ON THE ANALYSIS OF THE DISCRETE ELEMENT METHOD IN MULTI-CORE PROCESSORS

Luiz Carlos Facundo Sanches

lsanches@sjbv.unesp.br

São Paulo State University - UNESP

Av. Dr. Octávio da Silva Bastos, 2439, 13874-149, São Paulo, São João da Boa Vista, Brazil

Ronaldo Carrion

rcarrion@usp.br

University of São Paulo - USP

Av. Prof. Luciano Gualberto, travessa 3, 380, 05508-010, São Paulo, São Paulo, Brazil

Carlos Héracles Moraes de Lima

carlos.lima@sjbv.unesp.br

São Paulo State University - UNESP

Av. Dr. Octávio da Silva Bastos, 2439, 13874-149, São Paulo, São João da Boa Vista, Brazil

Abstract. *The Discrete Element Method (DEM) has been commonly used to simulate particle flows, though it is becoming a popular method to represent discontinuous medium lately. In computational fluids or solids mechanics, DEM is computationally expensive; however, reconfigurable computing can be used to make it less demanding. Recently, multi-core Central Processing Unit (CPU) and Graphics Processing Unit (GPU) have attracted a lot of attention in order to accelerate computer simulations in various applications. In fact, there exist architectures for multi thread parallel computation of DEM that accelerate the computation, while making effective use of the available memory. Although preliminaries results in literature show that multi-thread parallel programming performs similarly in GPUs and CPUs platforms, it is well known that DEM algorithms have higher scalability in the later. This paper describes the feasibility of using parallel computing to implement DEM on Personal Computer (PC) multi-core processors for granular hopper flow problem in silos. Preliminary results showed a speedup and almost linear scalability for scheme parallelization utilized. The values are in agreement with those encountered in literature.*

Keywords: *Discrete Element Method, Hopper flow, Parallel computing, Multi-core processors*

1 INTRODUCTION

The Discrete Element Method (DEM) initially described by Cundall and Strack (1979) has been widely used in numerical analyses of fluid and solid dynamic from an engineering perspective. The scheme is a *Lagrangian* approach where individual particles are calculated on the basis of *Newton's* second law of motion. Discrete elements enable us to investigate the granular flow of particles and their motion precisely. In this way, numerical computation becomes costly when the number of particles is very large. Novel modeling and parallel calculation techniques are needed to implement better practical DEM in large scale processes.

In the last years, powerful computer clusters and super computers have been used to simulate complex systems with large number of calculated particles using DEM. However, their excessive deployment costs limit their popularity. Similar results have been reported for other applications, e. g., computational fluid dynamics, structural analyses, and rigid body simulations. In these cases, the number of cores available on a CPU has, too, been steadily increasing. Multi-core processors for both graphics processing unit (GPU) (Wang, L., et al., 2013) and latest multi-core central processing unit (CPU) (Sakai et al., 2010), have emerged as a promising low-cost alternative to enable multi-thread parallel computations.

Some general strategies for parallelization are described by (Munjiza et. al, 2012). Recently, Lukas et. al (2014) presented a approach for the parallelization of two-dimensional finite-discrete element method aiming at clusters and desktop computers. In order to examine performance of an parallel scheme, this work presents an attempt to implement DEM on CPU multi-core processors for the granular hopper flow in silos.

2 DISCRETE ELEMENT MODELING

Cundall and Strack (1979) developed an explicit solution considering the dynamic equilibrium of individual particles rather than solving the entire system. The equations governing the translational and rotational dynamic equilibrium of a particle *i* with mass m_i is

$$\frac{d^2 x_i(t)}{dt^2} = \ddot{x}_i(t) = \frac{F_{x_i}(t)}{m_i}, \quad \text{and} \quad \frac{d^2\theta(t)}{dt^2} = \ddot{\theta}_i(t) = \dot{\omega}_i(t) = \frac{T_i(t) + M_i(t)}{I_i}, \tag{1}$$

where $x_i(t)$ are the particle centroid positions *i* arranged in a fixed Cartesian coordinate system, $\ddot{x}_i$ is the acceleration vector for *ith* particle, F_{xi} are the contact forces, m_i is the particle mass, $\dot{\omega}_i$ is the angular acceleration vector, T_i is the torque, M_i is the resultant moment acting through the centroid of the particle *i* and I_i is the moment of inertia. For a circular particle the moment of inertia is equal to $\rho \pi r_i^4 / 2$ where r_i is the radius and ρ is the mass density.

