Computing and Informatics, Vol. 41, 2022, 931–956, doi: 10.31577/cai_2022_4_931

THE PERFORMANCE ANALYSIS OF THE THERMAL DISCRETE ELEMENT METHOD COMPUTATIONS ON THE GPU

Ruslan Pacevič, Arnas Kačeniauskas

Department of Graphical Systems
Vilnius Gediminas Technical University
Vilnius 10223, Lithuania
e-mail: {ruslan.pacevic, arnas.kaceniauskas}@vilniustech.lt

Abstract. The paper presents a GPU implementation of the thermal discrete element method (TDEM) and the comparative analysis of its performance. Several discrete element models for granular flows, the bonded particle model and the TDEM are considered for quantitative comparison of computational performance. The performance measured on NVIDIA® Tesla™ P100 GPU is compared with that attained by running the same OpenCL code on Intel® Xeon™ E5-2630 CPU with 20 cores. The presented GPU implementation of the TDEM increases the computing time of the bonded particle model only up to 30.6 % of the computing time of the simplest DEM model, which is an acceptable decrease in the performance required for solving coupled thermomechanical problems.

Keywords: GPGPU computing, thermal discrete element method, bonded particle model, performance analysis, OpenCL

Mathematics Subject Classification 2010: 68-W10, 65-Y05, 70-04

1 INTRODUCTION

The discrete element method (DEM), originally developed by Cundall and Strack [1], is considered to be a powerful numerical technique to understand and model the

phenomena of the particulate media. The concept of the DEM presents the numerical methodology, providing the quantitative description of the discrete particulate media by considering the motion and deformation behaviour of individual particles in the frame of Newtonian mechanics. Traditionally, the DEM methodology is associated with the physical nature of granular materials [2]. However, the advanced DEM models are also applied to the heterogeneous multi-phase continuum, such as rock or concrete. Potyondy and Cundall [3] have suggested the bonded particle model (BPM) for the fracture behaviour. Thus, DEM can simulate the transition process from the continuum to discontinuum by changing the bond types between the particles without the need for specialized elements or remeshing lattices. Now, various DEM models are not limited to the analysis of granular materials [4, 5], but can be effectively applied to studying rock cutting [6], coupled multi-physical problems [7], sea ice failure [8] and the fracture of reinforced concrete [9].

The heat transfer between the contacting particles has been extensively investigated by the researchers in various fields. The heat conduction and thermomechanical problems can also be analysed using the thermal discrete element method (TDEM). In the case of particulate systems, Batchelor and O'Brien [10] have addressed the basic mechanisms of heat transfer by using the Hertz's contact theory and the approximate analytical solution. Vargas and McCarthy [11] have introduced the heat transfer algorithm into the DEM. Wu et al. [12] have used Voronoi cells to capture the influence of the packing structure on the heat transfer of monosized pebble beds. Wanne and Young [13] have performed the numerical simulation of thermally fractured granite based on the BPM model, but have not been able to capture the microscopic crack initiation and propagation processes at the cooling stage. All the above numerical studies have made a significant contribution and have proven that the DEM method is a promising approach to simulating the heat transfer and thermomechanical coupling [14]. However, further studies are still required to improve the reliability of the DEM models and to extend the field of their application.

Nevertheless, a long computational time of DEM simulations [15] based on complex models, including fracture or temperature, limits the analysis of industrial-scale applications. The selection of the efficient parallel solution algorithm is highly dependable on the specific characteristics of the considered problem and the numerical method used [16, 17, 18]. The emergence of general purpose GPU (GPGPU) computing seems to offer the possibility to simulate large-scale discrete particle systems, taking advantage of the massive parallel architecture of GPUs and the development of their programming tools, such as CUDA and OpenCL [19]. In general, using single-precision computations and simplified DEM models, not taking into account the tangential contact force or the time history-dependent friction model, results in a significant speedup on a GPU. NVIDIA SDK provided a sample code for DEM [20] and demonstrated the computing speed on the GPU which was over 40 times faster than that on the CPU. However, the contact force employed in the sample code had only a normal component and single-precision computations

were considered. Radeke et al. [21] have suggested an approach to using CUDA for investigating the size effects in granular flows with DEM, allowing for single-precision simulation of more than two million particles per gigabyte of the GPU memory. Govender et al. [15] have designed the modular high performance Blaze-DEMGPU framework for the GPU architecture. Twenty frames per second have been computed assuming the time history-independent friction model in the case of the industrial mill filled with 4 million particles. Considering the same assumption, Xu et al. [22] have achieved quasi-real-time simulation of an industrial rotating drum with 9.6 million particles. Longmore et al. [23] have taken into account particle shape by using multiple spheres representing a sand grain and could perform single-precision simulations of 256 000 tetrahedral granules at 120 milliseconds per time step on the GPU. Kelly et al. [24] have adopted an adimensionalization process combined with mixed-precision data to simulate 3D scenarios with up to 710 million spherical frictionless particles. Washizawa et al. [25] have demonstrated that the computing speed of the practical model, considering more forces between the interacting particles, is 7 times slower than that of the simplified model on the GPU.

Efficient GPU implementations of the practical DEM models are more challenging, because of the increase in the computing time and the required memory, which can reduce the number of the simulated particles. Yue et al. [26] have made a GPU version of the Trubal code and demonstrated its application in die filling. In 3D simulations, containing 20 000 particles, an average speedup of 19.66 has been achieved on NVIDIA Tesla K40c card. Pacevič et al. [9] have implemented the BPM based on the contact bond in the GPU code for simulating the damage and fracture of cohesive solids. Zheng et al. [27] have presented a GPU-based DEM-FEM computational framework implemented by CUDA FORTRAN for simulating the tire-sand interaction. To achieve a higher speedup ratio for a larger number of particles, a few efforts have been made to use the combined GPU and MPI technology [22, 28]. However, the communication overhead among GPUs significantly reduces the parallel performance because of the costly data transfer to the CPU memory and the MPI message passing among different nodes. Therefore, the efficient GPU codes with the implemented practical DEM models for simulating the industrial applications still present challenges, while the quantitative performance assessment remains of practical interest to researchers and engineers. Moreover, to the best of our knowledge, the implementation and evaluation of performance of TDEM on the GPU have not been presented in the literature.

The paper presents an OpenCL implementation of the TDEM and a quantitative comparison of its computational performance with that of various DEM models on the GPU. Other parts of the paper are organized as follows: Section 2 outlines the considered DEM models, Section 3 presents the developed GPU algorithm, Section 4 describes the solved applications, while Section 5 provides the performance analysis and Section 6 gives the concluding remarks.

2 DISCRETE ELEMENT MODELS

The DEM is a class of numerical techniques to simulate the motion of large numbers of particles. An arbitrary particle i in the system of N particles undergoes the translational and rotational motion described in time t as follows:

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

where m_i and I_i are the mass and the moment of inertia of the particle, respectively, while the vectors $\boldsymbol{x}_i$ and $\boldsymbol{\omega}_i$ determine the position of the centre of the particle i and the angular velocity around the particle's centre of mass. The vectors $\boldsymbol{F}_i$ and $\boldsymbol{M}_i$ represent the resultant force and the resultant moment, acting in the centre of the particle i, that can be computed by using the following formulas:

$$\boldsymbol{F}_i = m_i \boldsymbol{g} + \sum_{j=1, j\neq i}^{N_c} \left(\boldsymbol{F}^c_{ij,n} + \boldsymbol{F}^c_{ij,t} \right) + \sum_{k=1, k\neq i}^{N_b} \left(\boldsymbol{F}^b_{ik,n} + \boldsymbol{F}^b_{ik,t} \right), \tag{2}$$

$$\boldsymbol{M}_i = \sum_{j=1, j\neq i}^{N_c} \boldsymbol{M}^c_{ij} + \sum_{k=1, k\neq i}^{N_b} \boldsymbol{M}^b_{ik} \tag{3}$$

where $\boldsymbol{F}^c_{ij,n}$ and $\boldsymbol{F}^c_{ij,t}$ are the normal and tangential vector components of the contact force between the contacting particles that are indicated by subscript $j = 1,\ N_c$, $\boldsymbol{F}^b_{ik,n}$ and $\boldsymbol{F}^b_{ik,t}$ are the normal and tangential vector components of the bond force between the bonded particles that are indicated by subscript $k = 1,\ N_b$, $\boldsymbol{M}^c_{ij}$ and $\boldsymbol{M}^b_{ik}$ are moments of the contacting particles and the bonded particles, respectively, while $\boldsymbol{g}$ is the acceleration due to gravity. In the present work, the electromagnetic force [29], the aerodynamic force [30] and other external forces [31], except for the gravity force, are not considered.

