Large-scale Dynamic and Static Simulations of Complex-shaped Granular Materials Using Parallel Three-dimensional Discrete Element Method (DEM) on DoD Supercomputers

Beichuan Yan[a,*], Richard A. Regueiro[a]

[a] *Department of Civil, Environmental, and Architectural Engineering, University of Colorado Boulder*

Abstract

In this paper we present unprecedented three-dimensional (3D) DEM simulations of complex-shaped particles across five orders of magnitude of simulation scale (namely, number of particles) using up to 32,768 cores on U.S. Department of Defense (DoD) supercomputers. Firstly, we develop the parallel algorithm based on domain decomposition by following Foster's four-step design methodology and incorporating a number of advances, including concept of link-block (LB), border layer and migration layers, adaptive compute gridding technique, MPI transmission of C++ objects and pointers, etc. Secondly, performance analyses of the code including speedup, efficiency, scalability are provided as benchmarks and guidelines, as well as optimal computational granularity (CG) for each simulation scale. Thirdly, three full-scale simulations of sand pluviation, constrained collapse and particle shape effect are carried out to demonstrate the code capability as well as discover mechanical response of particle assemblies.

The parallel code enables us to simulate a wide range of dynamic and static laboratory and field tests in engineering applications that involve a large number of granular and geotechnical material grains, such as sand pluviation process, buried explosion in various soils, earth penetrator interaction with soil, influence of grain size, shape and gradation on packing density and

*Corresponding author. Fax: +1-303-492-7317

Email addresses: beichuan.yan@colorado.edu (Beichuan Yan), richard.regueiro@colorado.edu (Richard A. Regueiro)

shear strength, mechanical behavior under different gravity environments such as on the Moon and Mars, etc.

Keywords: large-scale, parallel computing, discrete element, complex-shaped, pluviation, collapse

---

1. INTRODUCTION

Since its introduction in the late 1970s by Cundall and Strack (1979), the Discrete Element Method (DEM) has been applied to study the mechanical and micro-mechanical behavior of particle assemblages for more than 40 years. However, application of 3D DEM to simulating practical problems involving granular and geomechanical materials is still limited in terms of problem size (namely, number of particles). For example, most applications involving complex-shaped particles such as axi-symmetric/revolution ellipsoids (Ng, 1994, 2004), three-axis ellipsoids (Yan et al., 2010), poly-ellipsoids (Peters et al., 2009), superellipsoids (Wellmann et al., 2008; Delaney et al., 2010), superquadrics (Williams and Pentland, 1992), constrain their number of particles to a few thousand. This is mainly due to the fact that the DEM poses high computational demands characterized by CPU-intensive interparticle contact detection and explicit time integration schemes.

Under many situations, the length scale and problem size of practical engineering problems cannot be circumvented even if a multiscale model is employed. For instance, to study the impact of blast waves on gravitationally deposited coarse-grained soils in which an explosive charge is ignited, a 40 cm x 40 cm x 40 cm specimen composed of ellipsoidal sand particles contains approximately 500 million 0.1$\sim$1 mm diameter particles depending on the particle shapes and size distribution. The limitation of problem size poses a barrier to simulating realistic engineering or laboratory problems involving complex-shaped particles that are the same size as physical particles.

It is a natural and indispensable trend to take advantage of modern supercomputers to perform large-scale computational tasks of three-dimensional (3D) DEM using parallel computing. This paper presents a study on the parallelization of 3D DEM, its performance measurement and analysis, and full-scale simulations of dynamic and static laboratory and engineering problems. It contains six sections. Section 1 has stated the motivation, mainly from the perspective of engineering applications; section 2 overviews the DEM framework, its computational features in neighbor search and contact

resolution, and recent interest in its parallelism; section 3 covers numerous concepts, techniques, optimizations and latest programming technologies in the process of MPI parallelization; section 4 presents performance analysis across five orders of magnitude of simulation scale using numerical experimental data collected from DoD supercomputers, including MPI profiling, optimal computational granularity (CG), formulation on execution time, communication time and parallel overhead percentage with regard to number of processors (strong scaling) and problem size (weak scaling), and FLOPS measurement using the Performance API (PAPI); section 5 presents three full-scale simulations of sand pluviation, constrained collapse and particle shape effect to discover mechanical response of particle assemblies that are impossible without large-scale parallel computing capability; the last section gives conclusion and outlook.

2. STATE-OF-THE-ART DEM AND ITS PARALLELISM

2.1. The DEM framework

A typical procedure of DEM analysis consists of three major computational steps in sequence, which are integrated in time using central difference method until a simulation is completed:

- contact detection between particles, including two phases:
  1. *neighbor search (neighbor estimate)*
  2. *contact resolution*
- contact force computation for each pair of particles in contact.
- particle motion update (translations and rotations) using Newton's second law.

The contact detection process is usually the major computational bottleneck, especially for a large number of complex-shaped particles. It is divided into two phases: neighbor search (or spatial reasoning) phase and contact resolution phase. Neighbor search identifies/estimates objects near the target object. It often uses an approximate geometry for the objects, such as bounding box or bounding sphere. The geometric contact resolution phase then uses a specific geometric representation of each body to resolve the contact geometry. For complex shapes such as ellipsoidal particles (three different semi-axis lengths) (Yan et al., 2010) or non-symmetric poly-ellipsoidal particles (Peters et al., 2009), the contact resolution between two particles

is much more expensive than spheres, increasing the floating point operations by several orders of magnitude due to the requirement of numerical accuracy and robustness. This is the most computationally challenging part of 3D DEM in addition to the non-linear and history-dependent mechanical models that describe interparticle interactions.

2.2. Neighbor search

In DEM simulations, there are three typical neighbor search algorithms with different time complexities: $O(n^2)$, coming from n-by-n simple search; $O(n\log n)$, resulting from *tree-based algorithms* (Jagadish et al., 2005; Muja and Lowe, 2009); and $O(n)$, rooted from *binning method* (Munjiza and Andrews, 1998; Williams et al., 2004) or link-cell (LC) method (Grest et al., 1989), where n denotes the number of particles.

2.3. Contact resolution

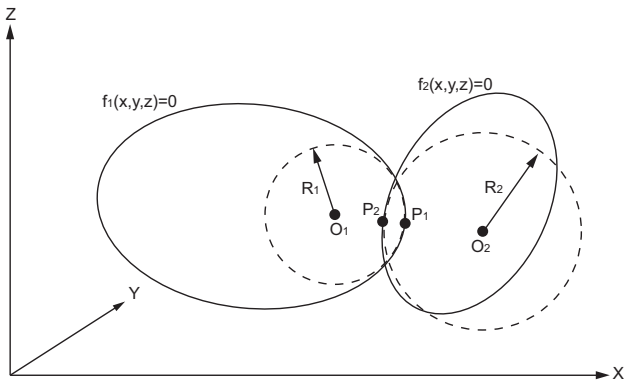
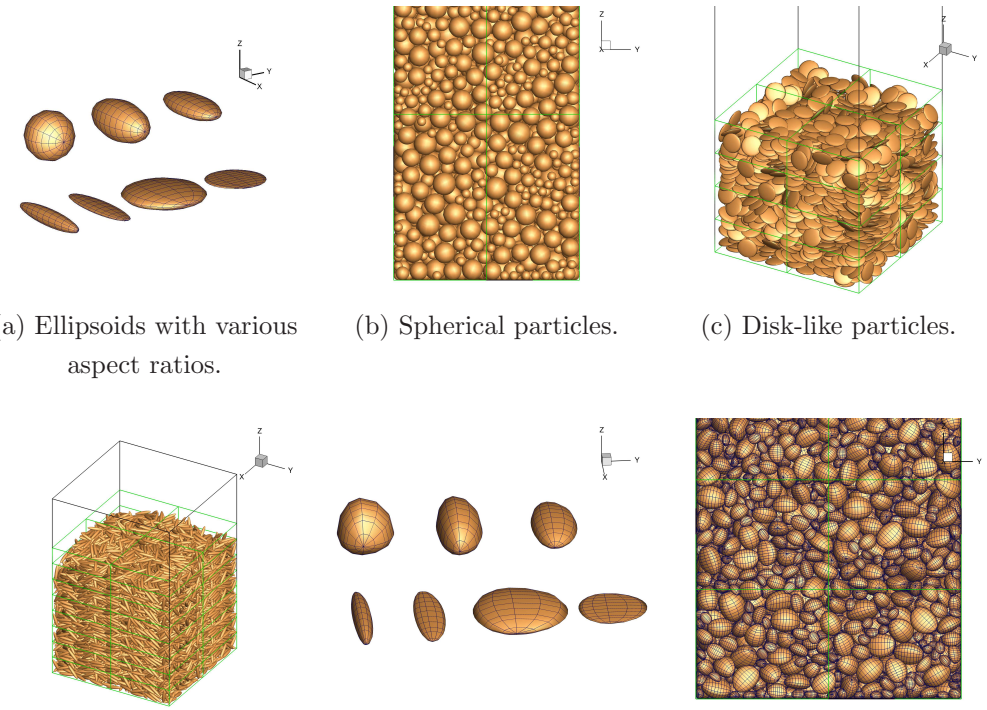


Figure 1: Contact between two ellipsoidal particles.

Yan et al. (2010) developed a robust contact resolution algorithm for three-axis ellipsoidal particles by constructing an *extreme value problem* of finding the deepest penetration of one particle into the other, as shown in Figure 1. Such an extreme value problem results in a sixth order polynomial equation. Conventional polynomial root finders cannot satisfy the high-precision numerical requirement in the 3D DEM computation. For example, the elastic overlap between two particles of typical quartz sand may vary between 10^{-8} to 10^{-5} meters depending on particle size, shape and external force, and a low-precision solver can lead to numerical instability or

spurious explosion of particles. Therefore, an iterative eigenvalue method is selected to find roots of the polynomial and determine the contact geometry. The algorithm and its implementation has been shown to be robust such that it is applicable to not only regularly bulky ellipsoidal shapes but also extreme-shaped ellipsoidal particles such as disks and needles, as shown in Figure 2(a∼d).



(a) Ellipsoids with various aspect ratios.

(b) Spherical particles.

(c) Disk-like particles.

(d) Needle-like particles.

(e) Poly-ellipsoids with various aspect ratios.

(f) Poly-ellipsoids.

Figure 2: Ellipsoids and poly-ellipsoids represent a wide variety of shapes in DEM.

Peters et al. (2009) proposed a non-symmetric poly-ellipsoid shape which joins eight component ellipsoids in eight different octants respectively to produce continuous surface coordinates, normal directions and intersections. It is more computationally expensive than a symmetric ellipsoid but it acts as a useful extension, as shown in Figure 2(e∼f). Zhang et al. (2017) described further details on poly-ellipsoid contact detection algorithm for improved computational efficiency.

2.4. Contact model and damping mechanism

A typical interparticle contact model is composed of Hertzian nonlinear normal contact model (Hertz, 1882), Mindlin's history-dependent shear model (Mindlin, 1949; Mindlin and Deresiewicz, 1953), Coulomb friction and contact damping mechanism, illustrated in Figure 3, where k_t and k_n denote tangential and normal stiffness between two particles in contact, respectively, and μ denotes coefficient of friction.

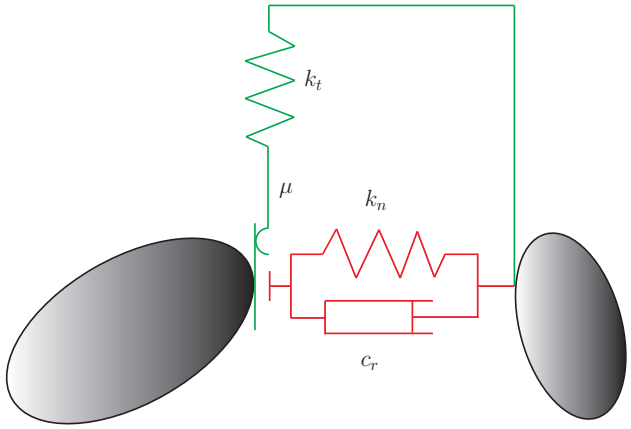


Figure 3: Model of contact interface.

The normal damping coefficient c_r (Onate and Rojek, 2004) can be taken as a fraction of the critical damping C_{cr} for the system of two rigid bodies with masses m_1 and m_2, connected with a spring of stiffness k_n (Taylor and Preece, 1992):

$$C_{cr} = 2\sqrt{\frac{m_1 m_2 k_n}{m_1 + m_2}} \quad (1)$$

$$c_r = \xi C_{cr} \quad (2)$$

where ξ is called damping ratio, which is usually $40 \sim 100\%$ for particles' interaction. In the tangential direction, Mindlin's hysteresis model and Coulomb friction work together to dissipate energy.

2.5. Weight of neighbor search

It is worth noting that the three neighbor search algorithms, $O(n^2)$, $O(n\log n)$ and $O(n)$, only affect the performance of neighbor search. They

have no bearing on contact resolution. The overall performance improvement resulting from these algorithms is highly limited for complex-shaped particles, because neighbor search only takes up a small fraction of floating point operations in the whole computation. For instance, contact resolution between a pair of three-axis ellipsoids is approximately 50 times as expensive as that of a pair of spheres, and contact resolution between a pair of poly-ellipsoids is nearly 300 times as expensive as that of a pair of spheres.

Both $O(n^2)$ and $O(n)$ algorithms are implemented and tested in the paper. The more complex the particle shapes are, the smaller the neighbor search fraction (NSF) is, whereby NSF is defined as the ratio of neighbor search time to the total contact detection time. With poly-ellipsoid computation by $O(n)$ algorithm, the NSF is as low as 0.7% whereas contact resolution takes up to 95.5%. Furthermore, the lower the computational granularity (CG), i.e., number of particles per process, the smaller the NSF.

Yan and Regueiro (2016a) pointed out: in both serial and parallel computing of complex-shaped 3D DEM, the $O(n^2)$ neighbor search algorithm is inefficient at coarse CG, however it executes faster than the $O(n)$ algorithm at fine CGs that are mostly employed in computational practice.

2.6. Efforts in developing parallel DEM

Firstly it is emphasized that this work is focused on DEM, not Molecular Dynamics (MD). MD and DEM are essentially very different computational methods in that: (a) MD simulations have become prominent because of the availability of accurate interatomic potentials for a range of materials, whereas granular particles in DEM generally interact with each other through direct contacts and friction. In DEM simulations the significant majority of floating point operations is consumed on contact geometry resolution, while that is not the case for MD whereby spheres are used; (b) rotation of granular particles plays a critical role in determining the deformation and strength of assemblies, while that is not the case with atoms or molecules in MD; (c) granular materials are usually frictional materials, for which particle size, shape, size distribution and change of boundary conditions have strong bearing on the assembly mechanical properties.

