Parallel execution of a parameter sweep for molecular dynamics simulations in a mixed GPU/CPU environment

Emmanuel N. Millán, Carlos García Garino, and Eduardo M. Bringa

Outline

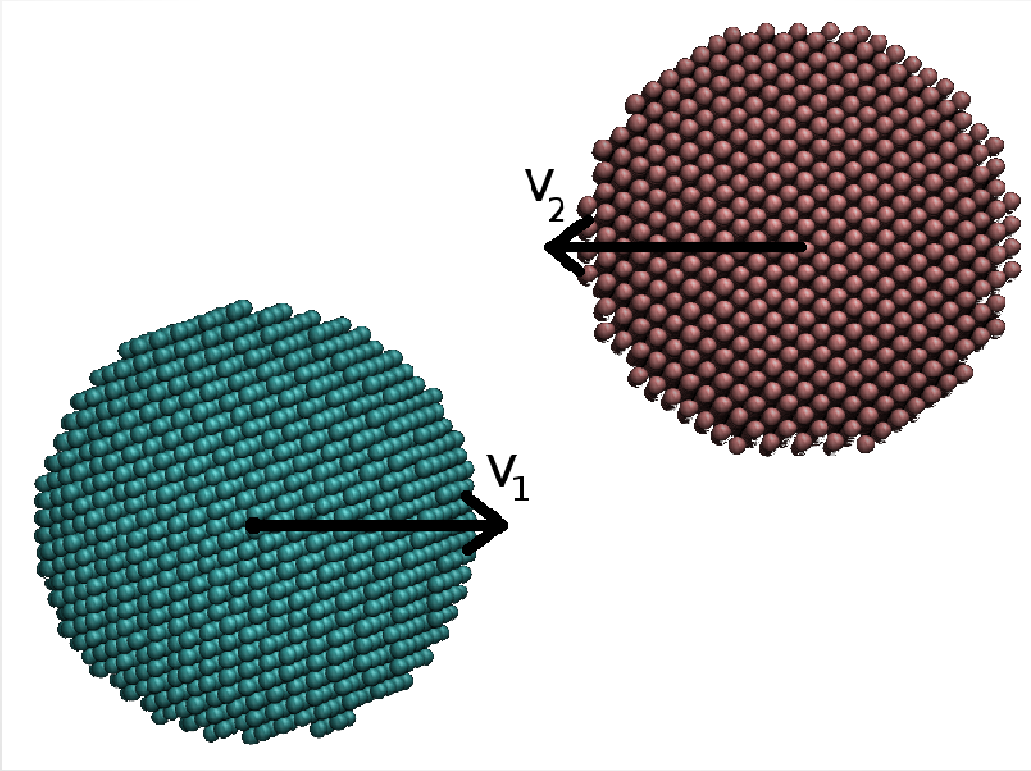
- Motivation & goals
- Parametric study of Collision of nanograins
- HW & SW (LAMMPS) details
- Experiments
- Results
- Conclusions

Motivation & Goals

- To perform parametric studies of nanograin collisions
- To improve makespan
- GPU & LAMMPS are nice candidates
- Parallel mixed CPU / GPU executions are considered.

Physical scenario: Collision of Nanograins

- The goal is to simulate the collision of spherical nanograins with f.c.c. (face centered cubic) structure.



Physical scenario: Collision of Nanograins

- Irreversible plastic deformation will occur for collision velocities above certain threshold.
- Accurate determinations of that threshold are not easily carried out, and grain size will affect strain rate and plastic yielding.
- We try to quantify how much plastic deformation occurs due to the collision.
- In f.c.c. solids, large stress leads to partial dislocations (PDis) travelling rapidly through the material.
- The partials run across the nanograins and are absorbed at the surfaces, leaving behind only stacking faults (SFs) which change the mechanical properties of the material.

