HPCLatAm 2013, pp. 89-100 (full paper)
Session: GPU Architecture and Applications
C. Garcia Garino and M. Printista (Eds.)
Mendoza, Argentina, July 29-30, 2013

A GPU implementation for improved granular simulations with LAMMPS.

Emmanuel N. Millán[1,2,3], Christian Ringl[6], Carlos S. Bederián[5], María Fabiana Piccoli[4], Carlos García Garino[2], Herbert M. Urbassek[6], and Eduardo M. Bringa[1,3]

[1] CONICET, Mendoza
[2] ITIC, Universidad Nacional de Cuyo
[3] Instituto de Ciencias Básicas, Universidad Nacional de Cuyo
[4] Universidad Nacional de San Luis
[5] Instituto de Física Enrique Gaviola, CONICET
[6] Fachbereich Physik und Forschungszentrum OPTIMAS, Universität Kaiserslautern, Erwin-Schrödinger-Straße, D-67663 Kaiserslautern, Germany
{emmanueln@gmail.com,cgarino@gmail.com,ebringa@yahoo.com}
http://sites.google.com/site/simafweb/

Abstract. Granular mechanics plays an important role in many branches of science and engineering, from astrophysics applications in planetary and interstellar dust clouds, to processing of industrial mixtures and powders. In this context, a granular simulation model with improved adhesion and friction, is implemented within the open source code LAMMPS (lammps.sandia.gov). The performance of this model is tested in both CPU and GPU (Graphics Processing Unit) clusters, comparing with performance for the LAMMPS implementation of another often used interaction model, the Lennard-Jones potential. Timing shows accelerations of ∼4-10x for GPUs versus CPUs, with good parallel scaling in a hybrid GPU-CPU cluster.

Keywords: GPU, Molecular Dynamics, granular mechanics

1 Introduction

Graphics Processing Units (GPUs) are being used in numerous scientific projects due to their high computing throughput and memory bandwidth. NVIDIA and AMD, the principal graphics card manufacturers, provide their own Software Development Kits (SDK) to use GPUs for general purpose computing: NVIDIA offers CUDA [1] and AMD provides Accelerated Parallel Processing (APP) [2]. Both SDKs can compile and execute code developed with OpenCL [3].

Atomistic simulations, including Molecular Dynamics (MD) simulations are one of the areas where application of GPUs has grown significantly. MD simulations solve Newton's equations of motion for the trajectories of interacting particles [4], and they have been extremely successful to model a variety of systems specially at the nanoscale [5]. These types of simulations solve the equations

of motion for a set of particles interacting through some potentials; for this type of simulations, the position and velocities of particles are known at discrete time steps, and can be used to calculate the strain, stress, and temperature of the system[4]. Therefore, MD is a versatile tool to study thermodynamic and mechanical properties of materials. One of the most used interaction models for particles is the Lennard-Jones potential [4], which is computationally efficient due to its simplicity, and it can capture complex behavior of materials, including the solid-liquid (melt) phase transition. Execution time for MD simulations with short-range interactions (i.e. excluding forces like electrical or gravitational forces), can scale linearly with system size. This is achieved by building neighbor lists for interacting particles and applying domain decomposition to spread the system amongst different processes/tasks. For execution in clusters, the communication overhead, due mostly to exchange of atoms in shared domains, can often be minimized in such a way that a system with N particles running in n processes, will have the same execution time that a system with $2N$ particles in $2n$ processes.

Several programs support the execution of MD simulations using GPUs. For example, in the area of chemistry and biology: Amber [6], NAMD [7] and GROMACS [8]. Moreover, in the area of physics and engineering simulation exist several programs like HOOMD [9], LAMMPS [10] and DL_POLY [11]. In this work we use the LAMMPS software, which is a mature, open source code in continuous development. One of the advantages of the LAMMPS code is its parallel efficiency. LAMMPS supports simulations in clusters using MPI [12, 13], multicore systems with OpenMP [14], and GPU clusters with CUDA or OpenCL (using the Geryon library [15, 16]).