Secondly, there have been considerable efforts in developing parallel DEM codes in recent years. Baugh Jr and Konduri (2001) presented a distributed computing system for DEM that is designed for loosely-coupled networks of workstations. The implementation is used to simulate a system with as many as 200,000 spherical particles using eight processors. As an example,

the speedup is close to 6 using 8 processors for the computation of 120k spherical particles.

Washington and Meegoda (2003) simulated a triaxial test using an algorithm titled "TRUBAL for Parallel Machines (TPM)" and showed its benefits over the serial version DEM code, TRUBAL. The TPM assigns each processor a multiple number of contact pairs (two spheres in contact) existing within an assembly of spheres using static memory arrangement. Two simulation scales are tested and compared, one with 403 spheres and the other with 1,672 spheres, on Connection Machine (CM-5) with 512 nodes at the Pittsburgh Supercomputing Center. The speedup is as low as 7.9 using 512 nodes for 403 spheres.

Henty (2000) chose to investigate the performance of a much smaller test code that implements precisely the same algorithm but has limited functionality rather than tackle a complete physics DEM application with all of its functionality and complexity. The superlinear speedup is observed in his tests.

Maknickas et al. (2006) described the DEMMAT_PAR code for simulation of visco-elastic frictional granular media, which has been created in the Parallel Computing Laboratory of Vilnius Gediminas Technical University. The code adopts a static spatial domain decomposition strategy, link-cell concept and MPI inter-processor communication. The speedup is approximately 11 on 16 processors for 100,000 spherical particles.

Munjiza et al. (2009) adopted parallel particle mesh (PPM) library to parallelize the DEM, and used a conservative lower bound to estimate the number of time steps between two Verlet list updates. In a sand avalanche simulation, 122 million spheres were used. In the figure generated from fixed-size problems that use up to 192 processors, superlinear speedup appears to occur when the number of processors increases to a certain value.

Vedachalam and Virdee (2011) used LAMMPS (large-scale atomic and molecular massively parallel simulator) and LIGGGHTS (LAMMPS improved for general granular and granular heat transfer simulations) to study the motion of snow particles, wherein the snow grains are assumed to be spherical particles of 5 mm diameter. With regard to the performance gain of parallelism, the authors wrote "on 480 processors for 75K particles, the speedup was 1.99, while on 960 processors for same number of particles speedup achieved was 2.52" in comparison to 120 processors.

These DEMs mentioned above share several weaknesses and challenges:

1. They only deal with spheres rather than complex-shaped particles, thus leading to several orders of magnitude less computational demand. For instance, the CPU demand of simulating 122 million spheres is merely equivalent to that of simulating 488k poly-ellipsoids, and the CPU demand of simulating 10 million ellipsoids approaches that of 500 million spheres, if the same interparticle contact mechanical models are used.
2. They highly simplify the complicated interparticle contact models and thus cannot capture the physical properties of granular material accurately. For instance, Munjiza et al. (2009) assumes constant normal and tangential elastic modulus, and LAMMPS assumes variable normal and tangential elastic modulus; however, both ignore the well-known history-dependent tangential behavior for granular particles represented by the Hertz-Mindlin contact model, which needs special implementation to keep track of the complex load-unloading-reloading path and history variables. Munjiza et al. (2009) updates the Verlet lists every 150 time steps, but interparticle contacts can generate and disappear, and lasting shear contact behavior can evolve, at every time step in conventional DEM, not to mention in high-fidelity simulations like soil-buried explosion.
3. They do not completely disclose or implement advanced parallelism requirements such as memory consumption management, dynamic load balancing technique, transmission of dynamically allocated objects between MPI processes, particle motion tracking mechanism across MPI processes, etc, which set a ceiling on computational performance gain and scalability. For example, the root process may deplete compute node memory in parallel computing, if it does not implement an effective memory deallocation mechanism during the process of time integration of millions of steps; a particle could disappear in computation when it moves across dynamically adaptive compute grids, if particle migration mechanism is not implemented correctly.
4. As presently reported in the literature, they achieve poor parallel speedup and scalability. As an example of LAMMPS, 4× number of processors gives rise to a low speedup of 1.99, and 8× number processors leads to a low speedup of 2.52, in the simulation of 75k spheres (Vedachalam and Virdee, 2011). Among these parallel DEMs, the algorithmic speedup and scalability challenges mainly come from: (i) strategy of static vs dynamic spatial domain decomposition; (ii) performance-critical details in design and implementation of the MPI transmission of adjacent particles from one process to another, which are thoroughly covered in Section 3.

Peters et al. (2009) divides the usage of DEMs into two classes of application: (i) prototype-scale simulations for engineering studies, and (ii) micromechanical studies for research of fundamental mechanics. They pointed out: "Prototype scale analyses require accurate bulk behavior of the medium, which presumably can be achieved without capturing details at the particle scale. In fact, in such studies both particle size and shape are sacrificed to obtain problem sizes suitable for practical computations. For micromechanical studies, greater fidelity to the actual particle characteristics is needed, because for simulation to be on equal footing with physical experiments, the particle-scale behavior must be correct. Simply reproducing bulk behavior is not sufficient if the intent is to develop generalizations about fundamental mechanics." Unfortunately the spherical DEMs mentioned above fall into the category of prototype-scale simulations, unless of course simple granular materials like spherical glass beads are studied fundamentally, not realistic sands or gravels.

3. MPI DESIGN, IMPLEMENTATION AND OPTIMIZATION

3.1. Four-step design and link-block

A concept of link-block (LB) is put forward for the design of the parallel algorithm, illustrated in Figure 4. With introduction of LB, the Foster's four-step methodology (Foster, 1995) can be readily applied:

Partitioning: The computational domain is divided into blocks. Each block may consist of many virtual cells. In Figure 4, there are 8 blocks numbered from 0 to 7, each containing 5 x 5 x 5 small virtual cells. The size of the virtual cells may be chosen to be the maximum diameter of the discrete particles.

Communication: Each cell, as a primitive task unit, can communicate with 26 possible surrounding ones to determine contact detection. However, the communication manner may be changed after the process of agglomeration.

Agglomeration: By combining 5 x 5 x 5 virtual cells into a block, communication overhead is lowered in that each block only needs to communicate with neighboring blocks through border/ghost layers, which are virtual cells marked by blue dots in Figure 4.

Mapping: There are choices of mapping a block of particles to a core, a CPU, multiCPUs within a node, or even a whole node. Very often each block is mapped to a whole compute node.

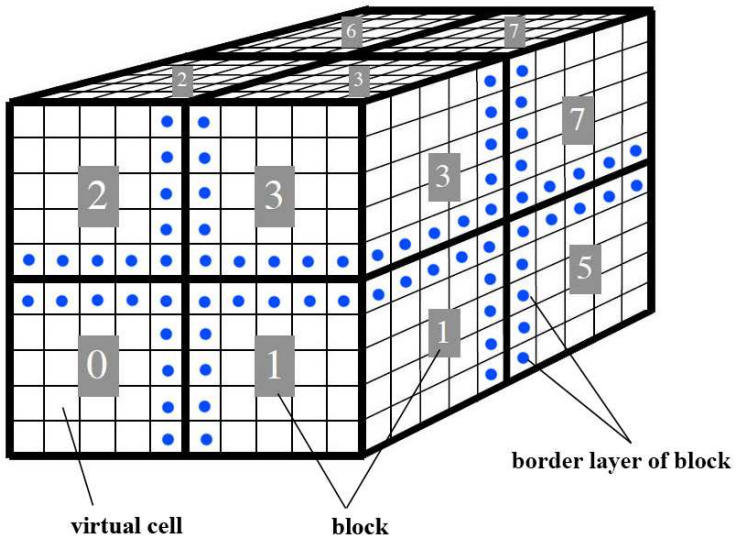


Figure 4: Schematic of link-blocks, virtual cells and border layers.

3.2. Flowchart of the parallel algorithm

The flowchart of the parallel code is designed as depicted in Figure 5. It exhibits twelve flow processes or steps, among which one is sequential, two are partially parallel, and nine are fully parallel. Ten of the twelve processes are integrated in time using time increment loops until a simulation is completed.

In comparison to the serial algorithm, the parallel one ends up with six more steps as follows:

1. step 2: 2-Root process divides and broadcasts info. This step only runs once so it does not cost much CPU time.
2. step 3: 3-All processes communicate with neighbors. This interprocess communication is the most important step in the parallel algorithm.
3. step 9: 9-All processes update compute grids. This step arises from consideration of computational load balance.
4. step 10: 10-All processes merge and output info. This step serves the goal of snapshotting simulation states. Beware that it does not execute at each time increment, otherwise it could cause unacceptable cost.
5. step 11: 11-All processes release memory of receiving particle info. This step arises from MPI transmission mechanism and must be carefully taken care of.

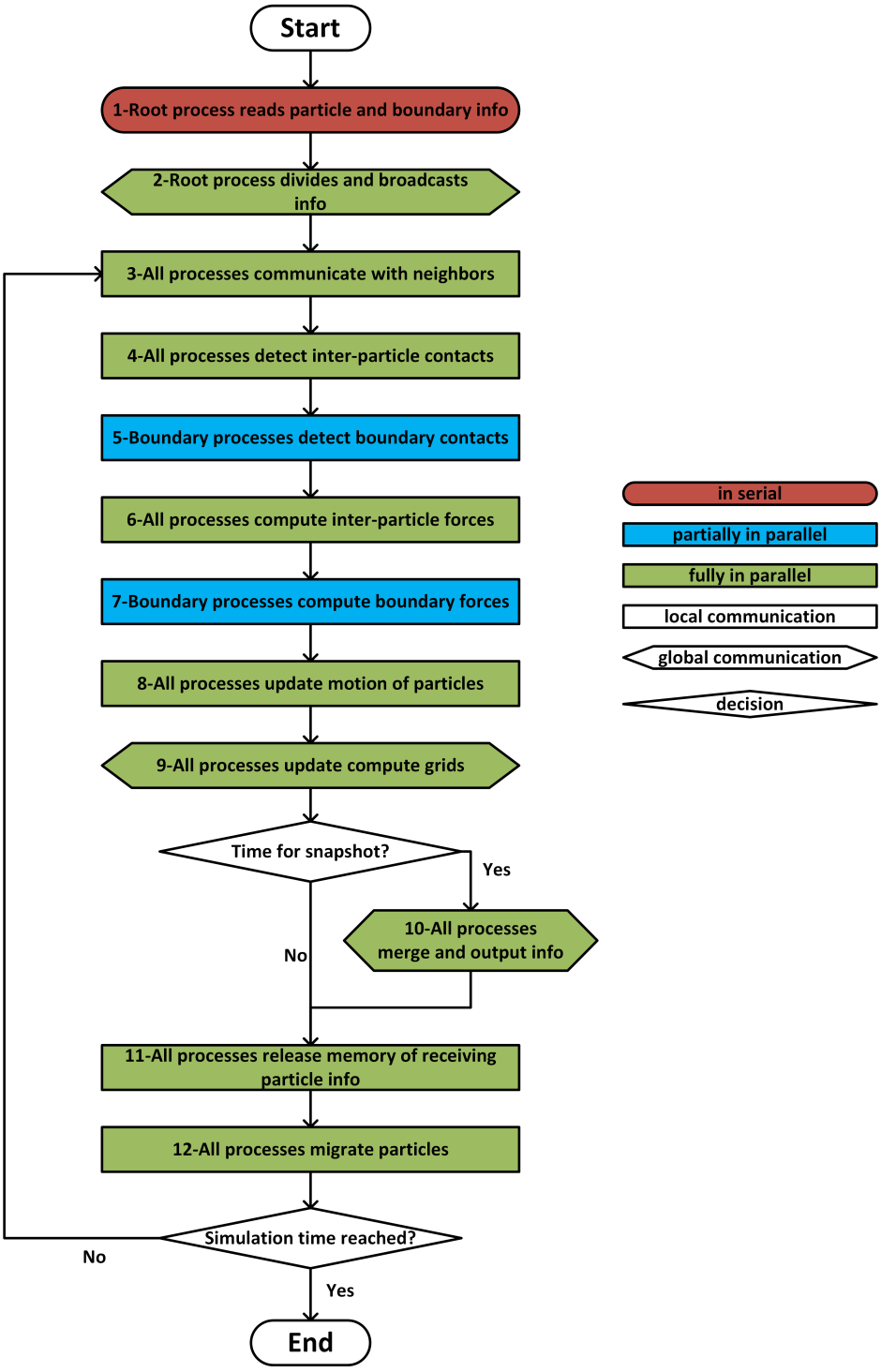


Figure 5: Flowchart of the parallel algorithm of 3D DEM.

6. step 12: 12-All processes migrate particles. This step handles the situation when particles move across block borders.

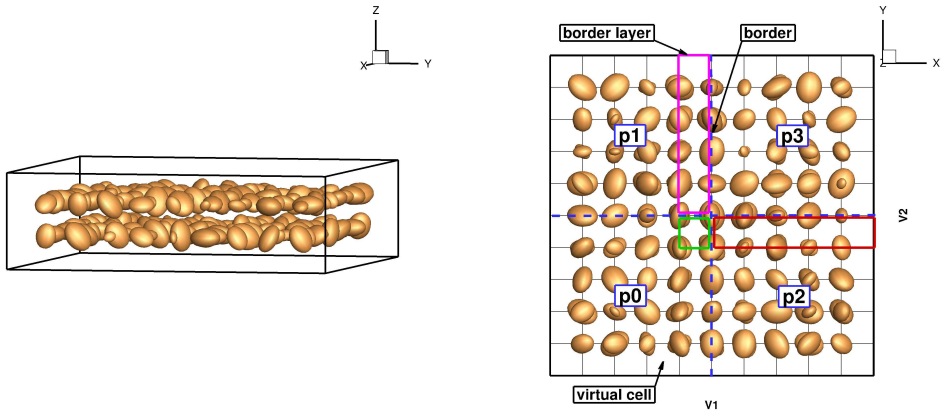
Note steps 5 and 7 are boundary processes: although they are partially parallel in spatial distribution (only boundary processes are running while other processes are idle), these two steps only perform two-dimensional operations on computational boundaries, therefore taking up a relatively small fraction of computational cost. Among the twelve steps, most of them only involve local communication, while three of them are associated with global communication.

