



HAL
open science

Grains3D, a flexible DEM approach for particles of arbitrary convex shape - Part II: Parallel implementation and scalable performance

Andriarimina Daniel Rakotonirina, Anthony Wachs

► **To cite this version:**

Andriarimina Daniel Rakotonirina, Anthony Wachs. Grains3D, a flexible DEM approach for particles of arbitrary convex shape - Part II: Parallel implementation and scalable performance. Powder Technology, 2018, 324, pp.18 - 35. 10.1016/j.powtec.2017.10.033 . hal-01857743

HAL Id: hal-01857743

<https://ifp.hal.science/hal-01857743>

Submitted on 17 Aug 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Grains3D, a flexible DEM approach for particles of arbitrary convex shape - Part II: parallel implementation and scalable performance

Andriarimina Daniel Rakotonirina^a, Anthony Wachs^{b,c,*}

^a*IFP Energies nouvelles, Fluid Mechanics Department, Rond-point de l'Echangeur de Solaize, BP 3, 69360 Solaize, France.*

^b*Department of Mathematics, University of British Columbia, 1984 Mathematics Road, Vancouver, BC, Canada, V6T 1Z2.*

^c*Departement of Chemical and Biological Engineering, University of British Columbia, 2360 East Mall, Vancouver, BC Canada, V6T 1Z3.*

Abstract

In [1] we suggested an original Discrete Element Method that offers the capability to consider non-spherical particles of arbitrary convex shape. We elaborated on the foundations of our numerical method and validated it on assorted test cases. However, the implementation was serial and impeded to examine large systems. Here we extend our method to parallel computing using a classical domain decomposition approach and inter-domain MPI communication. The code is implemented in C++ for multi-CPU architecture. Although object-oriented C++ offers high-level programming concepts that enhance the versatility required to treat multi-shape and multi-size granular systems, particular care has to be devoted to memory management on multi-core architecture to achieve reasonable computing efficiency. The parallel performance of our code Grains3D is assessed on various granular flow configurations comprising both spherical and angular particles. We show that our parallel granular solver is able to compute systems with up to a few hundreds of millions of particles. This opens up new perspectives in the study of granular material dynamics.

Keywords: Granular flow; Discrete Element Method; Angular particles; Parallel computing;

1. Introduction

Discrete Element Method (DEM) based simulations are a very powerful tool to simulate the flow of a granular media. The foundations of the method were introduced by Cundall and Strack [2] in the late seventies. Originally developed for contacts between spherical particles, the method was later extended to polyhedra by Cundall in 1988 [3].

*Corresponding author

Email address: wachs@math.ubc.ca (Anthony Wachs)

The conceptual simplicity combined with a high degree of efficiency has rendered DEM very popular. However, there are essentially still two bottlenecks in DEM simulations: (i) the non-sphericity of most real life particles and (ii) the generally large number of particles involved even in a small system.

In [1] we addressed issue (i), i.e., the non-sphericity of particles by reviewing the various existing techniques to detect collisions between two non-spherical particles and by suggesting our own collision detection strategy that enables one to consider any convex shape and any size. Issue (ii) can be tackled in two different and complementary ways. The former involves improving the computational speed of classical serial implementations of DEM. This can be achieved by a higher quality programming and smarter algorithms, but there is admittedly a limit in that direction, even with the most advanced implementations. The latter involves dividing the work load between different computing units and hence using distributed computing. Nowadays, there are two (potentially complementary) technologies for DEM distributed computing: CPU [4, 5, 6, 7, 8, 9] vs GPU [10, 11, 12, 13, 14, 15, 16, 9]. Both technologies have assets and drawbacks. Interestingly, the definition of large-scale computing fluctuates quite a lot from one publication to another publication as well as changes with time and fast-evolving supercomputing resources. While GPU is parallel in essence (multi-threaded), fast on-chip memory is limited in size and global memory access is very slow, which can result in a weak performance of the code [11]. Besides, the built-in parallelism of GPU is not yet fully designed for multi-GPU computations, which may limit the overall performance to that of a single GPU, in particular in terms of system size, i.e., number of particles. However, recent developments have shown that reasonable scalability can be achieved with single- and multi-GPU computations, as summarized in TAB. 1. Please note that, as emphasized by Shigeto and Sakai [9], the speed ratio 1 GPU/1 CPU in TAB. 1 and in general one-to-one GPU vs CPU comparisons might not always be fair.

CPU-based DEM codes generally exhibit no limit in number of communicating CPUs (cores) and hence no limit in number of particles, provided the scalability is maintained at a reasonable level. Communications between cores is achieved using MPI [17]. Although computations with up to a few tens of millions of particles are emerging with GPU-based implementations [10, 11, 15, 12], simulations with up to a few billions of particles can be envisioned with CPU-based implementations, provided computational practitioners have access to large supercomputers with many thousands of cores [6, 7, 8]. The forthcoming new GPU technology is likely to offer similar parallel computing capabilities as the CPU technology, either by improving inter-GPU communications without using CPUs or by

speeding up data exchange between GPUs and CPUs. At the time we write this article, this enhanced GPU technology is not available yet. Multi-CPU implementations already have or will soon have to address other challenges related to the evolution from multi-core to many-core technology, i.e., computing nodes have more CPUs that each have more cores. The current multi-core technology also poses tough challenges in terms of memory access and management (that we partly address in this work) but the next generation many-core technology will render these challenges even more crucial. One option to address them involves developing hybrid shared/distributed computing models, i.e., shared on a node with OpenMP and distributed among nodes using MPI [18]. These hybrid implementations might have been optional so far with 8, 16 or even 24 cores per node only, but may become mandatory in the future as the number of cores per node is likely to keep on increasing. Another option is to rethink the programming paradigm, both for many-CPU [19] and many-GPU [20] computing. This is an on-going effort in the scientific computing community.

DEM is used both in dry granular flow computations and in particle-laden flow computations. Collision detection and resolution is generally the most time-consuming part of a DEM computation. When particles are immersed in a fluid, hydrodynamic interactions reduce the number of collisions between particles and hence computations (corresponding to the DEM solver only, not the solution of the fluid mass and momentum conservation equations) for the same number of particles are faster or conversely a higher number of particles can be considered for the same computing time. Fluidized bed simulations are a typical case of a relatively low number of collisions with respect to the number of particles in the system once the bed is sufficiently fluidized, e.g., the inlet velocity is at least 2-3 times the minimum fluidization velocity. In a simple configuration of fluidization in a box, Pepiot and Desjardins [21] perform computations with up to 382 million of spheres on 4096 cores with a parallel efficiency of 85%. In the field of particle-laden flow simulations, let us mention the long term effort of the National Energy Technology Laboratory in the development of the open source code MFIx. In the past few years, MFIx, which has historically been known for its Two-Fluid model (MFIx-TFM) and Particle in Cell model (MFIx-PIC), has been enriched by an Eulerian/Lagrangian model (also called DEM-CFD) that relies on a DEM solver for the Lagrangian tracking of particles with collisions. The performance of the parallel DEM solver involved in the so-called MFIx-DEM model is analyzed by Gopalakrishnan and Tafti in [22]. Computations with up to 10 million of spheres in a fully 3D fluidized bed configuration on up to 256 cores with a reasonably satisfac-

tory parallel efficiency are presented. Using the MFIx-DEM model, Liu and Hrenya [23] also investigate its parallel performance in pseudo-2D rectangular fluidized beds. Computations with up to 10 million of spheres on up to about 80 cores show a satisfactory scalability provided the number of particles per core is 10^5 . Liu and Hrenya also address the important question of the physical time that can be computed versus the number of particles that can be computed and show that the balance is controlled by the domain size to particle size ratio, as smaller particles require smaller time steps to resolve collisions. MFIx possesses an active and large community of users and the literature comprises numerous works using MFIx to examine particle-laden flows and in particular fluidized bed configurations. Among many others, let us mention the recent work of Yang *et al.* [24, 25] using MFIx-DEM to study the flow dynamics in a double slot-rectangular spouted bed that contains around 2.6 million of spheres. Finally, Gel *et al.* [26] recently attempted to improve the parallel performance of MFIx by partly refactoring the code at a rather deep programming level to better fit modern high-performance computing architectures. Significant computing time reductions, up to 8 times improvement, are obtained. This is in line with our effort, presented later in this work, in refactoring our own code Grains3D to attain a satisfactory parallel performance.

Our goal in this paper is to elaborate on a simple domain decomposition based parallel extension of our granular code Grains3D and to assess its computing performance on systems of up to a few hundreds of millions of particles. Please note that most references given above considered spheres or spheroids. The main strength of our implementation is the ability to combine our simple but efficient parallel implementation to our collision detection strategy for non-spherical and angular particle shapes [1], and hence to target large-scale DEM computations of many millions of particles of, e.g., polyhedral shape. In *Section 2*, we quickly recall the features of our numerical model as already explained in [1]. We then present our parallel strategy in *Section 3*. In *Section 4* we measure the computing performance of our parallel implementation in various granular flow configurations (particle shape, particle load by core, weak scalability). Finally, we discuss parallel computing performances exhibited by Grains3D in *Section 5* and highlight the remaining intrinsic limitations of Grains3D and how to relax them.

2. Numerical model

The motion of the granular material is determined by applying Newton's second law to each particle $i \in \langle 0, N - 1 \rangle$, where N is the total number of particles. The rigid body

motion assumption leads to the decomposition of the velocity vector \mathbf{v} as $\mathbf{v} = \mathbf{U} + \boldsymbol{\omega} \wedge \mathbf{R}$, where \mathbf{U} , $\boldsymbol{\omega}$ and \mathbf{R} denote the translational velocity vector of the center of mass, the angular velocity vector of the center of mass and the position vector with respect to the center of mass, respectively. The complete set of equations to be considered is the following one:

$$M_i \frac{d\mathbf{U}_i}{dt} = \mathbf{F}_i \quad (1)$$

$$\mathbf{J}_i \frac{d\boldsymbol{\omega}_i}{dt} + \boldsymbol{\omega}_i \wedge \mathbf{J}_i \boldsymbol{\omega}_i = \mathbf{M}_i \quad (2)$$

$$\frac{d\mathbf{x}_i}{dt} = \mathbf{U}_i \quad (3)$$

$$\frac{d\boldsymbol{\theta}_i}{dt} = \boldsymbol{\omega}_i \quad (4)$$

where M_i , \mathbf{J}_i , \mathbf{x}_i and $\boldsymbol{\theta}_i$ stand for the mass, inertia tensor, center of mass position and angular position of particle i . \mathbf{F}_i and \mathbf{M}_i are the sum of all forces and torques applied on particle i , respectively, and can be further decomposed in purely granular dynamics (i.e., without accounting for any external forcing as e.g. hydrodynamic or electrostatic) into a torque-free gravity contribution and a contact force contribution as:

$$\mathbf{F}_i = M_i \mathbf{g} + \sum_{j=0, j \neq i}^{N-1} \mathbf{F}_{ij} \quad (5)$$

$$\mathbf{M}_i = \sum_{j=0, j \neq i}^{N-1} \mathbf{R}_j \wedge \mathbf{F}_{ij} \quad (6)$$

where \mathbf{F}_{ij} is the force due to collision with particle j and \mathbf{R}_j is a vector pointing from the center of mass of particle i to the contact point with particle j . In our contact force model, \mathbf{F}_{ij} comprises a normal Hookean elastic restoring force, a normal dissipative force and a tangential friction force. In particular, we do not consider any history term in our contact force model as, e.g., for tangential friction. The rationale for this intentional choice is as follows: a contact force model with a history term is significantly more computationally intensive than the selected simple contact force model without any history term. Computing contact forces is a serial task, hence using a simple contact force model that leads to speed up the serial parts of the code enables us to be more demanding on the parallel performance of our code (conversely, using a contact force model with a history term that requires more serial computations would artificially improve the parallel performance, although more data would need to be communicated by MPI).

The set of equations Eq. (1)-Eq. (4) is integrated in time using a second order leap-frog Verlet scheme. Rotations of particles are computed using quaternions for computational

efficiency as well as to avoid any gimbal lock configurations. The collision detection algorithm is a classical two-step process. Potential collisions are first detected via a linked-cell list and then actual collisions are determined using a Gilbert-Johnson-Keerthi (GJK) algorithm. The GJK algorithm is an iterative procedure that computes the minimal distance between two convex bodies. The distance function is based on a support function specific to each convex shape. The support function is known for a sphere, a polygon, a polyhedron, a cylinder and a cone, which enables us to consider a large variety of convex shapes and size. The whole collision detection algorithm is a 3-step procedure that operates as follows: (i) convex bodies that potentially collide are shrunk by their crust width, (ii) the GJK algorithm is used to compute the minimal distance between the shrunk bodies, and (iii) the bodies are swollen back to their original size and the contact features are reconstructed. For more detail about our GJK-based collision detection strategy, we refer the reader to Grains3D-Part I [1] and the references therein.

3. Domain Decomposition parallel strategy

Our parallel strategy is classical and is based on a domain decomposition technique. We consider below only the case of a constant in time domain decomposition, assuming that we know how to guarantee a reasonable load balancing of number of particles between subdomains over the whole simulation. The extension to dynamic load balancing in granular flows with large particle volume fraction heterogeneities will be shortly discussed in *Section 5* as an extension of this work.

We employ a cartesian domain decomposition. Each process hosts a single subdomain and we hence define a cartesian MPI communicator using the `MPI_Cart_create` command. It is then very convenient to identify the neighbouring subdomains on each subdomain as well as to implement multi-periodic boundary conditions. On each subdomain, we construct a cartesian linked-cell list with an additional layer of cells at the boundary with neighbouring subdomain to serve as an overlapping zone. This overlapping zone hosts clone particles used to compute collisions with particles located on a neighbouring subdomain (process). As a consequence, cells in a linked-cell list are tagged based on their location on the subdomain: 0 = interior, 1 = buffer and 2 = clone, as illustrated on FIG. 1(a). At each time step, clone particles are either created, deleted or updated. All particles are tagged based on the cell they belong to. Hence they consistently change status as they move in the subdomain. Corresponding operations are performed on neighbouring subdomains when a particle change status. For instance, if a particle moves from an interior cell (tag

= 0) to a buffer cell (tag = 1), a clone particle (tag = 2) is automatically created on the neighbouring subdomain.

The serial code is implemented in C++ which equips us with the required versatility to handle multiple particle shapes and sizes, based on inheritance mechanism, virtual classes and dynamic typing. Each particle is an instance of a C++ class and all active particles on a subdomain, including particles in buffer and clone zones, are stored in a primary list. Two additional separate lists for buffer and clone particles, respectively, are also created. As a consequence, when information of buffer particles needs to be sent to a neighbouring subdomain, we first loop on the list of buffer particles, extract the relevant information and copy it to a buffer memory container (a standard 1D array, i.e., a standard vector, of doubles or integers). Each subdomain keeps a list of reference particles corresponding to all the types of particle in the simulation. These reference particles store generic data as mass, moment of inertia tensor and geometric features, such that MPI messages contain velocity and position information only and their size is reduced to the minimum.