Granular simulations follow trajectories of particles with a finite volume which cannot be penetrated by other particles. Interactions amongst particles are typically short-ranged, and include contact and adhesive forces. There are several computational approaches to carry out granular simulations, and within an engineering or industrial context, granular simulations are generally carried out using the Discrete Element Method (DEM) [17]. Typical MD codes for point particles can also be adapted to run granular simulations efficiently, and LAMMPS includes the possibility of running a few different granular models both in CPUs and GPUs.

Recently, Ringl and Urbassek presented an improved model for granular simulations, and its implementation within the LAMMPS CPU code [18]. In this work, that CPU code is translated to GPU code. The new GPU code is validated through comparison to the CPU version, and then the parallel performance of the code is analyzed in several hardware configurations. Performance is also studied for an often-used interaction model, the Lennard-Jones potential.

2 Granular simulation details

In this section we describe the granular model, how the CPU code was implemented in the GPU version of LAMMPS and finally the hardware infrastructure and software details used to execute the simulations in this work.

2.1 Granular model

The granular theory implemented in this work is broadly discussed in [18] and [19]. Here we briefly describe the added features to the LAMMPS base code for granular simulations, for a description of the base features of LAMMPS granular model see [20], for an introduction to the physics of granular materials see [21]. In this case, granular simulations follow spherical grains, and the forces between two grains are classified as normal or tangential forces. Normal forces are divided in repulsive [22] and adhesive [23] forces. Tangential forces include several friction forces due to gliding [24], rolling [25] and torsional motion [25]. Each grain interacts only with other grains within an extremely short distance, leading to around 3-5 neighbors per grain in these particular simulations.

To test our implementation of the GPU version of the CPU code by Ringl and Urbassek [18], a large prismatic cell containing 70000 silica grains was used. Grains are arranged in a disordered, amorphous solid structure with high porosity. The box was elongated, with its length along the z axis twice its length along x and y axis. The top half of the cell was empty, to mimic empty space above a surface. This is a typical starting configuration to model impact processes, where material ejected from the surface moves towards the top of the box, and this was indeed the configuration used to model impacts by large grains in recent work [26]. For these performance tests, there is no impact, and all grains are given small, random, initial velocities.

2.2 GPU code

The LAMMPS code provides two packages with GPU support: the GPU package [16] and the USER-CUDA package developed mainly by Christian Trott. A detailed discussion of the packages can be found on-line (`http://lammps.sandia.gov/doc/Section_accelerate.html#acc_8`). Only the USER-CUDA package has the ability to execute granular simulations with the Granular-Hooke [27] [20] [28] pair interaction style. The CPU code developed in [18] is an extension of this pair style available for CPUs. For this reason, the GPU code developed here is an extension of the GPU version of the Granular-Hooke pair style available in the USER-CUDA package.

According to the LAMMPS code architecture, a typical pair style for interactions needs one file with the interaction details within the main *src* directory. For the USER-CUDA package an additional cuda kernel is needed within the lib/cuda directory. The new pair style is called granularEasy for both CPU and GPU. The new code follows the optimizations already applied in the original Granular-Hooke GPU pair style, such as using shared memory instead of global

memory to store force calculations. Documentation and code are being sent to the official LAMMPS repository for revision before being accepted as another USER package, once the code is accepted, it will be available to everyone through the official LAMMPS code source. Development and testing of the code was carried out with the LAMMPS version from 11 of March 2013.

To validate and check the correct functioning of the GPU code, we compared all the thermodynamic output against the corresponding output from the CPU code developed in [18]. Both versions should return exactly the same results given the same input configurations. In particular, for the total energy of the system, the maximum deviation observed for several runs, some lasting 50000 time steps, was at most in the 8th significant digit, given a relative error of ∼1e-6 %. The positions of the grains showed differences in the 6th significant digit only for few grains, after 10000 steps, with a relative error of ∼2.5e-4 %.