3.3. Interblock communication

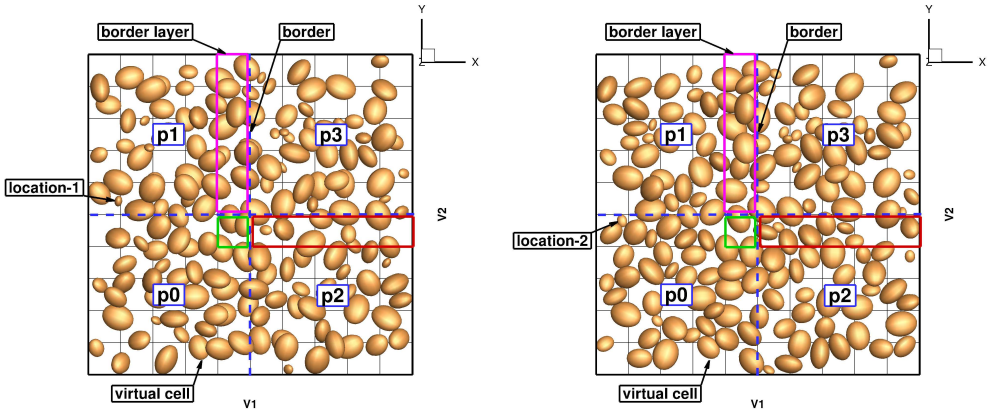
In Figure 4, beware that a border/ghost layer is not limited to constructing a surface layer between two adjacent blocks, as there are other forms. For example, block 3 communicates with block 1 through a surface border layer, block 3 communicates with block 0 through an edge border layer, while block 3 communicates with block 4 through a vertex border layer, as shown in Figure 4. Usually a block needs to exchange information with its neighbors through six surface border layers, twelve edge border layers and eight vertex border layers. The width of border layers is selected to be the radius of the largest particle and works well.

A "patch" test is designed using 162 ellipsoidal particles. The particle assemblage is composed of two layers of 81 particles, gravitationally deposited into a rigid container, illustrated in Figure 6(a). The container is partitioned into four blocks separated by blue dashed lines shown in Figure 6(b), which also represents the initial configuration of the randomly-sized particles as shown from top view.

Each block is mapped to and computed by an individual process, so there are four processes, p0 to p3. Each process needs to communicate with other processes to determine its own boundary conditions. For example, process p3 needs to know those particles from process p1 that are enclosed by the pink rectangular box, those from process p2 enclosed by the red rectangular box, and those from process p0 enclosed by the green square box. A detailed movie records how those particles move across the borders and collide with particles from other blocks, and it reveals that each process is able to determine its boundary conditions accurately. The overall motion of the 162 particles through parallel computing is observed to be the same as that observed in serial computing.



(a) 3D view of initial configuration (b) Top view of initial configuration



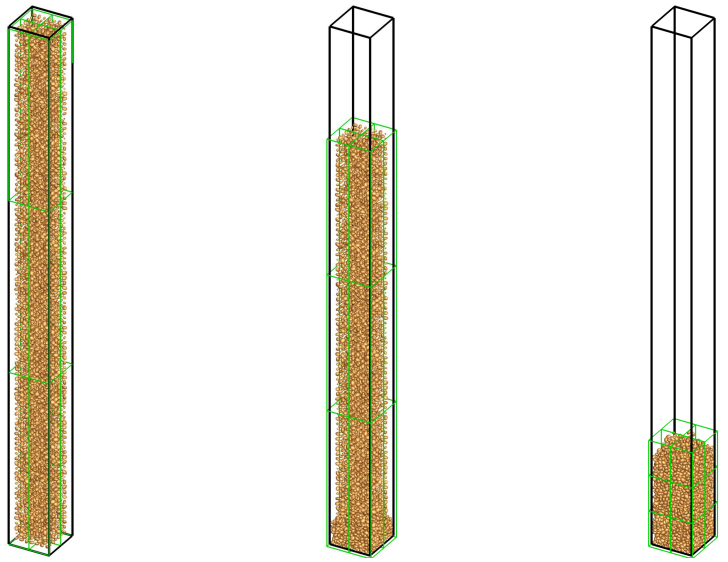
(c) Top view at time $t1$ during simulation (d) Top view at time $t2$ during simulation

Figure 6: Illustration of interblock communication.

3.4. Load balance and adaptive compute grids

In parallel computing it is important to maintain load balance between processes, otherwise some processes are busy computing while others could be hungry awaiting tasks. To this end, dynamically adaptive compute grids are developed in 3D DEM. In a test of pouring particles into a container via gravity, it can be clearly observed from the movie that the compute grids dynamically follow motion of the particles and redistribute across the space. Three snapshots of this process are captured and shown in Figure 7, where the 2 x 2 x 3 compute grids in x, y, z direction respectively are marked

by green boxes (differentiated from the fixed black box of the container). Note that compute grids must be distinguished from the physical container in terms of underlying data structure because compute grids are a dynamic data structure while a container uses a fixed one.



(a) Initial configuration of particle pluviation.

(b) Middle stage of particle pluviation.

(c) Final stage of particle pluviation.

Figure 7: Dynamically adaptive compute grids that achieve efficient load balance.

3.5. Across-border migration

Each process only knows its own space and associated particles, and a particle may enter or leave this space, which is called *across-block migration*. If a particle migrates across the block border, one process needs to delete this particle while another process needs to add this particle. Consider a small particle located at location-1 in process p1 at time $t1$ in Figure 6(c): it moves across the border of process p1 and enters the domain of process p0 at a later time $t2$, arriving at location-2 in Figure 6(d).

The algorithm to track how particles migrate across block/process borders is depicted by Figure 8. It looks similar to that of inter-process communication, however they are different or even the inverse of each other conceptually. In Figure 8, those rectangular and square boxes are a spatially outward extension of process p3, not the inward inclusion of process p0, p1,

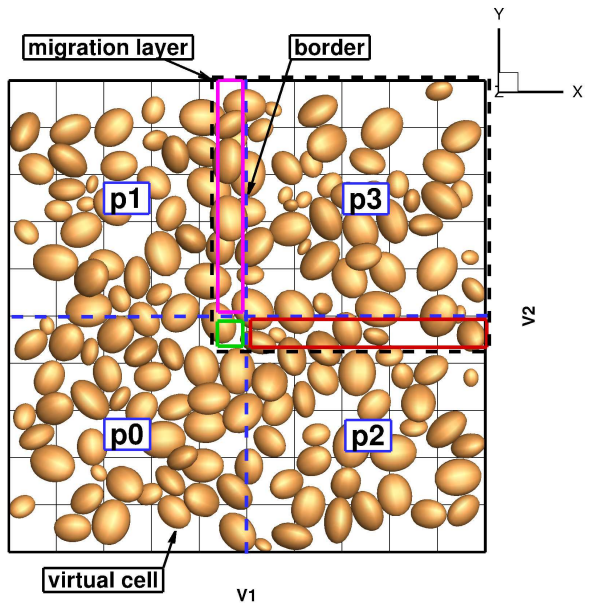


Figure 8: Particles migrate across blocks.

p2, respectively. They are called *migration layers* or *migration zones*. For example, p3 ought to check its three spatial migration layers to see if any of its particles move into the migration layers, and if yes, p3 should send such particles to its neighbors and delete them from its own space. The width of the migration layers is independent of the size of virtual cells, and it is actually determined by the velocities of the particles and time step used in current time increment, to ensure that no particle can leave the spatially outward extended block/process within the time step.

Furthermore, many performance-critical details are managed optimally in order to achieve the best performance, such as: optimizes memory management by minimizing global communication and avoiding large MPI memory consumption; achieves efficient MPI transmission of dynamic objects and pointers using Boost C++ libraries (Boost.MPI, Boost.Serialization, Boost Non-blocking communication, etc.); eliminates redundant contact information between adjacent MPI processes using high-performance data structures; and collects contact information between particles and partitioned boundaries using a special data structure with optimized MPI_Gather operations.

4. PERFORMANCE ANALYSIS OF 3D DEM

4.1. DoD Supercomputers

The target architectures in this work are four of the DoD supercomputers: Spirit, Excalibur, Thunder and Topaz, and their parameters are listed in Table 1.

Table 1: Four DoD supercomputers

supercomputer	Spirit	Excalibur	Thunder	Topaz
system	SGI ICE X	Cray XC40	SGI ICE X	SGI ICE X
compute nodes	4,590	3,098	3,216	3,456
cores per node	16	32	36	36
total cores	73,440	101,184	125,888	125,440
memory per node	32 GB	128 GB	128 GB	128 GB
CPU	Xeon E5-2670	Xeon E5-2698v3	Xeon E5-2699v3	Xeon E5-2699v3
core speed	2.6 GHz	2.3 GHz	2.3 GHz	2.3 GHz
interconnect	4x FDR InfiniBand	Cray Aries	4x FDR InfiniBand	4x FDR InfiniBand
peak PFLOPS	1.50	3.77	5.62	4.66
MPI	SGI MPT	Cray MPICH2	SGI MPT	SGI MPT

4.2. Types of DEM simulations

The serial DEM code for three-axis ellipsoids was initially developed by Yan (2008), and has been successfully parallelized and extended to perform nearly 20 types of simulation utilizing both ellipsoid and poly-ellipsoid shapes during the past five years, and it is now named `ParaEllip3d`, and hosted under the open source Tahoe Development Project (`tahoe.sourceforge.net`) with a BSD-3 license.

Overall, the problems that are modeled by 3D DEM fall into two main categories:

- static or quasi-static problems: laboratory tests such as oedometer compression, isotropic compression, conventional or true triaxial compression, in-situ Cone Penetration Test (CPT), static load test of cast-in-place piles, etc.
- dynamic problems: sand pluviation or deposition with gravity, collapse of particle assemblage, landslide under gravity, explosion beneath soil, installation of precast piles by means of hammers, sand dune movement, etc.

To cover the variations, sand pluviation ("raining") is selected as our representative test in evaluating parallel performance. Illustrated in Figure 7, the sand particles are generated based on a specific soil gradation curve (so that they have different sizes) and "floated" in space initially without interaction; during the process of gravitational pluviation, the bottom particles start to pack up and interparticle contacts should be detected; at the end, all particles come to rest and stay in a relatively "dense" state statically under gravity.

The static/quasi-static simulations can achieve excellent load balance, whereas in dynamic simulations, such as buried explosion in sand, it is difficult to achieve good load balance because the motion and distribution of soil grains is unknown. Even though each link-block contains the same number of particles, there could still be load imbalance, because the computational cost is not determined by the number of particles but instead by the number of interparticle contacts, which is unknown before the computation is performed.

4.3. MPI performance across multiple nodes

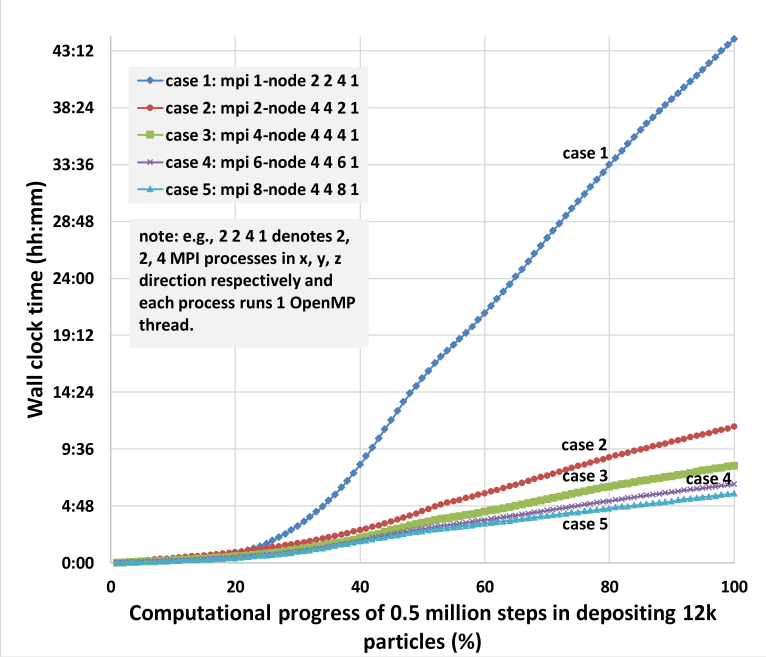
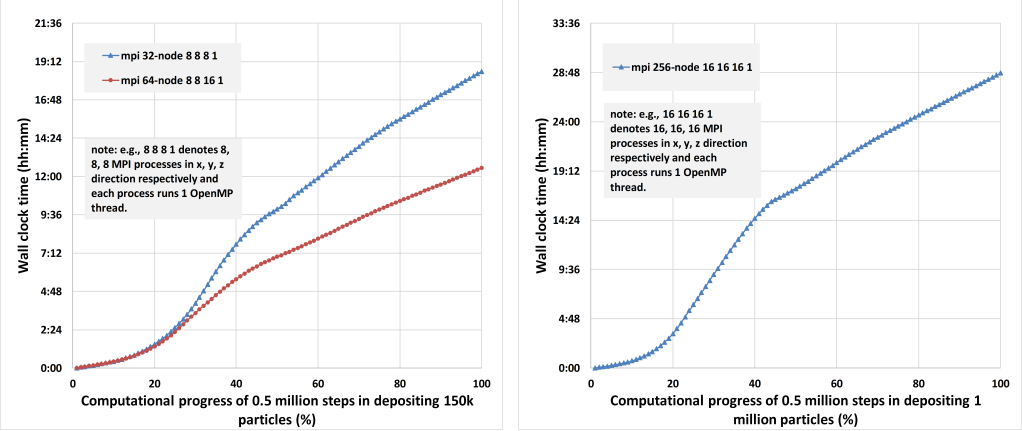


Figure 9: MPI performance across multiple nodes simulating 12k particles.

The simulation parameters are listed in Table 3. For sand pluviation of 12k particles, 1 to 8 compute nodes are utilized and their wall clock time versus computational progress curves are plotted in Figure 9; for 150k particles, 32 and 64 nodes are used to complete the simulation within 24 hours and plotted in Figure 10(a); for 1 million particles, 256 nodes are requested to run within 24 hours and plotted in Figure 10(b). Note that all of the times, including wall clock time and module execution times per step, are acquired using function `MPI_Wtime` for a high resolution measurement.



(a) 150k particles. (b) 1 million particles.

Figure 10: MPI performance of a large number of particles.

From all the simulations of 2.5k, 12k, 150k and 1 million particles, it is observed that wall time versus computational progress curves using different number of compute nodes exhibit larger gaps, which is particularly clear in Figure 10(a): the 32-node and 64-node curves overlap at the earlier stage of simulation when interparticle contacts have not accumulated, but deviate and keep increasing the gap at later simulation stage when more interparticle contacts have developed.