Parametric study of Collision of Nanograins

- In order to perform the parameter sweep study:
  - Three different velocities are considered for the grains.
  - Ten diferent lattice orientations are taken into account.
  - Five different impact parameter are modelled. Impact parameter measure the distance between the grain centers along the direction perpendicular to their relative velocity.
- Then a total of 150 different simulations have to be performed. ➔ 310000 seg – 85 horas

Molecular Dynamics: LAMMPS

LAMMPS characteristics:

- Mature.
- Open source code.
- In active development and with a fairly large user community.
- LAMMPS can work with MPI, OpenMP, CUDA and recently with OpenCL.

LAMMPS & Distributed Computing

LAMMPS supports several techniques to distribute the workload on the available hardware (a single workstation in this case):

- MPI library for different cores of a single workstation. The same technique is applied for nodes of a cluster.
- OpenMP for multicore workstations. *
- The complete workload can be executed in the GPU.
- The workload, via a pseudo domain decomposition can be assigned to CPU cores and the GPUs availables at a workstation. Dynamic or static load balancing can be considered.

* For more info on OpenMP support in LAMMPS see Axel Kohlmeyer web site: http://goo.gl/LQqAm .

HW & SW Infrastructure

- CPU: AMD Phenom 1055T six cores workstation running at 2.8 GHz, with 12 GB of RAM, 1 Tb 7200 RPM SATA hard drive.
- GPU: NVIDIA Tesla C2050 with 448 cores running at 1.15 GHz and 3 GB of ECC memory, it supports single/double precision.

- Slackware 13.37 64 bit
- Ruby 1.9.3p194
- LAMMPS version dated 10-Feb-2012, compiled with OpenMPI 1.4.2 and gcc 4.5.3 with -O2 optimizations, the GPU package with CUDA 3.2 and compute capability 2.0.

Experiments

The goal is to reduce the makespan of multiples jobs executing parallel processes both in the CPU and GPU.

An ad-hoc strategy is built-up in order to assign jobs to the CPU and GPU. This is a job scheduling problem, which is outside the scope of this work. Further investigation can improve this aspect.

Experiments

Different parallel modes of LAMMPS are considered:

- Homogeneous studies

  To process the parametric study on multicore CPU workstation. OpenMPI is used.

  To process a the parametric study on the GPU.

- Hybrid studies

  A Ruby script has been buit up in order to assign the workload both to CPU and GPU according a defined strategy. MPI is considered for this case as well.

Single Simulation

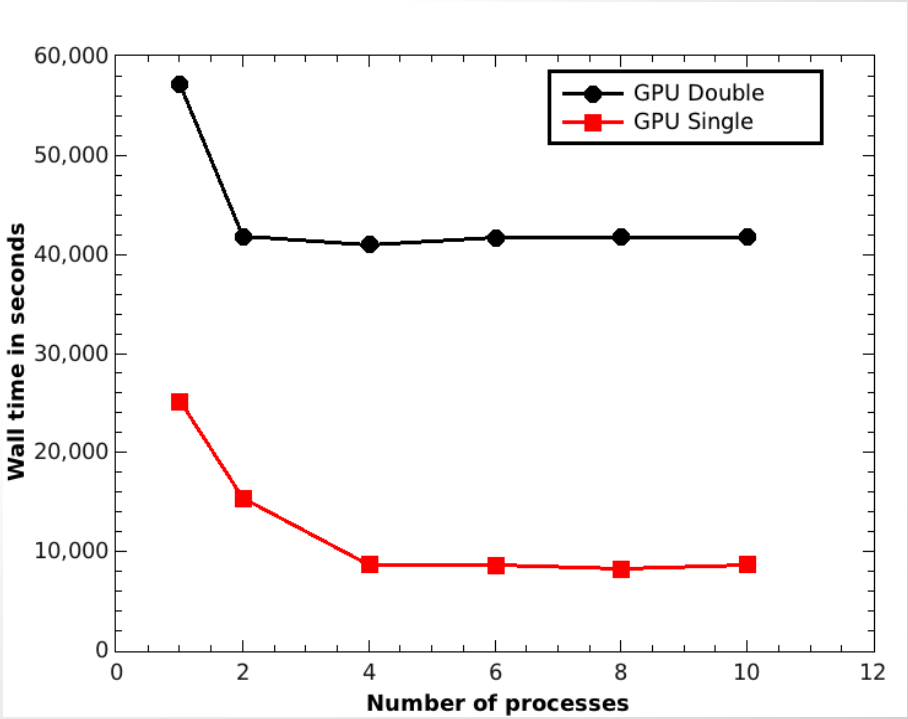
- In order to obtain reference execution times a single job is computed both in a CPU and GPU.
- For the CPU case domain decomposition and MPI are considered.
- For the GPU a pseudo domain decomposition is conducted. Different number of atoms are assigned to the GPU.