2.3 Hardware Infrastructure and Software Details

Simulations were executed in multiple hardware configurations.

(a) A workstation with six CPU cores and a single GPU (named Phenom). The Phenom workstation has an AMD Phenom 1055T six cores CPU at 2.8 GHz, with 12 GB of RAM, and 1 Tb 7200 RPM SATA hard drive.The operating system installed in the Phenom workstation is a Linux distribution, Slackware 13.37 64 bits with OpenMPI 1.4.2 and gcc 4.5.3.

(b) A single NVIDIA Tesla C2050 GPU, within the Phenom workstation. This GPU has 448 cores running at 1.15 GHz with 3 GB of ECC memory, it supports single and double precision and has Compute Capability (CC) of 2.0. The Compute Capability (CC) of the GPU indicates which features of CUDA the GPU can execute, for example: CC 1.3+ supports double precision, but previous versions can only execute code in single precision. The GPU is connected into a PCIe 2.0 x16 slot with 8 GB/s of maximum bandwidth.

(c) The two-node ICB-ITIC AMD Opteron cluster, with 64 CPU cores at 2.1GHz, 128 GB of RAM and dual Gigabit Ethernet in each node. The Linux distribution installed in the cluster is Rocks Cluster 5.5, with OpenMPI 1.4.3 and gcc 4.1.2.

(d) The UNC Mendieta heterogeneous (CPU+GPU) cluster, consisting of eight nodes with two Intel Xeon E5-2680 CPUs at 2.7GHz and 32 GB of RAM each, housing twelve NVIDIA Tesla M2090 GPUs with 6 GB GDDR5 memory (177 GBps) and 512 cuda cores at 1.3 GHz, and two NVIDIA Tesla C2075 GPUs with 6 GB GDDR5 memory (150 GBps) and 448 cuda cores at 1.15 GHz. GPUs are connected to PCIe 3.0 x16 slots, although the Tesla M2090 and C2075 are PCIe 2.0 x16 boards. The nodes are connected through a 20 Gbps InfiniBand DDR switch in a star configuration. We note that each node has only one InfiniBand card and, therefore, communications between GPUs from different nodes is shared between the 2 GPUs and the CPUs in each node. The Linux distribution installed in this cluster is CentOS 6.4 with MPICH 3.0.4.

According to [29] the use of InfiniBand in a cluster with LAMMPS provides up to 193% of higher performance than Ethernet using 14 cluster nodes. On some

test, four cluster nodes connected through InfiniBand give similar performance than 14 cluster nodes connected with Ethernet [29].

(a), (b), and (c), are part of the ICB-ITIC cluster hardware at the Universidad Nacional de Cuyo, Mendoza, Argentina, while (d) is located at the Universidad Nacional de Córdoba, Córdoba, Argentina,

The LAMMPS code used in all the hardware configurations was the same, version dated 11-Mar-2013 with CUDA 5.0. The LAMMPS code was compiled with -O2 optimizations and the USER-CUDA package with CC 2.0 and with double precision. We note that Tesla GPUs can bypass the CPU and do GPUDirect transfers to other GPUs either locally or through Infiniband to speed up MPI communication between processes, but this requires GPUDirect-aware code and a supporting MPI implementation (e.g. MVAPICH2). This is not the case for LAMMPS.

3 Benchmarks

We tested the CPU and GPU code with a simple simulation, a box with 70000 grains with different initial velocities. We ran the simulations for 1000 steps, printing the thermodynamic information every 100 steps. To simulate a greater number of grains we use the replicate command of LAMMPS to expand the box N times in each direction, for example, "replicate 2 2 2" will create 8 times more grains than the one presents in the initial box (see LAMMPS documentation, lammps.sandia.gov).