4.3.1. Constraint on performance gain

Figure 11 plots the speedup and experimentally determined serial fraction (EDSF) for a static simulation of 12k particles in terms of number of processors (namely, evaluated relative to the single processor performance). For a problem of fixed size, the speedup of a parallel computation typically increases while the efficiency typically decreases, as the number of processors

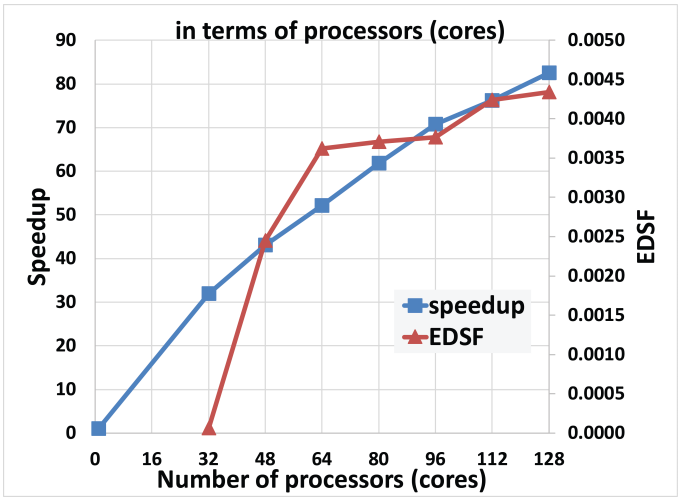


Figure 11: MPI speedup and EDSF of 12k particles.

increases; for the relatively small number of processors (128) used in these tests, the speedup exhibits a nonlinear relationship. MPI has achieved excellent speedup in this test; for example, it achieves a speedup of 83 using 128 cores.

Firstly, it is seen that the EDSF e values are between 0.01% and 0.43%, which indicates a very low serial fraction. According to Gustafson-Barsis's law, speedup $\psi \leq p + (1-p)s$, where p is the number of processors and s denotes the fraction of time spent in the parallel computation performing inherently sequential operations, if we let $s = e$ (which actually overestimates the value of s), it is obtained that $\psi \leq 128 + (1-128) * 0.43\% = 127.45$ using the data from 128 processors, therefore an excellent scaled speedup is achieved.

Secondly, it is shown that EDSF increases as the number of processors increases. This provides an important indication: the MPI performance gain is not constrained by inherently sequential code, but mostly by parallel overhead, which could be time spent in process startup, communication, or synchronization, or it could be an architectural constraint, as stated by the book *Parallel Programming in C with MPI and OpenMP* (Michael, 2003).

4.4. MPI profiling

For each scale of number of particles (2.5k, 12k, 150k, 1M and 10M), various numbers of compute nodes are used to test the speedup and efficiency

in static simulations. In particular, an excessive number of compute nodes may be employed for the following purpose: (1) observe how the speedup and efficiency respond to the increasing number of compute nodes; (2) test if there is an optimal computational granularity (number of particles per process) for each scale.

4.4.1. Speedup and efficiency

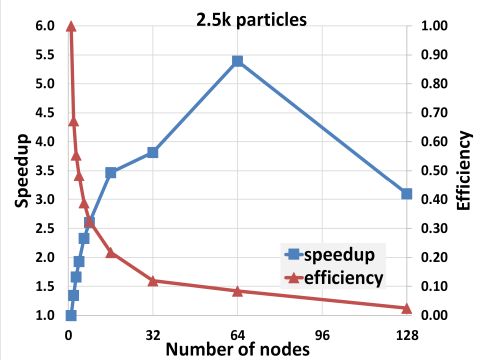
Figure 12 plots the speedup and efficiency of the five scales, each of which tests up to an excessive number of compute nodes. For example, the 2.5k-particle test requests up to 128 nodes which results in nearly 1 particle per process, and the 150k-particle test requests up to 512 nodes which results in nearly 18 particles per process.

Of the five scales, it can be discovered that the excessive number of compute nodes leads to a decrease of speedup, although the speedup exhibits a nonlinear increase within the range of adequate number of compute nodes. As an example, the 150k-particle test achieves a speedup of 92 using 128 nodes while it achieves a speedup of 76 using 256 nodes. It implies that for each scale of simulation there must be an optimization of computational resources, which we have defined as computational granularity, namely, the number of particles per process.

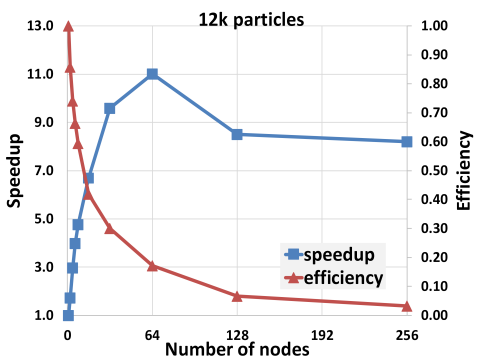
With regard to efficiency, it can become very low if an excessive number of compute nodes is used. For example, in the 12k-particle test, the efficiency is 0.60 (60%) using 8 nodes and 0.07 (7%) using 128 nodes. Low efficiency means low usage of computational resources, and should be avoided in parallel computing.

With adequate number of compute nodes, the speedup exhibits a monotonically increasing relationship with respect to the number of compute nodes at all scales, while the efficiency exhibits a monotonically decreasing trend. On the scale of 150k, 1M and 10M particles, higher-than-1 efficiency is observed. The superlinear speedup is pronounced; for example, the efficiency goes as high as 1.97 (197%) at 8 nodes in the 150k test; and 17.75 (1,775%) at 32 nodes, and 7.65 (765%) at 256 nodes in the 1 million particle test. It is worth noting that for all of the 1-node tests across the five scales, the memory size is sufficiently adequate to satisfy the computation and does not cause swap-out to hard drive.

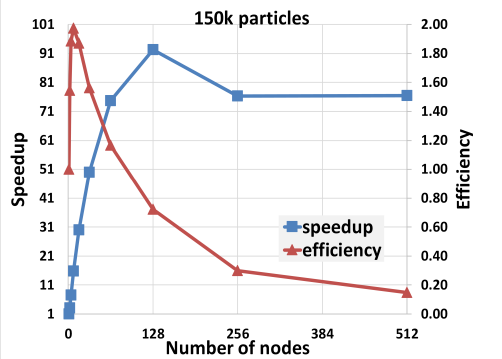
Yan and Regueiro (2016b,a) investigate the superlinear speedup in complex-shaped 3D DEM and concludes that: (1) Strong and weak scaling measurements show that cache miss rate is sensitive to the memory consumption



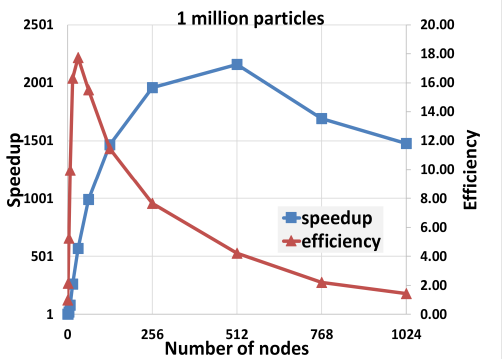
(a) 2.5k particles.



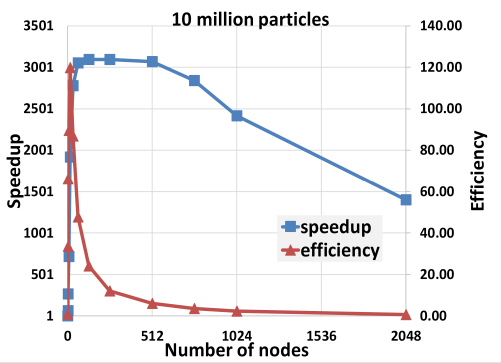
(b) 12k particles.



(c) 150k particles.



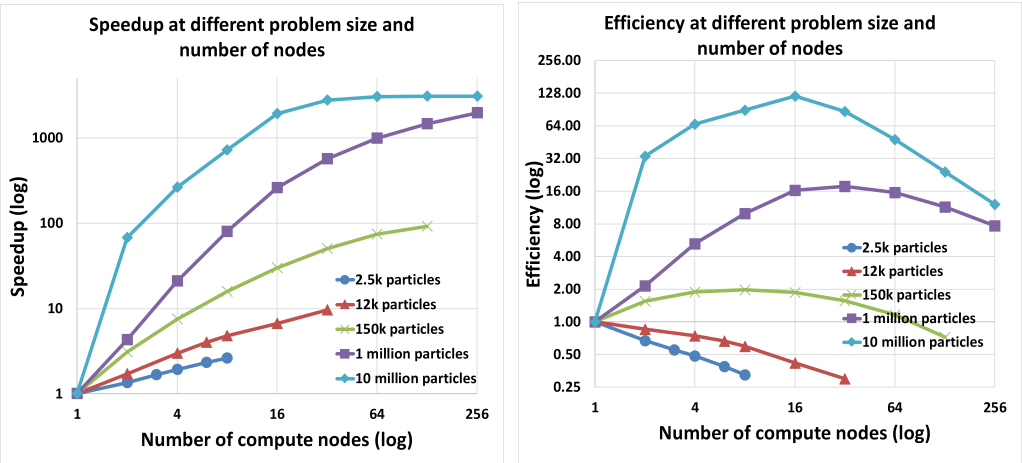
(d) 1 million particles.



(e) 10 million particles.

Figure 12: Speedup and efficiency across orders of magnitude of simulation scale in terms of number of particles.

reduction per processor, and the last level cache (LLC) contributes most significantly to the strong superlinear speedup among all of the three cache levels of modern microprocessors; (2) both $O(n^2)$ and $O(n)$ algorithms exhibit a strong superlinear speedup on large-scale simulations of complex-shaped 3D DEM. The $O(n)$ algorithm always exhibits a lower speedup than the $O(n^2)$ algorithm across all computational scales and granularities, mostly due to the base point measurement at a single compute node, whereby the $O(n)$ algorithm executes much faster than the $O(n^2)$ algorithm. On average, the speedup in $O(n)$ algorithm is reduced by approximately 1/3 relative to $O(n^2)$ algorithm on the simulation scale of 1 million ellipsoidal particles; (3) the superlinear speedup is commonplace for large-scale complex-shaped 3D DEM.



(a) Speedup across orders of magnitude simulation scale.

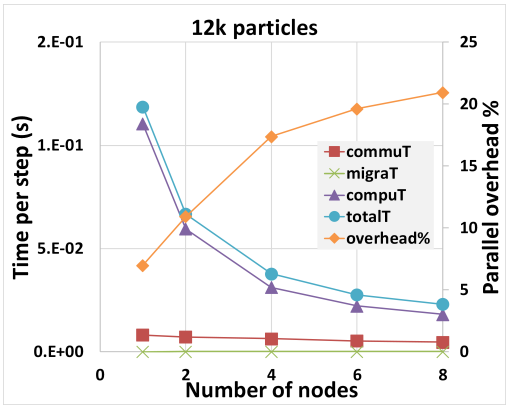
(b) Efficiency across orders of magnitude of simulation scale.

Figure 13: Speedup and efficiency across orders of magnitude of simulation scale in terms of number of particles.

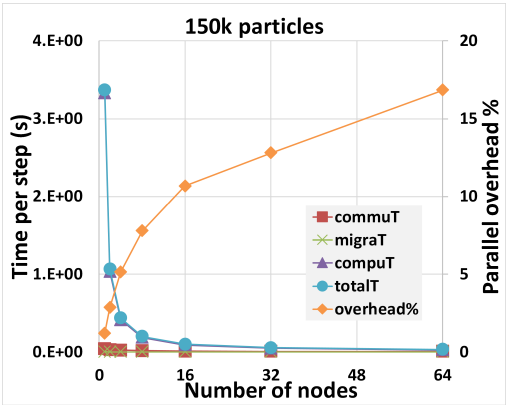
Figure 13(a) compiles all of the speedup data from static simulations at the five different scales. A loglog graph is plotted due to the wide range of problem size and number of processors. The Amdahl effect is pronounced: speedup is an increasing function of the problem size for any fixed number of processors.

Similarly, Figure 13(b) plots the efficiency across the five different scales using a log-log graph. Large-scale simulations such as 150k, 1M and 10M particles exhibit high efficiency above 1, while smaller scale simulations such as 2.5k and 12k particles show a lower-than-1 efficiency.

4.4.2. Module execution time and parallel overhead

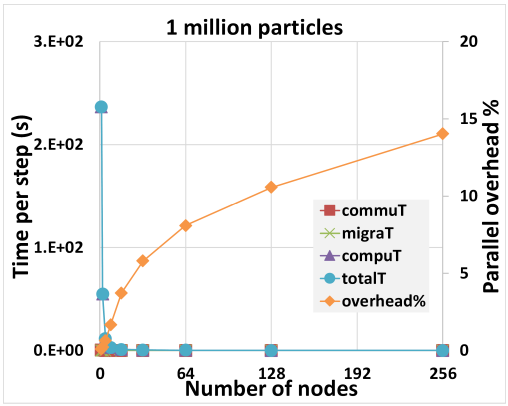


(a) 12k particles.

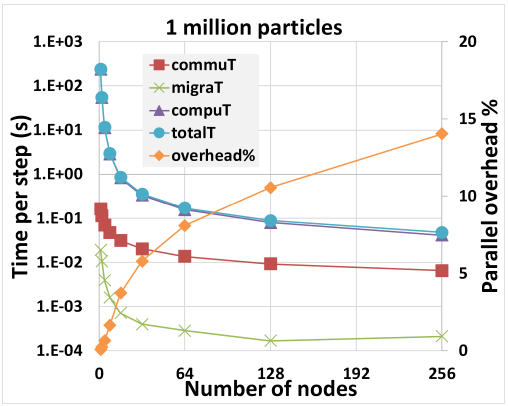


(b) 150k particles.

Figure 14: MPI profiling on modules for 12k and 150k particles.



(a) 1 million particles.



(b) 1 million particles, wall time in logarithmic scale.

Figure 15: MPI profiling on modules for 1 million particles.

Execution time of different modules is plotted with the percentage of parallel overhead using adequate number of compute nodes in Figures 14 and 15. Note that Figure 15(b) uses a logarithmic scale in wall time to distinguish between the close curves shown in Figure 15(a). Each module is described here again:

- commuT: communication time in step 3 (3-All processes communicate with neighbors) of the flowchart in Figure 5.