2.1 Granular Flow Model

Simulating granular flows, there are no bonded particles, therefore, N_b, $\boldsymbol{F}^b_{ik,n}$, $\boldsymbol{F}^b_{ik,t}$ and $\boldsymbol{M}^b_{ik}$ are always equal to zero in Equations (2), (3). The normal contact force $\boldsymbol{F}^c_{ij,n}$ can be expressed as the sum of the elastic and viscous components. In the present work, the normal contact force is computed according to the Hertz's contact model. The viscous counterpart of the contact force linearly depends on the relative velocity of the particles at the contact point.

The tangential contact force $\boldsymbol{F}^c_{ij,t}$ is divided into the parts of static friction and dynamic friction. The dynamic friction force is directly proportional to the normal component of the contact force. The static friction force [2, 32] is calculated by summing up the elastic counterpart and the viscous damping counterpart. The slip distance represented by the tangential displacement follows from the temporal integration of the tangential component of the relative velocity, starting at the time

instant, when the particles come into contact. It is worth noting that the length of the tangential displacement depends on the time history. Thus, the considered friction model is incremental or time history-dependent, which requires storing the values of the tangential displacement during the contact between the neighbouring particles in the memory.

Hereinafter, the simplest granular flow model, computing only the normal component of the contact force $\boldsymbol{F}^c_{ij,n}$, is abbreviated to GN. The DEM model, evaluating $\boldsymbol{F}^c_{ij,n}$ and the tangential component of the contact force $\boldsymbol{F}^c_{ij,t}$ with the time history-dependent friction, is termed GNT. The comprehensive granular flow model, which considers $\boldsymbol{F}^c_{ij,n}$, $\boldsymbol{F}^c_{ij,t}$ and the moment $\boldsymbol{M}^c_{ij}$ is abbreviated to GNTM. The details of the granular flow models can be found in the references [2, 4].

2.2 The Bonded Particle Model

The BPM based on parallel bond [3] is implemented for evaluating the performance of the damage and fracture simulations. The parallel bond can be envisioned as a set of elastic springs uniformly distributed over the circular cross-section in 3D, lying on the contact plane and centred on the contact point. Thus, it can transmit both the force and the moment between the bonded particles. The parallel bonds break instantaneously when the tensile stress σ exceeds the tensile strength σ_{lim} or the shear stress τ exceeds the shear strength τ_{lim}, leading to crack formation between two particles. It is worth noting that parallel bonds act in parallel with the granular portion of the force-displacement behaviour, when the depth of the overlap between the particles is more than zero. Moreover, after the breakage of the bonds, the contact forces are calculated according to the granular flow model. The details of the BPM can be found in [3].

2.3 A Thermal Discrete Element Model

The formulation of the TDEM is based on the assumption that the temperature difference inside the particles is negligible and the temperature within the particles can be considered uniform. This assumption is justified for the DEM, employing relatively small particles, and it is consistent with the formulation of the mechanical problem. The heat balance equation can be written for the particle i as follows:

$$m_i c_p \frac{dT_i}{dt} = \sum_{j=1, j\neq i}^{N_t} Q_{ij} \tag{4}$$

where T_i is the temperature of the particle i, c_p is the specific heat capacity of the particle's material, N_t is the number of the bonded particles added to the number of the contacting particles of the particle i, Q_{ij} is the heat conduction flux transmitted by the contact surface or bond between the two neighbouring particles i and j. The heat conduction flux between two particles i and j can be expressed as:

$$Q_{ij} = -H_{ij}^{cont}(T_i - T_j) \tag{5}$$

where H_{ij}^{cont} is the contact conductance coefficient, which might depend on the physical properties of the particles, the contact surface and the relative positions of the particles. In the present work, the contact conductance coefficient is computed by using the following formula [6, 33]:

$$H_{ij}^{cont} = \frac{\lambda A_{ij}^{cont}}{l_{ij}} \tag{6}$$

where λ is the heat conductivity of the material, A_{ij}^{cont} is the area of heat conduction or the contact surface between the particles, l_{ij} is the distance between the centres of the discrete particles i and j. However, the exact values of A_{ij}^{cont} can be obtained only in the particular cases. In the present work, it is assumed that the heat flux conducted through the contact area is equivalent to the heat flux in a bar with the radius equal to the arithmetic mean of the contacting particles' radii [6].

Thermal expansion is considered, calculating the radii of the particles:

$$R_i = R_i^{ini}[1 - \alpha(T_i - T_i^{ini})] \tag{7}$$

where R_i is the radius of the particle i, α is the linear thermal expansion coefficient, while R_i^{ini} and T_i^{ini} are the radius and temperature of the particle i at the initial (reference) time instant, respectively. The details of the implemented TDEM can be found in [6]. Hereinafter, the TDEM implemented with the BPM for simulating the temperature-dependent damage is abbreviated as TBPM.

3 THE DEVELOPED GPU ALGORITHM

The presented algorithm for shared memory architectures is developed to evaluate the computational cost of various DEM models implemented in the GPU code. The DEM code is programmed by using OpenCL [19] to run the same software on all shared memory architectures, including CPUs and GPUs of various vendors. In general, the contact search, the computation of forces and time integration are the most time-consuming procedures in double-precision DEM computations. The main attention is focused on the computation of forces because the implementation and evaluation of different forces is the main distinction of various DEM models. The flowchart of the developed GPU algorithm for double-precision DEM simulations is presented in Figure 1. The parallel algorithm for shared memory architectures can be outlined as follows. At the start of the simulation, preprocessing is performed on the CPU and the initial data are copied from the host memory into the global memory of OpenCL device. No further memory transactions between the CPU host memory and GPU global memory except for the result's storage are required.

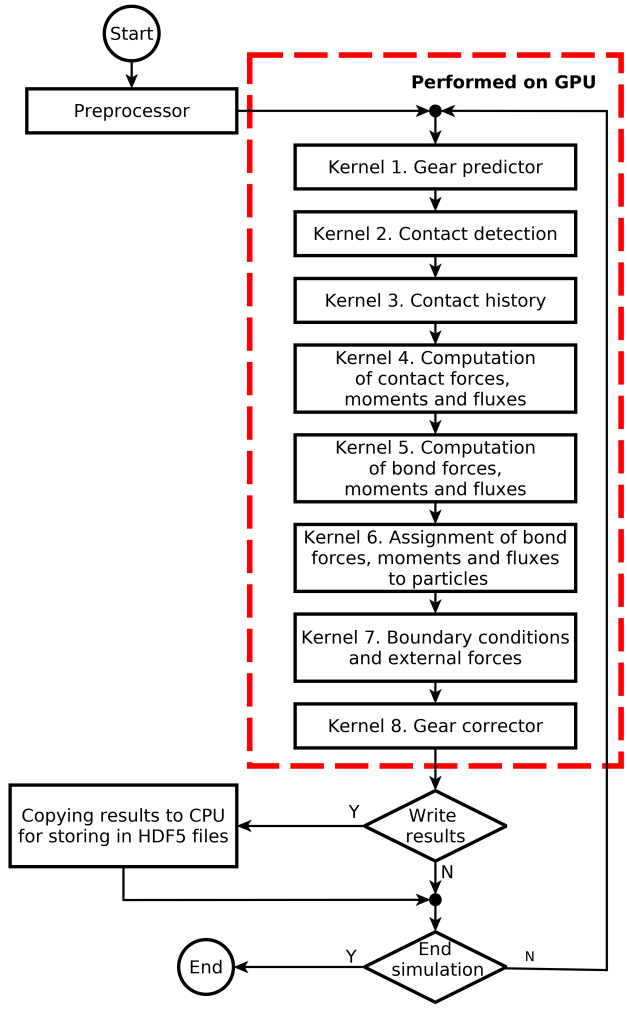

Figure 1. The flowchart of the GPU algorithm for DEM simulations

Kernel 1 starts the time integration of the particles' positions by using the accurate fifth-order Gear predictor-corrector algorithm, see [4] for details. The Gear predictor is performed on the thread per particle basis, which takes advantage of the massive parallel computation capabilities of modern GPUs and can be considered to be the most suitable parallelism in the case of DEM computations. In the loop through particles, the main arrays, such as the position, velocity and acceleration of processed particle, are accessed by using the coalesced pattern. At the beginning of the kernel, the variables of processed particle are copied from the GPU global memory to the thread private memory, which is usu-

ally mapped to registers. It is very efficient, when the copied data is used multiple times. The new values of the positions, velocities and accelerations of particles are predicted at the time increment by a simple series expansion up to the fifth-order of accuracy. The predicted values are copied from the private memory to the global memory at the end of this kernel. The similar data transfers between the GPU global memory and the private memory of threads are performed in all kernels.