Assorted communication strategies between processes (subdomains) can be designed, ranging from the simplest strategy to the most advanced (to the best of our knowledge for a cartesian MPI decomposition) strategy. We list below the different strategies we implemented and tested, ranked in growing complexity:

- the `AllGatherGlobal` strategy

All processes send information from their buffer particles to all other processes, regardless of their location in the MPI cartesian grid using a `MPI_Allgather` command. A huge amount of useless information is sent, received and treated by each process. It is however a good starting point and performs well up to 8 (maybe 16) processes maximum.

- the `AllGatherLocal` strategy

All processes send information from their buffer particles to all their neighbouring processes. The amount of useless information is reduced, but it is still far from optimal. This strategy performs reasonably well up to 16 (maybe 32) processes, but beyond the scalability markedly deteriorates.

- the `AllGatherLocal` strategy with non-blocking sending

The next level of sophistication involves performing the following among neighbouring processes: non-blocking sending of messages with the `MPI_Isend` command combined with classical blocking receiving with the `MPI_Recv` command. Incoming messages

are first checked with the `MPI_Probe` command and their size is detected with the `MPI_Get_count` command such that the receiving buffer is properly allocated for each received message [7, 8]. Using non-blocking sending speeds up communications as the MPI scheduler can initiate receiving operations even if sending operations are not completed, but still a lot of useless information is sent, received and treated.

- the adopted optimal strategy called `SendRecv_Local_Geoloc`

Not only cells (and hence particles belonging to these cells) are tagged in terms of their status (0 = interior, 1 = buffer and 2 = clone, see FIG. 1(a)) but cells in the buffer zone are also tagged in terms of their location with respect to the neighbouring subdomains using a second tag, named `GEOLoc` for geographic location, that takes the 26 following values (whose meaning is rather obvious on a 3D cartesian grid as can be seen in FIG. 1(b)): `EAST` & `WEST` in the x direction, `NORTH` & `SOUTH` in the y direction, `TOP` & `BOTTOM` in the z direction are the main neighbours, `NORTH_EAST`, `NORTH_WEST`, `SOUTH_EAST`, `SOUTH_WEST`, `NORTH_BOTTOM`, `NORTH_TOP`, `SOUTH_BOTTOM`, `SOUTH_TOP` are the edge neighbours, and `NORTH_WEST_TOP`, `NORTH_WEST_BOTTOM`, `SOUTH_WEST_TOP`, `SOUTH_WEST_BOTTOM`, `NORTH_EAST_TOP`, `NORTH_EAST_BOTTOM`, `SOUTH_EAST_TOP`, `SOUTH_EAST_BOTTOM` are the corner neighbours. The aftermath is an exact tailoring of sent messages with the appropriate information only, thus reducing the size of each sent message to the minimum.

Depending on the particle's `GEOLoc` tag, information from a buffer particle is copied to one or more buffer vectors to be sent to neighbouring subdomains. There are essentially three situations as illustrated below:

- a buffer particle with a main `GEOLoc` tag: for instance a particle tagged `SOUTH` is sent to the `SOUTH` neighbouring subdomain only (FIG. 2),
- a buffer particle with an edge `GEOLoc` tag: for instance a particle tagged `SOUTH_EAST` is sent to the `SOUTH`, `EAST` and `SOUTH_EAST` neighbouring subdomains only (FIG. 3),
- a buffer particle with a corner `GEOLoc` tag: for instance a particle tagged `SOUTH_WEST_TOP` is sent to the `SOUTH`, `WEST`, `TOP`, `WEST_TOP`, `SOUTH_WEST`, `SOUTH_TOP` and `SOUTH_WEST_TOP` neighbouring subdomains only.

Similarly to the `AllGatherLocal` strategy, exchange of information between neighbouring subdomains is performed by a combination of non-blocking sending opera-

tions using `MPI_Isend` and blocking receiving operations using `MPI_Recv`.

The buffer vectors sent and received by processes are of the C double type. A buffer vector contains for each particle the following data: particle identity number, particle reference type, MPI rank of sending process, velocity, position and orientation for a total of 29 numbers. Particle identity number, particle reference type and MPI rank of sending process are integer numbers and are cast into double numbers such that all features can be concatenated into a single vector of doubles. Hence each process sends to and receives from another neighbouring process a single message containing a vector of doubles with the `MPI_DOUBLE` data type (instead of sending and receiving separately in two different messages a vector of doubles with the `MPI_DOUBLE` data type and a vector of integers with the `MPI_INT` data type, respectively). Each message size is then 29 times the size of a double times the number of buffer particles with the appropriate `GEOLOC` tag. Due to the considerable latency involved in any MPI message, efficient parallel performance involves keeping the number of messages as low as possible. This explains why we cast integer to double as a way to avoid heterogeneous data types and/or twice more messages (the computing cost of the cast operation from integer to double when sending and back from double to integer when receiving is much smaller than the one associated to sending and receiving 2 messages instead of 1). Another option that we have not tried is to convert all data types to raw bytes and send a single vector of raw bytes using the `MPI_BYTE` data type. Each neighbouring process is then responsible to convert back the received raw byte messages to their original data types. This strategy has been successfully implemented in [7, 8].

At each time step, the full solving algorithm on each subdomain reads as follows:

1. for all particles with status 0 or 1: initialize force to gravity and torque to 0
2. for all particles:
 - (i) detect collisions
 - (ii) compute contact forces & torques
3. for all particles with status 0 or 1:
 - (i) solve Newton's law: EQ. (1) for translational velocity and EQ. (2) for angular velocity
 - (ii) update position EQ. (3) and orientation EQ. (4)

4. search for particles whose status changed from 0 to 1 add them to the list of buffer particles
5. MPI step using the `SendRecv_Local_Geoloc` strategy (in the 3D general case of 26 neighbouring subdomains):
 - (i) copy buffer particles features into the different buffer vectors of doubles depending on their `GEOLOC` tag,
 - (ii) perform non-blocking sendings of each of the 26 buffer vectors of doubles to the corresponding neighbouring subdomains,
 - (iii) for $j = 0$ to 25 (i.e., for each of the 26 neighbouring subdomains):
 - (I) perform a blocking receiving of the vector of doubles sent by neighbouring subdomain j ,
 - (II) Treat the received vector of doubles containing particles information
 - Create or update clone particles
 - Delete clone particles moved out of the subdomain
6. for all particles: based on their new position, update status and `GEOLOC` tags and the corresponding lists of buffer and clone particles

4. Computational performance

In this section, we assess the computational performance of our parallel DEM code `Grains3D` on assorted flow configurations in which load balancing in terms of number of particles per subdomain (process) is approximately constant over the whole simulation. All the test cases considered thereafter are fully three-dimensional. In all computations, each core hosts a single subdomain and a single process. Hence, the terms "per core", "per subdomain" and "per process" are equivalent. All parallel computations presented thereafter are performed on the supercomputer *ENER110*, IFPEN, Lyon, France. *ENER110* comprises 378 16-core nodes for a total of 6048 cores and a peak performance of 110Tflops. Processors are Xeon E5-2670 8C 2.6GHz and the interconnect is high-speed Infiniband FDR. The OS is Linux CentOS 6.4 and our code is compiled with IntelMPI-4.1 and GNU-4.8.5. Additional computations (not shown for the sake of conciseness) performed on the supercomputer *Occigen*, CINES, Montpellier, France and the supercomputer *Jasper*, WestGrid, Edmonton, Canada, gave parallel performances similar to those obtained on *ENER110*.

Our primary goal is to compute larger systems for a given computing time. We therefore assess the computational performance of Grains3D in terms of weak scaling. We compute the parallel scalability factor $S(n)$ by the following expression:

$$S(n) = \frac{T(1, N)}{T(n, N \times n)} \quad (7)$$

where $T(1, N)$ denotes the computing time for a problem with N particles computed on a single core or a single full node and $T(n, N \times n)$ denotes the computing time for a similar problem with $N \times n$ particles computed on n cores or nodes.

4.1. Assessing memory management on multi-core node architecture

4.1.1. Discharge flows in silos

The first test case is a discharge of particles from a silo. Before performing weak scaling tests, we validate our DEM solver versus experimental data. For that purpose, we select the work of González-Montellano *et al.* [27] as a reference because of its conceptual simplicity. Their study consists in comparing their own DEM simulation results to experimental data of spherical glass beads of 13.8 mm diameter discharging from a silo. The silo has a 0.5 m height (H) and 0.25 m sides (L) (FIG. 4(a)). The bottom has a truncated pyramid shape with a square hopper opening of 57 mm sides whose walls make an angle $\theta = 62.5^\circ$ with respect to the horizontal plane. In our simulations, we extend the bottom of the silo to collect all particles flowing through the opening of the hopper (see FIG. 4(b)). Obviously, this does not affect the discharge dynamics and rate.

As in [27], we fill the silo with 14000 spherical particles by performing a first granular simulation with the opening of the hopper sealed by a plate. In this preliminary simulation, we insert all particles together as a structured array in order to reduce the computing time (see FIG. 5 at $t = 0$). To this end, we extend the height of the silo in a way that all particles fit into the silo before they start to settle. The initial particles positions at the insertion time are actually slightly perturbed with a low amplitude random noise in order to avoid any artificial microstructural effect. Particles then settle by gravity and collide until the system reaches a pseudo steady state corresponding to a negligible total kinetic energy (see FIG. 5 at $t = T_{fill}$). As observed in [27], the 14000 spherical particles fill the silo up to $H_m \simeq 0.86H$. After the filling of the silo, the plate that blocks the particles is removed by imposing a fast translational frictionless displacement to start the discharge. Simulations are run until all particles have exited the silo (see FIG. 5 at $t = T_{dis}$).

As in [1] our contact model is the linear damped spring with tangential Coulomb friction for both particle-particle and particle-wall contacts. The magnitude of the parameters

involved in the silo discharge simulations is given in TAB. 2. In [1], we elaborated on the fact that the spring stiffness k_n in our contact model can be linearly related to the Young modulus E of the material. Since the contact duration is inversely proportional to k_n , a high E leads to a short contact duration, and hence a correspondingly small time-step Δt . For glass beads, the Young modulus E is approximately 50 GPa . It leads to a time step magnitude of the order of $\Delta t \sim 10^{-7} \text{ s}$, which would require to compute an unnecessary large number of time steps to simulate the whole discharge of the silo. In fact, as explained in [1], the stiffness coefficient k_n is generally not set in accordance with Hooke's law and Hertzian theory, but rather in a way to control the maximum overlap between particles as they collide. The meaningful parameters from a physical viewpoint are the coefficient of restitution e_n and the Coulomb friction coefficient μ_c . A smaller k_n enables us to use much larger time steps without affecting the whole dynamics of the system. This is rather customary in DEM simulations of non-cohesive materials. For more detail about how to determine k_n , the reader is referred to [1, 28, 29, 30] and the references therein. In TAB. 3, the meaningful physical parameters e_n and μ_c are set to exactly same values as those selected by González-Montellano *et al.* [27]. Using an estimate of the maximum collisional velocity of $v_{col} = 4.5 \text{ m/s}$, the selected value of k_n leads to a maximum overlap distance of 3% of the sphere radius. Please note that this estimate is highly conservative as $v_{col} = 4.5 \text{ m/s}$ is the free fall velocity of particles as they collide with the bottom wall of the collecting bin underneath the hopper opening. In fact, the collecting bin height is $\approx 1 \text{ m}$, hence we get $\sqrt{2 \times 9.81 \times 1} \approx \sqrt{20} \approx 4.5 \text{ m/s}$. In the dense discharging granular material above the hopper opening, the actual collisional velocity is much less. As a result, the maximum overlap between colliding particles in this part of the granular flow is less than 0.1% of the sphere radius, a value commonly deemed to be a very satisfactory (and almost over-conservative) approximation of rigid bodies in DEM simulations.

We report in TAB. 3 the values of the discharge time experimentally measured by González-Montellano *et al.* [27] together with our simulation result. González-Montellano *et al.* [27] carried out three times the same experiment but it seems that the observed deviation of the discharge time with respect to the mean value is very limited (of the order of 0.2%). In other words, the initial microstructure of the particles in the silo before removing the hopper gate is essentially similar and does not markedly affect the discharge process. Based on this observation, we perform a single discharge simulation. Our model shows a (even surprisingly) good agreement with the discharge time measured in the experiments of González-Montellano *et al.* [27]. Snapshots of the discharge process

also exhibit a highly satisfactory agreement between our simulations and the experiments in [27], as presented in FIG. 6. Although our goal in this work is not to carry out an extensive analysis of the discharge, it is computationally cheap and important to validate our model and gain confidence in the computed results. We are now in a sensible position to perform weak scaling tests and assess the scalability properties of our parallel DEM solver.

4.1.2. Parallel scalability

On the single-core architecture of the 90s, each core had its own levels of cache and its own random-access memory (RAM). The limitation of parallel implementations was hence essentially the communication overhead. This overhead depends on the MPI strategy (size of message, synchronous/asynchronous communication, blocking/non-blocking communication, etc). Since the early 2000s, the new emerging architecture relies on multi-core processors. In a supercomputer, these multi-core processors are bundled in computing nodes, i.e., a computing node hosts multiple processors that each hosts multiple cores. Cores share levels of cache on the processor they belong to and processors share RAM on the computing node they belong to. The aftermath is a more complex and competitive access to memory by all the cores of a computing node. Hence, parallel implementations running on modern supercomputers can be limited as much by the communication overhead as by the intra-processor and intra-node memory management and access. Our parallel DEM solver Grains3D is programmed in C++ and has gone through a deep refactoring along the following guidelines: (i) use object-oriented programming concepts at a very high level of design only, (ii) use standard old-fashioned C/F77-like containers whenever possible and (iii) slightly over-allocate memory and reduce to the absolute minimum dynamic memory management. There is still room for improvement in our implementation but we are now in a position to present acceptable parallel properties. In this section, we design two slightly different multi-silo discharge configurations in order to discriminate the computing overhead related to (i) memory competitive access and management and pure MPI communication latency from (ii) actual MPI communications and treatment of received information.