Figure 1 displays the simulations that were executed in the Phenom workstation [hardware configuration (a)], for different number of grains, from 7e4 to 4.48e6 grains. One curve for each number of CPU cores (1, 2, 4, and 6) and one curve for the NVIDIA Tesla c2050 GPU housed within the Phenom workstation [hardware configuration (b)]. The behavior of the wall clock time is roughly linear, as expected from typical MD codes. Two more curves are shown in the figure 1, the 7.5e4 curve fits the results obtained in [18] with a single processor AMD Opteron 275 running at 2.2 GHz. The 0.6e4 curve fits the results obtained with the Tesla c2050 GPU.

The largest number of grains which can be processed using the 3 GB memory of this GPU is approximately 4.5e6. The CPUs in the Phenom workstation have more RAM memory (12 GB) available, and they can process up to 24e6 grains. This is larger than the 18e6 grains expected from the GPU limit due to different memory management strategies. Table 1 contains the wallclock time for the Phenom workstation and the Tesla c2050 GPU for different system sizes. The average speedup obtained for the GPU vs one CPU core is 7.01 x. Comparing the GPU vs six CPU cores the average speedup is 2.95 x. These values of speed-up are similar to values reported for MD simulations using LAMMPS [16]. We note that running multiple cases for the same hardware configuration, for instance changing the random seed for initial velocities, changes the timing only within a few percent of the values listed in Table 1.

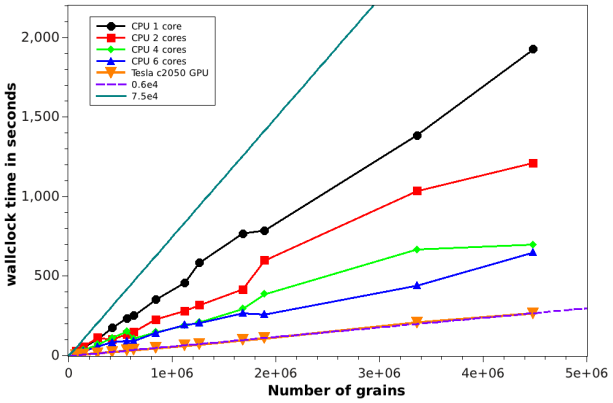

Fig. 1. Granular simulation with the GranularEasy pair style, 13 different amounts of grains, and 1000 steps, for different hardware configurations. A fit to the CPU results from [18] and a fit to the new GPU results are also included.

It is important to know not only the performance of the code for different system sizes, but the performance for a fixed size and different number of processes. Figure 2 shows the wall clock time for the granular simulations with 4.48e6 grains, for 1000 steps, and for several number of parallel processes. The benchmarks are performed for all hardware configurations described above (a-d). For an ideal parallelization strategy, timing should be proportional to 1/n, for n processes. This scaling would eventually decrease its slope due to communication amongst processes. This 1/n scaling is what is observed for small number of processes in Fig. 2, together with the levelling-off due to communications. It is clear that the Tesla M2090 GPU execution gives the best results. Executing the simulation in one M2090 GPU takes 184.48 s, two GPU processes in one node take 91.73 s , and four GPU processes in two nodes take 45.85 s. This last configuration gives the best result, because increasing the number of GPUs to 6 or 8 also significantly increases communication between them. The results obtained in CPU-only executions in the ICB-ITIC and Mendieta cluster did not achieve as good a performance as GPU executions. The best CPU-only time was obtained in the ICB-ITIC cluster using 16 CPU cores at 192.89 s.

The simulation box being elongated and half-empty leads to an imbalance in the assignation of processing work to each GPU, and deteriorates communication time, leading to a large increase in execution time for six processes in the M2090 GPU of the Mendieta cluster. The simulation time could be significantly reduced by eliminating the empty space in the system box and obtaining a roughly cubic box, removing the imbalance and at the same time minimizing communication. For this reason, a new simulation was executed using such homogeneous box, and the time for this new case is also shown in fig. 2 for six parallel processes.

Table 1. Granular simulations for different number of grains. Wall-clock time in seconds, with GPU vs CPU speedups also provided.

