LAMMPS on GPUs
A Tutorial

W. Michael Brown, Peng Wang, Paul S. Crozier, Steve Plimpton

Wednesday, February 24, 2010
Why run on GPUs?

- Technology paid for by gamers, but impact to scientific computing is now well-recognized
- Cheap, low-power (electrical) solution for data parallelism
  - 240+ cores on a GPU
  - High memory bandwidth
Porting LAMMPS to GPUs

- Still largely a research effort

Marc Adams (Nvidia)  
Pratul Agarwal (ORNL)  
Sarah Anderson (Cray)  
Mike Brown (Sandia)  
Paul Crozier (Sandia)  
Massimiliano Fatica (Nvidia)  
Scott Hampton (ORNL)  
Ricky Kendall (ORNL)  
Hyesoon Kim (Ga Tech)  
Axel Kohlmeyer (Temple)  
Doug Kothe (ORNL)  
Scott LeGrand (Nvidia)  
Ben Levine (Temple)  
Christian Mueller (UTI Germany)  
Steve Plimpton (Sandia)  
Duncan Poole (Nvidia)  
Steve Poole (ORNL)  
Jason Sanchez (RPI)  
Arnold Tharrington (ORNL)  
John Turner (ORNL)  
Peng Wang (Nvidia)  
Lars Winterfeld (UTI Germany)  
Andrew Zonenberg (RPI)
Currently Available in “Main” LAMMPS

- Lennard-Jones
  - Force/Neighbor
- Gay-Berne Potential
  - Force
- More capabilities soon...
How to Run LAMMPS on Your GPU
1. Do you have a GPU?

- For single precision
  - Currently need a CUDA-enabled GPU with compute capability >= 1.1

- For double precision
  - Currently need a CUDA-enabled GPU with compute capability >= 1.3

Windows: Device Manager
Linux: nvidia_settings or /sbin/lspci | grep nVidia

List of CUDA-enabled GPUs here:

Can use device query to get compute capability; more later...
2. Do you have CUDA?

- Need driver and toolkit only
- Need to have the `nvcc` compiler in your path
- *Pay attention to 32- or 64-bit*
- *No 64-bit on apple!*

```bash
set path = ( $path /usr/local/cuda/bin )
setenv LD_LIBRARY_PATH /usr/local/cuda/lib/

or

set path = ( $path /usr/local/cuda/bin )
setenv LD_LIBRARY_PATH /usr/local/cuda/lib64/
```
3. Edit LAMMPS GPU

Makefile

```
set LROOT = "/home/wmbrown/lammps-20Feb10"

cd $LROOT/lib/gpu

emacs Makefile.nvidia
```
3. Edit LAMMPS GPU Makefile (2)

BIN_DIR = .
OBJ_DIR = .
AR = ar
CUDA_CPP = nvcc -I/usr/local/cuda/include -DUNIX -O3 -Xptxas -v --use_fast_math
CUDA_ARCH = -arch=sm_13
CUDA_PREC = -D_SINGLE_SINGLE
CUDA_LINK = -L/usr/local/cuda/lib64 -lcudart $(CUDA_LIB)

For compute capability >= 1.3 can also use:

CUDA_PREC = -D_SINGLE_DOUBLE # Double precision accumulation
or
CUDA_PREC = -D_DOUBLE_DOUBLE # Double precision everything

For Apple, must compile 32-bit

CUDA_ARCH = -arch=sm_13 -m32
CUDA_LINK = -L/usr/local/cuda/lib -lcudart $(CUDA_LIB)

For compiler >= g++ 4.4 on Linux

CUDA_ARCH = -arch=sm_13 --compiler-bindir=/usr/bin/gcc-4.3
4. Make LAMMPS GPU lib

```bash
make -f Makefile.nvidia
./nvc_get_devices
```

<table>
<thead>
<tr>
<th>Device 0: &quot;GeForce GTX 295&quot;</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Revision number:</td>
<td>1.3</td>
</tr>
<tr>
<td>Total amount of global memory:</td>
<td>0.87 GB</td>
</tr>
<tr>
<td>Number of multiprocessors:</td>
<td>30</td>
</tr>
<tr>
<td>Number of cores:</td>
<td>240</td>
</tr>
<tr>
<td>Total amount of constant memory:</td>
<td>65536 bytes</td>
</tr>
<tr>
<td>Total amount of shared memory per block:</td>
<td>16384 bytes</td>
</tr>
<tr>
<td>Total number of registers available per block:</td>
<td>16384</td>
</tr>
<tr>
<td>Warp size:</td>
<td>32</td>
</tr>
<tr>
<td>Maximum number of threads per block:</td>
<td>512</td>
</tr>
<tr>
<td>Maximum sizes of each dimension of a block:</td>
<td>512 x 512 x 64</td>
</tr>
<tr>
<td>Maximum sizes of each dimension of a grid:</td>
<td>65535 x 65535 x 1</td>
</tr>
<tr>
<td>Maximum memory pitch:</td>
<td>262144 bytes</td>
</tr>
<tr>
<td>Texture alignment:</td>
<td>256 bytes</td>
</tr>
<tr>
<td>Clock rate:</td>
<td>1.24 GHz</td>
</tr>
<tr>
<td>Concurrent copy and execution:</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Device 1: "Tesla C1060"
cd $LROOT/src
emacs ./MAKE/Makefile.linux

If you are not 64-bit (or Apple)
gpu_SYSPATH = -L/usr/local/cuda/lib

If you are using Apple, compile LAMMPS 32-bit to link with GPU library

CC = g++ -m32
LINK = g++ -m32

make clean
6. Add GPU Package to LAMMPS

cd $LROOT/src
make yes-asphere
make yes-gpu
make linux
7. Modify your input script

```
 cd $LROOT/bench
 emacs in.lj
```

Must add `newton off` to beginning of script and `/gpu` to a supported pair_style

```
newton off
...
pair_style lj/cut/gpu one/node 0 2.5
```
7. Modify your input script (2)

- **GPU Selection Keyword**
  - **one/node** - single compute "node", which may have multiple cores and/or GPUs. *GpuID* should be set to the ID of the (first) GPU you wish to use with LAMMPS.
  - **one/gpu** - multiple compute "nodes" with one GPU per node. *GpuID* should be set to the ID of the GPU.
  - **multi/gpu** - multiple compute "nodes" on your system with multiple GPUs. *GpuID* should be set to the number of GPUs per node.
8. Run your input script

- Number of procs = number of gpus you want

```
mpirun -np 3 lmp_linux < in.lj
```

---

- Using GPGPU acceleration for LJ-Cut:

  GPU 1: Tesla C1060, 240 cores, 4 GB, 1.3 GHZ
  GPU 2: Tesla C1060, 240 cores, 4 GB, 1.3 GHZ
  GPU 3: GeForce GTX 295, 240 cores, 0.87 GB, 1.2 GHZ

---

GPU Time Stamps:

- Atom copy: 0.07111 s.
- Neighbor copy: 0.0004615 s.
- LJ calc: 0.1702 s.
- Answer copy: 0 s.
9. Speed-ups

- Depends on
  - Your CPU
  - Your GPU
  - Number of Particles
  - Cutoff

- More talks showing the GPU acceleration in LAMMPS to come...
Questions