OpenMP

Josh Alexander, University of Oklahoma
Ivan Babic, Earlham College
Andrew Fitz Gibbon, Shodor Education Foundation Inc.
Henry Neeman, University of Oklahoma
Charlie Peck, Earlham College
Skylar Thompson, University of Washington
Aaron Weeden, Earlham College
Sunday June 26 – Friday July 1 2011


Outline

- What is OpenMP?
- Why do we care about it?
- What does it take to *write* an OpenMP program?
- What does it take to *compile* an OpenMP program?
- What does it take to *run* an OpenMP program?

What is OpenMP?

- Shared Memory Parallelism
- Multithreaded code (what's a thread?)
- OpenMP == "Open Multi Processing"
- Just like MPI, OpenMP is a *standard*. This one consists of:
  - Compiler directives
  - Functions
  - Environment variables
- Since it's only a standard, we must have an implementation
  - Intel compilers, GNU compilers
- Only C/C++/Fortran

Compiler Directives

- Hints to the compiler; Suggestions for it to do something special with your code.
- Is the compiler *required* to listen to you? (Hint: it's not)
- These directives are called "pragmas" (i.e., *pragmatic*)
- The most common pragma looks like this:
  - #pragma omp parallel for
  - (We'll get more into what it actually means later)
- They will control *how* our code gets parallelized

Functions

- The OpenMP functions allow us to gather information about —and alter—the OpenMP runtime environment.
- Just like with MPI, we must include the OpenMP library:
  - #include <omp.h>
- Then we can use OpenMP functions like:
  - omp_get_thread_num();
- These functions will let us get, and use, the OpenMP equivalent of MPI's *rank* and *size*.

Environment Variables

- The Environment Variables provide a way for us to dictate certain features of the OpenMP runtime.
- For example, the number of threads:
  - setenv OMP_NUM_THREAD=8
  - This will tell the environment to run our program with 8 threads.
- Other options include:
  - OMP_SCHEDULE
  - OMP_STACKSIZE

Why do we care?

- MPI is hard
- OpenMP is **easy** (you'll see soon)
- No need to pass data around
- Avoid lots of concurrency issues and complications arising from complex MPI code
- "Instant" gratification

Why might OpenMP/SharedMemory be "bad"?

Write OpenMP

- Let's start with the canonical "Hello World":

```
int main () {
        printf("Hello World!\n");
}
```

Write OpenMP

- Now with an OpenMP pragma:

```
int main () {
        #pragma omp parallel
        {
        printf("Hello World!\n");
        }
}
```

Better Example

```
#include <stdio.h>

int main () {
        int i;

        #pragma omp parallel for
        for (i = 0; i < 10; i++) {
                printf("i=%d\n", i);
        }
}
```

Compile (and Run) OpenMP

- ***DO THIS ON AL-SALAM***

cd

./qsub-interactive-1-node

cp -r ~fitz/BWUPEP2011_OpenMP .

gcc -fopenmp -o for for.c

./for

- What happened?

Modifications

- What happens if you split up the printf inside the for loop?
  - printf("i=");
    printf("%d\n", i);
- What does the output look like? Why is this happening?
- Do we *want* to "fix" it?

Rank and size again

```
#include <omp.h>
#include <stdio.h>
#define WORKLOAD 1

int main () {
        int rank, size, i;
        #pragma omp parallel
        {
                rank = omp_get_thread_num();
                for(i=1; i<WORKLOAD; ++i);
                printf("Hello World from thread %d\n", rank);
                if ( rank == 0 ) {
                        size = omp_get_num_threads();
                        printf("There are %d threads\n",size);
                }
        }
        return 0;
}
```

Rank and size again

- What does it do? (Hint: not what we want)
- Thoughts for improving? (Hint: SMP)

Your turn…

- Take a look at the other code in the `BWUPEP2011_OpenMP` directory (timing.c). Read it, try to understand it, run it, see if you can change it to get different behavior.
- Take a look at your serial *calculatePI* code, try to add OpenMP to it (Hint: it should only take one pragma; think reduce too)
- Take a look at the serial NBody code. Remember the gprof'ing we did? Given that information, where do you think the best place for an OpenMP pragma is? Try it!