2.1 Updating Particle Position

From the dynamic equilibrium and knowing the resulting forces acting on them, it is possible to calculate the accelerations for *ith* particle. Particularly, for translation motion, we have:

$$m_i a_i^t = F_i^t, \tag{2}$$

where m_i is the inertia (mass) matrix, $a_i^t = \ddot{u}_i^t$ is the acceleration vector at time *t*, and F_i^t is the resultant force vector. To update such parameters it is necessary to implement integration methods, i.e. given by their first and second derivatives with respect to time.

With DEM, the relationship between the acceleration and velocity vector is:

$$a_i^t = \frac{1}{\Delta t}\left(v_i^{t+\Delta t/2} - v_i^{t-\Delta t/2}\right), \tag{3}$$

where $v_i^{t-\Delta t/2}$ and $v_i^{t+\Delta t/2}$ are the velocity at t - $\Delta t/2$ and t + $\Delta t/2$ respectively for the *ith* particle. Eq. (3) is also known as the position *Verlet* time integration scheme, and the velocity at time $t + \Delta t/2$ is then calculated as:

$$v_i^{t+\Delta t/2} = v_i^{t-\Delta t/2} + \Delta t \frac{1}{m_i}\left(F_i^t\right). \tag{4}$$

The velocity at time $t + \Delta t/2$ is equal to the average velocity within the interval from t to $t + \Delta t$. Then, we can calculate the updated particle position as:

$$x_i^{t+\Delta t} = x_i^t + \Delta t \times v_i^{t+\Delta t/2}, \tag{5}$$

where the particle position vector x gives the particle Cartesian coordinates and when required the total rotation about the principal axis. There are three basic types of interactions in 3D case: inter-particle interactions (collision), body forces (gravity and bulk viscosity), and particle-wall interactions. We have specified that the gravitational acceleration is 9.81 m/s in the negative x_2-direction. This is the usual value for simulations in which the x_2-axis is assumed to be vertical. The bulk viscosity (< 0.05 for elastodynamics problems and ~ 0.5 for quasi-static problems might be appropriate) is included as a damping force proportional to the instantaneous velocity of each particle acting in the direction that opposes motion.

2.2 Calculating Contact Forces

Translational and rotational motion (when required) of the particles were incremented using numerical integration procedure in which assembly configurations were generated at time intervals Δt. The contact forces consisted of normal and tangential components, each of which had a viscous dissipation term proportional to the component of the velocity in that direction. F_n denotes the normal component of the contact force, while the tangential component is denoted by F_t. The contact normal force can be calculated as (O'Sullivan, 2011)

$$F_n = F_N - c_n v_n, \tag{6}$$

where F_N is the elastic part of the normal interaction, c_n is the normal interaction damping coefficient and $v_n = (v \cdot n)n$ with n representing the unit vector in the interaction between the centers of the two particles and v the particle translational velocity vector.