Kernel 2 performs the contact detection on the thread per particle basis. The implementation [9] based on the hashed grid is employed in this research. To avoid storing the grid by using a dense array and to save the GPU memory, each cell is mapped into a fixed-size hash table. The fast implementation of the contact detection also exploits the private memory. However, the private memory size is strictly limited. If the private memory is overfilled, the GPU performance is drastically reduced, because overhead of the private memory is mapped to the global memory. In the workgroup, each thread holds information of the 27 neighbour cells in the local memory to decrease the usage of the private memory and to avoid the performance drops. The particles are sorted according to their hash values by using the fast radix sort method. Finally, the narrow phase of the contact search is accomplished, the collisions are identified, and the output of the kernel contains the contact list.

Kernel 3 handles the contact history, which consumes a considerable part of tangential contact force computations. The computation of the elastic counterpart of the static friction force requires the data on the length of the tangential displacement, which depends on the time history. Thus, the values of the tangential displacement during the contact between the neighbouring particles should be stored in the data arrays of the contacts, which considerably increases the consumed memory. In the case of granular flows, the contacts of the neighbouring particles can significantly change in time, which leads to intensive manipulation of the data arrays. Processing each particle, the indices of particle contacts and the coordinates of relevant contact points are copied from the global memory to the local memory of the workgroup to map the contacts of the previous time step to the newly detected contacts of the current time step. Finally, the kernel adds the newly computed increments to the accumulated values of tangential displacement in the relevant contacts.

Kernel 4 computes the contact forces and the moments between all overlapped particles. The normal and tangential components of contact forces include the elastic counterparts, viscous counterparts, the dynamic friction force or the static friction force. Moreover, it also calculates the heat conduction fluxes between the unbonded overlapped particles. All computations are performed based on a thread per particle. Thus, the kernel is executed in the loop through particles. Algorithm 1 presents the pseudocode of Kernel 4. In line 11, ID of the current particle processed by the thread is obtained by OpenCL functionality. The main computations are performed in the loop through particle neighbours, where N_n is the number of neighbours. Then, private variables $\boldsymbol{F}^{total}, \boldsymbol{M}^{total}, Q^{total}$ are initialized

by zero. In line 16, the ID of the neighbouring particle is read from the global memory array by the function *get_neighbour_id()*. The overlap of particles h_{ij} is calculated to perform contact check in the following condition. If the overlap is greater than zero, contact force, moment and heat flux of the processed contact are computed. In lines 23–25, the computed values of the processed contact are added to the private variables $\boldsymbol{F}^{total}, \boldsymbol{M}^{total}, Q^{total}$. At the end of the loop, values of the private variables are written to the output variables stored in the GPU global memory. Material properties extensively used for computation of forces and heat conduction fluxes are stored in the constant memory of GPU, which is very fast, but limited in size and scope. In the case of several materials and thermal dependencies, the array of material properties can occupy the large part of the available constant memory. The constant memory of the GPU is cached, and unlike the private and local memory, it does not need to be copied for each kernel call. Since constant memory can only be modified from the CPU, it is also useful for storage of constant numerical parameters, such as the time step for time integration.

The heat fluxes, moments, normal and tangential components of contact forces require 15, 57, 110 and 107 double-precision floating-point operations for each contact of the processed particle, respectively. It is worth noting that computations of the tangential contact force requires the value of the normal contact force, while-computations of moments use the value of the tangential contact force. Kernel 4 performs $289 \cdot N \cdot N_c$ floating-point operations in each time step. Each particle can have different number of contacts, N_c, which can also vary in time. In the case of monosized particles, the number of contacts cannot be greater than 12. Thus, the number of floating-point operations performed by the kernel is bounded by $3\,468 \cdot N$ or complexity of Algorithm 1 is $O(N)$. The GPU global memory traffic of Kernel 4 also depends on the number of particle contacts. In the case of maximal number of contacts, 2 256 bytes of the GPU global memory are transferred during execution of Kernel 4 for each processed particle. In the case of granular flows or unbonded particles, Kernel 4 can be treated as the main kernel because it computes the contact forces, the moments and heat conduction fluxes, thereby performing a large part of the work.

Kernel 5 computes the forces and the moments of the bonds between the particles. Moreover, it also calculates the heat conduction fluxes between the bonded particles for the TBPM. This kernel works on the thread per bond basis, which is very natural for damage and fracture simulations. Thus, the kernel is executed in the loop through bonds. The obtained increments of the bond forces and the moments as well as the computed values of the heat conduction fluxes (5), (6) are stored in the bonds arrays to avoid the memory writing conflicts, when threads of different bonds simultaneously write the results into the memory of the same particle. Algorithm 2 presents the pseudocode of Kernel 5. In line 13, ID of the current bond processed by the thread is obtained by OpenCL functionality. The following condition checks the bond state, because all computations of the kernel are performed only if the bond is not broken. In line 15, IDs of the bonded particles are read from

Algorithm 1 Computation of contact forces, moments and fluxes

1: **Input**
2: $\boldsymbol{x}$ Positions of particles
3: $\boldsymbol{v}$ Velocity of particles
4: $\boldsymbol{\omega}$ Angular velocity of particles
5: $\boldsymbol{T}$ Temperature of particles
6: **Output**
7: $\boldsymbol{F}$ Forces of particles
8: $\boldsymbol{M}$ Moments of particles
9: $\boldsymbol{Q}$ Heat fluxes of particles
10: **procedure** KERNEL4( )
11: $i \leftarrow get\_global\_id(0)$
12: $\boldsymbol{F}^{total} \leftarrow 0$
13: $\boldsymbol{M}^{total} \leftarrow 0$
14: $Q^{total} \leftarrow 0$
15: **for** $k \leftarrow 0$ to N_n **do**
16: $j \leftarrow get\_neighbour\_id(i,k)$
17: $h_{ij} \leftarrow compute\_overlap(\boldsymbol{x}[i], \boldsymbol{x}[j])$
18: **if** $h_{ij} > 0$ **then**
19: $\boldsymbol{F}^c_{ij,n} \leftarrow compute\_normal\_force(\boldsymbol{x}[i], \boldsymbol{v}[i], \boldsymbol{x}[j], \boldsymbol{v}[j])$
20: $\boldsymbol{F}^c_{ij,t} \leftarrow compute\_shear\_force(\boldsymbol{x}[i], \boldsymbol{v}[i], \boldsymbol{\omega}[i], \boldsymbol{x}[j], \boldsymbol{v}[j], \boldsymbol{\omega}[j])$
21: $\boldsymbol{M}^c_{ij} \leftarrow compute\_moments(\boldsymbol{x}[i], \boldsymbol{v}[i], \boldsymbol{\omega}[i], \boldsymbol{x}[j], \boldsymbol{v}[j], \boldsymbol{\omega}[j])$
22: $Q_{ij} \leftarrow compute\_heat\_fluxes(\boldsymbol{T}[i], \boldsymbol{T}[j])$
23: $\boldsymbol{F}^{total} \leftarrow \boldsymbol{F}^{total} + \boldsymbol{F}^c_{ij,n} + \boldsymbol{F}^c_{ij,t}$
24: $\boldsymbol{M}^{total} \leftarrow \boldsymbol{M}^{total} + \boldsymbol{M}^c_{ij}$
25: $Q^{total} \leftarrow Q^{total} + Q_{ij}$
26: **end if**
27: **end for**
28: $\boldsymbol{F}[i] \leftarrow \boldsymbol{F}^{total}$
29: $\boldsymbol{M}[i] \leftarrow \boldsymbol{M}^{total}$
30: $\boldsymbol{Q}[i] \leftarrow Q^{total}$
31: **end procedure**

the global memory array by the function *get_bond_end_points()*. In lines 16–19, the normal and tangential components of bond forces and moments are obtained adding the computed increments to the current values. In private variables stored values of forces and moments are necessary for computation of the tensile and shear stresses. In line 22, the criteria of the normal and tangential failure of the bond are checked. If the bond is broken zero values are written to output variables and the bond state array is updated. Otherwise, the normal and tangential components of bond forces, the normal and tangential components of bond moments and the computed heat conduction fluxes are written to output variables stored in the GPU global

