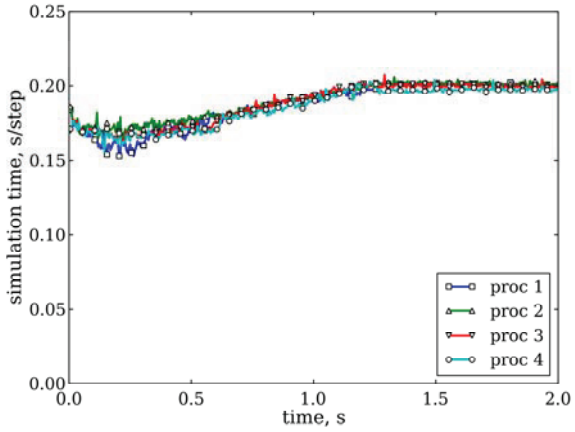

Figure 6. Load balance evaluated measuring simulation time of one time step

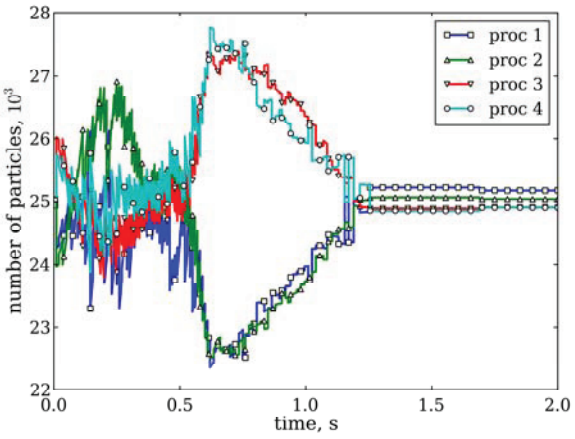

Figure 7. Time variation of the number of particles per processor

A highly efficient parallel implementation requires a well balanced workload among the processors. Figure 6 shows the dynamic load balance achieved by using four processors. The total workload increases slightly while particles rapidly change their positions. It can be explained by the fact that, initially, arrays of the particle data are ordered according to neighbourhood of the particles. This procedure was not repeated during computations, therefore, large changes of particles' positions lead to the cache miss. Figure 7 plots the time variation of the number of particles per processor. The workload is well balanced while processors work on very different numbers of particles. Comparison of these figures shows that the workload cannot directly be defined by the number of particles.

The discussed phenomena can be explained by the fact that the amount of computations strongly depends on the number of contacts between particles. Figure 8 plots time variation of particles' contacts, while Figure 9 illustrates the hopper discharge at different time instances. At the beginning of computations, it is difficult to balance the workload, because particles fall from the hopper and reach the bottom (Figure 9a). Thus, the number and the character of the contacts changes very quickly (Figure 8). At t=0.7s all falling particles belong to the upper subdomains (Figure 9b), which leads to the small number of contacts and the large number of particles. Figure 9c illustrates the moment when all particles are in the lower box. It is clear that at t=1.2s smaller number of particles and larger number of contacts are in the lower subdomains. Finally, it can be concluded that the number of particles is inversely proportional to the number of contacts in the processor, which leads to the balanced workload.

