COSC 462

OpenMP Basics: Directive Syntax

Piotr Luszczek

October 3, 2016

Example: Largest Value (OpenMP)

#pragma’s are compiler directives and are **not** like preprocessor directives such as #include or #ifdef

OpenMP created a namespace: omp

OpenMP allows parallel processing inside “parallel” regions

```
#pragma omp parallel for reduction(+:sum)
for (int i=0; i<N; ++i)
  sum += X[i];
```

Reductions need a target variable

OMP #pragma’s are most often applied to loops

OpenMP has fast built-in reductions based on arithmetic, bitwise, and logical operators

Separate OpenMP Pragma's

#pragma parallel marks a parallel region

```
#pragma omp parallel
{ // begin of parallel region

#pragma omp for reduction(+:sum)
for (int i=0; i<N; ++i)
  sum += X[i];

} // end of parallel region
```

There are a few OpenMP pragma's that may occur inside a parallel region

Running Multiple Threads

parallel regions don't have to use other OpenMP pragmas

```
#pragma omp parallel
{ // begin of parallel region

    printf( "Hello world!\n" );

} // end of parallel region
```

```
Hello world!
Hello world!
Hello world!
...
```

The output may be **scrambled** and each thread will print once

Dealing with I/O: One Thread for I/O

```
#pragma omp parallel
{ // begin of parallel region

    #pragma omp single
    printf( “Hello world!\n” );

} // end of parallel region
```

```
Hello world!
```

The output will not be **scrambled** and each thread will print once in any order

Dealing with I/O: Mutual Exclusion

```
#pragma omp parallel
{ // begin of parallel region

    #pragma omp critical
    {
    printf( “Hello world!\n” );
    fflush( stdout );
    }

} // end of parallel region
```

```
Hello world!
Hello world!
Hello world!
...
```

The output will not be **scrambled** and each thread will print once in any order