Algorithm 2 Computation of bond forces, moments and fluxes

Input
  $\boldsymbol{x}$ Positions of particles
  $\boldsymbol{v}$ Velocity of particles
  $\boldsymbol{\omega}$ Angular velocity of particles
  $\boldsymbol{T}$ Temperature of particles
Output
  $\boldsymbol{F}^n$ Normal forces of bonds
  $\boldsymbol{F}^t$ Shear forces of bonds
  $\boldsymbol{M}^n$ Normal directed moments of bonds
  $\boldsymbol{M}^t$ Shear directed moments of bonds
  $\boldsymbol{Q}$ Heat fluxes of bonds
procedure KERNEL5( )
  $id \leftarrow get\_global\_id(0)$
  if $check\_bond\_state(id)$ **then**
    $(i, j) \leftarrow get\_bond\_end\_points(id)$
    $\boldsymbol{F}^n_{ij} \leftarrow \boldsymbol{F}^n[id] + compute\_bond\_normal\_force(\boldsymbol{x}[i], v[i], \boldsymbol{x}[j], \boldsymbol{v}[j])$
    $\boldsymbol{F}^t_{ij} \leftarrow \boldsymbol{F}^t[id] + compute\_bond\_shear\_force(\boldsymbol{x}[i], \boldsymbol{v}[i], \boldsymbol{\omega}[i], \boldsymbol{x}[j], \boldsymbol{v}[j], \boldsymbol{\omega}[j])$
    $\boldsymbol{M}^n_{ij} \leftarrow \boldsymbol{M}^n[id] + compute\_bond\_normal\_moments(\boldsymbol{x}[i], \boldsymbol{v}[i], \boldsymbol{\omega}[i], \boldsymbol{x}[j], \boldsymbol{v}[j], \boldsymbol{\omega}[j])$
    $\boldsymbol{M}^t_{ij} \leftarrow \boldsymbol{M}^t[id] + compute\_bond\_shear\_moments(\boldsymbol{x}[i], \boldsymbol{v}[i], \boldsymbol{\omega}[i], \boldsymbol{x}[j], \boldsymbol{v}[j], \boldsymbol{\omega}[j])$
    $\sigma \leftarrow compute\_tesnsile\_stress(\boldsymbol{F}^n_{ij}, \boldsymbol{M}^t_{ij})$
    $\tau \leftarrow compute\_shear\_stress(\boldsymbol{F}^t_{ij}, \boldsymbol{M}^n_{ij})$
    if $\sigma > \sigma_{lim}$ or $\tau > \tau_{lim}$ **then**
      $\boldsymbol{F}^n[id] \leftarrow 0$
      $\boldsymbol{F}^t[id] \leftarrow 0$
      $\boldsymbol{M}^n[id] \leftarrow 0$
      $\boldsymbol{M}^t[id] \leftarrow 0$
      $\boldsymbol{Q}[id] \leftarrow 0$
      $mark\_bond\_as\_broken(id)$
    else
      $\boldsymbol{F}^n[id] \leftarrow \boldsymbol{F}^n_{ij}$
      $\boldsymbol{F}^t[id] \leftarrow \boldsymbol{F}^t_{ij}$
      $\boldsymbol{M}^n[id] \leftarrow \boldsymbol{M}^n_{ij}$
      $\boldsymbol{M}^t[id] \leftarrow \boldsymbol{M}^t_{ij}$
      $\boldsymbol{Q}[id] \leftarrow compute\_heat\_fluxes(\boldsymbol{T}[i], \boldsymbol{T}[j])$
    end if
  end if
end procedure

memory. 504 bytes of the GPU global memory are transferred during execution of Kernel 5 for each unbroken bond. The heat fluxes, moments, normal and tangential components of bond forces require 24, 65, 156 and 32 double-precision floating-point operations for each unbroken bond, respectively. Thus, Kernel 5 performs maximum $277 \cdot N_b$ floating point operations in each time step. The computational complexity of Algorithm 2 is $O(N_b)$. The number of bonds, N_b, remains constant during computations. However, the number of unbroken bonds can decrease in each subsequent time step.

Kernel 6 finishes the algorithm started by the previous kernel, assigning the contact forces, the moments and heat conduction fluxes, computed by the previous kernel in the bonds, to the processed particle. It processes the values of the forces and the moments available in the data arrays of the bonds and stores the results in the arrays of the particles. The values of the heat conduction fluxes are summed up and stored in a similar way. This kernel works on the thread per particle basis as opposed to the previous kernel, performing the computations on the thread per bond basis. Thus, the concurrent memory writing and atomic operations are avoided to ensure high parallel performance at the cost of the increased memory usage and the complexity of the algorithm.

Kernel 7 is aimed at computing the boundary conditions and the external forces. However, only the gravitational force (2) is required to solve the applications considered in the present study. The symmetry boundary conditions can be defined by fixing the positions of the particles located on the symmetry planes in the normal direction. The velocity of the boundary particles can be specified. The known values of the temperature or the heat conduction flux can be set in the boundary particles in the case of TDEM computations.

Kernel 8 completes the time integration, performing the Gear corrector on the thread per particle basis. The values of the positions, velocities, temperatures and other variables of the particles are copied from the global memory to the private memory, corrected and loaded back to the global memory. It is worth noting that time integration of the angular velocities (1) and temperatures (4) is performed by using the Euler's scheme in this kernel. At the end of the time step, the particles' data can be copied from the GPU global memory to the CPU host memory for storage on the hard disk drive in HDF5 format. It is recommended to transfer the data to the host memory as seldom as possible because it is a time-consuming process. GPU computations can be performed concurrently with the additional CPU thread writing results to HDF5 file. When all the time steps are finished, the DEM simulation is ended.

Computationally intensive code for calculating the contact and bond forces is grouped in several kernels, because very large kernels can lead to performance drops. In the previous version of the code, all computations of forces, moments and thermal conduction fluxes were performed by one large kernel. However, the kernel overfilled the private memory, and its overhead was automatically mapped to the global memory. The overall performance of the code was approximately reduced by 20 % of the execution time.

The implemented DEM models are tightly coupled, therefore, it is difficult to efficiently implement them in separate kernels. The TDEM is often required to simulate the granular flows, as well as the thermally induced damage. Thus, the computations of heat conduction fluxes are implemented to simulate the granular flow temperature in Kernel 4. Moreover, the values of the normal contact force can be used to compute the area of heat conduction between the particles calculated by using Hertz's contact theory [10, 11]. The heat conduction fluxes are computed to handle the temperature-dependent material damage by the TBPM in Kernels 5 and 6. Thermal expansion, which represents the direct coupling of thermal and mechanical problems, is evaluated by calculating the radii of the particles (7) at the end of the time step in Kernel 8. Thus, the computations of the TDEM are implemented in the main kernels, working with the forces, moments, time integration and boundary conditions.

4 THE CONSIDERED APPLICATIONS

Various applications are solved by the developed OpenCL code for the performance analysis and evaluation of the computational costs of various DEM models.

4.1 Gravity Packing

The gravity packing problem of granular material, falling under the influence of gravity into a container, is considered to investigate the computational costs of the DEM models because it often serves as a test problem for performance measurements [21, 28]. The solution domain is assumed to be a cubic container with the 2.0 m long edges. Half of the domain is filled with monosized particles, using a face-centred cubic structure. Granular material is represented by an assembly of 83 300, 686 000, 1 362 944 and 3 015 300 particles with the radii equal to 0.02 m, 0.01 m, 0.008 m and 0.0061 m, respectively. The initial velocities of the particles are defined randomly with uniform distribution, with their magnitudes being in the range of $[0.0; 0.1]$ m/s. The physical data of the particles of the artificially assumed material are as follows: the density is equal to $920\,\mathrm{kg/m^3}$, the Poisson's ratio is 0.352, the elasticity modulus is equal to $9.33 \cdot 10^6$ Pa and the friction coefficient is 1.0. Viscous damping coefficients in the normal and tangential directions are assumed to be equal to 1 500 and 150, respectively. The simulation of the granular material falling under the influence of gravity is interrupted after 1.5 s, when the particles reach the state of rest with the negligibly small average acceleration.