The first flow configuration consists in discharging particles from several silos using the previous configuration (FIG. 4). The multiple silos case is designed in a way that a silo is handled by a single core without any actual communication with neighbouring sub-domains (FIG. 7). In fact, silos are located far enough from each other to avoid the creation and destruction of clone particles. This flow configuration is hence illustrative of

case (i): memory competitive access and management and MPI communication latency. In fact, the code runs in MPI but messages are empty. The overhead coming from MPI is hence essentially related to the latency of the MPI scheduler to send and receive messages. We adopt a two dimensional domain decomposition ($N_{cores,x} \times N_{cores,y} \times 1 = N_{cores}$) to guarantee exact load balancing between the cores. We evaluate the scalability of our code by gradually increasing the size of our system. To this end, we perform discharge simulations of 2,000 cubic particles and 2,000, 14,000, and 100,000 spherical particles per silo, starting from one silo till 256 silos. Varying the load of particles per core changes the amount of memory allocated, managed and accessed by the code on each core. This enables us to discriminate further between memory management and MPI latency so that the effects of these two factors are not mixed up. In fact, MPI latency is independent of the particle load as the number of messages sent and received scales with the number of cores. The total number of particles N_T in the system is a multiple of that in a single core system and is defined as follows:

$$N_T = N_{p,1} \times N_{cores} \quad (8)$$

where $N_{p,1}$ and N_{cores} are respectively the number of particles on a single core system and the number of cores. The largest system comprises $100,000 \times 256 = 25,600,000$ of spherical particles. As the granular media is dense in most of the domain, the largest part of the computing time (more than 85%) is spent in computing interactions between particles, i.e., contact detection and contact forces. For the weak scaling tests, we run all discharge simulations over 300,000 time steps. Reference times on a single core job are listed in TAB. 4. A first interesting comment about TAB. 4 is that the computing time per particle and per time step is not constant and slightly increases with the size of the system. Even when running in serial mode, memory access is apparently not optimal as containers of larger size (as e.g. a larger list of particles) seem to slow down the computation. Some additional efforts in refactoring the serial implementation of the code are required but this is beyond the scope of the present paper.

The second flow configuration is very similar except that right now all silos are merged together into a big silo. The whole domain is thus shared by each core and actual communications (in the sense communications with non-empty messages) between sub-domains are exchanged (see FIG. 8). For this purpose, we performed discharge simulations of 10,000 cubic particles, 2,000, 14,000 and 100,000 spherical particles. As for the first flow configuration, a two dimensional domain decomposition is chosen such that each sub-domain has approximately the same number of particles as if the silos were independent. This hence

guarantees again an almost perfect load balancing between the cores.

FIG. 9 illustrates the scalability of our code of these two flow configurations. At first sight, results are very similar without (separate silos) and with (merged silos into a big silo) actual communications. We plot in FIG. 9(a) the parallel performance of Grains3D on the first test case, i.e., without any overlap between separate silos and empty MPI messages. This figure indicates that for low numbers of particles per core, the limiting factor is clearly MPI latency while for high numbers of particle per core, the serial computations per core prevail and the MPI latency becomes negligible. Hence, the loss of performance is primarily related to a yet non-optimal memory access and management on multi-core architectures. However, for a high enough number of particles per core as e.g. 100,000 spheres, the scaling factor $S(n = N_{cores})$ is independent of n up to $n = 256$ cores and is around 0.85. As the contact detection of convex bodies is more time-consuming than that of spheres (generally around 5 to 10 times longer for convex bodies than for spheres [1]), $S(n)$ for cubes is higher than $S(n)$ for spheres for the same number of particles per core. Hence, we expect that for more than 100,000 particles per core, the observed scaling factor of 0.85 for spheres is actually a lower bound and that the scaling factor for non-spherical particles should be higher. We plot in FIG. 9(b) the parallel performance of Grains3D on the second test case, i.e., a big silo split into sub-domains and non-empty MPI messages. The 2000 spheres per core is a special case as on each sub-domain there are almost as many particles on the actual sub-domain, i.e., interior and buffer zones, than in the clone layer, leading to a high global communication overhead (size of messages and treatment of information received). This is getting worse and worse as the number of cores increases (see blue line in FIG. 9(b)). The general outcome is in line with the first test case with empty messages: for a large enough number of particles per core, the scaling factor $S(n)$ is satisfactory (it is actually 0.78 for 100,000 spheres and is likely to be higher for 100,000 non-spherical particles). This is again emphasized in FIG. 10 where we compare the communication overhead to the serial computational task for a sphere and a polyhedron (here a cube but this statement applies to any polyhedron). The difference shown there is primarily due to the contact detection that requires to use a GJK algorithm for non-spherical particles while it is analytical (and hence faster) for spheres (see [1] for more detail). Interestingly, for 100,000 spheres, the scaling factor $S(n)$ drops from 0.85 with empty messages to 0.78 with non-empty messages and treatment of the received information. Therefore, the actual overall parallel overhead is around 7% and the rest of the loss of performance, i.e., the remaining 15%, is predominantly due to non-optimal memory access and management on

multi-core chips. For a dense granular flow with a minimum load of 100,000 of particles per core, we can expect a good overall parallel performance with a scaling factor $S(n) \gtrsim 0.75$ on up to 512 to 1024 cores. This is deemed to be very satisfactory for engineering and fundamental physics purposes. Systems with a low particles load per core, i.e., of the order of a few thousands, show an unsatisfactory, although not dramatically poor, parallel performance that exhibits the obvious tendency to degrade with the number of cores n .

4.2. Granular slumping

4.2.1. Dam break collapse

Granular column collapse is a very classical flow configuration to understand the fundamental dynamics of granular media [31, 32, 33, 34, 35]. The "dam break" configuration in a rectangular channel has been extensively studied by many authors, experimentally [34, 35, 32], analytically [32] and numerically [36], among others. The experimental set up is cheap and experiments are easy to conduct. The overall picture of granular column collapse has been described in many papers and books (and in particular in the aforementioned papers) but a fully comprehensive understanding is still lacking. To summarize, the macroscopic features of the collapse, i.e., the final height H_∞/H and the run-out distance $(L_\infty - L)/L = X_f/L$, scale with the initial aspect ratio $a = H/L$ of the column, where H and L denote the initial height and initial length of the column, respectively, and H_∞ and L_∞ denote the final height and final length of the column, respectively. It has been established and verified by many authors that H_∞/H and $(L_\infty - L)/L$ are essentially functions of a and vary as $H_\infty/H \simeq \lambda_1 a^\alpha$ and $(L_\infty - L)/L \simeq \lambda_2 a^\beta$, with $\alpha \approx 1$ for $a \lesssim 0.7$ and $\alpha \approx 1/3$ for $a \gtrsim 0.7$, and $\beta \approx 1$ for $a \lesssim 3$ and $\beta \approx 2/3$ for $a \gtrsim 3$, although Balmforth and Kerswell found slightly different exponents [32]. Anyhow, the constants λ_1 and λ_2 are largely undetermined. In the inertia dominated regime $a \gtrsim 3$, Lube *et al.* [35] suggested that $\lambda_2 = 1.9$. Although the qualitative description of granular column collapse in a rectangular channel is acknowledged by all contributors to the field, significant quantitative discrepancies can be found in terms of experimentally measured run-out distances between e.g. [35] and [32]. It is admitted that the problem is primarily governed by the initial aspect ratio a but the various existing studies also suggest that λ_1 and λ_2 might not be true constants but functions of the transverse dimension of the channel (narrow or wide slots), the type of material and the shape of the particles, although this functional dependence might be weak. In any case, the scaling analysis is assumed to be valid, which implies that the general behavior and hence H_∞/H and $(L_\infty - L)/L$ are independent of the dimensional system size.

In [36], we used Grains3D to carry out an extensive analysis of dam break granular collapses in a rectangular channel and satisfactorily reproduced the experimental data of Lajeunesse *et al.* [34]. Here our objective is twofold: (i) show that the scaling analysis is indeed valid by computing systems of increasing size but constant a and that the computed run-out distance is within the reported experimental range of values, and (ii) use the largest system as a reference point for weak scaling parallel tests.

4.2.2. Numerical simulation

Simulations are performed based on a well-known experimental set-up: a box with a lifting gate (see FIG. 11). The simulation procedure consists in filling the parallelepipedic reservoir of length L and width W up to a height H with granular media. Particles are inserted at the top of the reservoir. They settle by gravity and collide until the system reaches a pseudo steady state corresponding to a negligible total kinetic energy. Then, the gate is lifted over a time scale much smaller than that of the collapsing media in a sense that it does not affect the dynamics of the whole system. The moving gate is also chosen to be frictionless to avoid particle located close to the gate to be artificially lifted in the air. The lateral boundaries of our system are subjected to periodic conditions to mimic an infinite granular media in the transverse direction to the flow. Particles are assumed to have a mono-sized icosahedral shape that mimics quartz-sand grains. Icosahedral particles have an equivalent diameter d_p (diameter of a sphere of same volume as the icosahedron) of 3 mm . The magnitude of the parameters involved in the granular collapse simulations is given in TAB. 5. We take the free-fall settling velocity of the highest heap of particles (Size 5, $H = 0.905\text{ m}$) as an estimate of the maximum collisional velocity. We hence get $v_{col} = \sqrt{2 \times 9.81 \times 0.905} \approx 4.2\text{ m/s}$. The theoretical maximum overlap is of the order of maximum 5% of the particle equivalent radius as shown in Table 5. In practice, the average overlap and maximum overlap in all simulation are of the order of 0.1% and 1%, respectively.

We fix a to roughly 7.3 and select five systems of increasing dimensional size. The way we proceed is as follows: we set $W = 0.5\text{ m}$ and select $L = L_1 = 0.025\text{ m}$ has the length of the smallest system. We fill the reservoir with $N_1 = 98,000$ mono-disperse icosahedral particles and the resulting height is $H = H_1 = 0.187\text{ m}$. The 4 other systems have the following features: $i \in \{2, 5\}$, $L_i = iL_1$, $N_i = i^2N_1$, $H_i \approx iH_1$. The simulation of the filling process results in the following actual height and aspect ratio of the reservoir of particles for the different systems:

- Size 1: 98000 particles ($H = 0.187\text{ m}$, $L = 0.025\text{ m}$, $a = 7.475$)

- Size 2: 392000 particles ($H = 0.365m, L = 0.05m, a = 7.305$)
- Size 3: 882000 particles ($H = 0.547m, L = 0.075m, a = 7.296$)
- Size 4: 1568000 particles ($H = 0.731m, L = 0.1m, a = 7.31$)
- Size 5: 2450000 particles ($H = 0.905m, L = 0.125m, a = 7.238$)

The resulting aspect ratio a is $7.3 \pm 2.3\%$. The observed limited deviation of 2.3% is the aftermath of systems of slightly different compaction. In fact, the initial height is a result of the filling simulation and cannot be set a priori. It is only known after all particles have settled in the reservoir and the system exhibits a negligible total kinetic energy. It has been noticed that once the free fall phase of all particles is complete, the system relaxes and densifies extremely slowly over a time scale of a few seconds at least. Slow microstructural re-arrangements lead to a progressively more compact granular media in the reservoir. Actually, starting from a loose packing, the compaction of the system can be very slow, even with successive vertical taps [37]. In terms of computational cost, this situation may lead to an extremely long simulation time since the typical time step is of the order of a micro-second. We assume that these slight variations of the initial aspect ratio a and correspondingly of the initial volume fraction and microstructure of the granular media have a very low impact on the whole granular collapse. In the worst case, it will result in similar slight variations of the final height and the final run-out distance.

Measuring the run-out distance in an unbiased way is not straightforward as once the collapse is complete the front of the deposit of particles is diffuse (detached particles are spread out). We estimate the total length of the final deposit L_∞ as the minimal length that satisfies $\phi(L_\infty) \leq \phi_{min} = 0.1$, where $\phi(X)$ is the solid volume fraction in a box-like control volume $V_b = d_p \times W \times d_p$ that spans the whole transverse dimension of the flow domain and is centered at $(X, W/2, d_p/2)$. Note that changing ϕ_{min} from 0.1 to 0.05 or 0.025 does not change significantly L_∞ .

FIG. 12 and FIG. 13 illustrate the dynamics of the granular collapse and the time evolution of the free surface in a 2D $X - Z$ cut plane and in 3D, respectively, for case Size 4. As observed by [35], the early transient of the collapse correspond to a free-fall regime (FIG. 12 (a)-(c)) until the flow transitions to a phase over which the advancing front of the collapsing granular media reaches a quasi-constant velocity (FIG. 12 (d)-(f)), and finally the flow is friction-dominated and slows down to rest. Interestingly, over the second phase, the front of the collapsing granular media shows a rather chaotic dynamics. Although the front advances at a quasi-constant velocity, the singularity that the front represents leads to a high level of particles agitation with many particles being ejected/detached from the

mass to ballistically free-fly until they settle back on the deposit.

As experimentally observed by many authors, our computed results confirm that the overall dynamics and in particular the final height, run-out distance and cross-sectional profile of the deposit are independent of the size of the system and solely controlled by the initial aspect ratio a . We present in FIG. 14 a view from the top of the final deposit together with the scaled total length of the deposit L_∞/L obtained with the criterion $\phi \leq 0.1$ (red line) for all systems. The variation of the run-out distance $(L_\infty - L)/L$ is quantitatively plotted in FIG. 15. It is pretty obvious that $(L_\infty - L)/L$ is quasi-constant as a function of the size of the system. The limited variations obtained are primarily a result of the slight variations of a for the different sizes in the computations.

Finally, the final scaled cross-sectional profiles of the deposit for all system sizes nicely collapse on a unique master plot, as shown in FIG. 16, emphasizing once again the dependence to a and not to the dimensional size of the system. Let us complete this subsection by shortly discussing the value of the obtained run-out distance. [34] and [35] agree on the scaling exponents while [32] suggests slightly different values. Please note that all these works are experimental. For inertia-dominated regimes $a \gtrsim 3$, Lube *et al.* [35] even determine that the value of the constant λ_2 is around 1.9 and independent of the granular material properties and shape. Using their correlation $(L_\infty - L)/L \simeq 1.9a^{2/3}$, we get for $a \approx 7.3$, $(L_\infty - L)/L = 1.9 \times 7.3^{2/3} \simeq 7.15$, a value significantly less than our numerical prediction of ≈ 10.5 . In their experiments, Lube *et al.* have lateral walls while we have periodic conditions, i.e., no frictional resistance from any lateral walls. This difference in the flow configuration may qualitatively justify that our run-out distance is larger (less frictional resistance leads to a larger spread out of the granular media) but is probably not sufficient to quantitatively explain the discrepancy. The so-called Series B experiments of Lube *et al.* have the following features: (i) the channel is 20cm wide, (ii) they used coarse quartz sand of average size 1.5mm, hence the channel width to average grain size ratio is $W/d_p \simeq 133$, and (iii) the initial basal length for $a = 7.3$ is either 4.5cm or 8.3cm, hence the initial basal length to channel width ratio L/W is less than ~ 0.41 . In their experiments, Lube *et al.* reported that effects of the limiting channel walls are minimal. Our Size 2 case is the one that resembles the most Lube *et al.* 's experiments. Anyhow, in the 5 cases we consider, we have $W/d_p \simeq 167$ and L/W ranging from 0.05 to 0.25. We have run a series of additional simulations with lateral walls instead of periodic conditions. Lateral walls are made of the same material as the bottom wall. Corresponding run-out distances are also plotted in FIG. 15. As expected, additional friction with lateral

walls decreases the run-out distance as more energy is dissipated by friction with respect to periodic conditions, but our results confirm that changing lateral boundary conditions for a channel with such a small L/W has a limited quantitative impact on the run-out distance. However, FIG. 15 shows that the run-out distance decreases as L/W increases, emphasizing the fact that the relative importance of frictional resistance from lateral walls is getting stronger as L/W increases. In [32], Balmforth and Kerswell claim that λ_2 is a function of the granular material properties and shape, based on their own experimental results. Figure 11 in [32] suggests that for $a = 7.3$, the run-out distance roughly spans the range [7 : 13] for wide channels, with the largest value found for fine glass. Fine glass grains seem to look moderately angular (see Figure 3 in [32]) and could presumably be well represented by icosahedra. Our computed run-out distance hence falls almost in the middle of the range of values reported in [32]. Overall, our numerical prediction is in good agreement with the assorted experimental values reported in the literature. But additional simulations are required to further determine the right scaling and the potential dependence of that scaling to the granular material properties and shape.

