Faster Code for Free: Linear Algebra Libraries
Advanced Research Computing, 26 February 2016

1 Setup

1. Log into the `hpcXX` (e.g. `hpc02`) provided to you with the provided password:
   (a) Mac/Linux: Open a terminal and type
   ```
   ssh hpcXX@blueridge2.arc.vt.edu
   ```
   (b) Windows: Download PuTTY and in the Host Name field type: `hpcXX@blueridge2.arc.vt.edu`
2. Copy over the relevant files to your directory:
   ```
   cp /home/TRAINING/LA_Libraries/* .
   ```
3. Start an interactive job by running:
   ```
   ./interactive.sh
   ```

2 Review, build, and verify `mm_test.cc`

1. Using `cat`, `less`, `vim`, or `emacs`, review the contents of `mm_test.cc`:
   (a) Use of `extern ''C'' {}` to declare `dgemm` without looking for it
   (b) Locations of `A`, `B`, `transa`, `transb`, `m`, `n`, and `k` in the call to `dgemm()`
2. Build it (as `mm_test`):
   (a) OpenBLAS:
   ```
   module purge; module load gcc/5.1.0 openblas
   g++ -g -std=c++0x -L$OPENBLAS_LIB -lopenblas mm_test.cc -o mm_test
   ```
   (b) ATLAS:
   ```
   module purge; module load gcc/5.1.0 atlas/3.11.34
   g++ -g -std=c++0x -L$ATLAS_LIB -llapack -lptf77blas -ltatlas -lgfortran mm_test.cc -o mm_test
   ```
   (c) MKL:
   ```
   module purge; module load intel/15.3 mkl
   icpc -std=c++0x -L$MKL_LIB -mkl mm_test.cc -o mm_test
   ```
3. Run the program and make sure that it produces the correct output:
   ```
   ./mm_test 2 3 4
   ```

3 Review and build `mm_perf.cc`

1. Build it (as `mm_perf`):
   (a) OpenBLAS:
   ```
   module purge; module load gcc/5.1.0 openblas
   g++ -g -std=c++0x -L$OPENBLAS_LIB -lopenblas mm_perf.cc -o mm_perf
   ```
   (b) ATLAS:

```
module purge; module load gcc/5.1.0 atlas/3.11.34
g++ -g -std=c++0x -L$ATLAS_LIB -llapack -lptf77blas -ltatlas -lgfortran mm_perf.cc -o mm_perf
```

(c) MKL:

```
module purge; module load intel/15.3 mkl
icpc -std=c++0x -L$MKL_LIB -mkl mm_perf.cc -o mm_perf
```

You can try Intel's MKL Link Line Advisor for other build options.

2. Run it and view performance.

```
./mm_perf 4224
```

3. Play with the matrix dimensions (not too large!) to see how it scales. Theoretical max Gflops/s for a BlueRidge node is 332.8.

4. If you built with MKL or OpenBLAS, try choosing different numbers of threads (no more than 16 since that's the number of cores on a BlueRidge node) and see how performance scales:

```
export OPENBLAS_NUM_THREADS=16 #Number of threads used by OpenBLAS
export MKL_NUM_THREADS=16      #Number of threads used by MKL
```