The second term on the right-hand side of Eq. (6) represents the contact damping. There is also a tangential interaction, the frictional component which is given by

$$F_T = \mu F_N \left(1 - \left(1 - \left|\delta_t\right| / \delta_{máx}\right)^{3/2}\right), \tag{7}$$

where δ_t is the total tangential displacement between the two surfaces from the point were they initially came into contact. If $|\delta_t| > \delta_{máx}$ then gross sliding is deemed to have started and the frictional force assumes a constant value given by *Amonton's* law, $F_T = \mu F_N$, where μ is the coefficient of friction. In this case, the choice of $\delta_{máx}$ (maximum tangential displacement) is typically several orders of magnitude larger than that for a typical contact. Additionally, the total tangential interaction force at any time is

$$F_t = F_T - c_t v_t, \tag{8}$$

The second term on the right-hand side of Eq. (8) is the contact damping force proportional to the tangential component of the relative velocity, $v_t = v \times n$ and c_t is the tangential interaction damping coefficient. In the model above, unrealistic simplifications have been used. Firstly, was used non-rotational particles as our granular media. Other simplification is the use of a frictionless mesh wall. This will result in a wider flow pattern than usual, with particles sliding towards the outlet more easily.

3 DEM PARALLEL PC MULTI-CORE PROCESSORS

The DEM simulation includes registering particles and walls, collision detection, calculating the contact force, and updating particles. In this work, a modular open source object-oriented simulation code written in C++ was tested. According to Weatherley et al. (2014), spatial domain decomposition (in subdomains) is implemented using a master slave strategy with inter process communications using the Message Passing Interface (MPI). To do that, a *Verlet* list neighbor search algorithm for detecting neighboring particles and an explicit first-order finite difference time integration scheme are employed. Besides, a simple Application Programming Interface (API) allows to evaluate simulations via scripts written in Python programming language. During all iterations and loading step, each DEM ensemble runs independently. The parallelization is achieved by decomposing the problem domain into subdomains (Weatherley, et. al, 2014), where the interface of each sub-domain needs to be duplicated for information to be passed between two neighboring sub-domains. The computer simulations ran in Linux Ubuntu 14.02 and simulation was made in personal computer using multi-core CPU. Processing was implemented in an Intel Core i-5 4210U with Five-Core Processor, 2.40 GHz CPU clock and system with 8.0 GB RAM.

4 NUMERICAL ANALYSIS

Consider a cube of side 10 m containing particles whose radii range from 0.2 m to 0.5 m. Both elastic repulsion and frictional resistance between touching particles were incorporated. The normal stiffness coefficient is 1000.0 N/m, the tangential 100.0 N/m and the frictional coefficient is 0.6. For the walls, the normal stiffness coefficient is 10000.0 N/m. Run simulation for 10^5 interactions with a timestep increment of 10^{-4}s. POVray is used to record snapshots every 10^3 timesteps. Considering the geometry, a quarter of the 3D domain is modeled with non-rotational particles (Figure 1). The friction angles and the wall boundaries are set to zero and pressures are recorded at the middle segment to reduce the boundary effects. The scheme has been used by Weatherley et al. (2014) to model hopper flow.