Num of domains	CPU	GPU double	GPU single
1	2055 s	381 s	163 s
2	1031 s	346 s	146 s
4	668 s	382 s	184 s
6	554 s	487 s	268 s

GPU Multiple Simulations

The 150 independent jobs can't be concurrently processed due to memory limitations. It is not possible to execute more that 10 parallel jobs maximun.

Parallel simulations	GPU double	GPU single
1	57238 s	25233 s
2	41885 s	15409 s
4	41070 s	8755 s
6	41731 s	8659 s
8	41804 s	8304 s
10	41808 s	8760 s



Multiple Simulations

The minimum makespan is obtained submiting four different processes (37 independent jobs each one). Four CPU cores are required in order to manage each process.

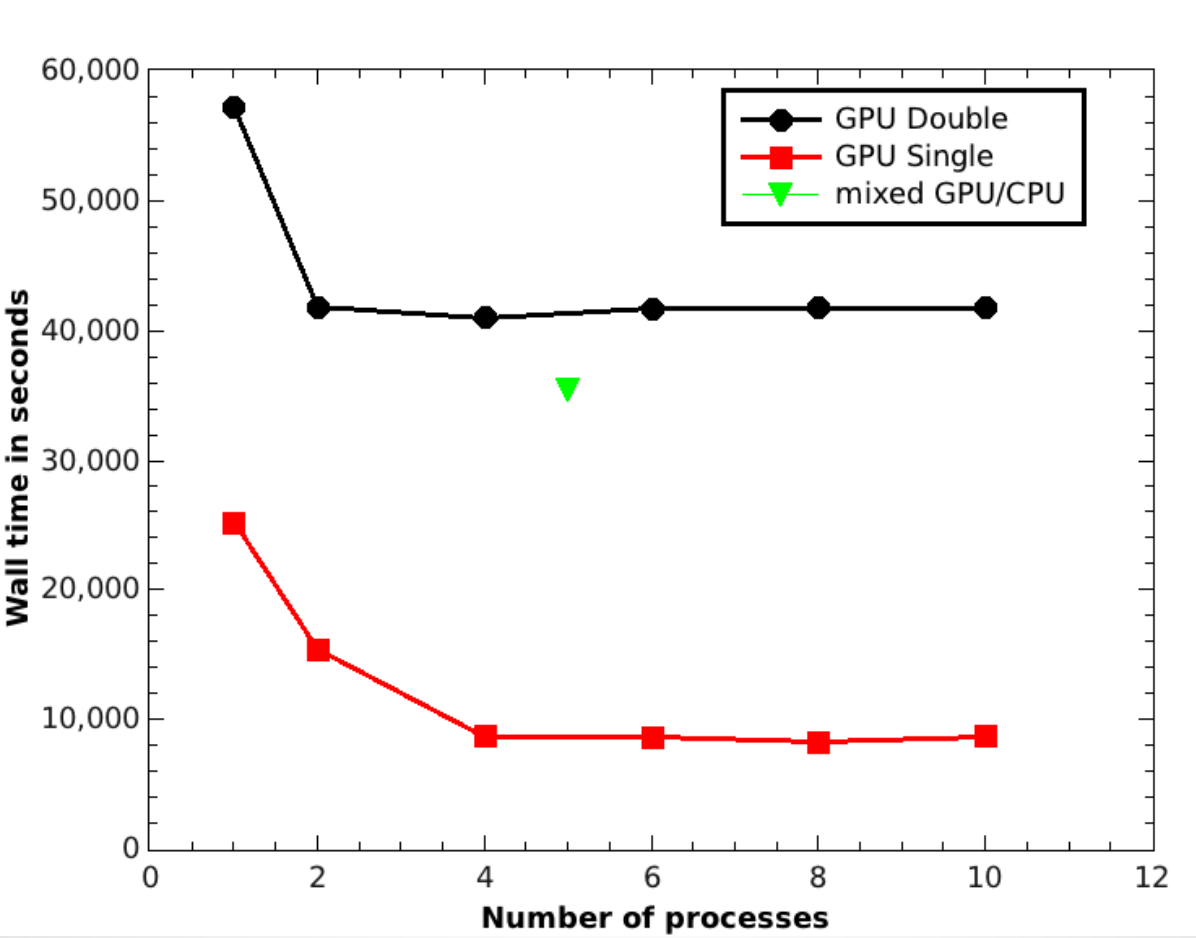
The remaining two CPU cores are used to execute simulations in parallel with the GPU.