4.2.3. Parallel scalability

We use the Size 5 granular column collapse flow configuration to perform weak scaling tests and further assess the parallel scalability of Grains3D. From *Section 4.1.2*, we learnt that a good parallel performance requires a minimum of $\approx 100,000$ particles per core. Therefore our reference case on a single core approximately corresponds to Size 5 case of *Section 4.2.2* but 24 times narrower. The system on a single core comprises $N_{p,1} = 101850$ icosahedra and its width is $W_1 = 0.021875m$. For parallel computing, we increase the system width and the number of particles accordingly. We adopt a 1D domain decomposition in the Y direction such that each core hosts approximately 101850 particles. Hence, a N_{cores} -core computation corresponds to a system with $N_T = N_{p,1} \times N_{cores}$ particles and of width $W = N_{cores} \times W_1$ as detailed in TAB. 6. The weak scaling tests are performed over the 20,000 first time steps of the collapse.

FIG. 17 shows the overall scalability of our code Grains3D. The code exhibits a very satisfactory performance for a particles load per core of $\approx 100,000$ of regular polyhedra. The scaling factor $S(n = N_{cores})$ is ≈ 0.93 on 512 cores for a system with a quasi-perfect load balancing. The plot seems to indicate a very slight degradation of the performance above 256 cores but the general trend suggests that $S(n)$ should still be $\gtrsim 0.9$ on 1024 cores for a system comprising more than 100,000,000 of regular polyhedra.

4.3. Coupling with a fluid in an Euler/Lagrange framework, application to fluidized beds

The final test case is a fluidized bed, i.e., a flow configuration in which the particles dynamics is not only driven by collisions but also by hydrodynamic interactions with the surrounding fluid. The model implemented here is of the two-way Euler/Lagrange or DEM-CFD type [38, 39, 40, 21]. The principle of the formulation is to write fluid porosity averaged conservation equations with an additional source representing the reaction of the particles on the fluid and to add a hydrodynamic force to the translational Newton's equation for the particles representing the action of the fluid on the particles. In our weak scaling tests below, our objective is primarily to evaluate the parallel scalability of the solid solver only and not to provide any new physical insight in the dynamics of fluidized beds (for the reader interested in physical analysis of fluidized beds performed with our model, please see [41, 42, 43, 44, 45]).

4.3.1. Formulation

The formulation of the set of governing equations dates from Anderson and Jackson [38] in the late 60s and was recently clarified in [46]. In essence, for the fluid part, the mass conservation equation and the momentum conservation equation are averaged by the local fluid porosity. In most formulations, the set of governing equations is integrated in control volumes larger than the particle diameter, although recent advances in this field have shown that it is possible to use a projection kernel disconnected from the grid size [21, 46]. Particles trajectories with collisions and hydrodynamic forces are tracked individually and computed by our granular dynamics code Grains3D. The two-way Euler/Lagrange formulation has been detailed many times in the past literature (see [39, 40, 47] among many others) and we shortly summarize the main features of our own two-way Euler/Lagrange numerical model.

The fluid is assumed to be Newtonian and incompressible. The set of governing equations for the fluid-solid coupled problem reads as follows:

- Fluid equations

We solve the following fluid porosity averaged mass and momentum conservation equations:

$$\frac{\partial \varepsilon}{\partial t} + \nabla \cdot \varepsilon \mathbf{u} = 0 \quad (9)$$

$$\rho_f \left(\frac{\partial(\varepsilon \mathbf{u})}{\partial t} + \nabla \cdot (\varepsilon \mathbf{u} \mathbf{u}) \right) = -\nabla p - \mathbf{F}_{fp} + 2\mu \nabla \cdot (\varepsilon \mathbf{D}) \quad (10)$$

where ρ_f , μ , ε and \mathbf{D} stand for the fluid density, the fluid viscosity, the fluid porosity (also referred to as fluid volume fraction) and the rate-of-strain tensor, respectively.

The pressure gradient term only contains the hydrodynamic pressure and \mathbf{F}_{fp} represents the fluid-particle hydrodynamic interaction force.

- Particles equations

We solve EQ. (1) and EQ. (2) with additional hydrodynamic interaction contributions \mathbf{F}_i and \mathbf{M}_i , respectively. The translational and angular momentum conservation equations of particle i hence read as follows:

$$M_i \frac{d\mathbf{U}_i}{dt} = M_i(1 - \rho_f/\rho_p)\mathbf{g} + \sum_{j=0, j \neq i}^{N-1} \mathbf{F}_{ij} + \mathbf{F}_{fp,i} \quad (11)$$

$$\mathbf{J}_i \frac{d\boldsymbol{\omega}_i}{dt} + \boldsymbol{\omega}_i \wedge \mathbf{J}_i \boldsymbol{\omega}_i = \sum_{j=0, j \neq i}^{N-1} \mathbf{R}_j \wedge \mathbf{F}_{ij} + \mathbf{M}_{fp,i} \quad (12)$$

where ρ_p , $\mathbf{F}_{fp,i}$ and $\mathbf{M}_{fp,i}$ stand for the particle density, the fluid-particle hydrodynamic interaction force exerted on particle i and the fluid-particle hydrodynamic interaction torque exerted on particle i , respectively.

The fluid-particle hydrodynamic interaction force $\mathbf{F}_{fp,i}$ exerted on particle i (and similarly for the torque) derives from the momentum exchange at the particle surface:

$$\mathbf{F}_{fp,i} = \int_{\partial\mathcal{P}_i} \boldsymbol{\tau} \cdot \mathbf{n} dS \quad (13)$$

where $\boldsymbol{\tau}$ denotes the point-wise fluid stress tensor and \mathbf{n} is the normal vector to the particle surface $\partial\mathcal{P}_i$. In the two-way Euler-Lagrange framework, point-wise variables are not resolved. A closure law is hence needed to compute the fluid-solid interaction at the position of each particle [39, 40, 21]. Following previous contributions to the literature, we assume that the dominant contribution to the hydrodynamic interaction is the drag and that the hydrodynamic torque is small enough to be neglected, i.e., we set $\mathbf{M}_{fp,i} = \mathbf{0}$. In our fluidized bed simulations, particles are spherical and we select the drag correlation proposed by Beetstra *et al.* [48, 49] which reads as follows:

$$\mathbf{F}_{i,fp} = \mathbf{F}_{d,i} = 3\pi d\mu(\mathbf{u} - \mathbf{U}_i)g(\varepsilon, \mathcal{R}e_{p,i}) \quad (14)$$

$$g(\varepsilon, \mathcal{R}e_p) = \frac{10(1 - \varepsilon)}{\varepsilon^2} + \varepsilon^2(1 + 1.5\sqrt{1 - \varepsilon}) + \frac{0.413\mathcal{R}e_p}{24\varepsilon^2} \left(\frac{\varepsilon^{-1} + 3\varepsilon(1 - \varepsilon) + 8.4\mathcal{R}e_p^{-0.343}}{1 + 10^{3(1-\varepsilon)}\mathcal{R}e_p^{-0.5-2(1-\varepsilon)}} \right) \quad (15)$$

$$\mathcal{R}e_{p,i} = \frac{\rho_f d_p \varepsilon |\mathbf{u} - \mathbf{U}_i|}{\mu}$$

To compute the reaction term $-\mathbf{F}_{fp}$ of the particles on the fluid flow, we need to use a projection operator from the Lagrangian description of the particles motion to the Eulerian description of the fluid flow. Here we use the simple embedded cube projection kernel

introduced by Bernard *et al.* [41, 42]. The fluid equations are discretized with a classical second-order in space Finite Volume/Staggered Grid discretization scheme and the solution algorithm is of the first-order operator splitting type. The two-way Euler/Lagrange model used here is implemented in the PeliGRIFF platform to which Grains3D is plugged to compute particles trajectories, see [50, 51] among others. For more detail about the formulation of the model and its implementation, the interested reader is referred to [41, 42].

The set of governing equations above can be easily made dimensionless by introducing the following scales: L_c for length, V_c for velocity, L_c/V_c for time, $\rho_f V_c^2$ for pressure and $\rho_f V_c^2 L_c^2$ for forces. In a dimensionless form, the governing equations contain the following dimensionless numbers: the Reynolds number $\mathcal{R}e_c = \frac{\rho_f V_c L_c}{\mu}$, the density ratio $\rho_r = \frac{\rho_p}{\rho_f}$ and the inverse Froude number $\mathcal{F}r = \frac{g L_c}{V_c^2}$.

4.3.2. Simulation set-up and parameters

We consider the fluidization of mono-disperse solid spherical particles in a simple box-like reactor. We use the uniform inlet velocity U_{in} as the characteristic velocity V_c and the spherical particle diameter d_p as the characteristic length L_c . The Reynolds and Froude numbers hence read as follows:

$$\mathcal{R}e_{in} = \frac{\rho_f U_{in} d_p}{\mu} \quad (16)$$

$$\mathcal{F}r_{in} = \frac{g d_p}{U_{in}^2} \quad (17)$$

Results hereafter are presented in a dimensionless form and dimensionless variables are written with a $\tilde{\cdot}$ symbol. Particles positions are initialized as a cubic array arrangement with a solid volume fraction of $\pi/6$. The computational domain is shown in FIG. 18. Inlet boundary condition corresponds to an imposed velocity $\mathbf{u} = (0, 0, 1)$ and outlet boundary condition corresponds to a standard free-flow condition with an imposed reference pressure. Lateral (vertical) boundaries are periodic.

The principle of our weak scaling tests is similar to the one adopted in the scaling tests of the previous sections except that here the reference case is a full node that comprises 16 cores. The domain is evenly decomposed and distributed in the horizontal $x - y$ plane to guarantee an optimal load balancing over the whole simulation, i.e., we adopt a $N_{cores,x} \times N_{cores,y} \times 1 = N_{cores}$ domain decomposition. The reference case on a full 16-core node has the following dimensionless size: $\tilde{L}_x = 200$, $\tilde{L}_y = 80$ and $\tilde{L}_z = 1500$ and initially hosts $200 \times 80 \times 300 = 4,800,000$ of spheres. With a $4 \times 4 \times 1$ domain decomposition, each sub-domain has the following dimensionless size $50 \times 20 \times 1500$ and hosts initially

$N_{p,1} = 50 \times 20 \times 300 = 300,000$ of spheres. The total number of particles in a system is $N_T = 300,000 \times N_{cores} = 4,800,000 \times N_{nodes}$. The initial height \tilde{H}_0 of the bed is 300, such that we also have $\tilde{L}_z/\tilde{H}_0 = 5$. The additional physical and numerical dimensionless parameters of our simulations are listed in TAB. 7. We use the same contact parameters for particle-bottom wall and particle-particle collisions.

Another important dimensionless parameter is the ratio of the inlet velocity U_{in} to the minimum fluidization velocity of the system U_{mf} . Here we select $U_{in}/U_{mf} = 3$ to run our weak scaling tests. To avoid a strong overshoot of the bed over the early transients, we first set $U_{in}/U_{mf} = 2$ for $\tilde{t} \in [0 : 1785]$ and then $U_{in}/U_{mf} = 3$ for $\tilde{t} > 1785$. The weak scaling tests are performed by increasing the length \tilde{L}_x of the system together with the number of particles, as shown in TAB. 8, with \tilde{L}_y and \tilde{L}_z kept unchanged. As \tilde{L}_x increases, the domain horizontal cross-section looks more and more like a narrow rectangle and the bed behaves like a pseudo-3D/quasi-2D bed, as transverse secondary instabilities in the y direction are artificially strongly damped by the narrow periodic length \tilde{L}_y while transverse secondary instabilities in the x direction are free to develop. This configuration is purposely selected to facilitate the visualisation of the bubbles dynamics inside the bed. As expected, the flow field does not vary much in the y direction (see FIG. 19). Note that this does not affect our weak scaling tests since with 4 sub-domains in the y direction and bi-periodic boundary conditions, each sub-domain has exactly 8 neighbors, regardless of the fact that the cross-section is a narrow rectangle or a square. In other words, the cartesian domain decomposition is fully 2D. The evaluation of the scaling factor is carried out over 20,000 time-steps as $U_{in}/U_{mf} = 3$ for $\tilde{t} > 1785$.

FIG. 19(a)-(d) illustrates the early transients for $U_{in}/U_{mf} = 2$ of the simulation with 19,200,000 of particles over which the primary streamwise (in the z direction) instability develops, as well documented in the literature. Then a secondary transversal (horizontal in the x direction) instability triggers, grows and leads to the creation of a first big bubble that eventually bursts. FIG. 19(e)-(i) shows the time evolution of the fluid porosity in a $x - z$ cut plane located at $\tilde{L}_y/2$ over the transition from $U_{in}/U_{mf} = 2$ to $U_{in}/U_{mf} = 3$. For $\tilde{t} > 1785$, the system progressively transitions to its bubbling regime. The level of intermittency decreases with time until the system reaches a pseudo-stationary bubbling regime. The presented results are qualitatively in line with the expected behavior of a fluidized in the selected flow regime [21].

FIG. 20 shows the parallel scalability of our granular solver Grains3D in our fluidized bed parallel simulations. The overall parallel efficiency of our granular solver is very satis-

factory. The scaling factor $S(n = N_{nodes})$ is 0.91 for the largest system investigated, i.e., for 230,400,000 of particles and 48 nodes/768 cores. This very high scalability for such a high number of particles derives from less frequent collisions between particles than in a dense granular media. Although collisions are constantly happening in the system, the presence of the fluid and the overall observed dynamics lead to particles often advancing over a few solid time steps without collide with another particle. We would like to emphasize that, in such a fluidized bed simulation, most of the computing time is spent in computing particles trajectories with collisions, i.e., in the granular solver. This has been shown as well in a companion paper [42]. So overall, measuring the parallel efficiency of the granular solver only in such systems still supplies a rather reliable indication of how the whole fluid-solid solver scales. Although FIG. 20 shows that the scaling factor seems to slightly degrade with increasing the number of nodes, the trend reveals that simulations with a 1 billion of particles on a few thousands of cores can be performed with a reasonably satisfactory scalability. This is indeed very encouraging.