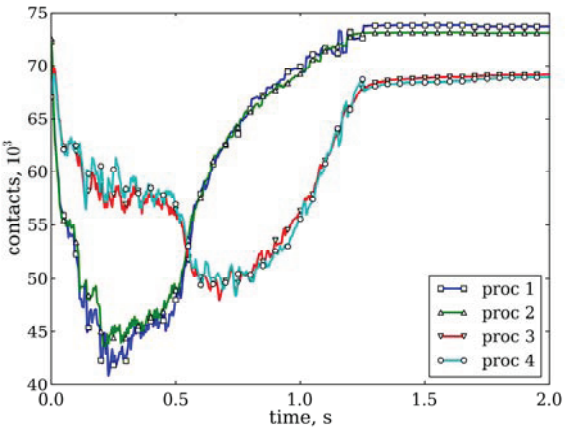

Figure 8. Time variation of particles' contacts

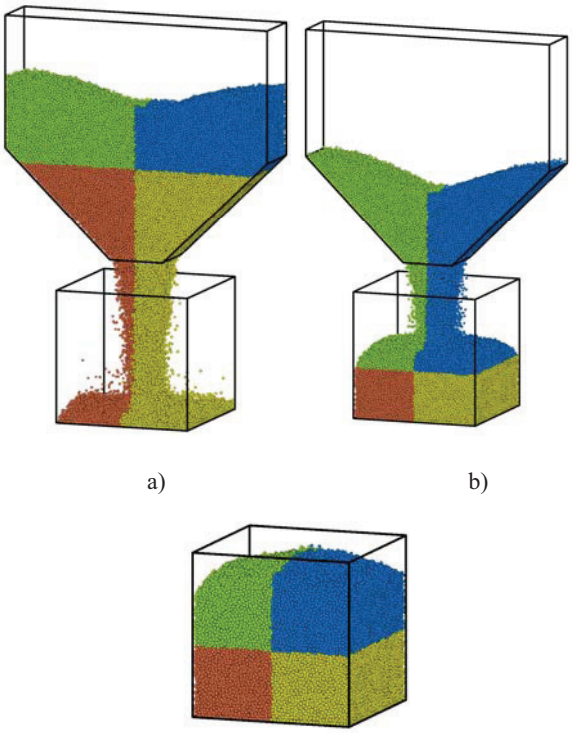

Figure 9. Hopper discharge simulated by 4 processors: (a) t=0.2s, (b) t=0.7s, (a) t=1.2s

5. Conclusions

In this paper, the development of parallel DEM software based on dynamic domain decomposition is described. The developed dynamic load balancing is applied to simulate the complex hopper discharge problem on the PC clusters. The quantitative comparison of the speed-ups measured performing static domain decomposition and dynamic domain decomposition is presented. Parallel speed-up analysis reveales that the developed dynamic decomposition is able to ensure proper load balancing and to simulate the challenging hopper discharge problem on the computer clusters based on multi-core architecture. Performed investigation shows that workload is dependent

on the granular flow character. Moreover, the load balance is significantly influenced by the number of contacting particles.

References

[1] **R. Balevičius, R. Kačianauskas, A. Džiugys, A. Maknickas, K. Vislavičius**. DEMMAT code for numerical simulation of multi-particle systems dynamics. *Information Technology and Control*, 2005, *Vol*. 34(1), 71–78.

[2] **P.W. Cleary**. Industrial particle flow modelling using discrete element method. *Engineering Computations*, 2009, *Vol*. 26(6), 698–743.

[3] **P.A. Cundall, O.D.L. Strack**. A discrete numerical model for granular assemblies. *Geotechnique*, 1979, *Vol*. 29(1), 47–65.

[4] **C.H. Dowding, O. Dmytryshyn, T.B. Belytschko**. Parallel processing for a discrete element program. *Computers & Geotechnics*, 1999, *Vol*. 25(4), 281–285.

[5] **F. Fleissner, P. Eberhard**. Parallel load-balanced simulation for short-range interaction particle methods with hierarchical particle grouping based on orthogonal recursive bisection. *Int. J. Numerical Methods in Engineering*, 2008, *Vol*. 74(4), 531–553.

[6] **G. Frenning**. An efficient finite/discrete element procedure for simulating compression of 3D particle assemblies. *Computer Methods in Applied Mechanics and Engineering*, 2008, *Vol*. 197(49–50), 4266–4272.

[7] **T.J. Goda, F. Ebert.** Three-dimensional discrete elementsimulations in hoppers and silos. *Powder Technology*, 2005, *Vol*. 158(1–3), 58–68.

[8] **K. Han, Y.T. Feng, D.R.J. Owen**. Performance comparisons of tree-based and cell-based contact detection algorithms. *Engineering Computations: Int. J. Computer-Aided Engineering and Software*, 2007, *Vol*. 24(2), 165–181.

[9] **B. Hendrickson, K. Devine**. Dynamic load balancing in computational mechanics. *Computer Methods in Applied Mechanics and Engineering*, 2000, *Vol*. 184(2–4), 485–500.

[10] **B. Hess, C. Kutzner, D. van der Spoel, E. Lindahl**. GROMACS 4: algorithms for highly efficient, load-balanced, and scalable molecular simulation. *J. Chemical Theory and Computation*, 2008, *Vol*. 4(3), 435–447.

[11] **J. Hutter, A. Curioni**. Dual-level parallelism for ab initio molecular dynamics: reaching teraflop performance with the CPMD code. *Parallel Computing*, 2005, *Vol*. 31(1), 1–17.