Then five parallel processes are executed. In order to obtain a minimun makespan different numbers of jobs are assigned to each process.

Multiple Simulations

Parallel simul. CPU	CPU processes (-np)	Parallel simul. GPU	GPU Double	Distr.	
				gpu	cpu
6	1	0	60334 s	0	150
0	0	4	41070 s	150	0
1	2	4	40038 s	145	5
			38932 s	140	10
			35511 s	130	20
			38658 s	120	30
			48218 s	110	40

Multiple Simulations



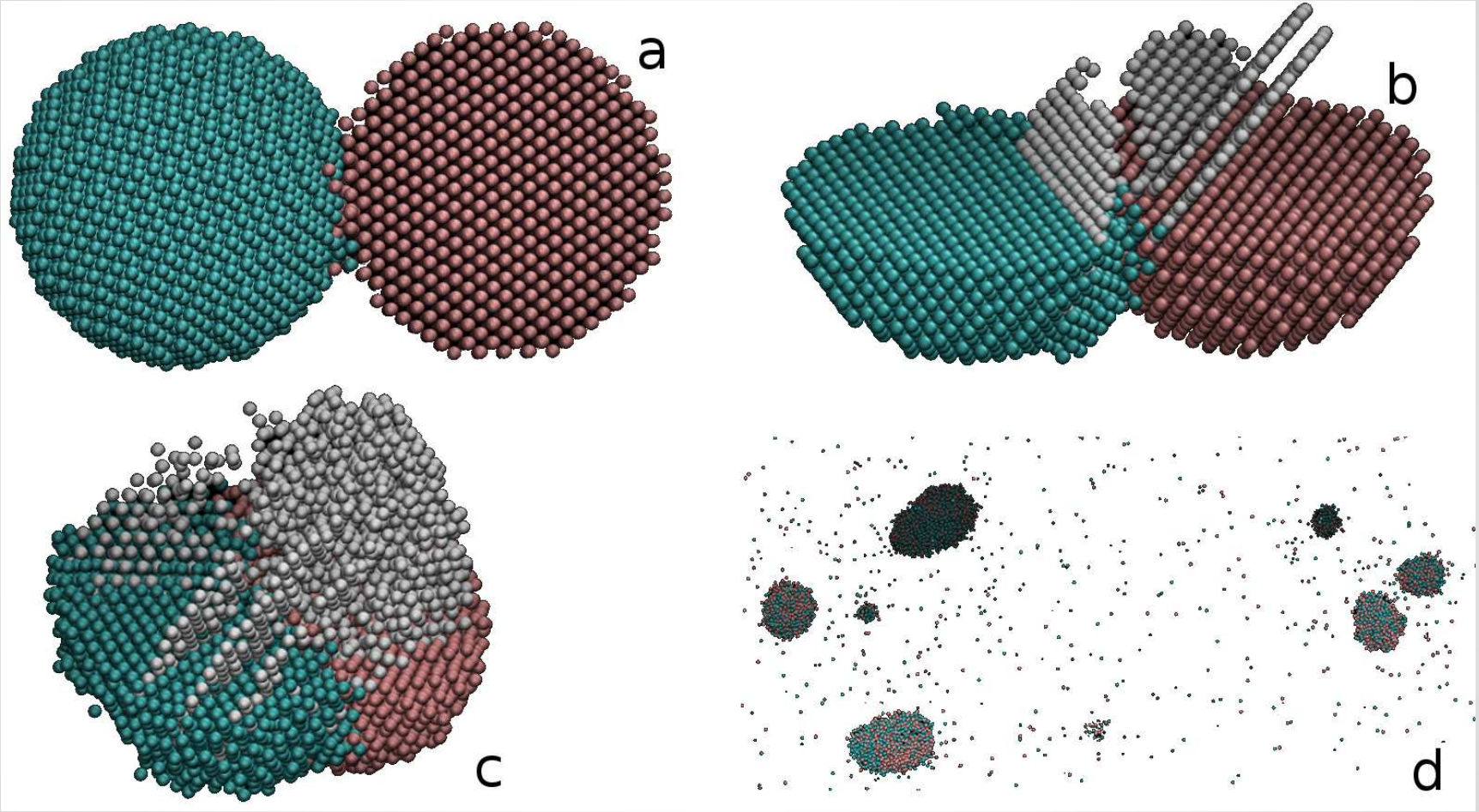
Collision of grains, 150 simulations. The mixed GPU/CPU test is for five processes running independent simulations in parallel using six CPU cores, one CPU process with two cores and four GPU processes.