- migraT: migration time in step 12 (12-All processes migrate particles) of the flowchart.
- compuT: numerical computation time, it equals totalT-commuT-migraT.
- totalT: total time at each step.
- overhead%: (commuT+migraT)/totalT, i.e., the overall parallel overhead percentage.

Note that the IO cost in 3D DEM is very limited. As a typical example, only 100 snapshots are taken for 5 million time increments of 3D DEM simulation. The synchronization overhead is less than 0.1% of the commuT across all of the simulation scales such that it is a negligible fraction of overall parallel overhead, of which the communication cost dominates.

Firstly, the ratio of migration time to communication time is as low as 1-6% across all simulation scales. This makes sense because there are no particles migrating across borders in static simulations, although step 12 must be executed and thus spends a very small fraction of time. Note that this ratio remains low even if it is evaluated in a dynamic simulation because use of the adaptive compute grids minimizes particle migration across borders.

Since step 3 (3-All processes communicate with neighbors) and step 12 (12-All processes migrate particles) of the flowchart employ the same design and implementation with different layer definitions, as shown in Sections 3.3, it can be approximately deduced that the actual interprocess communication spends about 95% of the overall communication overhead while the additional/redundant computation needed for the communication only takes about 5%.

Secondly, as the number of compute nodes increases, both the computation time and communication time (thus the total time) decrease, and the decrease rates are high at the very beginning and slow down later for a fixed problem size. This is the goal and anticipation from parallel computing. In addition, the communication time remains a small fraction relative to the computation time.

Thirdly, the parallel overhead consumes a low fraction of the wall time. On the scale of 12k particles it stays as high as between 11-21% using 2 to 8 nodes; on the 150k particles it ranges between 2.9-16.8% using 2 to 64 nodes; on the 1 million particles it ranges between 0.2% to nearly 14% when the number of nodes increases from 2 to 256. Overall, the parallel overhead percentage is nearly 10% for static simulations when an optimal number of compute nodes is used for computation. Considering that 6 steps are added in

order to parallelize the code, as described in Section 3.2, the overall parallel overhead (communication operations plus redundant computations) is low and acceptable.

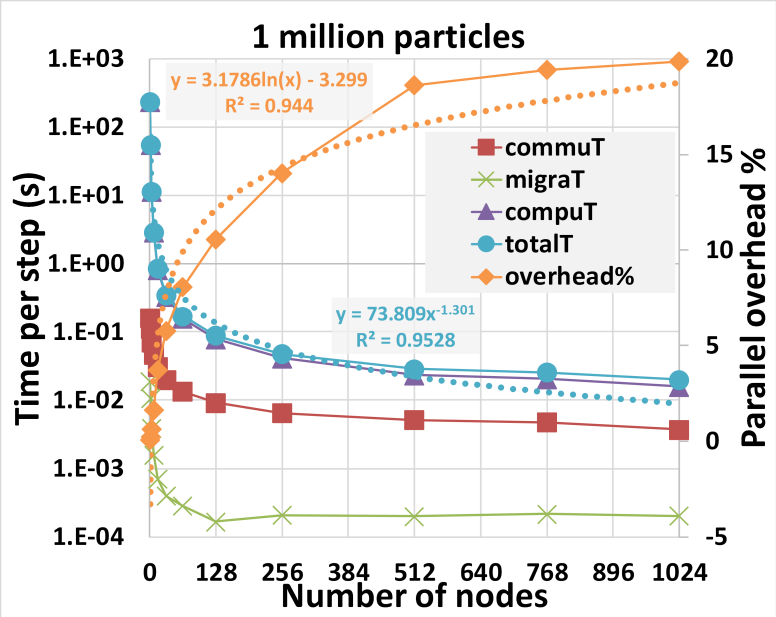


Figure 16: MPI profiling on modules for 1 million particles using excessive number of nodes.

Figure 16 depicts the log-log relationship between module time and parallel overhead for 1 million particles using 1 to 1,024 compute nodes excessively. In particular, a curve fitting is performed for the total execution time per step and parallel overhead percentage. It is seen that the relationship between the total execution time per step and number of nodes can be described in the form of a negative power function,

$$T(n,p) = O(p^{-k}), \tag{3}$$

where p is the number of nodes and k is a number greater than 1. The relationship between the parallel overhead percentage and number of nodes can be described in the form of a logarithmic function,

$$\text{overhead\%} \ = O(\log\, p). \tag{4}$$

$T(n,p)$ decreases quickly and overhead% increases slowly when p increases.

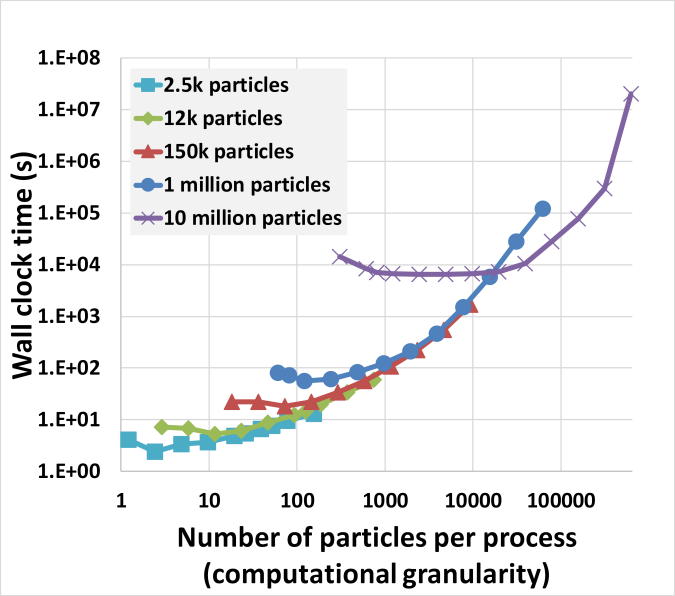


Figure 17: Computational granularity for various scales of simulation.

4.5. Computational granularity (CG)

Figure 17 plots the wall clock time versus number of particles per MPI process on different computational scales such that it is able to read the optimal computational granularity. Due to the wide range of number of particles and wall clock time per step, a log-log graph has been used.

Overall, as the number of compute nodes increases and thus the number of particles per process decreases, the wall clock time decreases, which indicates that a smaller computational granularity leads to faster computation for a fixed problem size. However, it can be observed that the wall time starts to increase when an excessive number of compute nodes is used and thus the computational granularity becomes too small. This occurs for all of the five scales: 2.5k, 12k, 150, 1M and 10M particles.

Although the speedup can keep increasing and thus wall clock time can keep decreasing until an extremely excessive number of compute nodes is used and thus leads to a speedup decrease and wall clock time increase eventually, the reasonable amount of computational resource (number of compute nodes) usually should not be requested too aggressively in performing practical computational tasks. For example, the 150k-particle simulation achieves a speedup of 74.6 (27.3 seconds per step) using 64 nodes, and 92.3 (25.3

seconds per step) using 128 nodes; then requesting 64 nodes may be a better choice than requesting 128 nodes.

As a guideline, the optimal computational granularity (CG), which can be estimated before submitting a job on supercomputers, is recommended in Table 2:

scale	1k	10k	100k	1 million	10 million
CG	20∼50	50∼100	100∼300	200∼500	5k∼10k

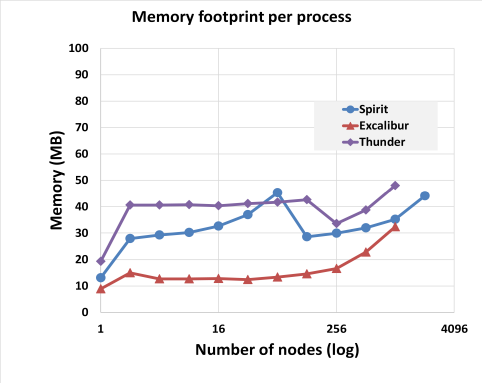
Table 2: Optimal computational granularities.

4.6. Weak scaling measurement

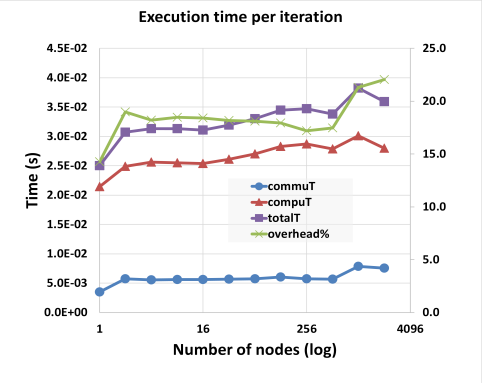
Weak scaling measurements are performed on multiple DoD supercomputers. Figure 18(a) plots the memory usage per process at different scale on the three supercomputers. Across all of the simulation scales it varies between 10∼50 MB, which is a very low memory consumption.

Figure 18(b), (c) and (d) plot the module execution time on Spirit, Excalibur and Thunder, respectively. On Excalibur the communication time stays constant as the workload is increased in direct proportion to the number of compute nodes, while on Spirit and Thunder the communication time nearly stays constant until the number of nodes reaches 512 or 1,024, whereby the increase is most likely attributed to the lack of hypercube interconnect (Rudi et al., 2015).

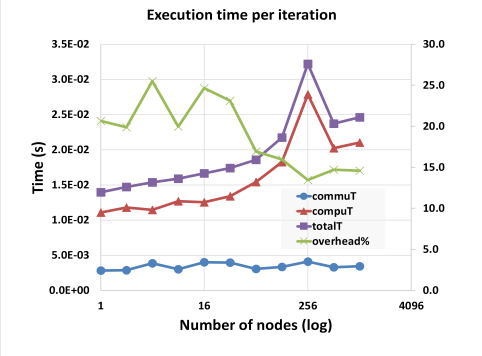
The computation time increases slightly on Spirit and Thunder while it increases more pronouncedly on Excalibur. The increase of computation time is not attributed to the global communication operations, namely, the step 9 (9-All processes update compute grids) in Figure 5, which is nominally included into computation time. Our measurements reveal that the global communication operations for updating compute grids normally take as low as 0.1% fraction of the computation time. The increase is actually related to the characteristics of memory consumption and cache hit/miss rate on these systems. This is justified by the cache miss statistics shown in Figure 18(e) and (f), wherein the total cache misses (TCM) of L3, L2 and L1 caches increase with the workload that is in direct proportion to the number of compute nodes. The `PAPI_L3_TCM` is plotted in Figure 18(f) to reveal more details.



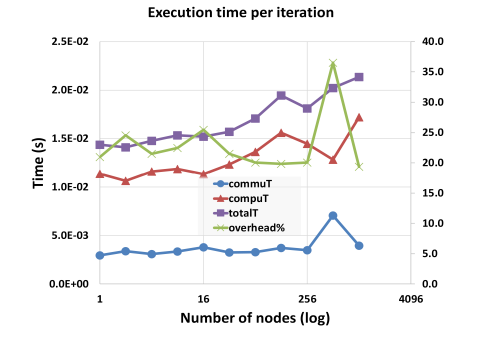
(a) Memory footprint.



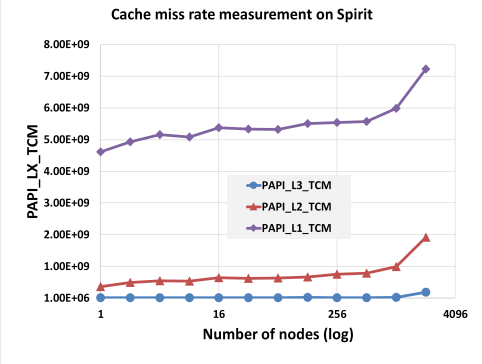
(b) Execution time on Spirit.



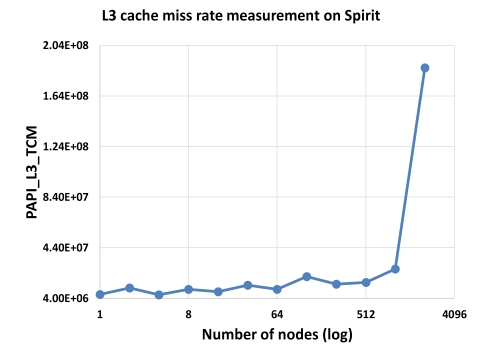
(c) Execution time on Excalibur.



(d) Execution time on Thunder.



(e) Cache miss measurement.



(f) L3 cache miss measurement.

Figure 18: Weak scalability measurement

4.7. Floating-point operation performance

The Performance API (PAPI) (Browne et al., 2000) is adopted in the C++ source code to measure the simulation performance by using performance counter hardware found in the microprocessors. Due to the limited access and allocation on the DoD supercomputers, we are only able to extend our simulations to 2,048 compute nodes and measure the floating-point operations on Spirit (note that floating point counters have been disabled in the Intel Haswell CPU architecture, on Excalibur and Thunder).

Figure 19 exhibits a linear relationship between FLOPS and number of compute nodes for the simulation of 1 million particles.

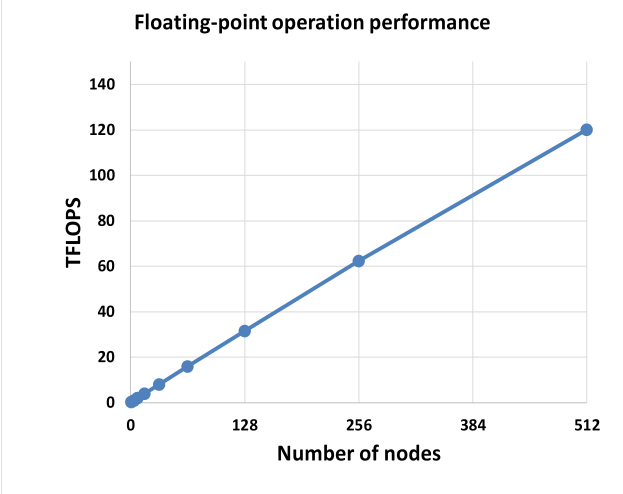


Figure 19: Floating-point operation performance of 1 million particles.

5. LARGE-SCALE SIMULATIONS OF PARTICLE MOTION

5.1. Sand pluviation and rebound

A dynamic simulation may exhibit characteristics that do not exist in a static one. For instance, from Figure 10 it can be observed that the curve changes its slope at different stages of sand pluviation: the time increment is fastest between 0-20%, it becomes slower between 20-40%, then it become fast again between 40-100%, though it is not as fast as that between 0-20%. Without combining the curve and corresponding snapshots of pluviation states and compute grids, as shown in Figure 20, it is nearly impossible to understand the change of curve slope.