[12] **A. Jabbarzadeh, J.D. Atkinson, R.I. Tanner**. A parallel algorithm for molecular dynamics simulation of branched molecules. *Computer Physics Communications*, 2003, *Vol*. 150(2), 65–84.

[13] **A. Kačeniauskas, R. Pacevič, A. Bugajev, T. Katkevičius.** Efficient visualization by using ParaView software on BalticGrid. *Information Technology and Control*, 2010, *Vol*. 39(2), 108–115.

[14] **R. Kačianauskas, A. Maknickas, A. Kačeniauskas, D. Markauskas, R. Balevičius**. Parallel discrete element simulation of poly-dispersed granular material. *Advances in Engineering Software*, 2010, *Vol*. 41(1), 52–63.

[15] **H. Kruggel-Emden, S. Rickelt, S. Wirtz, V. Scherer**. A numerical study on the sensitivity of the discrete element method (DEM) for hopper discharge. *J. Pressure Vessel Technology*, 2009, *Vol*. 131(3), 031211.

[16] **J.W. Landry, G.S. Grest, S.J. Plimpton**. Discrete element simulations of stress distributions in silos: crossover from two to three dimensions. *Powder technology*, 2004, *Vol*. 139(3), 233–239.

[17] **A. Maknickas, A. Kačeniauskas, R. Kačianauskas, R. Balevičius, A. Džiugys**. Parallel DEM software for simulation of granular media. *Informatica*, 2006, *Vol*. 17(2), 207–224.

[18] **D. Markauskas, R. Kačianauskas, A. Džiugys, R. Navakas**. Investigation of adequacy of multi-sphere approximation of elliptical particles for DEM simulations. *Granular Matter*, 2010, *Vol*. 12(1), 107–123.

[19] **D.R.J. Owen, Y.T. Feng**. Parallelized finite/discrete element simulation of multi-fracturing solids and discrete systems. *Engineering Computations*, 2001, *Vol*. 18(3–4), 557–576.

[20] **D.R. Parisi, S. Masson, J. Martinez**. Partitioned distinct element method simulation of granular flow within industrial silos. *ASCE J. Engineering Mechanins*, 2004, *Vol*. 130(7), 771–779.

[21] **S. Plimpton**. Fast parallel algorithms for short range molecular dynamics. *J. Computational Physics*, 1995, *Vol*. 117(1), 1–19.

[22] **C.A. Radeke, B.J. Glasser, J.G. Khinast**. Large-scale powder mixer simulations using massively parallel GPU architectures. *Chemical Engineering Science*, 2010, *Vol*. 65(24), 6435–6442.

[23] **M. Renouf, F. Dubois, P. Alart**. A parallel version of the non smooth contact dynamics algorithm applied to the simulation of granular media. *J. Computational and Applied Mathematics*, 2004, *Vol*. 168(1-2), 375-382.

[24] **B.C. Schäfer, S.F. Quigley, A.H.C. Chan**. Acceleration of discrete element method (DEM) on a reconfigurable co-processor. *Computers & Structures*, 2004, *Vol*. 82(20-21), 1707–1718.

[25] **J.H. Walther, I.F. Sbalzarini**. Large-scale parallel discrete element simulations of granular flow. *Engineering Computations*, 2009, *Vol*. 26(6), 688–697.

[26] **F. Wang, Y.T. Feng, D.R.J. Owen, J. Zhang, Y. Liu**. Parallel analysis of combined finite/discrete element systems on PC cluster. *Acta Mechanica Sinica*, 2004, *Vol*. 20(5), 534–540.

[27] **D. Zhang, C. Jiang, S. Li**. A fast adaptive load balancing method for parallel particle-based simulations. *Simulation Modelling Practice and Theory*, 2009, *Vol*. 17(6), 1032–1042.

[28] **H.P. Zhu, Z.Y. Zhou, R.Y. Yang, A.B. Yu**. Discrete particle simulation of particulate systems: theoretical developments. *Chemical Engineering Science*, 2007, *Vol*. 62(13), 3378–3396.

[29] **H.P. Zhu, Z.Y. Zhou, R.Y. Yang, A.B. Yu**. Discrete particle simulation of particulate systems: A review of major applications and findings. *Chemical Engineering Science*, 2008, *Vol*. 63(23), 5728–5770.

Received December 2010.