5. Discussion and Perspectives

We have suggested a simple parallel implementation of our granular solver Grains3D based on a fixed cartesian domain decomposition and MPI communications between sub-domains. The MPI strategy with tailored messages, non-blocking sendings and type conversion has proven to be particularly efficient when the flow configuration does not require any particular dynamic load balancing of the number of particles per core. In the three flow configurations investigated in this work, the parallel performance of the code is deemed to be more that acceptable, and even satisfactory to very satisfactory. For systems with more than 100,000 particles per core, the scaling factor $S(n)$ is consistantly larger than 0.75. In case particles are non-spherical, $S(n)$ is actually larger than 0.9 for computations on up to a few hundreds of cores.

We have also shown than the parallel performance is not only limited by the parallel overhead in terms of messages sent by and received from cores combined to copying the required information in buffers before sending and treating the information received, but also by the competitive access to and proper management of random-access memory on a multi-core architecture. The aftermath of this known limitation is the requirement to enhance even the serial parts of the code. This reprogramming task might be tedious but should be very beneficial on the long run as new architectures are likely to have more and more cores per processor and more and more processors per node. Although Grains3D

went through this refactoring process, there is still room for further improvement.

In its current state, Grains3D offers unprecedented computing capabilities. Systems with up to 100,000,000 of non-spherical particles can be simulated on a few hundreds of cores. Besides, the trend shown by the scaling factor as a function of the number of cores or nodes suggests that the milestone of a billion of particles is attainable with a decent parallel performance, without fluid or with fluid in the framework of a two-way Euler/Lagrange coupling method. This will create incentives to examine flow configurations that were beyond reach before and strengthen the position of numerical simulation associated to high performance computing as an indispensable tool to extend our comprehension of granular flow dynamics.

The next research directions that we will explore short-term on the purely computing side to further enhance the computing capabilities of Grains3D are the following ones:

- the development of a dynamic load balancing algorithm to supply a good parallel performance in flow configurations with high particle volume fraction heterogeneities and significant particle volume fraction time variations. We will focus first on purely granular systems, i.e, without any surrounding fluid, and will proceed in two steps. The problem of dynamic load balancing in particle-laden flows introduces another class of challenges related to balancing both the DEM solver and the fluid solver simultaneously. As a first step, we will implement an algorithm that dynamically balances the load of particles per core in one direction only and make sure this algorithm exhibits a good parallel performance. As a second step, we will extend this algorithm to dynamic load balancing in 3 directions. Conceptually, dynamic load balancing is not particularly complex but a parallel implementation that scales well is the true challenge,
- the intra-processor and intra-node limitation due to competitive access to memory and/or MPI latency may be partly corrected by moving to an hybrid OpenMP/MPI parallelisation instead of an all-MPI one, such as the one suggested by Berger *et al.* [18],
- as the number of cores attains a few thousands, the MPI latency as well as the number of messages sent and received might start to become a serious limitation, although we have not explored yet this range of number of cores. In case this should happen, our simple though very efficient so far MPI strategy might necessitate to be upgraded too, with at least improvements in the scheduling of messages or other

techniques,

- finally, although the ability to compute granular flows with non-spherical convex shape opens up fascinating perspectives to address many open questions in the dynamics of real life granular systems, this does not cover all possible particle shapes. In fact, many non-spherical particles are also non-convex. There is hence a strong incentive to devise a contact detection algorithm that can address granular media made of non-convex particles. We will examine this issue in *Grains3D-Part III: extension to non-convex particles*.

Acknowledgements

This work was granted access to the HPC resources of CINES under the allocations *2013-c20132b6699* and *2014-c20142b6699* made by GENCI. This research was enabled in part by support provided by WestGrid (www.westgrid.ca) and Compute Canada (www.computecanada.ca) through Prof. Wachs 2016's computing resource allocation *qpf-764-ab*. The authors would like to thank Dr. Manuel Bernard who developed the two-way Euler/Lagrange numerical model for the coupled fluid/particles computations.

References

- [1] A. Wachs, L. Girolami, G. Vinay, and G. Ferrer. Grains3D, a flexible DEM approach for particles of arbitrary convex shape - Part I: numerical model and validations. *Powder Technology*, 224:374–389, 2012.
- [2] P. A. Cundall and O. D. L. Strack. A discrete numerical model for granular assemblies. *Geotechnique*, 29(1):47–65, 1979.
- [3] P. A. Cundall. Formulation of a three-dimensional distinct element model—Part I. A scheme to detect and represent contacts in a system composed of many polyhedral blocks. *International Journal of Rock Mechanics and Mining Sciences & Geomechanics Abstracts*, 25(3):107–116, 1988.
- [4] D.A. Horner, J.F. Peters, and A. Carrillo. Large scale Discrete Element Modeling of vehicle-soil interaction. *Journal of Engineering Mechanics*, 127(10):1027–1032, 2001.
- [5] M. Lemieux, G. Léonard, J. Doucet, L.-A. Leclaire, F. Viens, J. Chaouki, and F. Bertrand. Large-scale numerical investigation of solids mixing in a V-blender using the Discrete Element Method. *Powder Technology*, 181(2):205–216, 2008.
- [6] J. H. Walther and I. F. Sbalzarini. Large-scale parallel discrete element simulations of granular flow. *Engineering Computations*, 26:688–697, 2009.
- [7] K. Iglberger and U. Rüde. Massively parallel rigid body dynamics simulations. *Computer Science-Research and Development*, 23(3-4):159–167, 2009.
- [8] K. Iglberger and U. Rüde. Large-scale rigid body simulations. *Multibody System Dynamics*, 25(1):81–95, 2011.
- [9] Y. Shigeto and M. Sakai. Parallel computing of Discrete Element Method on multi-core processors. *Particuology*, 9(4):398–405, 2011.
- [10] C. A. Radeke, B. J. Glasser, and J. G. Khinast. Large-scale mixer simulations using massively parallel GPU architectures. *Chemical Engineering Science*, 65:6435–6442, 2010.
- [11] N. Govender, D.N. Wilke, and Shalk K. Collision detection of convex polyhedra on the NVIDIA GPU architecture for the discrete element method. *Applied Mathematics and Computation*, 267:810–829, 2015.
- [12] S. Tsuzuki and T. Aoki. Large-scale granular simulations using Dynamic load balance on a GPU supercomputer. In *Proceedings of the 2014 ACM/IEEE conference on Supercomputing*, 2014.
- [13] T. Washizawa and Y. Nakahara. Parallel computing of discrete element method on GPU. *arXiv preprint arXiv:1301.1714*, 2013.
- [14] D. Jajcevic, E. Siegmann, C. Radeke, and J.G. Khinast. Large-scale CFD–DEM simulations of fluidized granular systems. *Chemical Engineering Science*, 98:298–310, 2013.
- [15] J.Q. Gan, Z.Y. Zhou, and A.B. Yu. A GPU-based DEM approach for modelling of particulate systems. *Powder Technology*, 301:1172–1182, 2016.
- [16] J. Xu, H. Qi, X. Fang, L. Lu, W. Ge, X. Wang, M. Xu, F. Chen, X. He, and J. Li. Quasi-real-time simulation of rotating drum using Discrete Element Method with parallel GPU computing. *Particuology*, 9(4):446–450, 2011.
- [17] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI (2Nd Ed.): Portable Parallel Programming with the Message-passing Interface*. MIT Press, Cambridge, MA, USA, 1999.
- [18] R. Berger, C. Kloss, A. Kohlmeyer, and S. Pirker. Hybrid parallelization of the LIGGGHTS open-source DEM code. *Powder Technology*, 278:234 – 247, 2015.

- [19] A. Vajda. *Programming many-core chips*. Springer Science & Business Media, 2011.
- [20] M. Steuwer and S. Gorlatch. *SkelCL: Enhancing OpenCL for High-Level Programming of Multi-GPU Systems*, pages 258–272. Springer Berlin Heidelberg, 2013.
- [21] P. Pepiot and O. Desjardins. Numerical analysis of the dynamics of two-and three-dimensional fluidized bed reactors using an Euler-Lagrange approach. *Powder Technology*, 220:104–121, 2011.
- [22] P. Gopalakrishnan and D. Tafti. Development of parallel DEM for the open source code MFIX. *Powder Technology*, 235:33–41, 2013.
- [23] P. Liu and C.M. Hrenya. Challenges of DEM: I. Competing bottlenecks in parallelization of gas–solid flows. *Powder Technology*, 264:620–626, 2014.
- [24] S. Yang, K. Luo, K. Zhang, K. Qiu, and J. Fan. Numerical study of a lab-scale double slot-rectangular spouted bed with the parallel CFD–DEM coupling approach. *Powder Technology*, 272:85–99, 2015.
- [25] S. Yang, K. Zhang, and J.W. Chew. Computational study of spout collapse and impact of partition plate in a double slot-rectangular spouted bed. *AIChE Journal*, 61(12):4087–4101, 2015.
- [26] A. Gel, J. Hu, E. Ould-Ahmed-Vall, and A.A. Kalinkin. Modernization and optimization of a legacy open-source CFD code for high-performance computing architectures. *International Journal of Computational Fluid Dynamics*, 31(2):122–133, 2017.
- [27] C. González-Montellano, A. Ramirez, E. Gallego, and F. Ayuga. Validation and experimental calibration of 3D discrete element models for the simulation of the discharge flow in silos. *Chemical Engineering Science*, 66(21):5116–5126, 2011.
- [28] P. W. Cleary and M. L. Sawley. DEM modelling of industrial granular flows: 3D case studies and the effect of particle shape on hopper discharge. *Applied Mathematical Modelling*, 26(2):89–111, 2002.
- [29] P. W. Cleary. Industrial particle flow modelling using discrete element method. *Engineering Computations*, 26(6):698–743, 2009.
- [30] P. W. Cleary. DEM prediction of industrial and geophysical particle flows. *Particuology*, 8(2):106–118, 2010.
- [31] A. Ritter. Die fortpflanzung de wasserwellen. *Zeitschrift Verein Deutscher Ingenieure*, 36(33):947–954, 1892.
- [32] N. J. Balmforth and R. R. Kerswell. Granular collapse in two dimensions. *Journal of Fluid Mechanics*, 538:399–428, 9 2005.
- [33] C. Ancey, R. M. Iverson, M. Rentschler, and R. P. Denlinger. An exact solution for ideal dam-break floods on steep slopes. *Water Resources Research*, 44(1):n/a–n/a, 2008. W01430.
- [34] E. Lajeunesse, J. B. Monnier, and G. M. Homsy. Granular slumping on a horizontal surface. *Physics of Fluids*, 17(10), 2005.
- [35] G. Lube, H. E. Huppert, R. S. J. Sparks, and A. Freundt. Collapses of two-dimensional granular columns. *Phys. Rev. E*, 72:041301, Oct 2005.
- [36] L. Girolami, V. Hergault, G. Vinay, and A. Wachs. A three-dimensional discrete-grain model for the simulation of dam-break rectangular collapses: comparison between numerical results and experiments. *Granular Matter*, 14(3):381–392, 2012.
- [37] J.B. Knight, C.G. Fandrich, C. N. Lau, H. M. Jaeger, and S. R. Nagel. Density relaxation in a vibrated granular material. *Phys. Rev. E*, 51:3957–3963, May 1995.
- [38] T. B. Anderson and R. Jackson. Fluid mechanical description of fluidized beds. equations of motion. *Industrial & Engineering Chemistry Fundamentals*, 6(4):527–539, 1967.

- [39] T. Kawaguchi, T. Tanaka, and Y. Tsuji. Numerical simulation of two-dimensional fluidized beds using the discrete element method (comparison between the two- and three-dimensional models). *Powder Technology*, 96(2):129–138, 1998.
- [40] T. Tsuji, K. Yabumoto, and T. Tanaka. Spontaneous structures in three-dimensional bubbling gas-fluidized bed by parallel DEM-CFD coupling simulation. *Powder Technology*, 184(2):132–140, 2008.
- [41] M. Bernard. *Multi-scale approach for particulate flows*. PhD thesis, Institut National Polytechnique de Toulouse, 2014.
- [42] M. Bernard, A. Wachs, and E. Climent. Controlling the quality of two-way Euler/Lagrange numerical modeling of bubbling and spouted fluidized beds dynamics. *Industrial & Engineering Chemistry Research*, 56(1):368–386, 2017.
- [43] A. Esteghamatian, M. Bernard, A. Lance, A. Hammouti, and A. Wachs. Micro/meso simulation of a fluidized bed in a homogeneous bubbling regime. *International Journal of Multiphase Flow*, 92:93–111, 2017.
- [44] A. Esteghamatian, A. Hammouti, M. Lance, and A. Wachs. Particle resolved simulations of liquid/solid and gas/solid fluidized beds. *Physics of Fluids*, 29(3):033302, 2017.
- [45] A. Esteghamatian, F. Euzenat, A. Lance, A. Hammouti, and A. Wachs. A stochastic formulation for the drag force based on multiscale numerical simulation of fluidized beds. *in revision in International Journal of Multiphase Flow*, 2017.
- [46] J. Capecelatro and O. Desjardins. An Euler–Lagrange strategy for simulating particle-laden flows. *Journal of Computational Physics*, 238:1–31, 2013.
- [47] B. H. Xu and A. B. Yu. Numerical simulation of the gas-solid flow in a fluidized bed by combining discrete particle method with computational fluid dynamics. *Chemical Engineering Science*, 52(16):2785–2809, 1997.
- [48] R. Beetstra, M. A. Van der Hoef, and J. A. M. Kuipers. Drag force of intermediate Reynolds number flow past mono- and bidisperse arrays of spheres. *AIChE Journal*, 53(2):489–501, 2007.
- [49] R. Beetstra, M. A. Van der Hoef, and J. A. M. Kuipers. Numerical study of segregation using a new drag force correlation for polydisperse systems derived from lattice-Boltzmann simulations. *Chemical Engineering Science*, 62(1):246–255, 2007.
- [50] A. Wachs. A DEM-DLM/FD method for direct numerical simulation of particulate flows: Sedimentation of polygonal isometric particles in a Newtonian fluid with collisions. *Computers & Fluids*, 38(8):1608–1628, 2009.
- [51] A. Wachs, G. Vinay, and A. Hammouti. PeliGRIFF Home Page. <http://www.peligriff.com>, 2007-2016.