Number of grains	CPU 1 core	CPU 6 cores	GPU	Speedup GPU vs 1 core	Speedup GPU vs 6 cores
70000	27.631 s	15.3 s	4.71 s	5.87	3.25
140000	53.88 s	29.91 s	8.74 s	6.16	3.42
280000	106.59 s	59.52 s	16.79 s	6.35	3.54
420000	175.74 s	86.7 s	24.77 s	7.09	3.50
560000	236.11 s	93.85 s	33.01 s	7.15	2.84
630000	254.24 s	94.42 s	36.79 s	6.91	2.57
840000	354.46 s	143.3 s	48.93 s	7.24	2.93
1120000	460.92 s	195.31 s	64.95 s	7.10	3.01
1260000	585.72 s	207.96 s	72.91 s	8.03	2.85
1680000	768.01 s	268.8 s	98.92 s	7.76	2.72
1890000	788.59 s	261.11 s	111.46 s	7.08	2.34
3360000	1386.8 s	443.12 s	209.35 s	6.62	2.12
4480000	1929.38 s	650.57 s	266.71 s	7.23	2.44
			AVG speedup	7.01	2.95

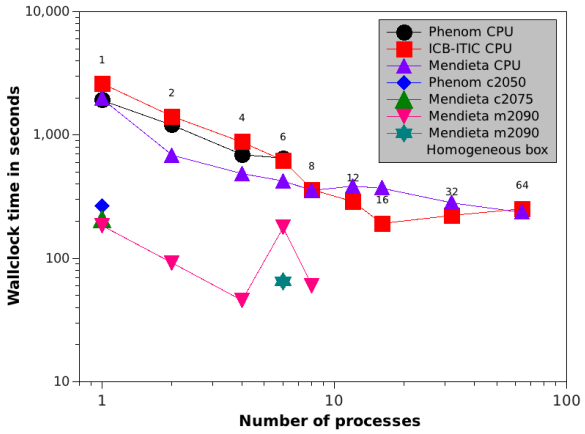

Fig. 2. Granular simulation with the GranularEasy pair style, with 4.48e6 grains and 1000 steps, for different number of parallel processes, in multiple hardware configurations, as described in text.

Figure 3 shows the total execution time for several granular simulations, divided into several contributions: communication time, calculation of pair forces, building of neighbor list, and time for other calculations (like output time). This figure clearly shows the increase of communication time with increasing number of processes, 4 GPU processes are executed in 2 cluster nodes (2 GPUs in each

node), 6 GPU processes in 3 nodes and finally 8 GPU processes are executed in 4 nodes.

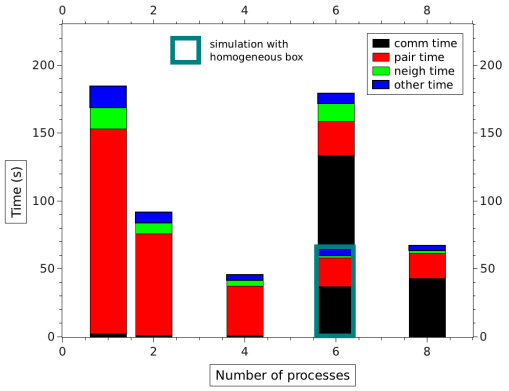

Fig. 3. GPU Granular simulation, 1000 steps with 4.48e6 grains, for different number of processes. For six processes, two simulations are shown, one has an elongated, half-empty heterogeneous box, and the second has a cubic homogeneous box filled with grains.

Because the granular simulation is set for a heterogeneous box (elongated and half-empty), the LAMMPS domain decomposition will assign GPUs to empty regions, decreasing performance. This is the case for 6 processes using the default domain decomposition. To improve this situation, we execute the same simulation using the homogeneous box also used for Fig. 2, shown with a thick border line. Communication time decreases significantly, as expected.