4.2 Uniaxial Tension of Reinforced Concrete

The tensile test of reinforced concrete prisms instrumented with the internal strain gauges [34] is used for evaluating the computational performance of the BPM. The reinforced concrete prisms of $150.0 \times 150.0 \times 270.0$ mm and the reinforcing bars

with the diameter of 20.0 mm are tested. The height, width and pitch of the ribs are equal to 1.3 mm, 2.0 mm and 120°, respectively. The distance between the ribs is 8.0 mm. Due to the axial symmetry, only one eighth of the reinforced concrete specimen is modelled. The particles located at the end of the steel reinforcing bar are moved with the specified constant velocity in y direction, which simulates the uniaxial tension. The symmetry boundary conditions are defined on the opposite plane, setting the displacement of concrete and steel particles to zero in y direction. The normal components of the displacements of the particles are also fixed on the other two symmetry planes, crossing each other on the y axis. The reinforced concrete is represented by an assembly of 88 872, 627 264, 1 377 576 and 3 000 848 monosized spherical particles. The curve, representing the reinforcing bar with ribs, is rotated about the y axis to define the interface surface between the concrete and the reinforcing steel bar. Sufficiently fine discretization is required to accurately describe the interface surface, which leads to large numbers of particles [9].

4.3 Heating of Refractory Concrete

Sharp temperature changes on the refractory concrete lining of thermal equipment, such as biomass-fired furnaces, cause a large temperature gradient, which leads to the formation of a high stress in the refractory material of the lining. Despite the advances in experimental measurements of the temperature and numerical methods, the prediction of the lining's lifespan still remains an important challenge. Therefore, the heating and cooling of refractory concrete is considered to help validate the TBPM results and to study the computational performance of its GPU implementation. $200 \times 200 \times 200$ mm specimens of refractory concrete were produced for the experiment. The time of curing of the specimens is 48 h. The specimen is mounted on the bottom of the laboratory furnace, where drying and firing of concrete take place to avoid cracking caused by vapour formation during the sharp heating. The mechanical properties of concrete are as follows: density is equal to 2 250 kg/m^3, cold compressive strength is equal to 106 MPa, bending strength is 13.5 MPa and the elasticity modulus is equal to 23.2 GPa. The thermal properties of concrete are as follows: specific heat capacity is 900 J/kgK, the heat conductivity is 1.5 W/mK and the thermal expansion coefficient is equal to $0.66 \cdot 10^{-6} C^{-1}$. To compare the performance with that of other DEM models the specimen is represented by an assembly of 87 808, 665 500, 1 372 000 and 3 014 284 monosized particles. In the TBPM, the above-mentioned macro-level values of the specific heat capacity, heat conductivity and the thermal expansion coefficient are used along with the calibrated parameter relevant to the area of heat conduction (6) or the bar radius.

Heating is applied to the upper surface of the specimen, measuring the temperature by a thermocouple. A cycle of heating and cooling is performed according to the following mode: the temperature rises to 900°C at 300°C/h, while retaining the temperature of 900°C for 30 minutes and cooling is performed with the open cover of the furnace (400°C/h). Figure 2 shows the comparison of the experimental measurements with the numerical results of the TBPM. The measured temperature

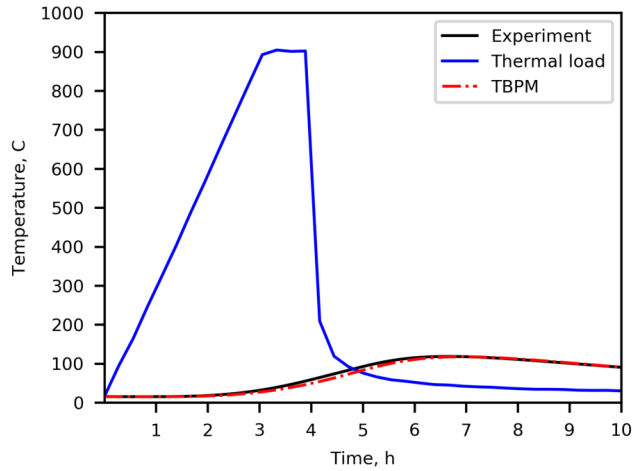

Figure 2. The comparison of the experimental measurements and the results of the TBPM

(the curve "Thermal load" in Figure 2) is specified as the boundary condition on the upper square surface of the solution domain. All sides of the specimen are thermally isolated, therefore, a zero heat flux is specified as the boundary condition on the relevant surfaces of the solution domain. The bottom surface of the specimen is surrounded by the air of the temperature reaching 20°C. Thus, at the bottom of the solution domain the mixed boundary conditions are specified, which leads to heat convection from the heated surface to the air of the ambient temperature of 20°C. Figure 2 shows a good agreement of the TBPM results (the curve "TBPM") with the experimental temperature measurements (the curve "Experiment") at the bottom of the specimen.

5 THE PERFORMANCE ANALYSIS

The implementations of the considered DEM models have been validated comparing the obtained solutions with the experimental measurements. The implemented BPM has been validated solving the uniaxial tension of reinforced concrete problem in [9]. The validation of the TBPM against the experimental measurements of temperature has been presented in Figure 2.

Thus, the applications of gravity packing, uniaxial tension of reinforced concrete and heating of refractory concrete are considered to evaluate the computational performance of various DEM models implemented on the GPU. All double-precision computations are performed on the NVIDIA® Tesla™ P100 GPU Computing Accelerator (56 Streaming Multiprocessors, 1 792 FP64 CUDA Cores, 12 GB HBM2, 549 GB/s memory bandwidth). Hardware characteristics of the workstation

used for quantitative comparison of parallel performance are listed below as follows: Intel®Xeon™ E5-2630 2.20 GHz 2 × CPU, 32 GB DDR4 2 133 MHz RAM.

The host code is compiled with GCC 7.3.0 and -O3 optimization flag. OpenCL 2.0 is employed for CPU computations, while OpenCL 1.2 is used for GPU computations, because NVIDIA® Tesla™ P100 does not support higher OpenCL version. The default compilation and optimization flags are used for compilation of kernels. No floating-point arithmetic optimization is enabled to preserve accuracy of numerical solutions. The HDF5 version 1.8.20 is used to store results on the hard disk drive. The actual computation time of 100 000 time steps is measured to investigate the computational efficiency of the developed OpenCL code in the case of various numbers of discrete particles. Preprocessing, visualization and computations are performed on the computational infrastructure [35] of the Vilnius Gediminas Technical University.

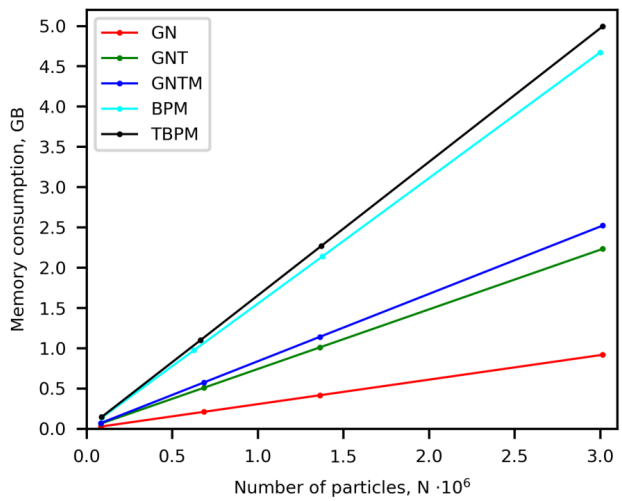

Figure 3. The memory consumed by the considered DEM models

Figure 3 shows the amount of memory consumed by various DEM models on the GPU. In the legend, the names of curves "GN", "GNT", "GNTM", "BPM" and "TBPM" match the abbreviations of the considered models. As expected, the simplest granular flow model GN requires the smallest amount of memory, which does not reach 1 GB in the case of double-precision simulation of 3 015 300 particles. The evaluation of the tangential component of the contact force with the time history-dependent friction model is very expensive, therefore, the GNT model consumes 2.4 times the memory of the GN model. The computation of the moments is not so expensive in terms of the used memory as it adds only 13.0 % of the memory used by the GNT model.

The applications of gravity packing, the uniaxial tension of reinforced concrete and the heating of refractory concrete are solved by using various numbers of particles. However, the linear dependency of the consumed memory on the number of particles can be observed in Figure 3 and, therefore, linear interpolation is applied. The BPM significantly increases the used memory up to 86.2 % of the memory required for the comprehensive model of the granular flows (GNTM) or up to 237.2 % of the memory used by the GN model. On the contrary, the TBPM increases the memory used by the BPM only up to 33.8 % of the memory used by the GN model. However, the most expensive TBPM model, including the computations of the bonded particles, as well as granular flows, requires 5.4 times the memory of the simplest GN model, evaluating only the normal component of the contact force. It is worth noting that a large amount of memory is required by the DEM models performing the operations and storing the results in the data arrays of the bonds or contacts between the neighbouring particles.

