OpenMP Overview

Cheng-Zhong Xu
Wayne State University

What is OpenMP?

- A new API for multi-threaded, SAS programming on shared memory machines.
- A set of compiler directives in the source program.
- Portable: compiler directives do not affect sequential code;
  - pragram’s in C / C++
  - (special) comments in Fortran code.
- Standardized: rapidly gaining acceptance among vendors and application writers.
- Has specific support for scientific application needs (unlike Pthreads).
- See http://www.openmp.org for more info.
OpenMP Example

Sequential Matrix Multiplication
for( i=0; i<n; i++ )
 for( j=0; j<n; j++ ) {
    c[i][j] = 0.0;
    for( k=0; k<n; k++ )
      c[i][j] += a[i][k]*b[k][j];
}

//OpenMP Code
#pragma omp parallel for
for( i=0; i<n; i++ )
 for( j=0; j<n; j++ ) {
    c[i][j] = 0.0;
    for( k=0; k<n; k++ )
      c[i][j] += a[i][k]*b[k][j];
}

OpenMP Programming Model

- Thread-based, Explicit Parallelism
- Fork-Join Model
- Compiler directive for parallelism exploitation
- Nested parallelism is supported
  - Parallel constructs inside of other parallel constructs
- Dynamic parallelism is supported
General OpenMP Code Structure

```c
#include <omp.h>
main () {
    int var1, var2, var3;
    Serial code ... 

    // Beginning of parallel section. Fork a team of threads.
    // Specify variable scoping
    #pragma omp parallel private(var1, var2) shared(var3)
    {
        Parallel section executed by all threads ... 
        All threads join master thread and disband
    }
    Resume serial code
    ... 
}
```

OpenMP Example Usage (1 of 2)

- OpenMP is more than an API
  - OpenMP compiler
  - OpenMP runtime library
Compiler Switch

- If you give sequential switch,  
  - comments and pragmas are ignored.
- If you give parallel switch,  
  - comments and/or pragmas are read, and  
  - cause translation into parallel program.  
  - e.g. IBM XLF compiler  
    `xlf_r -qsmp=omp example.f`
- Ideally, one source for both sequential and parallel program (big maintenance plus).

OpenMP Directives

- Parallelization directives:  
  - `parallel region`
  - `parallel for`
- Data environment directives:  
  - `shared`, `private`, `firstprivate`, `threadprivate`,  
    `reduction`, etc.
- Synchronization directives:  
  - `barrier`, `critical`, `flush`
General Rules about Directives

- They always apply to the next statement, which must be a structured block.
- Examples
  - `#pragma omp …
    statement`
  - `#pragma omp …
    { statement1;
      statement2;
      statement3; }`

OpenMP Parallel Region

- `#pragma omp parallel`
  - A number of threads are spawned at entry
    - set by `omp_set_num_threads()`
    - `OMP_NUM_THREADS` env variable
    - Implementation default
  - Each thread executes the same code.
  - Each thread waits at the end.
  - Very similar to a number of create/join’s with the same function in Pthreads.
Explicit Thread Control

- Get thread ID and total number (as in pthreads).
  
  ```c
  int omp_get_thread_num()
  int omp_get_num_threads()
  ```

  Example:
  ```c
  #pragma omp parallel
  {
    if( !omp_get_thread_num() )
      master();
    else
      slave();
  }
  ```

Parallel Region Example

```c
#include <omp.h>
main () {
  int nthreads, tid;
  /* Fork a team of threads giving them their own copies of variables */
  #pragma omp parallel private(nthreads, tid)
  {
    /* Obtain and print thread id */
    tid = omp_get_thread_num();
    printf("Hello World from thread = %d\n", tid);
    /* Only master thread does this */
    if (tid == 0) {
      nthreads = omp_get_num_threads();
      printf("Number of threads = %d\n", nthreads);
    }
  } /* All threads join master thread and terminate */
}
Work Sharing Directives

- A work-sharing construct divides the execution of the enclosed region among threads
  - Always occur within a parallel region directive.
  - Launch NO new threads
- Two principal ones are
  - parallel for
  - parallel section
- Example
  ```c
  #pragma omp parallel
  #pragma omp for
  for( ... ) { ... }
  - Each thread executes a subset of the iterations.
  - All threads wait at the end of the parallel for.
  ```

Example: work-sharing

```c
#include <omp.h>
#define CHUNKSIZE 100
#define N 1000
main () {
  int i, chunk;
  float a[N], b[N], c[N];
  /* Some initializations */
  for (i=0; i < N; i++)
    a[i] = b[i] = i * 1.0;
  chunk = CHUNKSIZE;

  #pragma omp parallel shared(a,b,c,chunk) private(i)
  {
    #pragma omp for schedule(dynamic,chunk) nowait
    for (i=0; i < N; i++)
      c[i] = a[i] + b[i];
  } /* end of parallel section */
} /* end of main */
```

- `schedule`: determines how loop iteration be divided: dynamic, static guided, etc
- `nowait`: no need for sync at the end of parallel loop
Multiple Work Sharing Directives

• May occur within a single parallel region
  
  ```
  #pragma omp parallel
  {
    #pragma omp for
    for ( ; ; ) { … } /* i-loop */
    #pragma omp for
    for ( ; ; ) { … } /* j-loop */
  }
  ```

• All threads wait at the end of i-loop.

Conditional Parallelism:

```
#pragma omp parallel if( expression )
#pragma omp for if( expression )
#pragma omp parallel for if( expression )
```

Execute in parallel if expression is true, otherwise execute sequentially.

```for( i=0; i<n; i++ )
    #pragma omp parallel for if( n-i > 100 )
    for( j=i+1; j<n; j++ )
        for( k=i+1; k<n; k++ )
            a[j][k] = a[j][k] - a[i][k]*a[i][j] / a[j][j]
```
Types of Work-Sharing Directives

- for, section, single

Synchronization Directives

- critical. E.g.
  ```
  #pragma omp parallel shared(x)
  {
    #pragma omp critical
    x = x+1;
  } // end of parallel section
  ```

- barrier. E.g.
  ```
  if (x==0)
  {
    #pragma omp barrier
  }
  ```

- atomic, ordered, flush
OpenMP Example: Synchronization

```c
#pragma omp parallel private(mydiff)
for( ; diff > delta; ) {
    #pragma omp for nowait
    for( i=from; i<to; i++ )
        for( j=0; j<n, j++ ) { ... }
    diff = 0.0; mydiff = 0.0;
    #pragma omp barrier
...
#pragma omp for nowait
for( i=from; i<to; i++ )
    for( j=0; j<n; j++ ) {
        mydiff = max(mydiff,fabs(grid[i][j]-temp[i][j]);
        grid[i][j] = temp[i][j];
    }
#pragma critical
    diff = max( diff, mydiff );
#pragma barrier
}
```

History

- In the early 90’s: popularity of SMP and DSM machines
- OpenMP started in 1997
  - Oct 1997, Fortran Ver 1.0
  - Late 1998, C/C++ Ver 1.0
  - June 2000, Fortan Ver 2.0
  - April 2002, C/C++ Ver 2.0
- Vendors:
  - HP, Intel, IBM, Sun, SGI, etc
  - Many other software vendors
  - Application developers