3.1 Benchmarks for Lennard-Jones interaction

Lennard-Jones (LJ) systems have also been used to model grains [30] [31]. Here the melt example for L-J particles from the LAMMPS distribution is used to study performance in the hardware configurations used for granular benchmarks. In this example, the starting configuration is a cubic box, with particles in perfect crystal positions using a cubic unit cell, as shown in Figure 4(a). A large velocity is assigned to all atoms, leading to melting of the initial solid. This results in the liquid structure seen in Fig. 4(b). Note that a liquid and an amorphous solid of the same substance, can generally only be differentiated by calculation of particle diffusivity, which would be much larger in the liquid. The amorphous structure used in the granular simulations can be seen in Fig. 4(c).

In Figure 5 one can observe the melt simulation running in the Mendieta cluster, in GPUs and CPUs, for two different values of the interaction cut-off values. Simulations were executed for 1000 timesteps with 256e3 atoms. The interaction cut-off gives the radius of the sphere, centered around each particle,

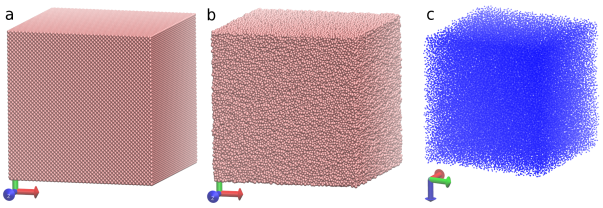

Fig. 4. Different Molecular Dynamics snapshots. (a) Melt simulation for 256e3 Lennard-Jones particles, at timestep 0. (b) Same simulation shown in (a), at timestep 10000, showing an amorphous liquid resulting from the destruction of the initial crystalline order. (c) Granular simulation with the GranularEasy pair style with 7e4 grains at timestep 10000, showing an amorphous solid structure.

within which other particles would be considered to interact with that center particle. We are considering the L-J interactions as short-range in the sense that they decrease rapidly with distance. Neglecting interactions beyond certain distance (the radial cut-off) might be computationally efficient without compromising the accuracy of the results. For the simulations here, a cut-off of 2.5 (in LJ length units) would be fairly standard, and would include $\sim$ 75 neighbors. A cut-off of 5.0 would be rarely used, and would include $\sim$ 526 neighbors. LJ simulations provide a better opportunity to study communication effects, given that the number of neighbors is much larger than in granular simulations and it can be easily changed with large variations.

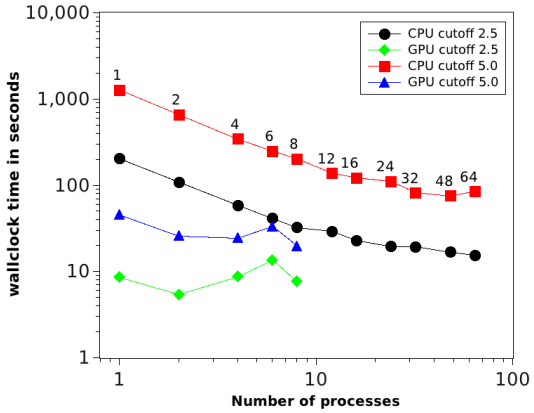

Fig. 5. Lennard-Jones melt simulation, for 1000 steps and 256e3 atoms, for different number of processes in the Mendieta cluster with CPUs and GPUs.

As for the granular simulations, the GPUs clearly outperforms the CPUs, even for 64 cores. The speed-up can reach nearly factors larger than 20. The

timing for the smaller cut-off is smaller, as expected due to communication increase for larger neighbor lists.

Figure 6 shows the total execution time for several LJ simulations divided into several contributions, as in Fig 3. The time values also clearly show increase of communication time with increasing number of processes, and a tremendous difference in communication and force calculation time with the longer cut-off, as expected.