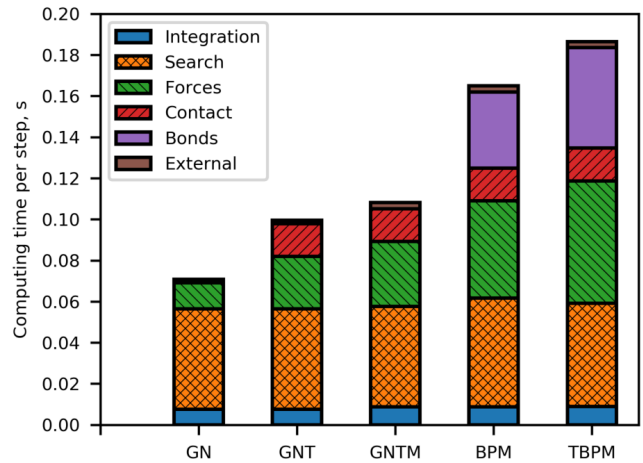

Figure 4. The contribution of computational procedures to the total benchmark time

Figure 4 shows the contribution of the execution time of computational procedures implemented as GPU kernels (Figure 1) to the total time of the considered benchmarks based on various DEM models. The columns represent the computing time of the performed benchmarks in the case of the largest numbers of particles, exceeding three million. The abbreviations of the models denote the same things as the curves in Figure 3. The names of the columns ("Integration", "Search", "Forces", "Contact", "Bonds" and "External") represent the time integration (Kernels 1 and 8), the contact detection (Kernel 2), the computations of the contact forces, the moments and heat fluxes (Kernel 4), the handling of the contact history (Kernel 3), the computation and assignment of the forces, the moments and heat

fluxes of the bonds (Kernels 5 and 6) as well as the evaluation of the external forces and the boundary conditions (Kernel 7).

The impact of the computation of the gravity force and the boundary conditions on the benchmark time is very small. It depends on the solved application and varies between 1.5 % and 2.8 % of the relevant execution time. It can be observed that time integration always consumes a similar amount of time. However, increasing the benchmark time decreases the consumed time percentage from 10.7 % (the GN column) to 4.8 % (the TBPM column) of the relevant benchmark time. The contact search takes 68.9 % of the benchmark time in the case of the GN model. However, in the case of the TBPM model, the consumed time percentage is much lower, reaching 26.3 % of the benchmark time. Handling the contact history required for the computation of the tangential force component always needs a similar period of time, except for the GN model, which considers only the normal component of the contact force. Kernel 3 performs intensive manipulation of data arrays, storing the information of the contact history, which might significantly change in time. Its percentage of the execution time varies from 16.0 % to 8.6 % of the relevant varying benchmark time. The computations of the contact forces, the moments and heat fluxes take from 18.0 % to 31.9 % of the relevant benchmark time. The time consumed by this kernel strongly depends on the DEM model. In the case of the GNT model, the computing time of the contact forces is two times as long as that of the GN model. In the case of the GNTM model, the computation of the moments adds 24.5 % of the computing time of the forces. In the case of the BPM based on the parallel bond, the computation of the bond forces and moments by the kernels 5 and 6 is rather expensive and takes 22.4 % of the benchmark time. Moreover, in the case of the TBPM model, the kernels 5 and 6 also compute the heat fluxes of the bonds and consume 26.2 % of the benchmark time. It can be concluded that the time required for computing the forces, the moments and the heat fluxes varies most strongly in the context of various DEM models.

Figure 5 shows the scaling performance of the considered DEM models and the quantitative comparison of the performance measured on the GPU and the CPU. The average execution time of the same OpenCL code of the time step is measured on NVIDIA® Tesla™ P100 (GPU) and dual Intel® Xeon™ E5-2630 (CPU) with 20 physical cores to evaluate the speedup ratio of CPU to GPU. In the legend, the abbreviation “GPU” represents the execution time measured on the GPU, while the abbreviation “CPU” and the dotted lines denote the benchmark time measured on the 20 cores of CPU. The abbreviations of the DEM models have the identical meanings given in Figure 3. In the case of the granular flow model (GNTM), the speedup ratio of CPU to GPU varies from 3.0 to 7.2 for various numbers of particles. In the cases of more time-consuming models (BPM and TBPM), the speedup ratio is up to 7.4. It is worth noting that OpenCL intensively uses the advanced vector extensions on the sufficiently powerful dual Intel®Xeon™ E5-2630 CPU with 20 physical cores, which considerably reduces the measured speedup ratio. In absolute values, the measured Cundall number (FPS×number of particles) varies from $1.62 \cdot 10^7$ to $4.45 \cdot 10^7$, depending on the complexity of the DEM model implemented on the

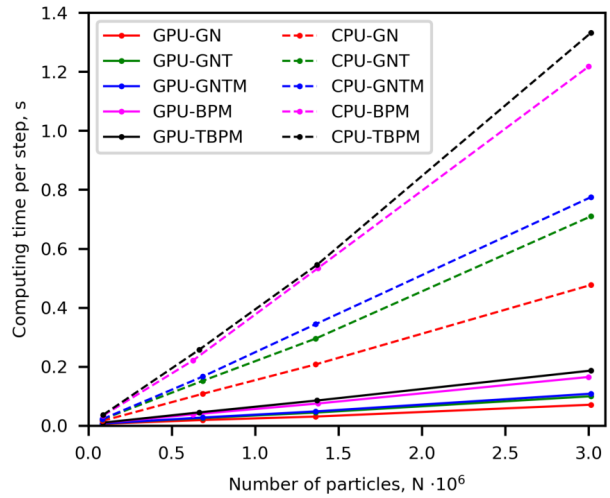

Figure 5. The scaling performance of the considered DEM models

GPU. As expected, the largest Cundall number equal to $4.45 \cdot 10^7$ is achieved in the case of the simplest GN model. The comprehensive granular flow model GNTM allows for obtaining the Cundall number equal to $2.83 \cdot 10^7$. The lowest numbers $1.84 \cdot 10^7$ and $1.62 \cdot 10^7$ can be observed in the cases of the complex BPM and TBPM models, respectively.

Figure 6 shows the computational overhead of the implemented DEM models and supplements the results presented in Figure 6 in the case of the largest numbers of the particles, exceeding three million. The overhead is calculated according to the execution time of the simplest GN model, which computes only the normal component of the contact force. The abbreviations of the DEM models used in the legend have the meanings identical to those given in Figure 3. The performance analysis reveals that the simplest DEM model (GN) requires the smallest amount of computational resources. The computation of the tangential component of the contact force with the time history-dependent friction model is very expensive and increases the computing time up to 40.5 % of the time required for the GN model on the GPU. The computation of the moments is less expensive, adding to the GNT model up to 12.1 % of the execution time of the GN model, which results in the total overhead equal to 52.6 % of the execution time of the GN model on the GPU. In the case of the CPU, this overhead is higher and is equal to 62.4 % of the GN model's execution time.

The nearly linear dependency of the computing time on the number of particles can be observed in Figure 5, therefore, linear interpolation is applied to computing the time of various applications solved by using slightly varying numbers of particles. The application of the BPM results in another significant increase in the

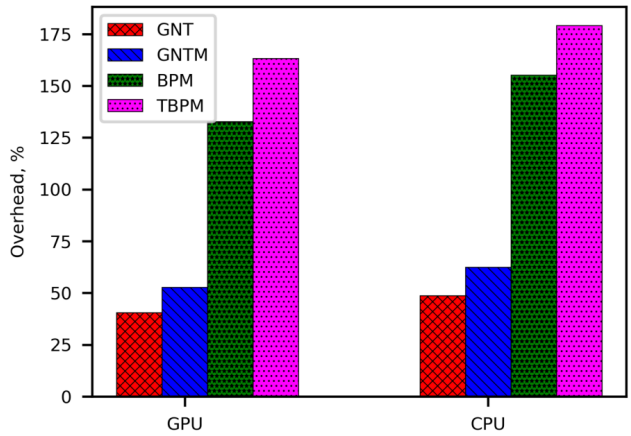

Figure 6. The computational overhead of the implemented DEM models compared to the normal contact force model GN

computing time, which is qualitatively similar to the increase caused by handling the contact history, which is required for computing the tangential contact force. Assuming linear dependency, the BPM adds to the granular flow model GNTM 81.2 % of the execution time of the GN model, which results in the total overhead equal to 133.6 % of the execution time of the GN model. It is worth noting that the parallel bond is very expensive in terms of computational performance compared to simpler bond implementations [9]. Moreover, to perform the quantitative comparison with other DEM models the contact search is performed in each time step, which is rarely required for damage simulations. The computations of temperature and heat fluxes is not so costly, therefore, the TBPM adds to the computing time of the BPM 30.6 % of the benchmark time of the GN model. However, the most expensive TBPM model, including the computations of granular flows and bonded particles, adds the overhead equal to 164.5 % of the execution time of the simplest GN model on the GPU. In the case of the CPU, the measured overhead is even larger, reaching 180.5 % of the execution time of the simplest GN model. Moreover, the percentage of the increased memory required by the complex DEM models is even higher.

