Hands-On Exercises for Performance Tuning Workshop

Judy Gardiner
Ohio Supercomputer Center

July 27, 2017

These exercises go along with the Performance Workshop at the Ohio Supercomputer Center (OSC). They are based on a simple Laplace solver code with different optimizations. The code is written in Fortran, but the principles are the same for C/C++. Loops would have to be interchanged for C/C++ because arrays are laid out differently than in Fortran (row-major vs. column-major order).

Code for the examples is on Owens in the directory: `~judithg/training/Performance`

Three versions of the code are used to illustrate the effect of various optimizations. If appropriate optimization flags are set, the compiler can automatically perform many of the optimizations.

laplace-good	Code accessing memory with unit stride for good cache utilization and vectorization.
laplace-slow	Loops are incorrectly ordered for poor cache utilization (bad example).
laplace-omp	OpenMP version for multicore usage.

1. Start an interactive batch job on a 28-core node with a 4-hour walltime limit.

   ```
   qsub –I –X –l nodes=1:ppn=28 –l walltime=4:00:00
   ```

   [Wait for prompt]

   ```
   cd $PBS_O_WORKDIR
   module load allinea
   ```

2. Compile the good code with recommended flags and time its execution.

   ```
   ifort -O3 -xHost –o laplace-good laplace-good.f
   /usr/bin/time ./laplace-good
   ```

3. Generate and view a performance report.

   ```
   perf-report --no-mpi ./laplace-good
   firefox [name of file]
   ```

   Note: You can find the file name for the performance report by using the command "`ls`".
   Example: `laplace-good_1p_1n_2017-07-06_14-21.html`

4. Compile, displaying information about loop optimizations. This generates the same executable as exercise #1.

```
ifort –O3 –xHost –qopt-report –o laplace-good laplace-good.f
cat laplace-good.optrpt
```

5. Compile the good code with no optimizations and time its execution. Also generate a performance report.

```
ifort –O0 –o laplace-good-O0 laplace-good.f
/usr/bin/time ./laplace-good-O0

perf-report --no-mpi ./laplace-good-O0
firefox [name of file]
```

6. Compile the slow code with no optimizations and time its execution. Also generate a performance report.

```
ifort –O0 –o laplace-slow-O0 laplace-slow.f
/usr/bin/time ./laplace-slow-O0

perf-report --no-mpi ./laplace-slow-O0
firefox [name of file]
```

7. Compile the slow code with recommended flags, displaying loop optimization information, and time its execution.

```
ifort -O3 –xHost –qopt-report –o laplace-slow laplace-slow.f
cat laplace-slow.optrpt
/usr/bin/time ./laplace-slow
```

8. Compile OpenMP code and time its execution on different numbers of cores.

```
ifort –O3 –xHost -qopenmp –o laplace-omp laplace-omp.f
export OMP_NUM_THREADS=1
/usr/bin/time ./laplace-omp
export OMP_NUM_THREADS=2
/usr/bin/time ./laplace-omp
export OMP_NUM_THREADS=4
/usr/bin/time ./laplace-omp
```