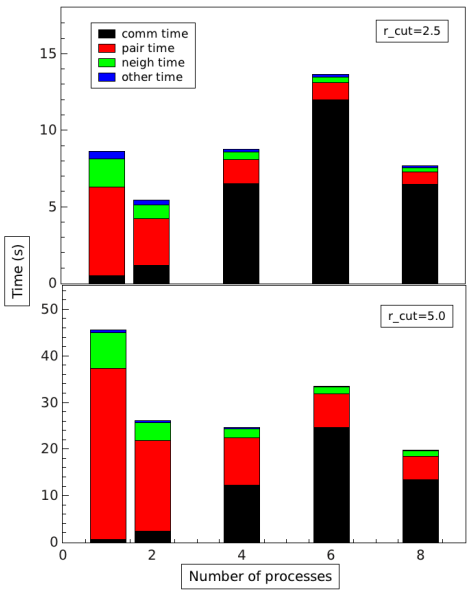

Fig. 6. GPU Melt simulation, 1000 steps with 256e3 atoms, for different number of processes.

4 Summary and Conclusions

Granular simulations are useful to understand the behavior of various systems of interest for both basic science and technological applications. In this work an improved granular model recently presented for CPUs [18], was implemented to run in GPUs within the USER-CUDA package of the LAMMPS Molecular Dynamics (MD) software distribution [10].

The granular model was validated by detailed comparison to results from the CPU version. Speedups of $\sim$7 x (compared to one CPU core) and $\sim$3 x (compared to six CPU cores) were obtained, and compared well with other speedups reported for MD simulations [16]. Benchmarks performed in different hardware configurations, including one CPU cluster and a hybrid CPU-GPU cluster,

showed that the GPU code is always faster that the CPU version. To compare with typical MD simulations, we also report performance of the Lennard-Jones (LJ) interaction potential in the same hardware configurations.

We also tested the difference between granular simulations for a cubic homogeneous box, versus an elongated, half-filled heterogeneous box, containing the same grains. The large decrease in execution time for the homogeneous box points to the relevance of a careful simulation design, taking into account both domain decomposition possibilities and communication expenses due to particular domain choices.

For a single GPU, a linear scaling is observed with increasing system size. An inverse linear scaling is also observed for a fixed system size (4.48e6 grains), increasing the number of processes, using up to 4 GPUs. For more processes the timing does not decrease further due to communication increase. Communications are typically related to the number of neighbor particles/grains which has to be considered for interactions. In granular simulations there are very few neighbors, while in LJ simulations there are 70-500 neighbors, depending on the chosen interaction cut-off.

For granular simulations, a Tesla c2050 GPU is similar in performance to 16 CPU cores in the ICB-ITIC cluster. The Mendieta cluster using Tesla M2090 GPUs give the best performance in our tests, since using 4 GPUs in two cluster nodes gives a speedup of $\sim$4.2 x against the best CPU result (ICB-ITIC cluster with 16 CPU cores). These figures point out to the good performance of atomistic codes in GPUs, with future improvements expected from the new Kepler GPU architecture, as already reported for the code HOOMD [32].

5 Acknowledgements

E. Millán acknowledges support from a CONICET doctoral scholarship. E.M. Bringa thanks support from a SeCTyP2011-2013 grant, PICT2009-0092 and PICT2009-1325.

References

1. CUDA from NVIDIA: `http://www.nvidia.com/cuda`
2. AMD Accelerated Parallel Processing (APP) `http://developer.amd.com`.
3. OpenCL, The open standard for parallel programming of heterogeneous systems: `http://www.khronos.org/opencl/`
4. Allen, M.P., Tildesley, D.J.: Computer simulations of liquids. Oxford Science Publications, Oxford, (1987)
5. Anders, C., Bringa, E.M., Ziegenhain, G., Graham, G.A., Hansen, J.F., Park, N., Teslich, N.E., Urbassek, H.M.: Why Nanoprojectiles Work Differently than Macroimpactors: The Role of Plastic Flow. Phys. Rev. Lett. **108** (2012) 027601
6. AMBER: `http://ambermd.org/`
7. NAMD: `http://www.ks.uiuc.edu/Research/namd`
8. GROMACS: `http://www.gromacs.org/`