Figure 7 presents the basic roofline model, considering the GPU global memory traffic. The horizontal axis represents the arithmetic intensity (Flops/byte), which means the number of double-precision floating-point operations per byte of GPU global memory transfers incurred during the execution of a kernel. The vertical axis represent double-precision floating-point operations performed by a kernel per second (Flops/s). Both axes are in logarithmic scale. The curve “NVIDIA

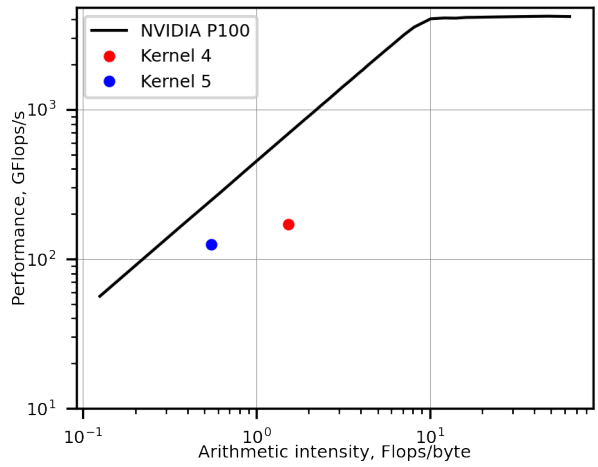

Figure 7. The basic roofline model

P100" represents the peak bandwidth ceiling (diagonal line) and the peak performance ceiling (horizontal line) measured by mixed operational intensity OpenCL kernels of GPU synthetic benchmark suite Mixbench [36]. The points "Kernel 4" and "Kernel 5" represent performance of the computationally intensive kernels for calculating the forces, the moments and the heat conduction fluxes of the TBPM model. Kernel 4 computes the contact forces between all overlapped particles, while Kernel 5 computes the bond forces between bonded particles. It is evident that all TBPM computations are memory bound because they use many arrays stored in the GPU global memory. Kernel 5 accesses even more global memory than Kernel 4 because the number of bonds is larger than the number of particles. Finally, Kernel 4 performs more double-precision floating-point operations per second than Kernel 5. However, the performance of Kernel 5 is closer to the performance of the synthetic GPU benchmark than that of Kernel 4.

6 CONCLUSIONS

The paper presents the OpenCL implementation of the thermal discrete element method and the quantitative comparison of its computational performance with that of various DEM models on shared memory architectures. Based on the performed study, some observations and concluding remarks may be drawn as follows:

- The performance analysis reveals that a relatively high CPU to GPU speedup ratio up to 7.4 has been achieved in spite of the intensive usage of the advanced vector extensions by the OpenCL on the 20 core CPU.

- The time required for the computation of the forces, the moments and heat fluxes of the bonds and the contacts between the particles is highly dependent on the complexity of the considered DEM model, while all relevant Kernels 3, 4, 5 and 6 can generally consume from 18 % to 66.7 % of the benchmark execution time.
- The GN model, considering only the normal component of the contact force, requires the shortest computing time because of the simple computations performed only on the thread per particle basis.
- The GNT model is very expensive in terms of the memory and the computing time because of the time history-dependent length of the tangential displacement, which requires the storing and processing of the contact lists of variable size.
- The difference in the computing time between the BPM based on the parallel bond and the comprehensive granular flow model GNTM varies from 30.7 % to 81.2 % of the execution time of the GN model, which is the largest increase in computational resources among the investigated DEM models.
- The DEM models, performing the operations and storing the results in the data arrays of the bonds or contacts between the neighbouring particles, require the large amount of memory and long computing time.
- The presented implementation of the TBPM increases the used memory and the computing time of the BPM model up to 33.8 % and 30.6 % of the memory and the execution time of the simplest GN model, respectively, which is an acceptable increase in the computational resources required for modelling the additional coupled field.

Acknowledgement

The present research is part of the project No. 09.3.3-LMT-K-712-02-0131, funded under the European Social Fund measure "Strengthening the Skills and Capacities of Public Sector Researchers for Engaging in High Level R & D Activities", administered by the Research Council of Lithuania.

REFERENCES

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

[2] Džiugys, A.—Peters, B.: An Approach to Simulate the Motion of Spherical and Non-Spherical Fuel Particles in Combustion Chambers. Granular Matter, Vol. 3, 2001, No. 4, pp. 231–266, doi: 10.1007/PL00010918.

[3] Potyondy, D. O.—Cundall, P. A.: A Bonded-Particle Model for Rock. International Journal of Rock Mechanics and Mining Sciences, Vol. 41, 2004, No. 8, pp. 1329–1364, doi: 10.1016/j.ijrmms.2004.09.011.

[4] Kačeniauskas, A.—Kačianauskas, R.—Maknickas, A.—Markauskas, D.: Computation and Visualization of Discrete Particle Systems on gLite-Based Grid. Advances in Engineering Software, Vol. 42, 2011, No. 5, pp. 237–246, doi: 10.1016/j.advengsoft.2011.02.007.

[5] Tiscar, J. M.—Escrig, A.—Mallol, G.—Boix, J.—Gilabert, F. A.: DEM-Based Modelling Framework for Spray-Dried Powders in Ceramic Tiles Industry. Part II: Solver Implementation. Powder Technology, Vol. 377, 2021, pp. 795–812, doi: 10.1016/j.powtec.2020.08.095.

[6] Rojek, J.: Discrete Element Thermomechanical Modelling of Rock Cutting with Valuation of Tool Wear. Computational Particle Mechanics, Vol. 1, 2014, pp. 71–84, doi: 10.1007/s40571-014-0008-5.

[7] Kačianauskas, R.—Rimša, V.—Kačeniauskas, A.—Maknickas, A.—Vainorius, D.—Pacevič, R.: Comparative DEM-CFD Study of Binary Interaction and Acoustic Agglomeration of Aerosol Microparticles at Low Frequencies. Chemical Engineering Research and Design, Vol. 136, 2018, pp. 548–563, doi: 10.1016/j.cherd.2018.06.006.

[8] Long, X.—Ji, S.—Wang, Y.: Validation of Microparameters in Discrete Element Modeling of Sea Ice Failure Process. Particulate Science and Technology, Vol. 37, 2019, No. 5, pp. 550–559, doi: 10.1080/02726351.2017.1404515.

[9] Pacevič, R.—Kačianauskas, R.—Kačeniauskas, A.—Kaklauskas, G.—Barauskas, R.: Fast GPU Simulation of Reinforced Concrete at the Scale of Reinforcement Ribs by the Discrete Element Method. Archives of Mechanics, Vol. 71, 2019, No. 4-5, pp. 459–488, doi: 10.24423/aom.3148.

[10] Batchelor, G. K.—O'Brien, R. W.: Thermal or Electrical Conduction Through a Granular Material. Proceedings of The Royal Society A: Mathematical, Physical and Engineering Sciences, Vol. 355, 1977, pp. 313–333, doi: 10.1098/rspa.1977.0100.

[11] Vargas, W. L.—McCarthy, J. J.: Heat Conduction in Granular Materials. AIChE Journal, Vol. 47, 2001, No. 5, pp. 1052–1061, doi: 10.1002/aic.690470511.

[12] Wu, H.—Gui, N.—Yang, X.—Tu, J.—Jiang, S.: Numerical Simulation of Heat Transfer in Packed Pebble Beds: CFD-DEM Coupled with Particle Thermal Radiation. International Journal of Heat and Mass Transfer, Vol. 110, 2017, pp. 393–405, doi: 10.1016/j.ijheatmasstransfer.2017.03.035.

