LAMMPS with CUDA for Dummies

I. Sticco[*] and F. Cornes[†]
Departamento de Física, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, Pabellón I, Ciudad Universitaria, 1428 Buenos Aires, Argentina.

G.A. Frank[‡]
Universidad Tecnológica Nacional, Facultad Regional Buenos Aires, Av. Medrano 951, 1179 Buenos Aires, Argentina.

(Dated: March 5, 2018)

This article is intended as a starting point in the art of simulation. We will try to make it easy...

PACS numbers: DuMMy, 1.2.me

I. WHERE ARE WE STANDING?

We assume that you managed to configure a basic system (that is, a single multi-core machine) with one Graphic Processing Unit (GPU). The GPU is supposed to be CUDA capable and is currently ready for running code. We further assume that you are already familiar with LAMMPS. If you are not in this situation, we recommend our tutorials "Computer configuration for Dummies (from scratch)" and "Lammps for Dummies" (both documents) .

The challenge is now to run LAMMPS on the CUDA capable GPU. LAMMPS people explain that four configuration steps are needed in order to run LAMMPS's scripts for CUDA. The steps are as follows

(1) Build the LAMMPS GPU library and LAMMPS files.

(2) Set the number of GPU's per node and the `mpirun` options.

(3) Change the LAMMPS scripts accordingly.

A. Build the LAMMPS files

LAMMPS does not ship with *all* the library and configuration files ready to use. This is because LAMMPS needs specific information on the system that is running. At least three parameters are required: the compute capability of the installed GPU, the precision, number of streaming multi-processors (SM), the `g++` current version and the value of the CUDA environmental variables.

In order to get the compute capability of your GPU, look into

```
http://www.nvidia.com/object/cuda_gpus.html
```

We will continue with the same example as in "Computer configuration for Dummies (from scratch)", that is, with a GeForce GTX980 graphic card installed in the system. Thus, it is straight forward from this website that the compute capability equals 5.2. You can further search in the web for the file

```
GeForce_GTX_980_Whitepaper_FINAL.PDF
```

and check that the SM number is 16.

The environmental variables can be checked through the `env` or `printenv` commands. The `LD_LIBRARY_PATH` variable equals `/usr/local/cuda-9.1/lib64`. The `lib64` means that the GPU operates in 64 bits (double precision).

The `g++` current version can be obtained by typing `g++ --version`. The returned version number is `6.3.0 20170516`.

A summary of all this information is listed below

```
compute capability: 5.2
streaming multi-processors (SM): 16
presicion: 64 bits
g++ version: 6.3.0 20170516
PATH=/usr/local/cuda-9.1/bin
CUDA_HOME=/usr/local/cuda-9.1
LD_LIBRARY_PATH=/usr/local/cuda-9.1/lib64
```

This information is enough (till now) to build the LAMMPS GPU library. The library is in

```
/home/me/mydir/lammps-31Mar17/lib/gpu
```

Move to this directory. You will find many files, but the important ones (at this stage of the configuration) are

```
Makefile.linux
Makefile.linux.double
Makefile.linux.mixed
```

[*] ignaciosticco@gmail.com
[†] fercornes@gmail.com
[‡] guillermo.frank@gmail.com

You may choose `Makefile.linux.double` as the source file for building the library since the precision of the current GPU is double (64 bits). Open this file, comment the line `CUDA_ARCH = -arch=sm_21` and replace it with the line `CUDA_ARCH = -arch=sm_52` since the compute capability is `5.2`. You may further check that `sm_52` is an allowed values by typing `nvcc --help`. Look for the lines

```
--gpu-architecture <arch>
...
Allowed values for this option:  'compute_30',
'compute_32','compute_35', 'compute_37',
'compute_50','compute_52','compute_53',
'compute_60','compute_61', 'compute_62',
'compute_70','compute_72','sm_30', 'sm_32' ,
'sm_35','sm_37','sm_50','sm_52','sm_53','sm_60',
'sm_61','sm_62','sm_70','sm_72'.
```

(Warning: if this option is not correctly set, you will experience some kind of error when running a LAMMPS script)

The `CUDA_HOME` line and the `CUDA_PRECISION` line remains the same since the default values are compatible with the collected information. Thus, you are now able to build the library by typing

```
make -f Makefile.linux.double
```

You will see lots of compilation information running through the screen. Just wait a few minutes (be patient).

Check if the files `libgpu.a` and `Makefile.lammps` were created. That's fine! Also try `./nvc_get_devices` and to produce the following report

```
-----------------------------
Using platform:
NVIDIA Corporation NVIDIA CUDA Driver
CUDA Driver Version:                           9.10

Device 0: "GeForce GTX 980"
Type of device:                                GPU
Compute capability:                            5.2
Double precision support:                      Yes
Total amount of global memory:                 3.94153 GB
Number of compute units/multiprocessors:       16
Number of cores:                               3072
Total amount of constant memory:               65536 bytes
Total local/shared memory per block:           49152 bytes
Total registers available per block:           65536
Warp size:                                     32
Maximum number of threads per block:           1024
Maximum group size (threads per block)
 1024 x 1024 x 64
Maximum item sizes (threads per dim)
 2147483647 x 65535 x 65535
Maximum memory pitch (bytes):                  2147483647
Texture alignment:                             512 bytes
Clock rate:                                    1.2155 GHz
Run time limit on kernels:                     No
Integrated:                                    No
Support host page-locked memory mapping:       Yes
Compute mode:                                  Default
Concurrent kernel execution:                   Yes
Device has ECC support enabled:                No
-----------------------------
```

Everything looks fine! Congratulations, you are almost done! Just include the GPU package as usual. We choose to include this package into the `mpi` build (do not try the `serial` build!). Type

```
cd /home/me/mydir/lammps-31Mar17/src
make yes-gpu
make mpi
```

Notice from the given report that a lot of code for the GPU has been added to `lmp_mpi`. That is all! Installation finished!

