CmpE 275, Fall 2015 – OpenMP Lab

C++ OpenMP Threading examples

The intent of the three coding examples it to help understand the effects of memory, ordering, and independent execution in distributed processes (threads in this case). These short examples showcase both the ease of parallel coding and the problems one encounters. There are many sources of documentation, examples, slideshows on using OpenMP (e.g., http://bisqwit.iki.fi/story/howto/openmp/).

The following comments assist in exploring options with the samples

Hello – The hello examples (2) look at the classic, ubiquitous, code fragment seen in most language introductions. Yet, our examples are inconsistent. Really, how much complexity an exist in a one line of code? This is what makes this example a great learning point when examining dependencies between threads and a common or shared memory space. There is only one line of code difference that separates the two examples, a print() and std::cout. So, why is it that running the two codes the output is different. The key question of this example is Why are the results different concerning these two lines?

```
printf("Hello (%d)\n",ID);

std::cerr << "Hello (" << ID << ") " << std::endl;
```

What is occurring with these statements is the key point. The execution of the std::cerr is a collection operator overloaded functions (<<) each function call gives an opportunity for context switching. Secondly, the implementation of std::cout and std::cerr use a shared buffer. These two conditions create a race condition.

How do you fix this code to produce the desired results? One option is to use critical sections (`#pragma omp critical`) – What is the cost of using a critical section? Try timing both the correct and incorrect versions to understand the cost. Why is this so?

PI – The four implementations to calculate PI extend the hello example to look at loop-level parallelism and how to manage shared memory dependencies. Correct implementations (one is not correct) show stylistic viewpoints. What is common to each threaded example is the loop dependency to sum a value, this creates shared memory between thread executions.

The Java version is provided to benchmark against the C++ versions.

Matrix – The third code example is provided without OpenMP pragma statements. It is intended to provide a code to try your hand at optimizing some code. Before adding any pragma statements, be sure to run a few tests to act as a baseline for comparing your changes.

Notes:

1. There are a number of tools available to you. There is the IDE's graphical debugger which allows you to step through the execution or look at call stacks.
2. Also available are analysis tools like Google's gperftools (https://github.com/gperftools/gperftools) and Valgrind's cachegrind (http://valgrind.org).
3. Have you read about cache line?