

OpenMP Lab

https://github.com/ResearchComputing/USGS_2016/

February 9, 2016

Timothy Brown



Research Computing
UNIVERSITY OF COLORADO **BOULDER**

Source Code

All of the source code is online, in a github repository:

https://github.com/ResearchComputing/USGS_2016_02_09-10

Login to Yeti

1. Log in to Yeti.

```
laptop ~$ ssh yeti.cr.usgs.gov
```



2. Clone the git repository

```
yeti-login01 ~$ git clone \  
https://github.com/ResearchComputing/USGS_2016_02_09-10
```

3. Start a compute job.

```
yeti-login01 ~$ sinteractive -A training \
                        -p UV           \
                        -t 01:00:00 -n 1 \
                        --cpus-per-task=4 \
                        --reservation=workshop
```



4. Load the Intel parallel studio module

```
compute80 ~$ module load intel/psxe-2015
```

This contains the C, C++ and Fortran compilers as well as the Intel MPI library.

Language	Compiler
C	icc
C++	icpc
Fortran	ifort

Number of threads

Figure out how many threads we have.
Using the `omp_get_thread_num()` library call.

Example	Source
Fortran	<code>thread_num/thread_num_f.f90</code>
C	<code>thread_num/thread_num_c.c</code>

► Compiling the program.

Intel `icc -qopenmp -o thread_num_c thread_num_c.c`

GCC `gcc -fopenmp -o thread_num_c thread_num_c.c`

PGI `pgcc -mp -o thread_num_c thread_num_c.c`

IBM `xlc -qsmp=omp -o thread_num_c thread_num_c.c`

► Execute the program, specifying different numbers of threads.

1. `./thread_num_c`

2. `env OMP_NUM_THREADS=1 ./thread_num_c`

3. `env OMP_NUM_THREADS=64 ./thread_num_c`

► What is the output?

- Threads printed out their identification number.
- Random order of numbers. Threads execute independently and in general order will be random.

Preprocessor

Using the pre-processor with C and Fortran code.

- ▶ Using `ifdef`'s within the source code file, means you can compile your code with OpenMP directives without using OpenMP.

Example	Source
Pre-processor in Fortran	<code>ifdef/if.F90</code>

Loops

Example	Source
Array addition using a for loop	loop/for.c
First private in a parallel region	loop/first.c
Last private in a parallel region	loop/last.c
Reduction operator	loop/total.c

Sections

Example	Source
Distribute independent work	<code>sections/sections.f90</code>

Synchronization

Example	Source
Synchronization using a barrier	<code>sync/barrier.c</code>
An ordered reduction	<code>sync/ordered.c</code>

Alignment

Data alignment is crucial when using SIMD directives for vectorization.

Example	Source
Alignment in C	<code>align/align_c.c</code>
Alignment in Fortran	<code>align/align_f.F90</code>

Note `icc`, `gcc` and `gfortran` pre-processors have `__BIGGEST_ALIGNMENT__` defined. However `ifort` does not (it is currently an open feature request). So the `Makefile` defines it for Fortran.

Target

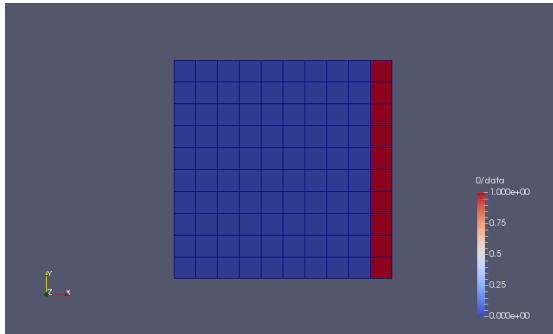
Using the OpenMP v4.5 target directives for off-loading to an accelerator.

Example	Source
Target in C	target/target_c.c

Jacobi

The Jacobi method for stencil-based iterative solvers. The example is for a 3D diffusion equation.

Example	Source
Fortran	<code>jacobi</code>



- ▶ Load the HDF5 module for I/O.
`compute80 ~$ module load tools/hdf5-1.8.15-intel`
- ▶ Compile the serial jacobi program
`compute80 ~$ make`
- ▶ Execute the program with a $100 \times 100 \times 100$ domain and 1000 iterations. These are the defaults in the `input.nl` file.

```
compute80 ~$ ./jacobi -i input.nl
```

Questions?

Online Survey

<Timothy.Brown-1@colorado.edu>

License

This work is licensed under the Creative Commons Attribution 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by/4.0/>

When attributing this work, please use the following text:
“OpenMP Lab”, Research Computing, University of Colorado
Boulder, 2015. Available under a Creative Commons
Attribution 4.0 International License.

