Gromacs Summary

Intro

Gromacs is a molecular dynamics code, that runs on both cpus and gpus.

This will be a sort of a summary of where we are with gromacs, and should be enough to bootstrap anyone to some familiarity.

Runtime

Gromacs works on with a hybrid of mpi ranks/omp threads. It suggests that having between 2 and 6 threads per rank, so common options on 28 cores are:

- 4 ranks, 7 threads
- 6 ranks 4 threads
- 8 ranks 3 threads
- 2 ranks 14 threads.

Also, in general there is a flag -npme which determines how much processing power is given to the electrostatic force calculation, and one can tune this using g_tune_pme, but our benchmark doesn't have electrostatics, so it doens't matter.

The ranks must be even when using two gpus per node, and I'm not sure about only using one.

There are other options, like -nstlist 100, which might have a slight improvement, but not much.

We found that 4 ranks, 7 threads seemed to perform the best.

▼ Run script

```
module load cmake/3.14.3
module load gcc/5.5.0
source /home/mbeukman/intel/2018/parallel_studio_xe_2018.4.057/psxevars.sh
module load cuda/10.0
source ../install/bin/GMXRC
which gmx_mpi

procs=${1:-4}
args=${2:-""}
filename=run-$procs-${args// /-/g}".log"

mpirun -np $procs gmx_mpi mdrun -s lignocellulose-rf.tpr  -gpu_id 0 -nsteps 400 $args  2>&1 | tee logs/$filename
```

Building

The build procedure is a relatively straight forward cmake procedure.

▼ Build script on the v100

```
module load cmake/3.14.3
module load gcc/5.5.0
source /home/mbeukman/intel/2018/parallel_studio_xe_2018.4.057/psxevars.sh
module load cuda/10.0

mkdir build
cd build

cmake ..  -DREGRESSIONTEST_DOWNLOAD=ON \
 -DGMX_SIMD=AVX_256 \
 -DCMAKE_INSTALL_PREFIX=`pwd`/../install \
 -DCMAKE_C_COMPILER=icc -DCMAKE_CXX_COMPILER=icpc \
 -DGMX_MPI=on -DGMX_GPU=on -DGMX_FFT_LIBRARY=mkl 2>&1 | tee cmake.log
make -j28 2>&1 | tee make.log
make check 2>&1 | tee makecheck.log
make install 2>&1 | tee makeinstall.log
```

Notes on the flags:

- -DGMX_SIMD=AVX_256: Use AVX_256 where you can, this is the highest that is supported on the skylakes.
- DCMAKE_C_COMPILER: icc
- -DGMX_MPI: Build the mpi version. This is for across nodes.
  - **When running only on one node, maybe the dgx, use -DGMX_THREAD_MPI=on instead. This has slightly better performance on one node.**
- -DGMX_GPU=on: Use gpus
- -DGMX_FFT_LIBRARY=mkl: Use mkl as fft library. This is not needed, since the fft is done on the gpus, so no mkl.

Stuff gathered from presentation on 2020/05/22

- The presentation (find the video and slides here: https://mellanox.box.com/s/bl9uzc6vkuhisp010p3idzorvt7f5874)
- The new gromacs 2020 version has ported basically everything to the gpu, so it's much faster
- The following environment variables activate the new optimizations
  - `export GMX_GPU_DD_COMMS=true`
    - This communicates directly between gpus (using cuda direct and nvlink and such) for the halo exchanges twixt PP tasks
  - `export GMX_GPU_PME_PP_COMMS=true`
    - GPU coms for pp ↔ pme tasks
  - `export GMX_FORCE_UPDATE_DEFAULT_GPU=true`
    - Puts all of the updating constraints (i.e. set new particles positions on the gpu rather than cpu)
    - can also use `-update gpu` option to mdrun (equivalent)
  - `-nb gpu -bonded gpu -pme gpu`
    - This tells mdrun to put all of the force calculations on the gpu

- On multi gpu, pass `-npme1` to mdrun, to limit pme interactions to a single gpu

- His recommendations
  - The full set of mdrunoptions used when running the above 4XGPU performance comparisons are as follows:

```
gmxmdrun-v -nsteps 100000 -resetstep 90000 -noconfout \
-ntmpi4 -ntomp10 \
-nb gpu -bonded gpu -pme gpu -npme1 \
-nstlist400
```

    - Explanation
      - The first line instructs Gromacsto run 100,000 steps for this relatively short benchmark test, with the timing counters reset at step 90,000 to avoid initialization costs being included in the timing since these will not be important for typical long production runs.
      - These specific values have been chosen to allow Gromacs long enough to automatically perform PME tuning, where the size of the PME grid is tuned to give an optimal load balance between PP and PME tasks.
      - Similarly, the -noconfoutoption instructs Gromacsnot to write output files at the end of this short benchmarking run to avoid artificially high I/O overheads.
      - The second line specifies that 4 thread-MPI tasks should be used (i.e. one per GPU), with 10 OpenMP threads per thread-MPI task such that a total of 40 OpenMP threads are in use to match the number of physical CPU cores in our server.
      - The third line offloads all force calculations to the GPU
      - And the final -nstlist400 option instructs Gromacs to update the neighbor list with frequency of 400 steps –this can be adapted without any loss of accuracy (when using the Verletscheme), and we found this value to give the best performance through experimentation.

Multiple Compute nodes (not useful for dgx, but good for our testing)

- Use this patch: https://gitlab.com/gromacs/gromacs/-/tree/cuda-aware-mpi
- Build MPI version of Gromacs in standard way (`-DGMX_MPI=ON`)
  - Except add a cmake option `-DGMX_CUDA_AWARE_MPI=ON` to enable new code path.
    - *Note that multi-node scaling will be very limited due to bottleneck on single PME task.*

Kind of useful supplementary info

- **Forces**
  - Non-bondedforces: (short range) -particles within a certain cutoff range interact directly
  - PME: long-range forces accounted for through a “Particle Mesh Ewald” scheme, where Fourier transforms are used to perform calculations in Fourier space, which is much cheaper than calculating all interactions directly in real space
  - Bonded forces: required due to specific behaviour of bonds between particles, e.g. the harmonic potential when two covalently bonded atoms are stretched
- For Things with PME, use 3 gpus on PP tasks, and the fourth only for pme (-npme 1)

Some questions

- Why do some benchmarks not have PME?
  - He said, there are some, not too common, but they exist
- With what parameters/options could we possibly experiment with, in addition to what you specified, to affect the performance?
  - look at profiles
  - what is code doing
  - maybe look at code optims
  - runtime
    - he gave the best ones.

- What does the -dd option (domain decomposition grid) to mdrun mean and how does it affect the run?
  - gromacs uses heuristics, -dd overwrites it
  - bad to use, in general
    - possibly if very unbalanced, then use.
- what does resetstep do?
  - reset step is what you need to include to time, i.e. -nsteps =100k, -resetstep 90k means only the last 10k steps will count for the time
-
- when we run gromacs, it runs non bonded forces twixt particles in a cut-off
  - neighbour list
  - needs to be recalculated every so often
    - nstlist how often should we recompute the list?

Conclusion

Gromacs is a difficult benchmark to optimise, and I'm not sure how to improve it's performance.

Our current best results are with:

- intel 2018 compiler
- avx2_256 simd instruction set
- GPU
- MPI on two nodes, or thread MPI on one node
- FFT doesn't matter since the gpu takes care of it.

Resources