9. Anderson, J.A., Lorenz, C.D., Travesset, A.: General Purpose Molecular Dynamics Simulations Fully Implemented on Graphics Processing Units. Journal of Computational Physics **227** 5342–5359
10. Kelchner, C.L, Plimpton, S.J., Hamilton, J.C.: Dislocation nucleation and defect structure during surface indentation. Phys Rev B, **58** (1998) 11085-11088
11. DL POLY Molecular Simulation Package: `http://www.cse.scitech.ac.uk/ccg/software/DLPOLY/`
12. OpenMPI: `http://www.open-mpi.org`
13. MPICH2: `http://www.mcs.anl.gov/research/projects/mpich2/`
14. The OpenMP API specification for parallel programming: `http://openmp.org/`
15. Geryon: `http://users.nccs.gov/~wb8/geryon/index.htm`
16. Brown, W.M., Wang, P., Plimpton, S.J., Tharrington, A.N.: Implementing molecular dynamics on hybrid high performance computers short range forces. Computer Physics Communications **182** (2011) 898–911
17. Carmona, H.A., Wittel, F. K., Kun, F., Herrmann, H. J.: Fragmentation processes in impact of spheres. Phys. Rev. E **77** (2008) 051302
18. Ringl, C., Urbassek, H.M.: A LAMMPS implementation of granular mechanics: Inclusion of adhesive and microscopic friction forces. Computer Physics Communications **183** (2012) 986–992
19. Ringl, C., Bringa, E.M., Bertoldi, D.S., Urbassek, H.M.: Collisions of porous clusters: a granular-mechanics study of compaction and fragmentation. Astrophysical Journal **752** (2012) 151
20. Silbert, L.E., Ertas, D., Grest, G.S., Halsey, T.C., Levine, D., Plimpton, S.J.: Granular flow down an inclined plane: Bagnold scaling and rheology. Physical Review E **64** (2001) 051302
21. Duran, J.: Sands, powders, and grains: an introduction to the physics of granular materials (Springer Verlag, 2000).
22. Pschel, T., Schwager, T: Computational Granular Dynamics: Models and Algorithms. (Berlin: Springer). (2005)
23. Blum, J.: Dust agglomeration. Adv. Phys., 55. (2006) 881–947
24. Burnham, N. and Kulik, A. A: Handbook of Micro/Nano Tribology, ed. B. Bhushan (2nd ed.; Boca Raton, FL: CRC Press) (1999) 247 .
25. Dominik, C. and Tielens, A. G. G. M.: The Physics of Dust Coagulation and the Structure of Dust Aggregates in Space. ApJ, **480** (1997) 647
26. Ringl, C., Bringa, E.M., Urbassek, H.M.: Impact on porous targets: Penetration, crater formation, target compaction, and ejection. Phys. Rev. E **86**, (2012) 061313.
27. Brilliantov, N.V., Spahn, F., Hertzsch, J.-M., Pschel, T.: Model for collisions in granular gases. Physical review E **53** (1996) 5382.
28. Zhang, H.P., Makse, H.A.: Jamming transition in emulsions and granular materials. Physical Review E **72** (2005) 011301
29. LAMMPS Performance Benchmark and Profiling: `http://www.hpcadvisorycouncil.com/pdf/LAMMPS_Analysis_and_Profiling_Intel.pdf`
30. Kalweit, M., Drikakis, D.: Collision dynamics of nanoscale Lennard-Jones clusters. Phys. Rev. B **74** (2006) 235415
31. Seung-Chai Jung, Jong-Geun Bang, Woong-sup Yoon: Applicability of the macroscale elastic contact theories for the prediction of nano-scaled particle collision with a rigid flat surface under non-adhesive and weakly-adhesive conditions, Journal of Aerosol Science **50**, (2012) 26–37
32. HOOMD-BLUE Benchmarks: `http://codeblue.umich.edu/hoomd-blue/benchmarks.html`