Speed up & Summary

	Makespan	Speedup
CPU 1 process	308250 s	
CPU 6 processes	60334 s	5.1x
GPU double 4 processes	41070 s	7.5x
GPU 4 processes / CPU 1 process x 2 cores	35511 s	8.6x

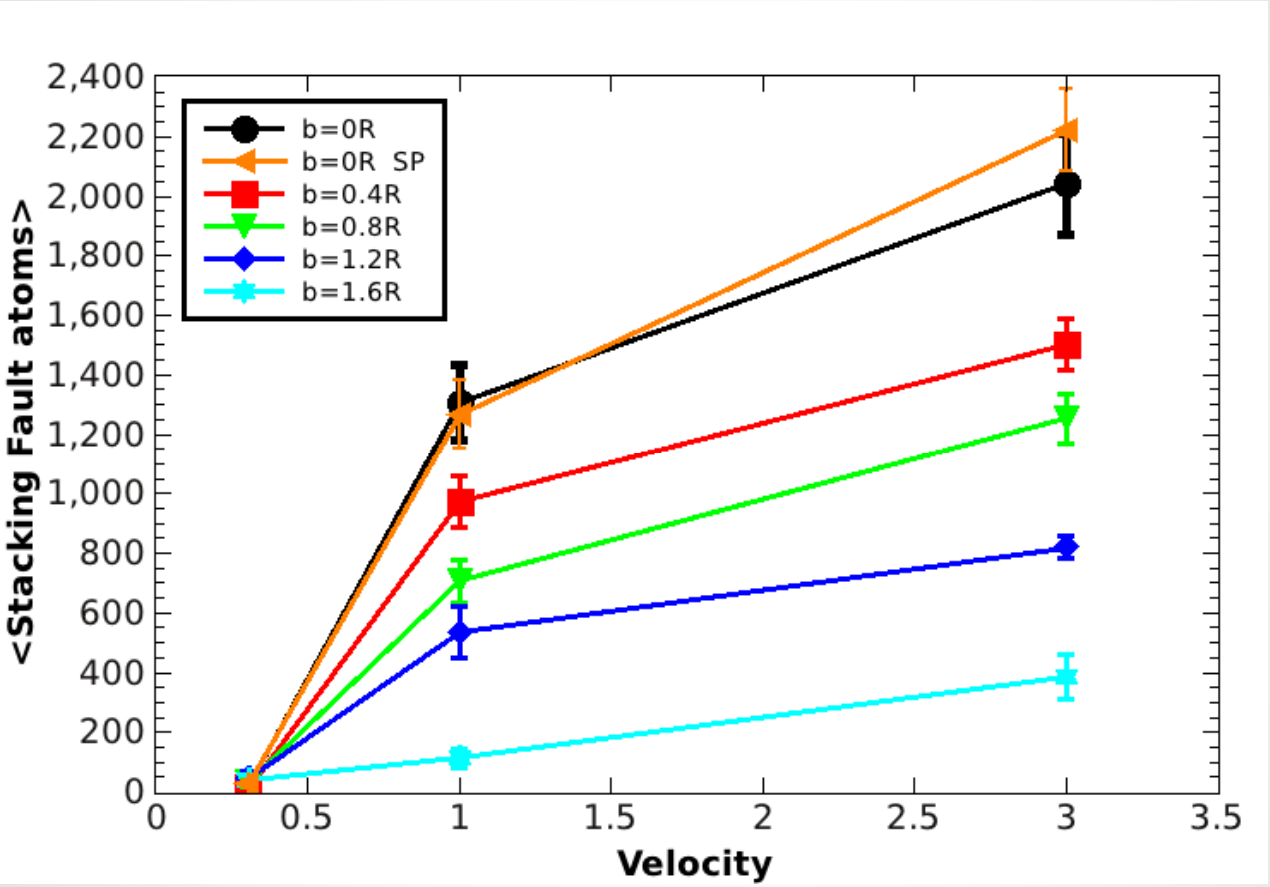
- For 1 process GPU is faster than CPU
- GPU performance similar to multicore CPU
- GPU features have to be stressed
- Concurrent (MPI) processes submitted to GPU 7.5 x
- GPU / CPU hybrid scheme provides additional speedup 1.15 x

Results



Snapshot showing the last step of the simulation for different velocities V (LJ units): a) V=0.3, b) V=1, c) V=3, and d) V=6. Only part of the atoms are shown, to allow the visualization of stacking fault atoms (grey), arranged in planar structures. Grains are showed in teal and red, to display the lack of mixing at low velocities.

Results



Average of stacking fault atoms for 10 orientations versus velocity (LJ units), b indicates the impact parameter, and R is the radius of the spherical grains (R=6.7 σ). SP indicates single precision.

Conclusions

- An ad-hoc strategy has been designed in order to conduct parameter sweeps for a MD problem. A Ruby script to has been designed.
- The performed tests shown that running independent simulations in parallel in both CPU and GPU improves makespan.
- Parametric studies can be solved in a reasonable completation time even for modest hardware like the used in this experiments.

Conclusions

- Using double precision, the GPU is 1.46 times faster than the six CPU cores, launching 4 simulation in parallel in the GPU.
- Mixing LAMMPS processes in GPU and CPU gives better performance. We obtained a speedup of 1.15x comparing the best result from double precision in GPU (41070 s) to the best result of mixed CPU/GPU (35511 s).

Conclusions

- The number of atoms in Stacking Faults (SFs) grows with velocity. Then plasticity effects are greater for larger grain velocities.

- We locate a threshold for plasticity, averaged over different impact orientations, for velocities between 0.3 and 1 (LJ units).

- Beyond V=3 we observe significant fragmentation of the grains, and a measure of plasticity is no longer meaningful.

- Such information is important for astrophysical applications and will be expanded in the near future.

Thank you!

Questions?