Faster Code for Free: Linear Algebra Libraries

Justin Krometis, Advanced Research Computing, 27 October 2017

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@cascades2.arc.vt.edu
   ```
   (b) Windows: Download PuTTY and in the Host Name field type: `hpcXX@cascades2.arc.vt.edu`
2. Copy over the relevant files to your directory:
   ```
   cp /home/TRAINING/LA/20171027/* .
   ```
3. Start an interactive job by running:
   ```
   ./interactive.sh
   ```

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

1. Using `cat`, `less`, `vim`, or `emacs`, review the contents of `mm_test.c`:
   (a) Use of `extern` 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 openblas
   gcc -L$OPENBLAS_LIB -lopenblas -o mm_test mm_test.c
   ```
   (b) ATLAS:
   ```
   module purge; module load gcc atlas
   gcc -L$ATLAS_LIB -llapack -lptf77blas -ltatlas -lgfortran mm_test.c -o mm_test
   ```
   (c) MKL:
   ```
   module purge; module load intel mkl
   icc -L$MKL_LIB -mkl mm_test.c -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.c`

1. Build it (as `mm_perf`):
   (a) OpenBLAS:
   ```
   module purge; module load gcc openblas
   gcc -lrt -L$OPENBLAS_LIB -lopenblas mm_perf.c -o mm_perf
   ```
   (b) ATLAS:

```
module purge; module load gcc atlas
gcc -lrt -L$ATLAS_LIB -llapack -lptf77blas -ltatlas -lgfortran mm_perf.c -o mm_perf
```

(c) MKL:

```
module purge; module load intel mkl
icc -lrt -L$MKL_LIB -mkl mm_perf.c -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 Cascades node is 870.

4. If you built with MKL or OpenBLAS, try choosing different numbers of threads (no more than 32 since that's the number of cores on a Cascades node) and see how performance scales:

```
export OPENBLAS_NUM_THREADS=32 #Number of threads used by OpenBLAS
export MKL_NUM_THREADS=32      #Number of threads used by MKL
```

4 Eigenvalue Solver `eig_test.c`

1. Using `cat`, `less`, `vim`, or `emacs`, review the contents of `eig_test.c`:

   (a) Use of `extern` to declare `dsyev` and `dgemm`

   (b) Structure of the `dysev` call. Note that the matrix gets replaced by eigenvectors since we set `jobz` to `v`

   (c) We use a matrix multiply (dgemm) to check our answers. Note where transposes are and are not used.

2. Build it (as `eig_test`):

   (a) OpenBLAS:

```
module purge; module load gcc openblas
gcc -L$OPENBLAS_LIB -lopenblas -o eig_test eig_test.c
```

   (b) ATLAS:

```
module purge; module load gcc atlas
gcc -L$ATLAS_LIB -llapack -lptf77blas -ltatlas -lgfortran eig_test.c -o eig_test
```

   (c) MKL:

```
module purge; module load intel mkl
icc -L$MKL_LIB -mkl eig_test.c -o eig_test
```

3. Run the program and make sure that it produces the correct output:

```
./eig_test 5
```