A full-scale simulation of depositing one million ellipsoidal particles ranging over 2∼5 mm diameters is carried out using 256 compute nodes on Spirit. The particles are generated in an initially floating-in-air state that occupies a volume of 30 cm × 30 cm × 170 cm in x, y, z direction respectively, and start to free fall gravitationally into an enclosed rigid container. The parameters are listed in Table 3.

Young's modulus E (Pa)	2.9×10^{10}
Poisson's ratio ν	0.25
specific gravity G_s	2.65
interparticle coef. of friction μ	0.5
interparticle contact damping ratio ξ	0.85
particle radii (m)	$0.001 \sim 0.005$
particle shape (aspect ratio)	1:0.8:0.6
time step $\triangle t$ (sec)	1.0×10^{-6}
simulation time (sec)	1.2
computational wall time (hours)	29

Table 3: Numerical parameters used in pluviation simulation.

High-resolution movies generated from the simulation are displayed at YouTube:

- 3d view: `https://www.youtube.com/watch?v=_XY73KKMSv8`
- side view: `https://www.youtube.com/watch?v=JCyLot7mZpM`
- zoomed-in side view: `https://www.youtube.com/watch?v=ORiotH7ajPw`

Illustrated by the pluviation states in Figure 20, there are not many interparticle contacts between 0-20% stage, so the time increment computation goes quickly; between 20-40%, interparticle contacts increase more quickly while most compute grids are consumed by spatially scattered upper particles without contacts, thus time increment computation slows down; after 40%, nearly all particles are settled and most compute grids are used by the settled particles, therefore the time increment computation speeds up again.

It is a bit surprising to discover that nearly 50% of the time increment loops (0-50% stage) are spent on the overall gravitational pluviation, while the other 50% of the time increment loops (50-100% stage) are spent on the rebound phenomenon at the top of the settled particle assemblage. The rebound is clearly shown in the movie and can also be seen during the 50-70%

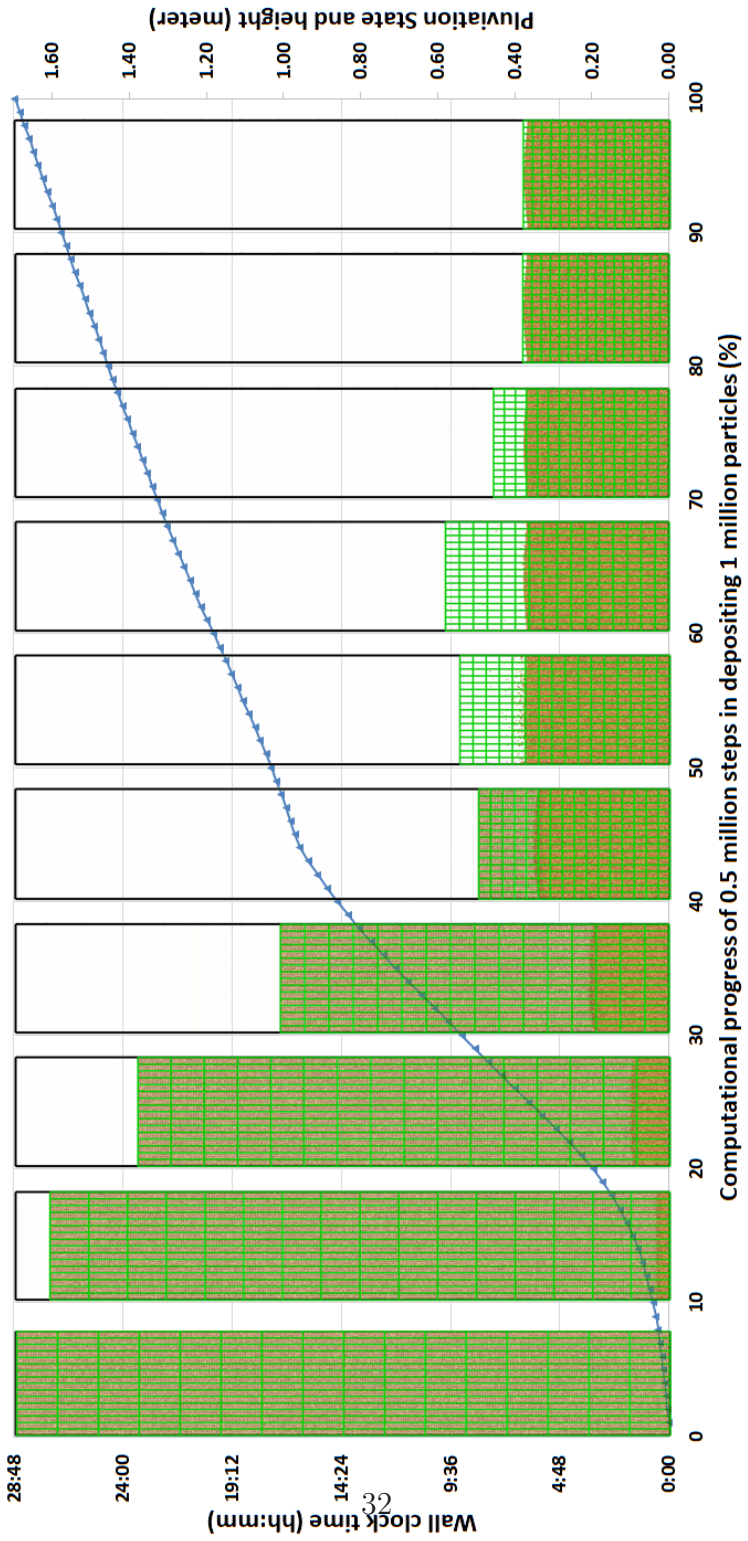


Figure 20: Sand pluviation of one million particles.

stage in Figure 20. It is captured at a certain instant and zoomed-in on the left side of Figure 21, and on the right is the rested particle assemblage.

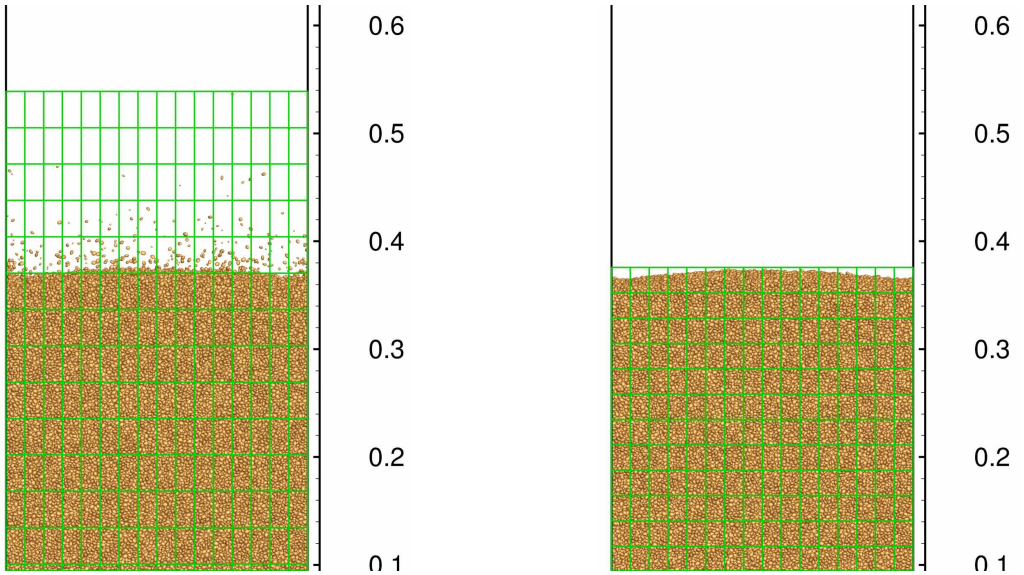


Figure 21: Sand rebound and rested state.

Simulating the rebound phenomenon in DEM is a challenging task in that:

- The relative impact velocity between the rebounding and falling particles could be much higher, and it requires more strict time step control.
- The overall rebound height of 20 cm in the numerical simulation agrees with that in our in-situ experiments of free falling 2∼5 mm diameter gravel particles at a 1.7-meter height. Note the particles involved in the rebound are those that initially are positioned close to the 1.7 meter such that experimental release and the numerical release are close in terms of potential energy.
- Some rebounding particles, especially those small particles, could rebound at an abnormally high velocity (translational and sometimes rotational) and in an arbitrarily sideways direction (hitting the container's side wall). This has been frequently observed in our in-situ experiments of free falling 2∼5 mm diameter gravel particles at a 1.7-meter height.

- The migration of particles from one compute grid to the other occurs frequently in the rebound area, as shown in the simulation movie, thus the parallel algorithm must be able to handle the across-border migrations accurately.

In this sense, the rebound phenomenon in sand pluviation may serve as a valuable testing problem in justifying the DEM model and the parallel algorithm. In addition, it should be particularly helpful to compare with high speed imaging results for the purpose of calibration and verification.

It is worth noting that although the rebound of top particles leads to inefficiency in compute grids allocation and computational load imbalance, fortunately the time increment speed is not significantly affected, which can be justified by the linear segment between the 50-100% stage.

5.2. Constrained collapse

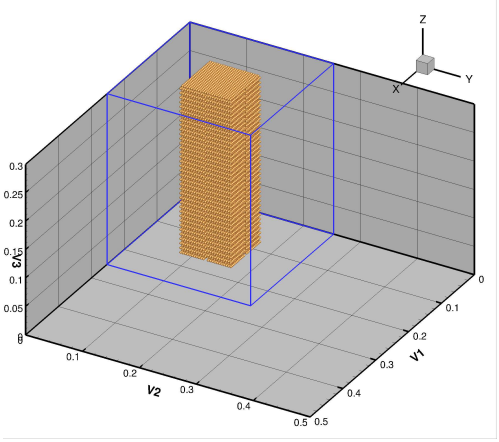
A series of numerical simulations are carried out on Topaz to study the deposition and collapse effect of particle assemblies, and the numbers of particles used in the series are 7.4k, 36k, 56k, 76k, 104k, 130k, 167k, 267k, 384k, 535k, 703k and 1M, respectively. Of all the 12 simulations the particles are initially floated between the elevation of $0 \sim 30$ cm, and start to free fall into a rigid container whose four side walls are always 8 cm away from the initial particle assembly. Figure 22 demonstrates the initial and rested states of 36k and 384k particles, respectively, whereby the container boundaries are plotted by blue lines.

The parameters of these simulations are listed in Table 4. The simulation time is extended to 2.0 seconds such that particle rolling motions along the assembly surface are sufficiently observed even if the overall assembly is settled.

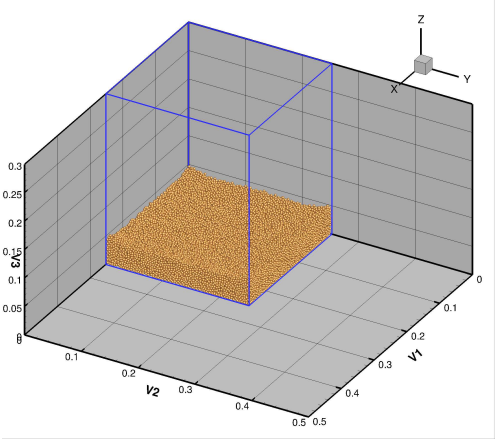
High-resolution movies generated from the simulations are provided at YouTube playlists:

- 3d view of the 12 movies: `https://www.youtube.com/playlist?list=PL0Spd0Mtb6vW1q_h4R_uYVcsh8mu9JtcS`
- side view of the 12 movies: `https://www.youtube.com/playlist?list=PL0Spd0Mtb6vUfMW2cx6XxvWeW-z_ZJg2q`

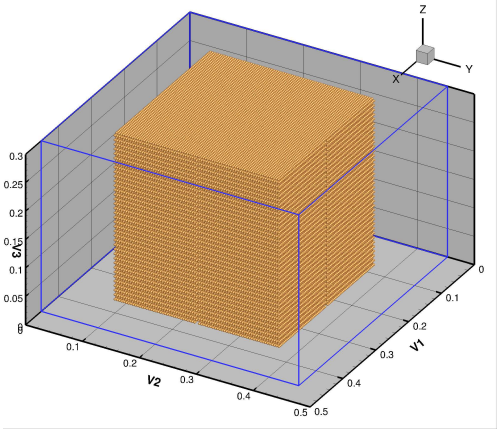
Figure 23 demonstrates the collapse process of 535k particles with 8 snapshots in sequential order of time. Note the scattered particles in the central



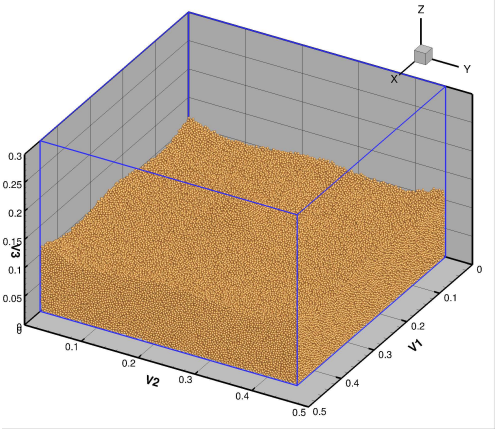
(1a) 36k particles initial.



(1b) 36k particles rested.



(2a) 384k particles initial.



(2b) 384k particles rested.

Figure 22: Initial and rested states of particles in 3D view.

Young's modulus E (Pa)	8.5×10^9
Poisson's ratio ν	0.25
specific gravity G_s	2.65
interparticle coef. of friction μ	0.5
interparticle contact damping ratio ξ	1.00
sphere radii (m)	0.002
time step $\triangle t$ (sec)	5.0×10^{-7}
simulation time (sec)	2.0

Table 4: Numerical parameters used in collapse simulation.

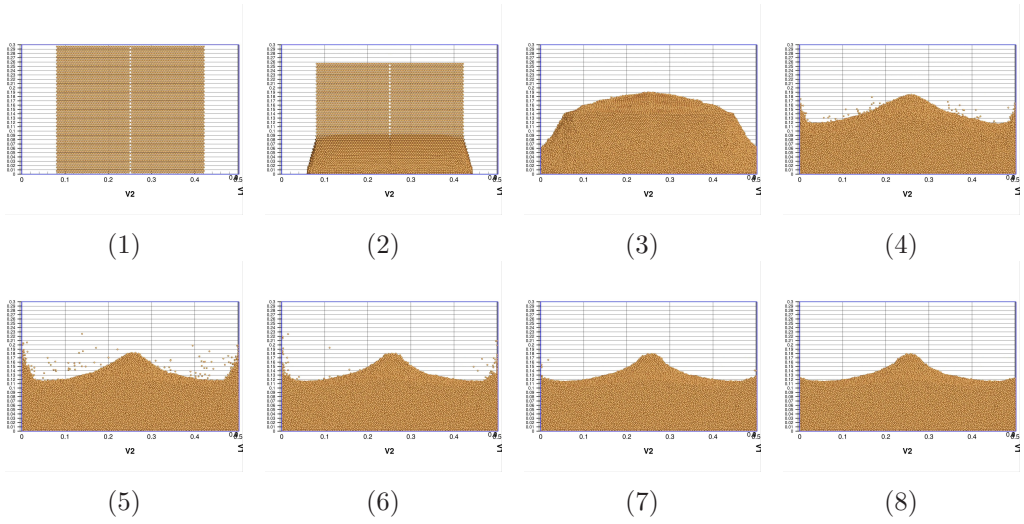


Figure 23: Side view of the collapse process of 535k particles.