B. Test the LAMMPS GPU configuration

This section corresponds to the items (2) and (3) mentioned at the beginning of the document. Both items deal with either the scripts writing and the command line switches required for running the script in the GPU. However, we are allowed to replace the command line switches by some extra lines inside the scripts.

The very first example to try corresponds to the following script

```
units        lj
atom_style   atomic
lattice      sc 0.1111 origin 0.5 0.5 0.5
region       box block 0 30 0 30 0 30 units box
create_box   1 box
create_atoms 1 box
mass         1 1.0
velocity     all create 2.0 87287 dist gaussian

pair_style   lj/cut 2.5
pair_coeff   1 1 1.0 1.0 2.5

neighbor     0.2 bin
neigh_modify every 10 delay 0 check no

fix          1 all nve

thermo       50
run          2500
```

This is a simple Lennard-Jones(12,6) system with periodic boundary conditions. The total number of particles is 2744. The simulation runs through "neighbor lists". To run this script, type (assuming that the LAMMPS src directory is in the path)

```
lmp_mpi -i in.lammps
```

where `in.lammps` is the name of the script file. You can further type

```
mpirun -n 4 lmp_mpi -i in.lammps
```

in order to run four simultaneous processes through the `mpi` protocol.

To run this script through the single GPU in the system, it is necessary to introduce two changes

1. Add the following first line `package gpu 1`
2. Replace the command `pair_style lj/cut 2.5` by `pair_style lj/cut/gpu 2.5`

You should receive the following report

```
LAMMPS (31 Mar 2017)
Lattice spacing in x,y,z = 2.08015 2.08015 2.08015
Created orthogonal box = (0 0 0) to (30 30 30)
  1 by 1 by 1 MPI processor grid
Created 2744 atoms

--------------------------------------------------------------------------
- Using acceleration for lj/cut:
-  with 1 proc(s) per device.
--------------------------------------------------------------------------
Device 0: GeForce GTX 980, 16 CUs, 3.9/3.9 GB, 1.2 GHZ (Double Precision)
--------------------------------------------------------------------------

Initializing Device and compiling on process 0...Done.
Initializing Device 0 on core 0...Done.

Setting up Verlet run ...
  Unit style    : lj
  Current step  : 0
  Time step     : 0.005
Per MPI rank memory allocation (min/avg/max) = 2.084 | 2.084 | 2.084 Mbytes
Step Temp E_pair E_mol TotEng Press
       0            2  -0.13584096            0    2.8630657   0.17591932
      50    2.0835998  -0.26207998            0    2.8621807    0.2012439
     100    2.2340769    -0.489748            0    2.8601462    0.2284795
...
...
    2400    2.2949948  -0.58940312            0    2.8518346   0.23073706
    2450    2.2993636  -0.59553711            0    2.8522514   0.22313226
    2500    2.2993162  -0.59495139            0     2.852766   0.22164144
Loop time of 1.9205 on 1 procs for 2500 steps with 2744 atoms

Performance: 562352.161 tau/day, 1301.741 timesteps/s
20.6% CPU use with 1 MPI tasks x no OpenMP threads

MPI task timing breakdown:
Section |  min time  |  avg time  |  max time  |%varavg| %total
---------------------------------------------------------------
Pair    | 1.5275     | 1.5275     | 1.5275     |   0.0 | 79.54
Neigh   | 0.00041532 | 0.00041532 | 0.00041532 |   0.0 |  0.02
Comm    | 0.1416     | 0.1416     | 0.1416     |   0.0 |  7.37
Output  | 0.0028753  | 0.0028753  | 0.0028753  |   0.0 |  0.15
Modify  | 0.19741    | 0.19741    | 0.19741    |   0.0 | 10.28
Other   |            | 0.05069    |            |       |  2.64

Nlocal:    2744 ave 2744 max 2744 min
Histogram: 1 0 0 0 0 0 0 0 0 0
Nghost:    1692 ave 1692 max 1692 min
Histogram: 1 0 0 0 0 0 0 0 0 0
Neighs:    0 ave 0 max 0 min
Histogram: 1 0 0 0 0 0 0 0 0 0

Total # of neighbors = 0
Ave neighs/atom = 0
Neighbor list builds = 250
Dangerous builds not checked


---------------------------------------------------------------------
      Device Time Info (average):
---------------------------------------------------------------------
Data Transfer:   0.0670 s.
Data Cast/Pack:  0.1318 s.
Neighbor copy:   0.0003 s.
Neighbor build:  0.1685 s.
Force calc:      0.0732 s.
Device Overhead: 0.0679 s.
Average split:   1.0000.
Threads / atom:  4.
Max Mem / Proc:  3.98 MB.
CPU Driver_Time: 0.2851 s.
CPU Idle_Time:   0.4523 s.
---------------------------------------------------------------------


Please see the log.cite file for references relevant to this simulation

Total wall time: 0:00:02
```

Great! You made it! The GPU configuration is up and running!.

Just in case that you receive an error message or you want to compile again the GPU package, remember to do

```
cd /home/me/mydir/lammps-31Mar17/src
make no-gpu
cd /home/me/mydir/lammps-31Mar17/lib/gpu
make -f Makefile.linux.double clean
```

before you compile the package. Enjoy!