[13] Wanne, T. S.—Young, R. P.: Bonded-Particle Modeling of Thermally Fractured Granite. International Journal of Rock Mechanics and Mining Sciences, Vol. 45, 2008, No. 5, pp. 789–799, doi: 10.1016/j.ijrmms.2007.09.004.

[14] Chen, L.—Wang, C.—Moscardini, M.—Kamlah, M.—Liu, S.: A DEM-Based Heat Transfer Model for the Evaluation of Effective Thermal Conductivity of Packed Beds Filled with Stagnant Fluid: Thermal Contact Theory and Numerical Simulation. International Journal of Heat and Mass Transfer, Vol. 132, 2019, pp. 331–346, doi: 10.1016/j.ijheatmasstransfer.2018.12.005.

[15] Govender, N.—Rajamani, R. K.—Kok, S.—Wilke, D. N.: Discrete Element Simulation of Mill Charge in 3D Using the BLAZE-DEM GPU Framework. Minerals Engineering, Vol. 79, 2015, pp. 152–168, doi: 10.1016/j.mineng.2015.05.010.

[16] Bystrov, O.—Kačeniauskas, A.—Pacevič, R.—Starikovičius, V.—

Maknickas, A.—Stupak, E.—Igumenov, A.: Performance Evaluation of Parallel Haemodynamic Computations on Heterogeneous Clouds. Computing and Informatics, Vol. 39, 2020, No. 4, pp. 695–723, doi: 10.31577/cai_2020_4_695.

[17] Šešok, D.—Belevičius, R.—Kačeniauskas, A.—Mockus, J.: Application of GRID Computing for Optimization of Grillages. Mechanika, Vol. 82, 2010, No. 2, pp. 63–69.

[18] Govender, N.: Study on the Effect of Grain Morphology on Shear Strength in Granular Materials via GPU Based Discrete Element Method Simulations. Powder Technology, Vol. 387, 2021, pp. 336–347, doi: 10.1016/j.powtec.2021.04.038.

[19] Du, P.—Weber, R.—Luszczek, P.—Tomov, S.—Peterson, G.—Dongarra, J.: From CUDA to OpenCL: Towards a Performance-Portable Solution for Multi-Platform GPU Programming. Parallel Computing, Vol. 38, 2012, No. 8, pp. 391–407, doi: 10.1016/j.parco.2011.10.002.

[20] Green, S.: Particle Simulation Using CUDA. NVIDIA Whitepaper, 2010, 12 pp.

[21] Radeke, C. A.—Glasser, B. J.—Khinast, J. G.: Large-Scale Powder Mixer Simulations Using Massively Parallel GPU Architectures. Chemical Engineering Science, Vol. 65, 2010, No. 24, pp. 6435–6442, doi: 10.1016/j.ces.2010.09.035.

[22] Xu, J.—Qi, H.—Fang, X.—Lu, L.—Ge, W.—Wang, X.—Xu, M.—Chen, F.—He, X.—Li, J.: Quasi-Real-Time Simulation of Rotating Drum Using Discrete Element Method with Parallel GPU Computing. Particuology, Vol. 9, 2011, No. 4, pp. 446–450, doi: 10.1016/j.partic.2011.01.003.

[23] Longmore, J. P.—Marais, P.—Kuttel, M. M.: Towards Realistic and Interactive Sand Simulation: A GPU-Based Framework. Powder Technology, Vol. 235, 2013, pp. 983–1000, doi: 10.1016/j.powtec.2012.10.056.

[24] Kelly, C.—Olsen, N.—Negrut, D.: Billion Degree of Freedom Granular Dynamics Simulation on Commodity Hardware via Heterogeneous Data-Type Representation. Multibody System Dynamics, Vol. 50, 2020, No. 4, pp. 355–379, doi: 10.1007/s11044-020-09749-7.

[25] Washizawa, T.—Nakahara, Y.: Parallel Computing of Discrete Element Method on GPU. Applied Mathematics, Vol. 4, 2013, No. 1A, pp. 242–247, doi: 10.4236/am.2013.41A037.

[26] Yue, X.—Zhang, H.—Ke, C.—Luo, C.—Shu, S.—Tan, Y.—Feng, C.: A GPU-Based Discrete Element Modeling Code and Its Application in Die Filling. Computers and Fluids, Vol. 110, 2015, pp. 235–244, doi: 10.1016/j.compfluid.2014.11.020.

[27] Zheng, Z.—Zang, M.—Chen, S.—Zeng, H.: A GPU-Based DEM-FEM Computational Framework for Tire-Sand Interaction Simulations. Computers and Structures, Vol. 209, 2018, pp. 74–92, doi: 10.1016/j.compstruc.2018.08.011.

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

[29] Tumonis, L.—Kačianauskas, R.—Kačeniauskas, A.—Schneider, M.: The Transient Behavior of Rails Used in Electromagnetic Railguns: Numerical Investi-

gations at Constant Loading Velocities. Journal of Vibroengineering, Vol. 9, 2007, No. 3, pp. 15–19.

[30] Stupak, E.—Kačianauskas, R.—Kačeniauskas, A.—Starikovičius, V.—Maknickas, A.—Pacevič, R.—Staškūnienė, M.—Davidavičius, G.—Aidietis, A.: The Geometric Model-Based Patient-Specific Simulations of Turbulent Aortic Valve Flows. Archives of Mechanics, Vol. 69, 2017, No. 4-5, pp. 317–345.

[31] Liu, G.—Marshall, J. S.—Li, S. Q.—Yao, Q.: Discrete Element Method for Particle Capture by a Body in an Electrostatic Field. International Journal for Numerical Methods in Engineering, Vol. 84, 2010, No. 13, pp. 1589–1612, doi: 10.1002/nme.2953.

[32] Mindlin, R. D.—Deresiewicz, H.: Elastic Spheres in Contact Under Varying Oblique Forces. Journal of Applied Mechanics, Vol. 20, 1953, No. 3, pp. 327–344, doi: 10.1115/1.4010702.

[33] Terreros, I.—Iordanoff, I.—Charles, J. L.: Simulation of Continuum Heat Conduction Using DEM Domains. Computational Materials Science, Vol. 69, 2013, pp. 46–52, doi: 10.1016/j.commatsci.2012.11.021.

[34] Jakubovskis, R.—Kaklauskas, G.: Bond-Stress and Bar-Strain Profiles in RC Tension Members Modelled via Finite Elements. Engineering Structures, Vol. 194, 2019, pp. 138–146, doi: 10.1016/j.engstruct.2019.05.069.

[35] Kačeniauskas, A.—Pacevič, R.—Starikovičius, V.—Maknickas, A.—Staškūnienė, M.—Davidavičius, G.: Development of Cloud Services for Patient-Specific Simulations of Blood Flows Through Aortic Valves. Advances in Engineering Software, Vol. 103, 2017, pp. 57–64, doi: 10.1016/j.advengsoft.2016.01.013.

[36] Konstantinidis, E.—Cotronis, Y.: A Quantitative Roofline Model for GPU Kernel Performance Estimation Using Micro-Benchmarks and Hardware Metric Profiling. Journal of Parallel and Distributed Computing, Vol. 107, 2017, pp. 37–56, doi: 10.1016/j.jpdc.2017.04.002.

Ruslan Pacevič is Associated Professor at the Department of Graphical Systems of Vilnius Gediminas Technical University (VGTU), Lithuania. He is also the postdoctoral research fellow at the Department of Applied Informatics of Kaunas University of Technology. He received his Ph.D. in informatics engineering from VGTU in 2015. He is the co-author of several scientific papers and participant of several European and national research projects. His research interests include distributed, grid and cloud computing, DEM, development of SaaS and middleware components, OpenStack, Eucalyptus, visualization software, GPGPU, OpenCL.

Arnas Kačeniauskas is the Director of the Institute of Applied Computer Science of Vilnius Gediminas Technical University (VGTU), Lithuania, and the Chief Researcher at the Laboratory of Parallel Computing. He is also Professor and Ph.D. supervisor at the Department of Graphical Systems at VGTU. He received his Professor in informatics engineering and Ph.D. in mechanical engineering at VGTU. He is the R & D Project Manager, author and co-author of 38 scientific journal papers indexed in Clarivate Analytics WoS database journals. His research interests include parallel, distributed, grid and cloud computing, DEM, high-performance computing, performance of SaaS, Linux containers, CFD, haemodynamics, coupled problems in multiphysics, GPGPU.