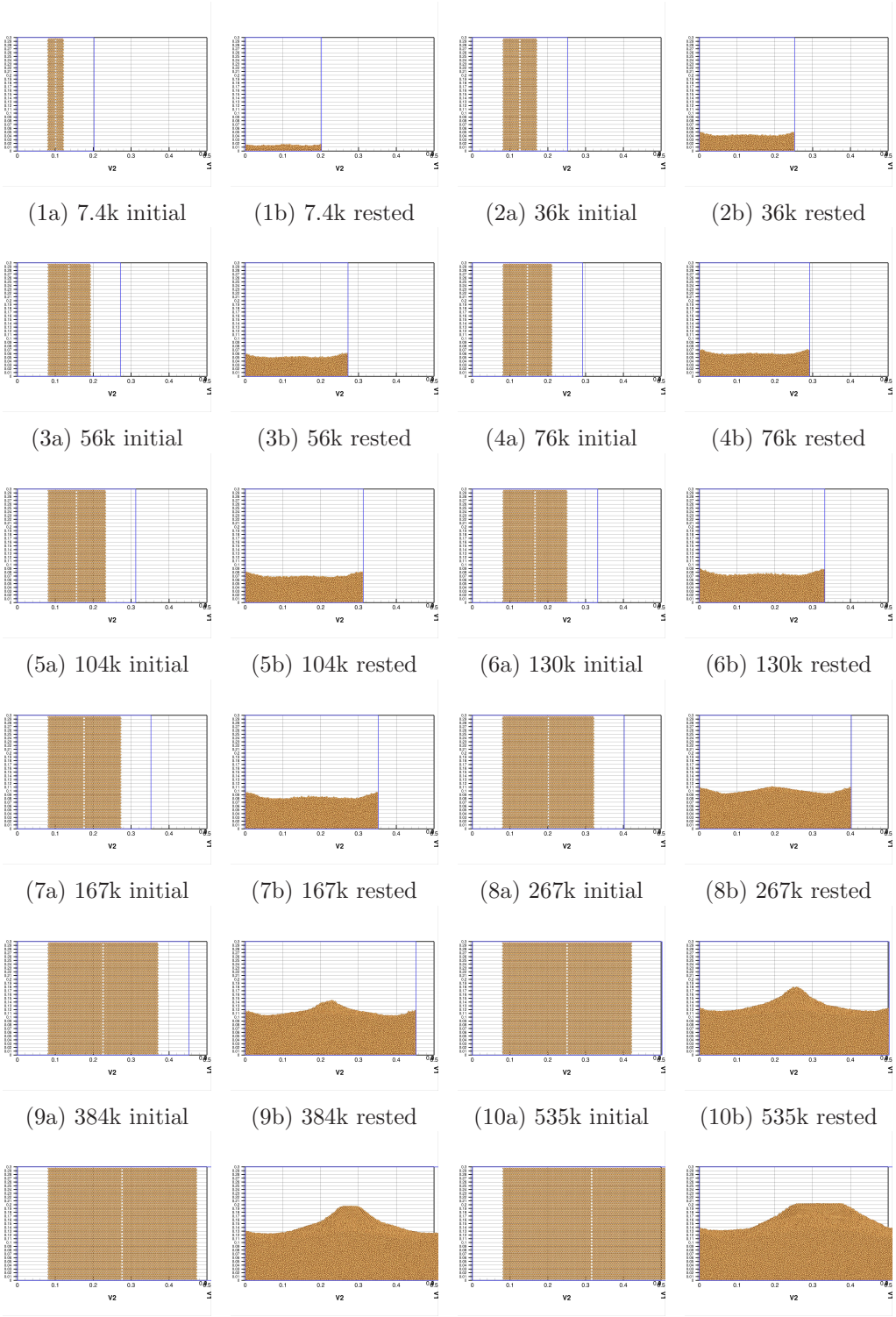
area of Figure 23 ($4 \sim 6$) are rebounded from the front/back walls, not from the left/right walls.

Figure 24 plots side view of the initial and rested states of the 12 simulations. It is clear that the horizontal side length or perimeter of the initial particle assembly "column" increases with an increasing number of particles while retaining a constant distance of 8 cm to the container walls. There are several observations that are worth noting:

Firstly, the 4 corners of the container seem to pile more particles than the 4 sides of the container. This can also be observed in Figure 22 or from the movies at YouTube. This is not a surprise as the spreading particles are reflected towards the corners when hitting the wall.

Secondly, for the first 6 cases (7.4k to 130k particles) it is seen that the rested particle assemblies display a nearly flat or very slightly bulging surface in the central area; for the last 6 cases (167k to 1M particles) it is observed that the rested particle assemblies exhibit a clearly raised or lumped "hill" in the central area. The more particles used in the simulation, the clearer the hill shape is formed. This is not surprising as there is less space for the particles to move with increasing number of particles.

Thirdly, the lumped "hill" can even form a top platform, similar to a Mesoamerican pyramid, in the case of 703k and 1M particles. This is particularly clear if observing the 3D YouTube movies.



(1a) 7.4k initial (1b) 7.4k rested (2a) 36k initial (2b) 36k rested

(3a) 56k initial (3b) 56k rested (4a) 76k initial (4b) 76k rested

(5a) 104k initial (5b) 104k rested (6a) 130k initial (6b) 130k rested

(7a) 167k initial (7b) 167k rested (8a) 267k initial (8b) 267k rested

(9a) 384k initial (9b) 384k rested (10a) 535k initial (10b) 535k rested

(11a) 703k initial (11b) 703k rested (12a) 1M initial (12b) 1M rested

Figure 24: Side view of initial and rested states of 12 simulations.

Quantitative analyses will be further investigated for these simulations, such as velocity field distribution, energy and its conversion process, packing density and its distribution, angle of repose vs number of particles, etc.

5.3. Particle shape effect

Although particle shape plays an insurmountably important role in determining the assembly mechanical behavior, there are no practical numerical tools to study it systematically before large-scale simulations are made possible. In this section, numerical collapse experiments of 535k different-shaped particles are performed on Topaz to obtain an initial impression on effect of particle shape. The particles shapes are chosen to be spheres and ellipsoids (aspect ratio 1:0.8:0.6) for the purpose of comparison, of which the maximum particle semi-axis length is 2 mm. The numerical parameters are listed in Table 5.

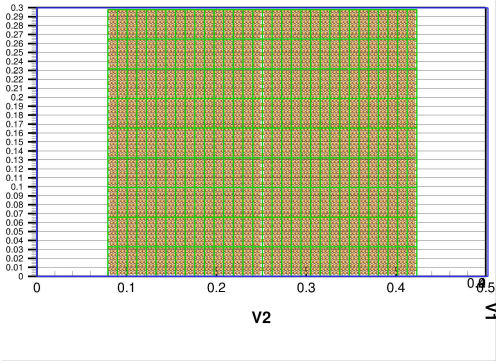
Young's modulus E (Pa)	8.5×10^9
Poisson's ratio ν	0.25
specific gravity G_s	2.65
interparticle coef. of friction μ	0.5
interparticle contact damping ratio ξ	1.00
particle maximum radii (m)	0.002
particle shape (aspect ratio)	1:1:1, 1:0.8:0.6
time step $\triangle t$ (sec)	5.0×10^{-7}
simulation time (sec)	1.5

Table 5: Numerical parameters used in particle shape simulation.

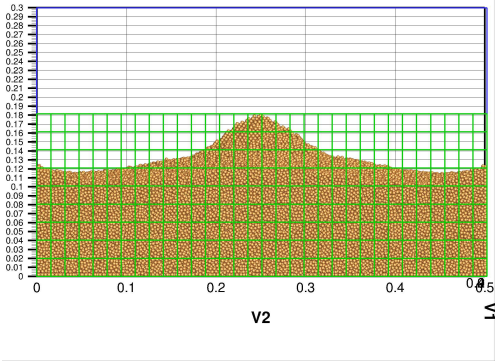
5.3.1. Volume and motion

Figure 25 plots the side view of initial and rested states of the spherical and ellipsoidal particles in the collapse simulation. Obviously they exhibit different rested bulk volume because the individual spherical particles have a larger volume than that of individual ellipsoidal particles (with a volume ratio of 1:0.48). More details can be observed via movies that are uploaded to YouTube:

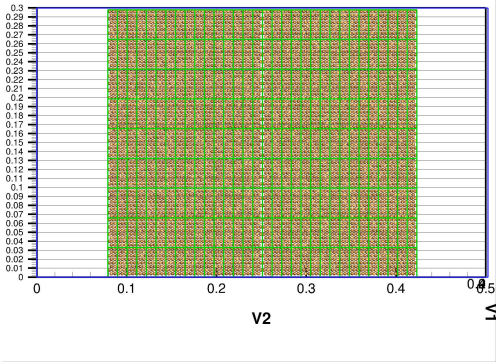
- 3d view of spheres: `https://www.youtube.com/watch?v=ZVX4-fREy2M`
- 3d view of ellipsoids: `https://www.youtube.com/watch?v=nrUUp1rz9OY`
- side view of spheres: `https://www.youtube.com/watch?v=ThzDJ19CkUw`



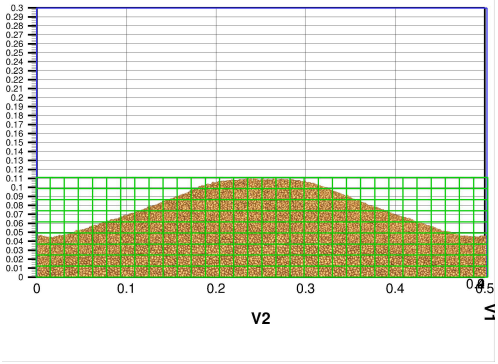
(1a) Spheres initial.



(1b) Spheres rested.



(2a) Ellipsoids initial.



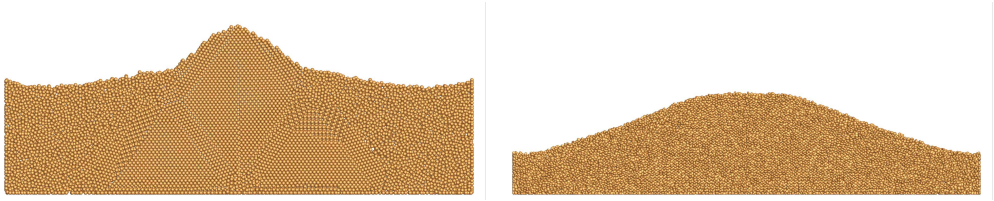
(2b) Ellipsoids rested.

Figure 25: Shape effect between spherical and ellipsoidal particles.

- side view of ellipsoids: `https://www.youtube.com/watch?v=0OnviFsK1GY`

Figure 26 shows the cross-sectional view of the sphere and ellipsoid assemblies at rest. Details of the particle motion along the cross sections can be observed via the following movies at YouTube:

- 3d view of spheres: `https://www.youtube.com/watch?v=slmczgvyOEc`
- 3d view of ellipsoids: `https://www.youtube.com/watch?v=kDc-dYykrfg`
- side view of spheres: `https://www.youtube.com/watch?v=nVuv_TT3J5w`
- side view of ellipsoids: `https://www.youtube.com/watch?v=sFBT2Pc2Cfk`



(1 Spheres.) (2) Ellipsoids.

Figure 26: Cross-sectional view of sphere and ellipsoid assemblies at rest.

The main difference lies in that the ellipsoidal particle assembly tends to "mobilize" much less than spherical particle assembly:

1. The sphere assembly impacts the container wall at a much higher speed, especially with surface particles running into container corners and causing strong reflection.
2. The ellipsoid assembly exhibits a much lower horizontal speed in the collapse, leading to a weak reflection from the container walls.
3. Counterintuitively, the sphere assembly forms a central cone shape while the ellipsoid assembly ends up with a flatter platform.

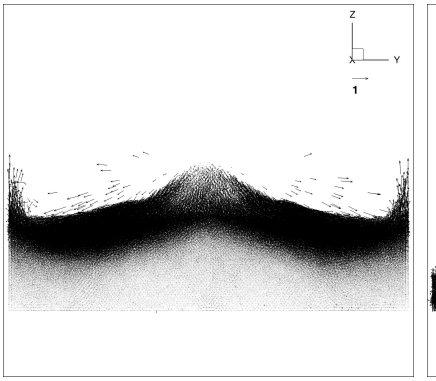
The difference stems mainly from the stronger rolling resistance of the ellipsoid shape. Quantitative evaluation of the particle shapes will be further investigated and compared with high-speed imaging results.