List of Tables

1	Summary of recent contributions to DEM computations on GPUs. "np" stands for "not reported".	32
2	Contact force model parameters, estimate of contact features at $v_{col} = 4.5 \text{ m/s}$ and time step magnitude used in the silo discharge simulation. . . .	33
3	Comparison between experimental data of [27] and our simulation results with Grains3D for the discharge time of the silo.	34
4	Silo discharge for different systems: reference times of a serial job over 300,000 time steps of 10^{-5} s	35
5	Contact force model parameters, estimate of contact features at $v_{col} = 4.2 \text{ m/s}$ and time step magnitude used in the dam break simulations.	36
6	System size in granular dam break weak scaling tests.	37
7	Fluid and particles physical and numerical dimensionless parameters.	38
8	System size in the fluidized bed weak scaling tests. Each node hosts 16 cores, i.e., $N_{cores} = 16 \times N_{nodes}$, and each core initially hosts $N_{p,1} = 300,000$ of spheres, thus $N_T = 300,000 \times N_{cores} = 4,800,000 \times N_{nodes}$	39

Authors	Max Number of GPUs	Max Number of particles	Speed ratio 1 GPU/1 CPU
Xu <i>et al.</i> [16]	270	10,000,000	nr
Washizawa and Nakahara [13]	1	131,072	6
Shigeto and Sakai [9]	1	1,280,000	0.87 – 3.4
Tsuzuki and Aoki [12]	512	129,000,000	nr
Gan <i>et al.</i> [15]	34	10,000,000	10

Table 1 Summary of recent contributions to DEM computations on GPUs. "nr" stands for "not reported".

Parameter	Value
Particle-Wall	
k_n ($N m^{-1}$)	1×10^6
e_n, μ_n (s^{-1})	$0.62, 3.63 \times 10^3$
μ_c	0.3
k_{ms}	1×10^{-5}
δ_{max} (m), δ_{max}/R	$2.25 \times 10^{-4}, 0.033$
T_C (s)	1.85×10^{-4}
Particle-Particle	
k_n ($N m^{-1}$)	7.2×10^5
e_n, μ_n (s^{-1})	$0.75, 1.87 \times 10^3$
μ_c	0.3
k_{ms}	1×10^{-5}
δ_{max} (m), δ_{max}/R	$1.92 \times 10^{-4}, 0.028$
T_C (s)	1.55×10^{-4}
Δt (s)	1×10^{-5}

Table 2 Contact force model parameters, estimate of contact features at $v_{col} = 4.5 m/s$ and time step magnitude used in the silo discharge simulation.

Repetition	Experiments (<i>s</i>) [27]	Grains3D (<i>s</i>)
1	29.32	29.36
2	29.28	
3	29.2	
Mean discharge time (<i>s</i>)	29.27	29.36

Table 3 Comparison between experimental data of [27] and our simulation results with Grains3D for the discharge time of the silo.

Configuration/core	Total computing time	Computing time per particle and per time step
2000 spheres	59min 56s	$6\mu s$
14000 spheres	12h 24min 27s	$10\mu s$
100000 spheres	104h 41min 12s	$12\mu s$
2000 cubes	3h 58min 32s	$24\mu s$
10000 cubes	30h 30min 16s	$36\mu s$

Table 4 Silo discharge for different systems: reference times of a serial job over 300,000 time steps of $10^{-5}s$.

Parameter	Value
Particle-Box (PB) & Particle-Gate (PG)	
k_n ($N m^{-1}$)	1×10^5
e_n, μ_n (s^{-1})	0.75 , 6.86×10^3
$\mu_{c,PB}, \mu_{c,PG}$	0.5, 0
k_{ms}	0
δ_{max} (m) , δ_{max}/R	7.16×10^{-5} , 0.048
T_C (s)	5.91×10^{-5}
Particle-Particle	
k_n ($N m^{-1}$)	1×10^5
e_n, μ_n (s^{-1})	0.75 , 6.86×10^3
μ_c	0.5
k_{ms}	0
δ_{max} (m) , δ_{max}/R	4.87×10^{-5} , 0.0325
T_C (s)	4.19×10^{-5}
Δt (s)	2.5×10^{-6}

Table 5 Contact force model parameters, estimate of contact features at $v_{col} = 4.2 m/s$ and time step magnitude used in the dam break simulations.

N_{cores}	1	16	32	64	128	256	512
$W(m)$	0.021875	0.35	0.7	1.4	2.8	5.6	11.2
N_T	101,850	1,627,500	3,255,000	6,510,000	13,020,000	26,040,000	52,080,000

Table 6 System size in granular dam break weak scaling tests.

Parameter	Value
Fluid	
ρ_r	2083.333
$\mathcal{R}e_{in}$	79.333
$\mathcal{F}r_{in}$	6.927×10^{-3}
$\Delta\tilde{t}_f$	0.0119
Particle	
e_n	0.9
μ_c	0.1
k_{ms}	0
$\tilde{\delta}_{max}$	0.025
$\Delta\tilde{t}_p$	0.00595

Table 7 Fluid and particles physical and numerical dimensionless parameters.

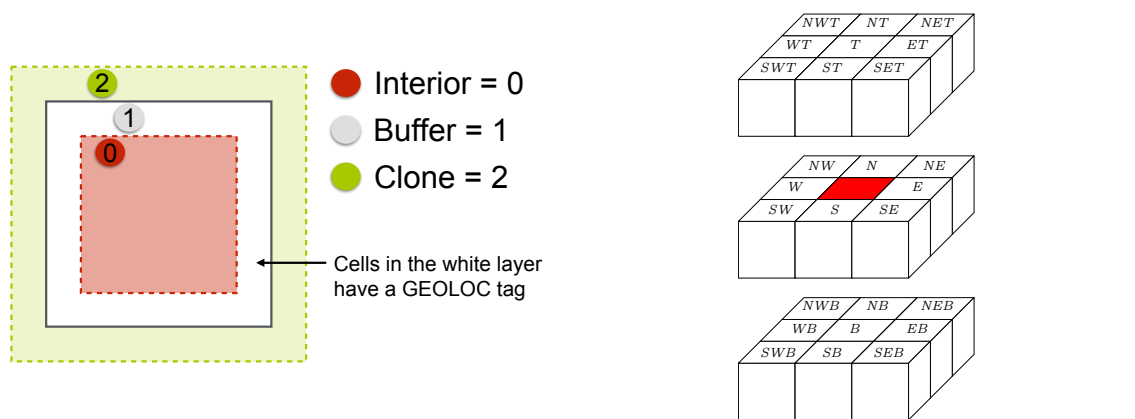
\tilde{L}_x	200	400	800	1600	3200	9600
N_{nodes}	1	2	4	8	16	48
N_{cores}	16	32	64	128	256	768
N_T (million)	4.8	9.6	19.2	38.4	76.8	230.4

Table 8 System size in the fluidized bed weak scaling tests. Each node hosts 16 cores, i.e., $N_{cores} = 16 \times N_{nodes}$, and each core initially hosts $N_{p,1} = 300,000$ of spheres, thus $N_T = 300,000 \times N_{cores} = 4,800,000 \times N_{nodes}$.

List of Figures

1	Tags (status and geolocalisation) of particles in the linked-cell grid.	42
2	2D illustration of inter-process communication for a particle tagged SOUTH.	43
3	2D illustration of inter-process communication for a particle tagged SOUTH_EAST.	44
4	Shape and dimensions of the 3D silo.	45
5	Simulation results of filling and discharge of the 3D silo with Grains3D. Coloured by the particle velocity magnitude.	46
6	Comparison between experimental data of [27] and our simulation results with Grains3D: snapshots of discharge dynamics at different times.	47
7	Multi-silo simulation set-up without overlap between silos (communications with empty messages between sub-domains).	48
8	Multi-silo simulation set-up with all silos merged (connected hoppers) into one big silo (actual communications with non-empty messages between sub-domains). Each hopper corresponds to a sub-domain.	49
9	Weak scaling parallel performance of Grains3D in the multi-silo configurations with (a) disconnected silos and (b) merged silos into one big silo.	50
10	Ratio between parallel overhead and serial tasks for systems made of spherical particles and polyhedral particles.	51
11	Granular dam break set-up. The granular media is made of icosahedral particles.	52
12	3D view of the granular dam break flow for Size 4 case.	53
13	2D view of the granular dam break flow for Size 4 case. (a)-(f) correspond to snapshots every 0.1s.	54
14	Variation of L_∞/L . L_∞ (red) for $\varepsilon \leq 0.1$. Blue dots are particles positions.	55
15	Variation of run-out distance $(L_\infty - L)/L$ with dimensional size of the system for $a \approx 7.3$. Complementary results with lateral walls instead of periodic conditions are plotted in green.	56
16	Final scaled profiles of the deposit as a function of dimensional size of the system for $a \approx 7.3$. All profiles collapse on a single master profile.	57
17	Weak scaling parallel performance of Grains3D in granular dam break computations.	58
18	Fluidized bed computational domain.	59

19	Fluidized bed dynamics in the case $N_{cores} = 64$, $N_T = 19,200,000$: (a)- (d) $U_{in}/U_{mf} = 2$, $\varepsilon = 0.75$ fluid porosity contours colored by pressure magnitude, velocity contours in a $x - z$ cut plane located at $\tilde{y} = \tilde{L}_y$ and pressure contours in a $y - z$ cut plane located at $\tilde{x} = 0$, (e)-(i) porosity field ε in a $x - z$ cut plane located at $\tilde{y} = \tilde{L}_y/2$ over the transition from $U_{in}/U_{mf} = 2$ to $U_{in}/U_{mf} = 3$	60
20	Weak scaling parallel performance of Grains3D relative to a full 16-core node in fluidized bed computations.	61



(a) 2D illustration of the status of a particle depending on their location on the domain, i.e., depending on the tag of the cell it belongs to (b) GEOLOC tag of cells in the buffer zone. N, S, W, E, T and B denote respectively the North, South, West, East, Top and Bottom directions.

Figure 1 Tags (status and geolocalisation) of particles in the linked-cell grid.

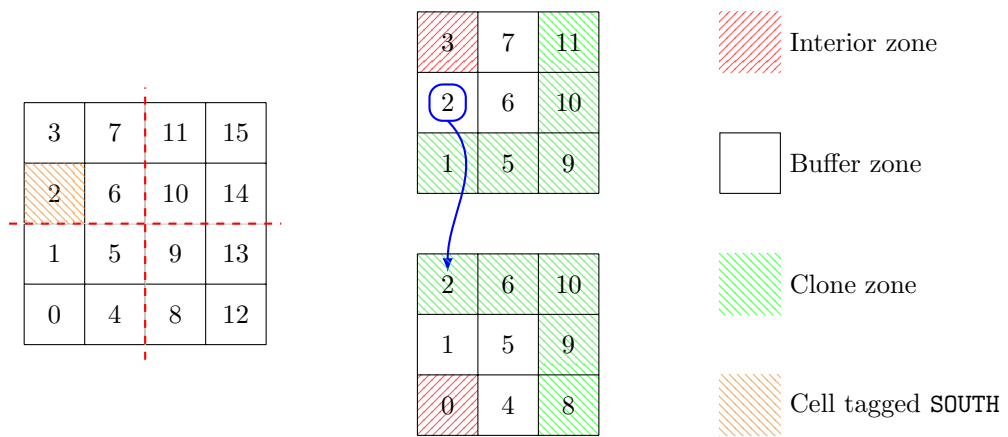


Figure 2 2D illustration of inter-process communication for a particle tagged SOUTH.

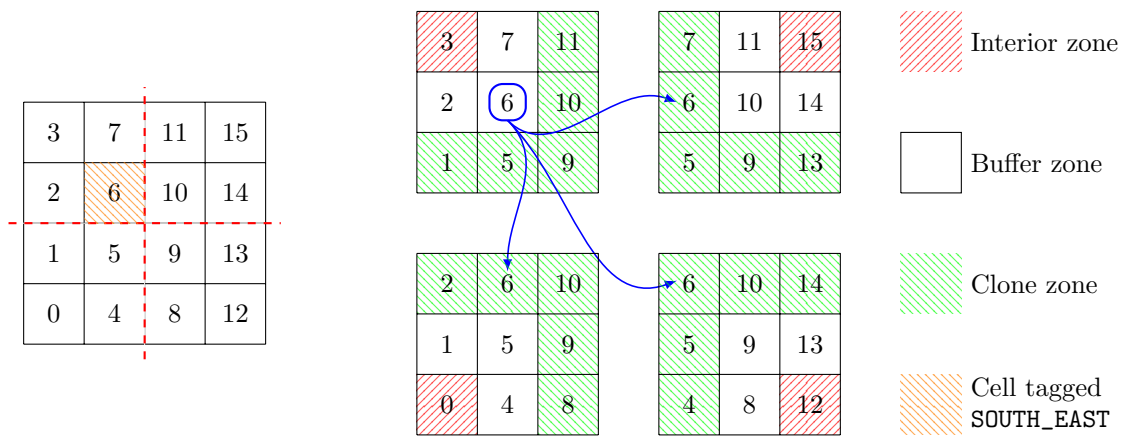
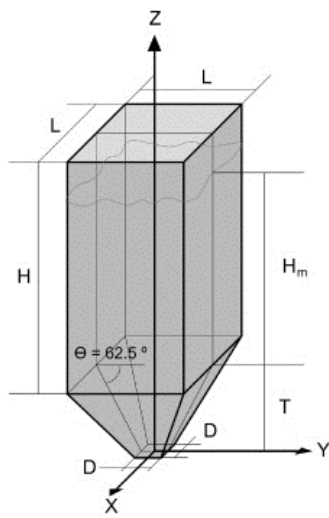
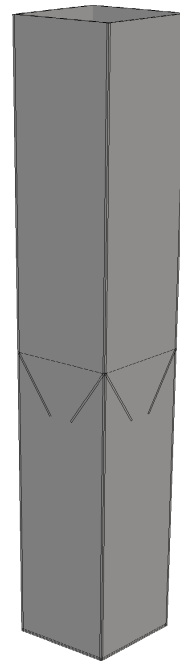


Figure 3 2D illustration of inter-process communication for a particle tagged SOUTH_EAST.



(a) From González-Montellano *et al.* [27]



(b) Equivalent extended silo in our simulations

Figure 4 Shape and dimensions of the 3D silo.

