Spring 2017: Advanced Topics in Numerical Analysis: High Performance Computing

Compiling and running code on CIMS machines[1]

We have different versions of GNU and Intel compilers installed on CIMS machines. To compile a serial code such as those in the `lecture1` directory, you can use

```
gcc inner.c -o inner
```

If you prefer using an Intel compiler, can use using Intel's `icc` compiler:

```
icc inner.c -o inner
```

To get faster code, you can use optimization flags, e.g.:

```
icc inner.c -O2 -o inner
```

Alternatives to `-O2` are `-O1` (less aggressive optimization) or `-O3` (more aggressive optimization). You can get a newer Intel compiler by loading the corresponding module:

```
icc --version
module load intel-2016
icc --version
```

Compiling distributed memory code requires that MPI (Message Passing Interface) is available, which is the case by default for newer Intel compilers (i.e., it required to load the intel module).

```
mpicc inner-mpi.c -o inner-mpi
```

Then you can run the program on, say, 3 processor cores in parallel:

```
mpirun -np 3 ./inner-mpi 30000 100
```

The output should look something like that:

```
rank 0/3 reporting for duty
rank 1/3 reporting for duty
rank 2/3 reporting for duty
Time elapsed is 0.007342 seconds.
Inner product is 60000.000000.
13.074883 GB/s
0.817180 GFlops/s
```

You can ask for more cores than your computer has, and MPI will run several processes on a single physical compute core.
Finally, lets try to run the shared memory version of our program. The newer versions of the GNU compilers (as well as Intel compilers) support OpenMP, i.e., the framework that allows running several processes with a shared memory. You have to explicity link against the openmp libraries:

[1] This is for machines running CentOS, which should be most of them at that point as far as I know.

```
gcc inner-omp.c -fopenmp -o inner-omp
```

To run the shared memory version of our program, you can use:

```
./inner-omp 10000 100
```

Here, the system will decide how many threads to use. On my desktop, it uses 8 threads. You can also choose how many threads you want the system to use for your shared memory parallel computation:

```
OMP_NUM_THREADS=16 ./inner-omp 1000 1000
```

Running MPI across several CIMS machines

To run accross several CIMS machines, possibly including our compute servers in the basement[2], you first need to first ensure password-free ssh access, for instance following the steps described here[3]. Since your home directly is the same on all CIMS machines, this just amounts to copying your public key into the authorized_keys file (which is in the same directory). I usually just do a manual copy-and-paste, adding to the end of that file. Then, the command

```
mpirun -np 20 -hosts crunchy1,crunchy3 ./inner-mpi
```

will run the inner-mpi program on overall 20 cores of `crunchy1` and `crunchy3`. You can also create a file that contains the names of all the hostnames you want to be using, and pass this file to `mpirun` via `-f FILENAME` instead of listing the hosts on the command line.

[2] http://cims.nyu.edu/webapps/content/systems/resources/computeservers
[3] https://www.digitalocean.com/community/tutorials/how-to-set-up-ssh-keys--2