5.3.2. Velocity field

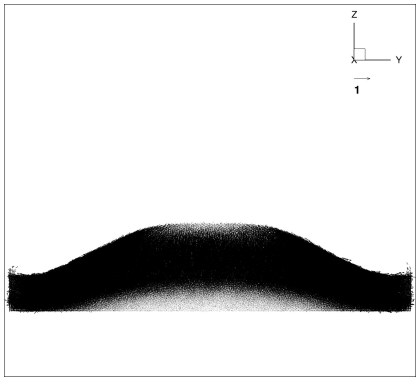
The velocity fields are also recorded by the following movies at YouTube:

- side view of spheres: `https://www.youtube.com/watch?v=7rroK70sc8E`
- side view of ellipsoids: `https://www.youtube.com/watch?v=AocRAlbqMt0`
- top view of spheres: `https://www.youtube.com/watch?v=Q5U-0sQG58Y`
- top view of ellipsoids: `https://www.youtube.com/watch?v=_-uHT8gwY-Y`

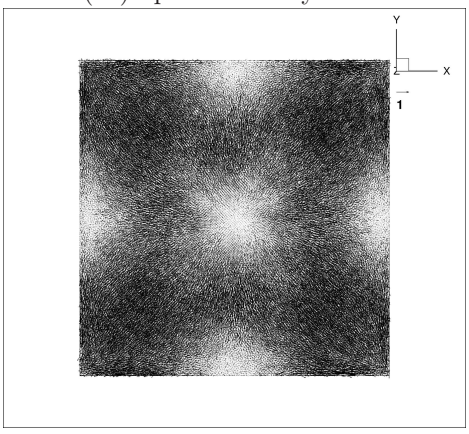
Figure 27 captures the velocity field at the instant of 0.35 seconds for the two assemblies. Note again the scattered velocity vectors observed in the central area are rebounded from the front/back walls, not from the left/right walls. It is read from Figure 27 (1a) and (1b) that the particle vertical velocities are approximately 1.0 m/s and 0.2 m/s for sphere and ellipsoid assemblies, respectively, for the frontal particles that impact the container walls. Figure 27 (2a) and (2b) plots the top view of velocity field at the



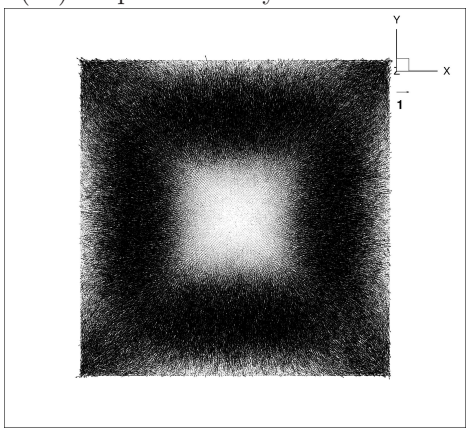
(1a) Sphere velocity side view.



(1b) Ellipsoid velocity side view.



(2a) Sphere velocity top view.



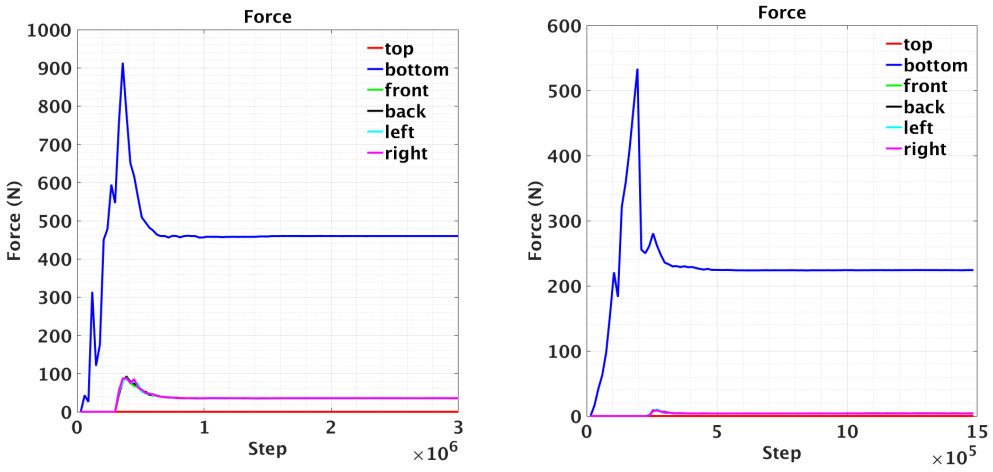
(2b) Ellipsoid velocity top view.

Figure 27: Velocity field of spherical and ellipsoidal particles.

instant of 0.35 second, when the collapse nears the rested state but there are still many particles sliding along the assembly surface. It is interesting to observe that the ellipsoid assembly tends to have a higher horizontal speed of those sliding particles than the sphere assembly at this stage. This phenomenon is also clearly observed from the aforementioned 3D movies at YouTube.

5.3.3. Impact and verification

The interaction between particles and the bottom and side walls of the rigid container is recorded in the collapse process. Figure 28 plots the process of impact forces between particles and walls. Firstly an important physical verification of the static force at the bottom wall is performed for the



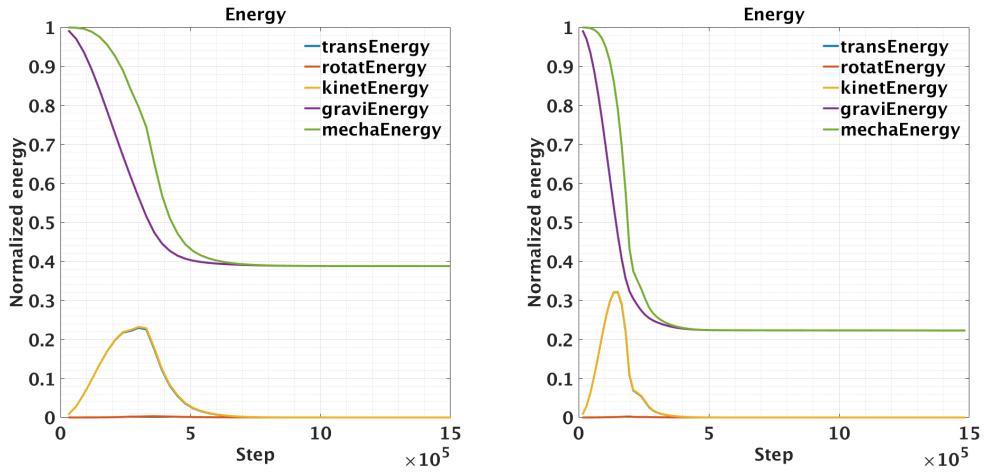
(a) Spheres. (b) Ellipsoids.

Figure 28: Impact forces that particles act onto the container walls.

complex-shaped ellipsoidal particles: it reads 223.9 N from Figure 28(b) as the result of parallel computing using 256 compute nodes. In comparison to the physical weight, 223.4 N, of all the ellipsoidal particles, the computational error is merely 0.2%. In this sense our integration of selected interparticle contact models, contact resolution algorithm for complex-shaped particles, and parallel design and implementation, delivers an accurate computational framework that covers both dynamic and static DEM simulations.

Secondly, the peak forces acting on the bottom wall are 913 N and 535 N (1:0.58) for the sphere and ellipsoid assembly, respectively. The peak forces acting on the side walls are averagely 90 N and 10N (1:0.11) for the sphere and ellipsoid assembly, respectively. The gap between the lateral ratio 1:0.11 and the vertical ratio 1:0.58 indicates the particle shape effect of more rolling resistance, interlocking-prone and internal energy dissipation by ellipsoidal particles.

Figure 29 (a) and (b) plot the energy process of the sphere and ellipsoid assemblies, respectively. Each energy term is normalized to the initial mechanical energy of the particle assembly. Note the initial mechanical energy equals the initial gravitational potential energy as particles have not started to move at that instant. It is clearly shown that gravitational energy



(a) Spheres. (b) Ellipsoids.

Figure 29: Particle assembly energy in the process of collapse.

of the particle assembly is converted to kinetic energy, and the mechanical energy (sum of gravitational potential energy and kinetic energy) dissipates effectively by DEM friction and damping mechanisms. It is also observed that the rotational kinetic energy takes a very small fraction relative to the translational kinetic energy, and that the normalized kinetic energy of ellipsoids goes higher than that of spheres even though spheres tend to be more "mobilized".

6. CONCLUSION AND OUTLOOK

The parallelized 3D DEM of complex-shaped particles features negligible serial fraction and low parallel overhead when executing on contemporary multiprocessing supercomputers, and delivers high speedup and efficiency. The performance analyses reveal that communication time is a decreasing function of the number of compute nodes, and superlinear speedup is a common phenomenon for large-scale 3D DEM simulations of complex-shaped particles.

These days DEM researchers are still struggling in simulating a few to many thousand complex-shaped particles, whereas we have advanced the

scale to 10 million, and used them for full-scale simulations such as gravitational deposition and buried explosion (through DEM Computational Fluid Dynamics) in sandy soils for a DoD project successfully. With this significant computational capacity extension for complex-shaped particles, the 3D DEM opens the door to simulating many natural phenomena, engineering problems or laboratory tests such as sand dune movement, quick sand phenomenon, soil liquefaction in earthquakes, soil-tire interaction, landslide in geotechnical engineering, precast pile installation/penetration and load bearing capacity mechanisms in civil engineering, internal shear band evolution in soil mechanics, etc.

Acknowledgments

We would like to acknowledge the support provided by ONR MURI grant N00014-11-1-0691, and the DoD High Performance Computing Modernization Program (HPCMP) for granting us the computing resources required to conduct this work. We declare that there is no conflict of interest.

John W Baugh Jr and RKS Konduri. Discrete element modelling on a cluster of workstations. *Engineering with Computers*, 17(1):1–15, 2001.

Shirley Browne, Jack Dongarra, Nathan Garner, George Ho, and Philip Mucci. A portable programming interface for performance evaluation on modern processors. *International Journal of High Performance Computing Applications*, 14(3):189–204, 2000.

Peter A Cundall and Otto DL Strack. A discrete numerical model for granular assemblies. *Geotechnique*, 29(1):47–65, 1979.

Gary W Delaney, Paul W Cleary, Matt D Sinnott, and Rob D Morrison. Novel application of dem to modelling comminution processes. In *IOP Conference Series: Materials Science and Engineering*, volume 10, page 012099. IOP Publishing, 2010.

Ian Foster. *Designing and building parallel programs*. Addison Wesley Publishing Company Reading, 1995.

Gary S Grest, Burkhard Dünweg, and Kurt Kremer. Vectorized link cell fortran code for molecular dynamics simulations for a large number of particles. *Computer Physics Communications*, 55(3):269–285, 1989.

David S Henty. Performance of hybrid message-passing and shared-memory parallelism for discrete element modeling. In *Proceedings of the 2000 ACM/IEEE conference on Supercomputing*, page 10. IEEE Computer Society, 2000.

Heinrich Hertz. Ueber die Berührung fester elastischer Körper. [On the fixed elastic body contact]. *Journal für die reine und angewandte Mathematik (Crelle)*, 92:156–171, 1882.

Hosagrahar V Jagadish, Beng Chin Ooi, Kian-Lee Tan, Cui Yu, and Rui Zhang. idistance: An adaptive b+-tree based indexing method for nearest neighbor search. *ACM Transactions on Database Systems (TODS)*, 30(2): 364–397, 2005.

Algirdas Maknickas, Arnas Kačeniauskas, Rimantas Kačianauskas, Robertas Balevičius, and Algis Džiugys. Parallel dem software for simulation of granular media. *Informatica*, 17(2):207–224, 2006.

J Quirm Michael. *Parallel Programming in C with MPI and OpenMP*. New York: McGraw-Hill Press, 2003.

R.D. Mindlin. Compliance of elastic bodies in contact. *Trans. ASME, J. App. Mech.*, 16(3):259–268, 1949.

R.D. Mindlin and H. Deresiewicz. Elastic spheres in contact under varying oblique forces. *Trans. ASME, J. App. Mech.*, 20(3):327–344, 1953.

Marius Muja and David G Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)*, 2:331–340, 2009.

A. Munjiza and K.R.F. Andrews. Nbs contact detection algorithm for bodies of similar size. *International Journal for Numerical Methods in Engineering*, 43(1):131 – 149, 1998.

Antonio Munjiza, Jens H Walther, and Ivo F Sbalzarini. Large-scale parallel discrete element simulations of granular flow. *Engineering Computations*, 26(6):688–697, 2009.

Tang-Tat Ng. Numerical simulations of granular soil using elliptical particles. *Comput. Geotech.*, 16(2):153 – 169, 1994. ISSN 0266-352X.

Tang-Tat Ng. Triaxial test simulations with discrete element method and hydrostatic boundaries. *Journal of Engineering Mechanics*, 130(10):1188 – 1194, 2004.

E. Onate and J. Rojek. Combination of discrete element and finite element methods for dynamic analysis of geomechanics problems. *Comp. Meth. App. Mech. Engr.*, 193(27-29):3087 – 3128, 2004. ISSN 0045-7825.

John F. Peters, Mark A. Hopkins, Raju Kala, and Ronald E. Wahl. A polyellipsoid particle for nonspherical discrete element method. *Engineering Computations*, 26(6):645–657, 2009.

Johann Rudi, A Cristiano I Malossi, Tobin Isaac, Georg Stadler, Michael Gurnis, Peter WJ Staar, Yves Ineichen, Costas Bekas, Alessandro Curioni, and Omar Ghattas. An extreme-scale implicit solver for complex pdes: highly heterogeneous flow in earth's mantle. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, page 5. ACM, 2015.

Lee M. Taylor and Dale S. Preece. Simulation of blasting induced rock motion using spherical element models. *Engineering Computations (Swansea, Wales)*, 9(2):243 – 252, 1992. ISSN 0264-4401.

Vinodh Vedachalam and Davy Virdee. Discrete element modelling of granular snow particles using liggghts. *M. Sc., University of Edinburgh*, 2011.

David W Washington and Jay N Meegoda. Micro-mechanical simulation of geotechnical problems using massively parallel computers. *International journal for numerical and analytical methods in geomechanics*, 27 (14):1227–1234, 2003.

Christian Wellmann, Claudia Lillie, and Peter Wriggers. A contact detection algorithm for superellipsoids based on the common-normal concept. *Engineering Computations*, 25(5):432–442, 2008.

John R Williams and Alex P Pentland. Superquadrics and modal dynamics for discrete elements in interactive design. *Engineering Computations*, 9 (2):115–127, 1992.

John R Williams, Eric Perkins, and Ben Cook. A contact algorithm for partitioning n arbitrary sized objects. *Engineering Computations*, 21(2/3/4): 235–248, 2004.

Beichuan Yan. Three-dimensional discrete element modeling of granular materials and its coupling with finite element method. Technical report, Thesis (Ph.D.)–University of Colorado, 2008.

Beichuan Yan and Richard A. Regueiro. Comparison Between $O(n^2)$ and $O(n)$ Neighbor Search Algorithm and its Influence on Superlinear Speedup in Parallel 3D Discrete Element Method (DEM) for Complex-shaped Particles. In review, 2016a.

Beichuan Yan and Richard A. Regueiro. Superlinear Speedup Phenomenon in Parallel 3D Discrete Element Method (DEM) Simulations of Complex-shaped Particles. In review, 2016b.

Beichuan Yan, Richard A. Regueiro, and Stein Sture. Three dimensional ellipsoidal discrete element modeling of granular materials and its coupling with finite element facets. *Engineering Computations*, 27(4):519–550, 2010.

B. Zhang, R.A. Regueiro, A.M. Druckrey, and K. Alshibli. Construction of poly-ellipsoidal grain shapes from smt imaging on sand, and the development of a new dem contact detection algorithm. *Engineering Computations*, 2017.