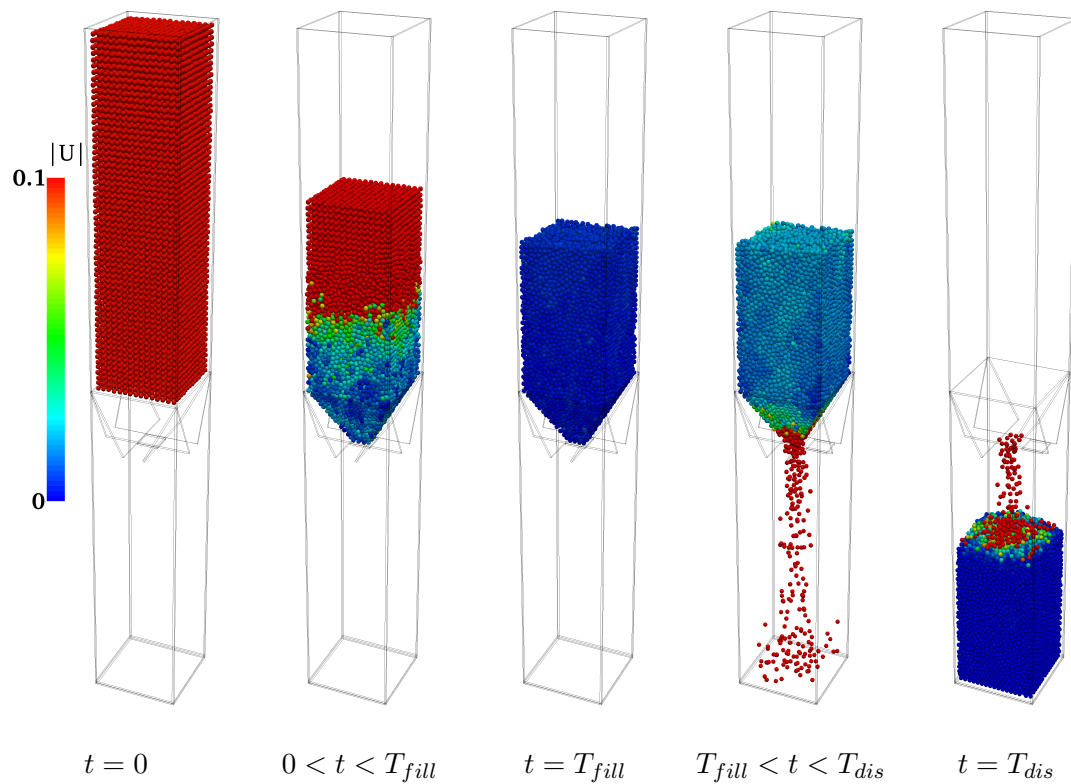


Figure 5 Simulation results of filling and discharge of the 3D silo with Grains3D. Coloured by the particle velocity magnitude.

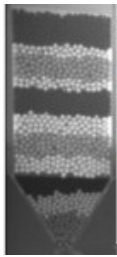
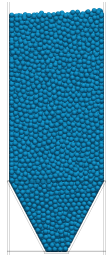
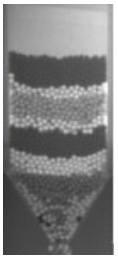
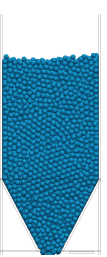

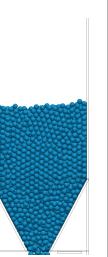

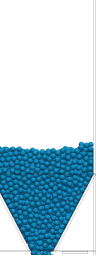
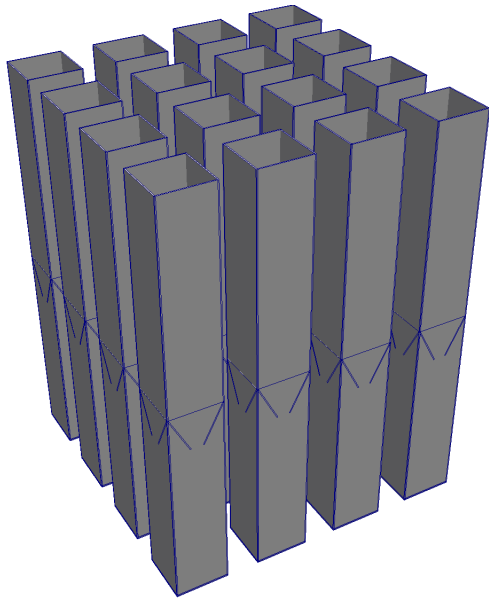
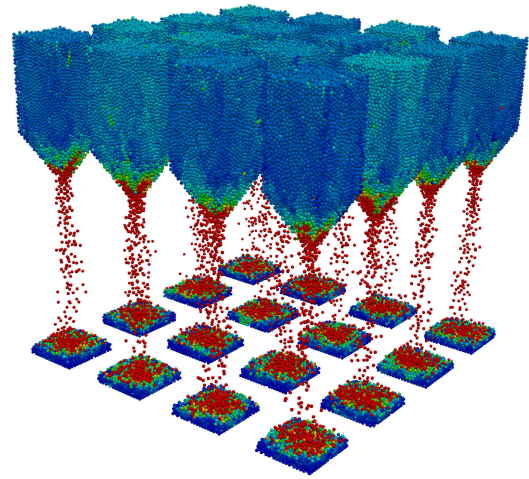
Test	DEM	Test	DEM	Test	DEM	Test	DEM
							
$t = 0T$		$t = 0.25T$		$t = 0.5T$		$t = 0.75T$	

Figure 6 Comparison between experimental data of [27] and our simulation results with Grains3D: snapshots of discharge dynamics at different times.

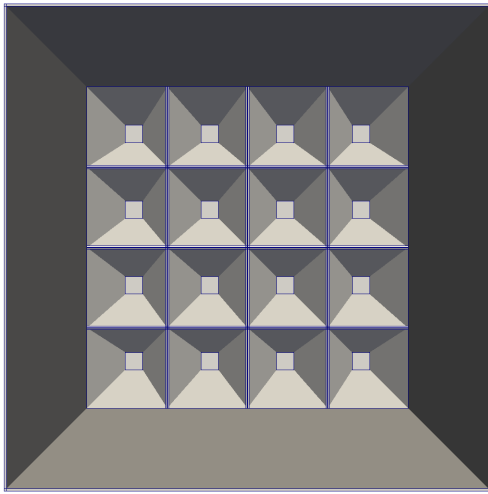


(a) Decomposition of the domain into 16 sub-domains. Each silo is handled by a single core.

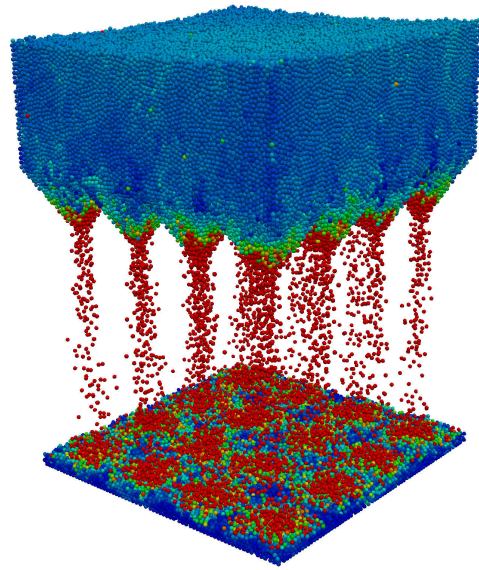


(b) Discharge of 14000 spherical particles per silo from 16 independent silos. Silos are hidden. Coloured by the particle velocity magnitude (blue = min, red = max).

Figure 7 Multi-silo simulation set-up without overlap between silos (communications with empty messages between sub-domains).

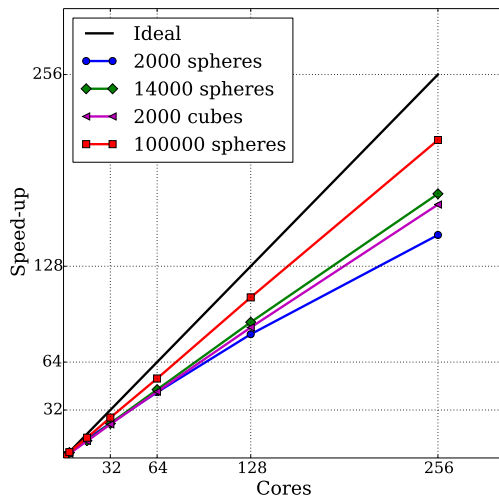


(a) Top view of the simulation domain in which 16 silos are merged into one big silo.

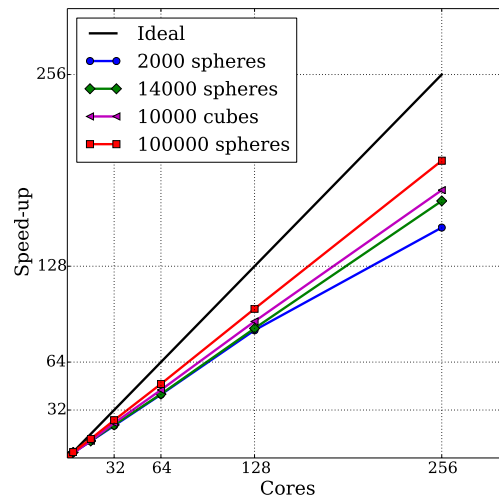


(b) Discharge of 16000 spherical particles per sub-domain from 16 connected hoppers. Hoppers are hidden. Coloured by the particle velocity magnitude (blue = min, red = max).

Figure 8 Multi-silo simulation set-up with all silos merged (connected hoppers) into one big silo (actual communications with non-empty messages between sub-domains). Each hopper corresponds to a sub-domain.



(a) Communication disabled.



(b) Communication enabled.

Figure 9 Weak scaling parallel performance of Grains3D in the multi-silo configurations with (a) disconnected silos and (b) merged silos into one big silo.

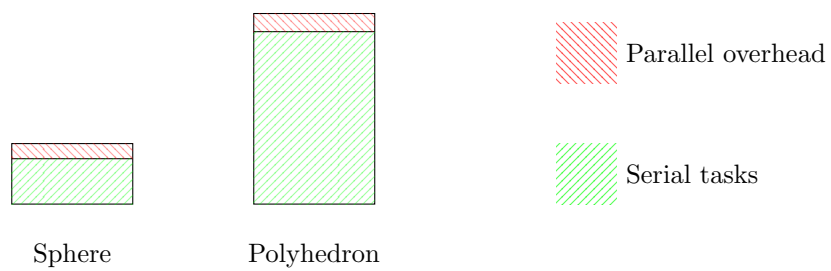


Figure 10 Ratio between parallel overhead and serial tasks for systems made of spherical particles and polyhedral particles.

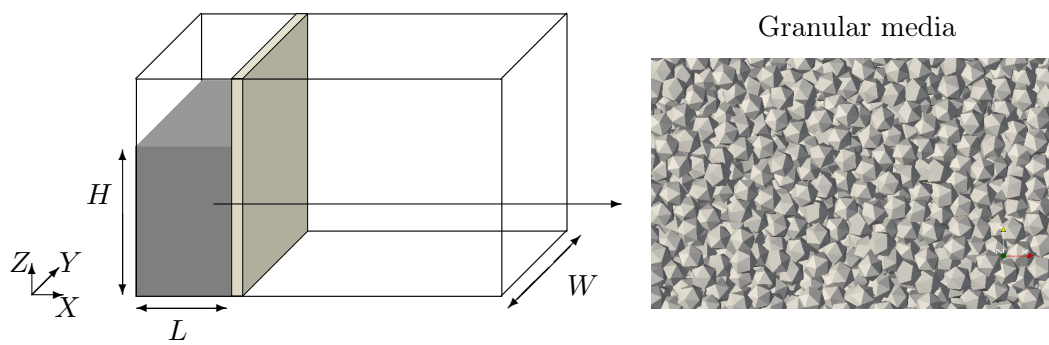


Figure 11 Granular dam break set-up. The granular media is made of icosahedral particles.

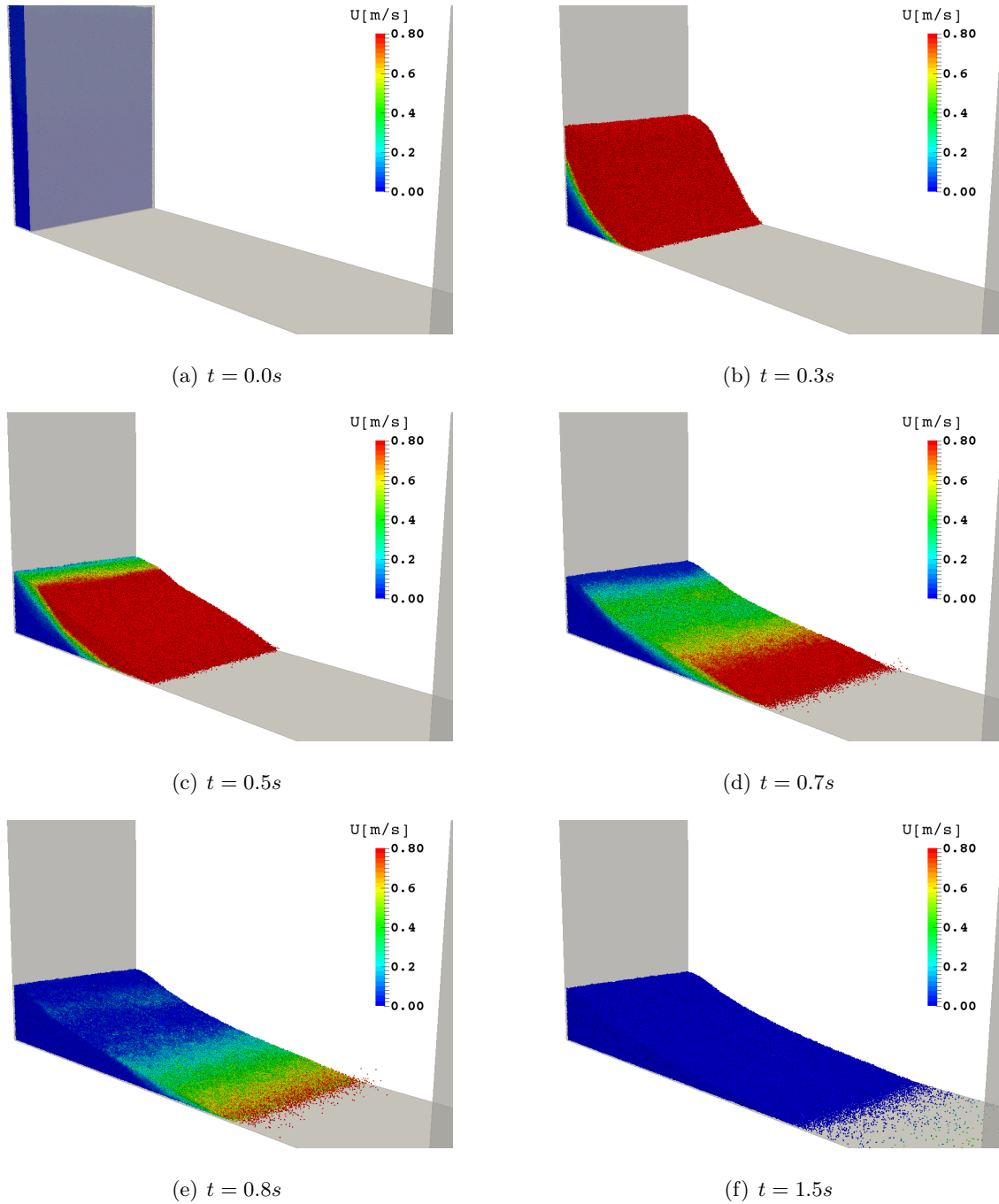


Figure 12 3D view of the granular dam break flow for Size 4 case.