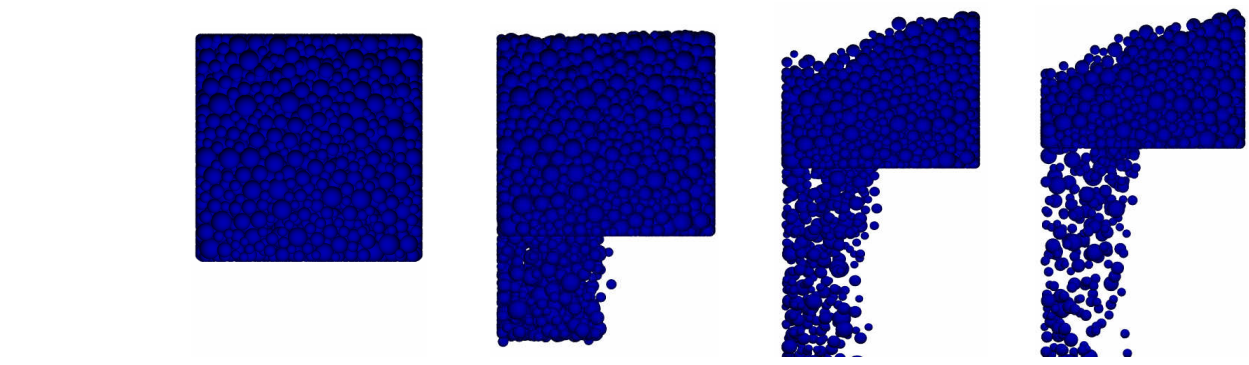

Figure 1. A quarter of the domain for a granular hopper flow in silos

For a fixed number of the particles in the cube, we record the computational time and compare the respective speedups using different number of processors n_p, which specifies the number of MPI paralell processes used in the simulation. The scalability of the parallel analysis was also investigated. It was considered up to four physical CPU cores.

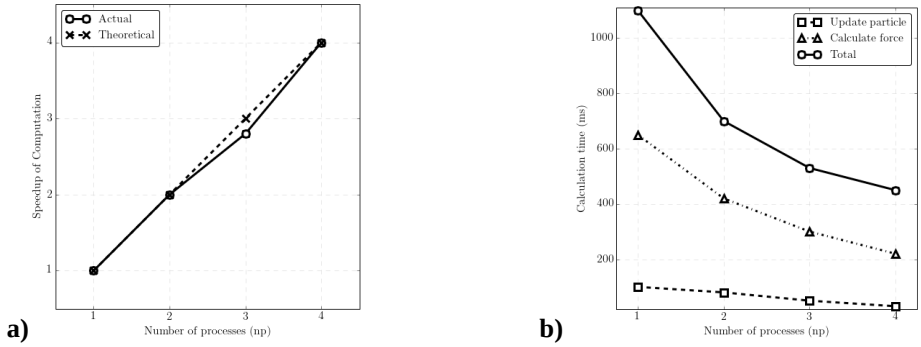

Figure 2. a) Speedup of calculus thought parallelization; b) Scalability of cores on CPU

The results are shown in Fig. 2. For a certain processor number the speedup is defined as the executing time using one single processor divided by that of specific number of processors. Premature values showed a speedup and almost linear scalability for our parallelization.

5 CONCLUSIONS AND FUTURE WORK

We have made a parallel application using open multi core processing and message passing interface code. The code was tested on personal computer with up to four CPU multi-core processor to model a granular hopper flow problem in silos. Preliminary results are in agreement with those encountered in literature. Nevertheless, future investigation needs to be carried out to examine the effect of both CPU and GPU based parallel computing method.

REFERENCES

Cundall, P. A., Strack, O. D. L., 1979. A discrete numerical model for granular assemblies. *Geotechnique*, v. 29, pp. 47–65.

Lukas, T., D'Albano, G. G. S., Munjiza, A. A., 2014. Space decomposition based parallelization solutions for the combined finite-discrete element method in 2D. *Journal of Roch Mechanics and Geotechnical Engineering.* v. 6, pp. 607-615

Munjiza, A. A., Knight, E. E., Rougier, E., 2012. *Computational Mechanics of Discontinua*. Wiley Series in Computational Mechanics. Chichester, UK: Wiley.

O'Sullivan, C., 2011. *Particulate Discrete Element Modelling: a Geomechanics Perspective*. New York: Spon Press.

Sakai, M., Yamada, Y., Shigeto, Y., Shibata, K., Kawasaki, V. M., Koshizuka, S., 2010. Large-scale DEM modeling in a fluidized bed. *Int. J. Num. Meth. in Fluids*, v.64, pp. 1319–35

Wang, L., Li, S., Zhang, G., Ma, Z. Zhang, L., 2013. A GPU-Based Parallel Procedure for Nonlinear Analysis of Complex Structures Using a Coupled FEM/DEM Approach. *Mathematical Problems in Engineering*, v. 2013. Article ID 618980.

Weatherley, D. K., Boros, V. E., Hancock, W. R., Abe, S., 2014. *ESyS-particle Tutorial and User Guide Version 2.3.1*. The University of Queensland. Australia.