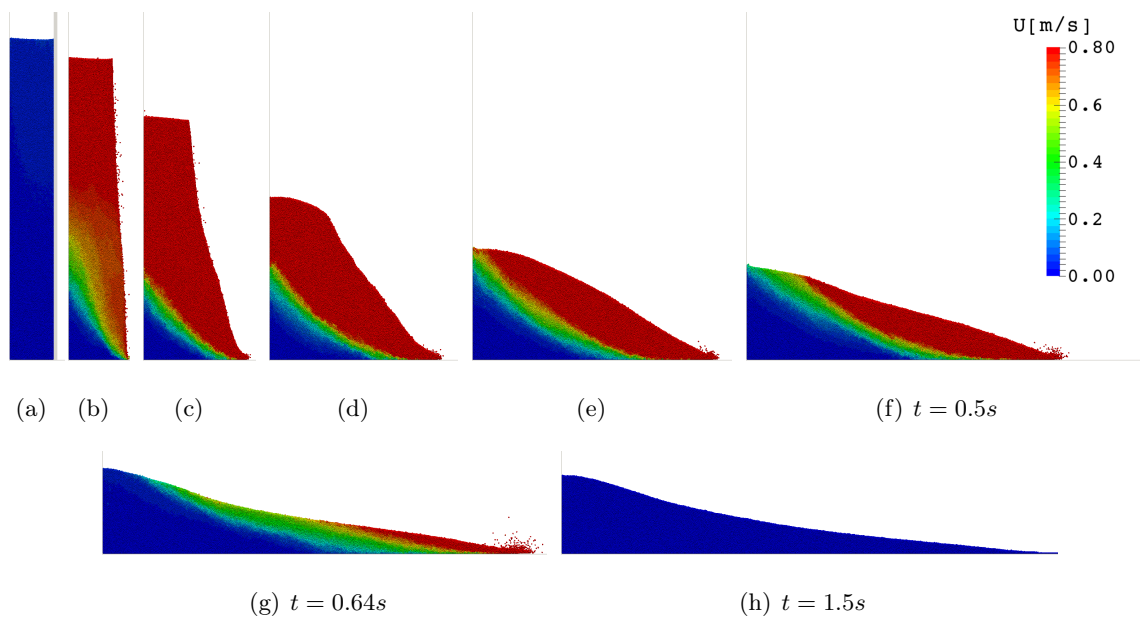


Figure 13 2D view of the granular dam break flow for Size 4 case. (a)-(f) correspond to snapshots every 0.1s.

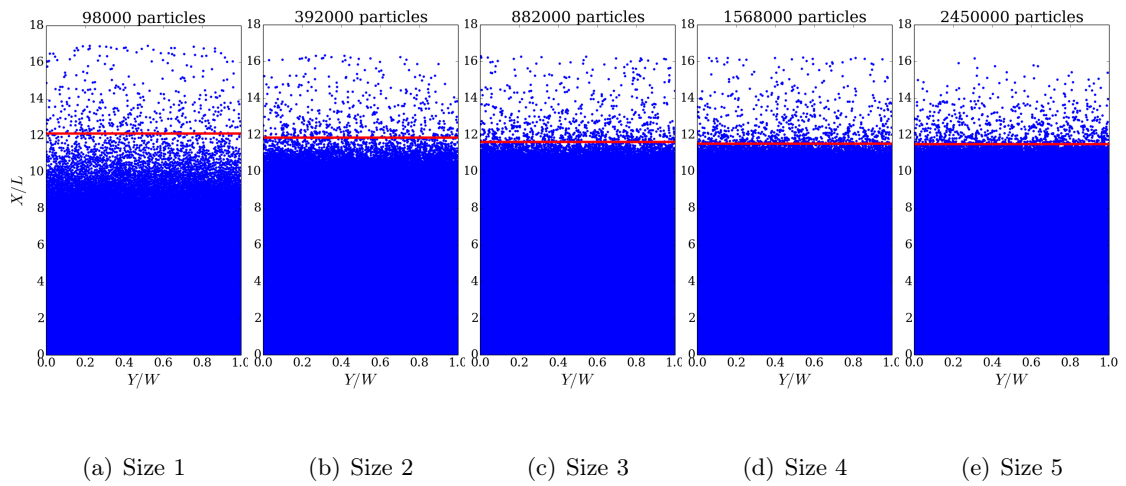


Figure 14 Variation of L_∞/L . L_∞ (red) for $\varepsilon \leq 0.1$. Blue dots are particles positions.

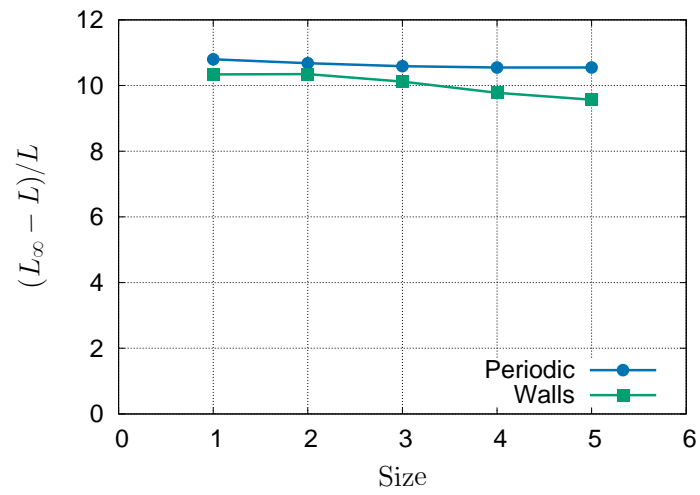


Figure 15 Variation of run-out distance $(L_\infty - L)/L$ with dimensional size of the system for $a \approx 7.3$. Complementary results with lateral walls instead of periodic conditions are plotted in green.

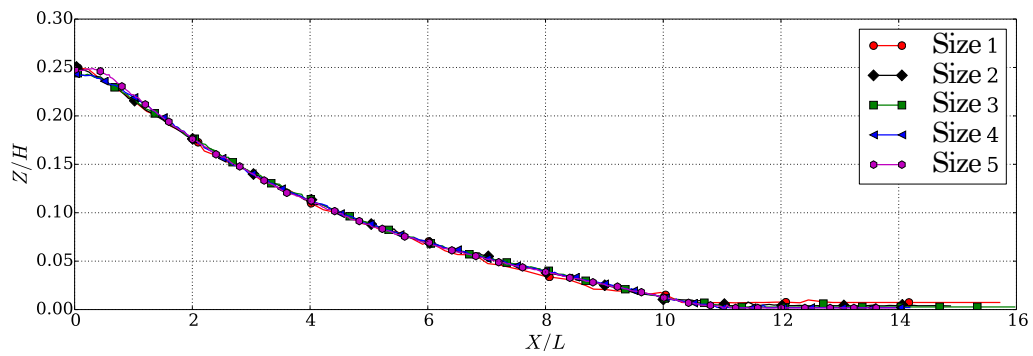


Figure 16 Final scaled profiles of the deposit as a function of dimensional size of the system for $a \approx 7.3$. All profiles collapse on a single master profile.

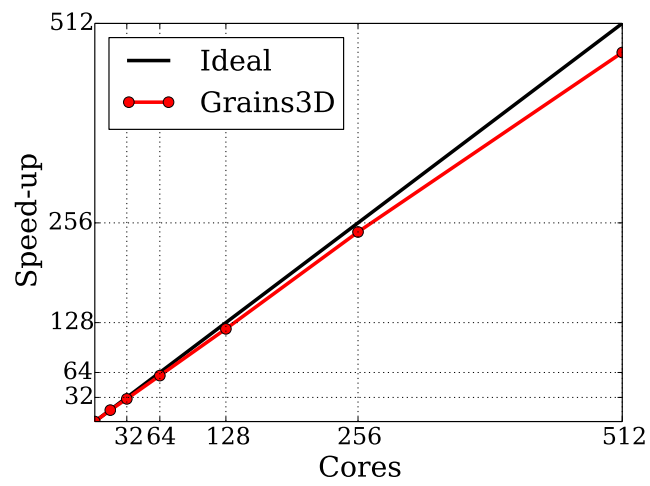


Figure 17 Weak scaling parallel performance of Grains3D in granular dam break computations.

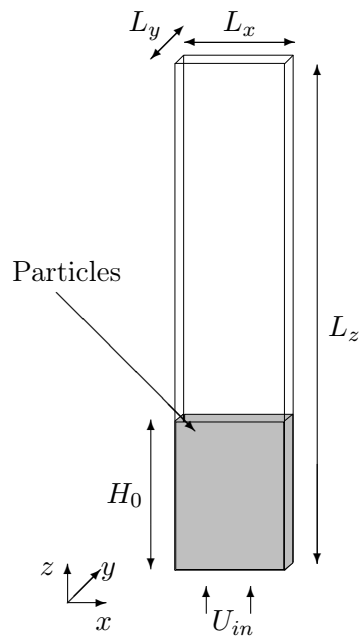


Figure 18 Fluidized bed computational domain.

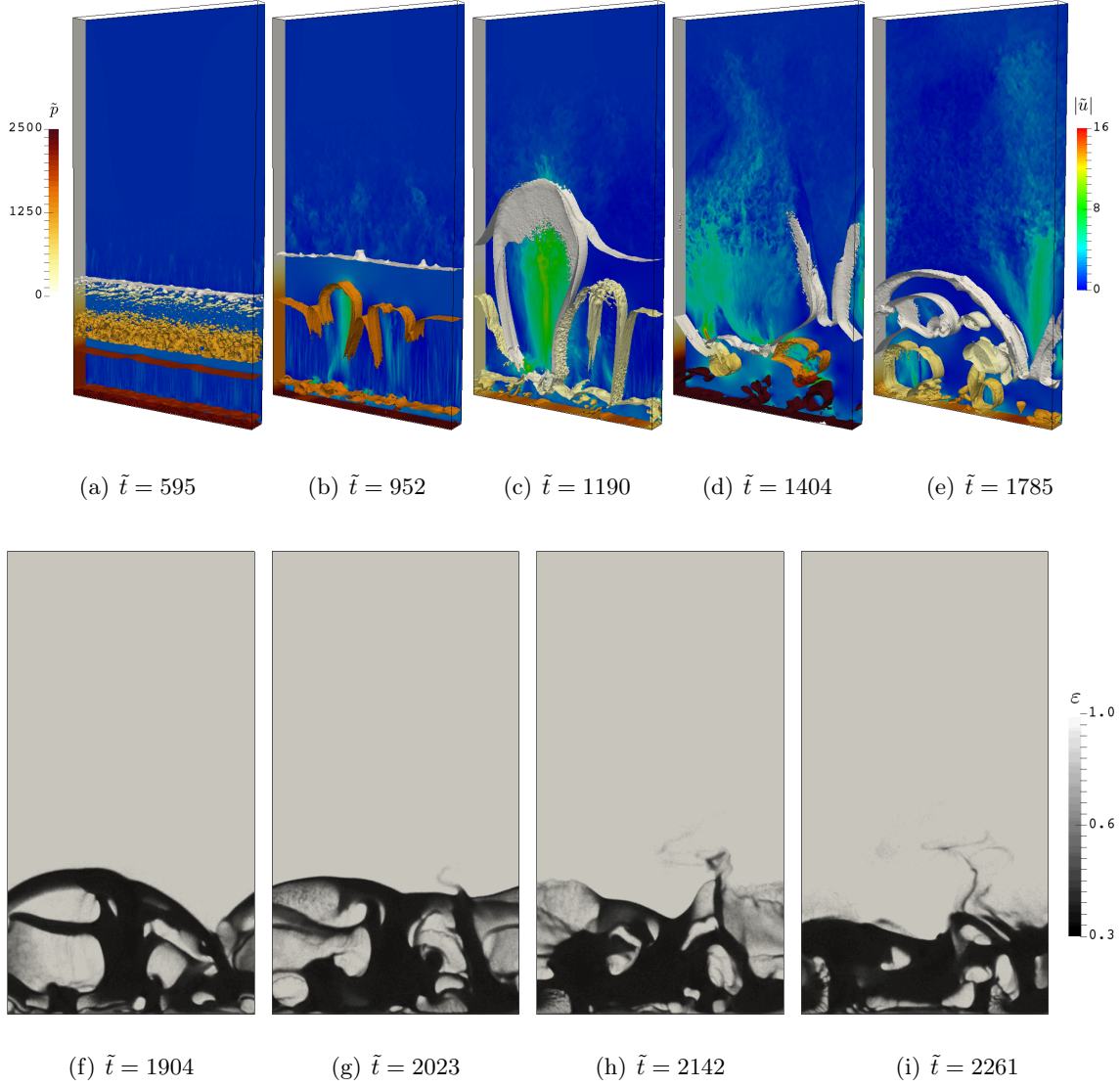


Figure 19 Fluidized bed dynamics in the case $N_{cores} = 64$, $N_T = 19,200,000$: (a)-(d) $U_{in}/U_{mf} = 2$, $\varepsilon = 0.75$ fluid porosity contours colored by pressure magnitude, velocity contours in a $x - z$ cut plane located at $\tilde{y} = \tilde{L}_y$ and pressure contours in a $y - z$ cut plane located at $\tilde{x} = 0$, (e)-(i) porosity field ε in a $x - z$ cut plane located at $\tilde{y} = \tilde{L}_y/2$ over the transition from $U_{in}/U_{mf} = 2$ to $U_{in}/U_{mf} = 3$.

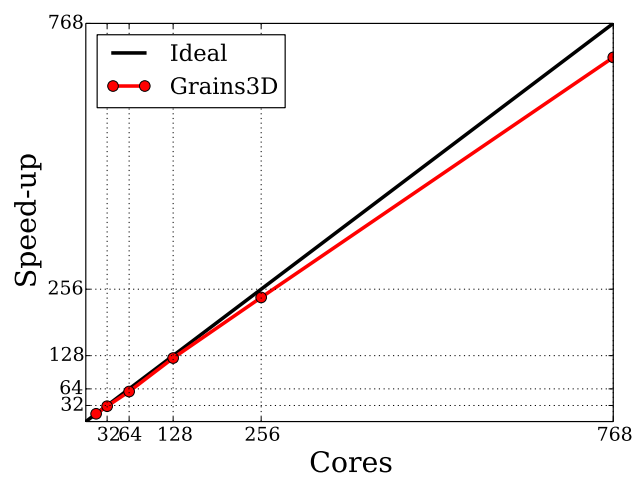


Figure 20 Weak scaling parallel performance of Grains3D relative to a full 16-core node in fluidized bed computations.