PC to HPC

Xiaoge Wang
ICER
Jan 27, 2016
About This Series

• Format: talk + discussion
• Focus: fundamentals of parallel computing
  – (i) partitioning: data partition and task partition;
  – (ii) communication: data sharing or message passing;
  – (iii) coordination between parallel tasks.
• Tools:
  – (i) Shell script
  – (ii) OpenMP
  – (iii) MPI
  – (iv) Matlab
Objective

• Introduction to parallel programming
• Hear from MSU research computing community
Seminar 2:

Parallel Programming with OpenMP
Outline

• Fundamentals for parallel programming
• Overview of OpenMP
• Example of HPC using OpenMP
Outline

• **FUNDAMENTALS FOR PARALLEL PROGRAMMING**
  • Overview of OpenMP
  • Example of HPC using OpenMP
Fundamentals

• Partitioning:
  – Data partition
  – Task partition

• Communication
  – Data sharing
  – Message passing

• Coordination between parallel tasks.
  – Dependency
  – Workflow
Fundamentals in OpenMP

• Partitioning:
  – *Data partition*
  – *Task partition*

• Communication
  – *Data sharing*
  – Message passing

• Coordination between parallel tasks.
  – *Synchronization*
Outline

• Fundamentals for parallel programming

• **OVERVIEW OF OPENMP**

• Example of HPC using OpenMP
Programming Model

• Fork-Join model: OpenMP programs execute serially until they encounter the “parallel” directives.

• The directives are responsible for creating a group of threads
Memory Model

- Relaxed-consistency shared memory
- Each thread has its temporary view of memory
- Each thread has access to private memory
OpenMP Stack

OpenMP Basic defs: Solution Stack

- End User
- Application
  - Directives, Compiler
  - OpenMP library
  - Environment variables
- OpenMP Runtime library
  - OS/system support for shared memory and threading
- Shared Address Space
  - Proc1
  - Proc2
  - Proc3
  - ProcN
OpenMP API

- Directives
- Functions
- Environment variables
Parallel Directive

• Prototype:

```c
#include <omp.h>

#pragma omp parallel [clause list]
/* structured block */
```

• Clause list is used for specifying:
  – Conditional parallelization
  – Degree of concurrency
  – Data handling
Clause list

- **if(scalar-expression)**
- **num_threads(integer-expression)**
- **default(shared | none)**
- **private(list)**
- **firstprivate(list)**
- **shared(list)**
- **copyin(list)**
- **reduction(operator: list)**
General Form of Directives

• C/C++

\# pragma omp directive-name [clause[[], clause]...] new-line

• Fortran

!$OMP |C$OMP |*$OMP directive-name [clause[[[,] clause]...] new-line
#include <omp.h>
main () {
    int var1, var2, var3;
    /* Serial code */
    ......

    /* Beginning of parallel section. */
#pragma omp parallel private(var1, var2) shared(var3) {
        /* Parallel section executed by all threads . . . */
        ......
    }

    /* Resume serial code . . . */
    ......
}
Runtime Library Functions

• Execution environment routines
  – Control number of threads and processors;
  – Control and monitoring thread creation

• Lock routines
  – Mutual exclusion

• Timing routines
  – Performance measurement
Execution Environment

• Control the number of threads and processors

#include <omp.h>

void omp_set_num_threads(int num_threads);
int omp_get_num_threads();
int omp_get_max_threads();
int omp_get_thread_num();
int omp_get_num_procs();
int omp_in_parallel();
......
Execution Environment

- Control and monitoring thread creation

```c
#include <omp.h>

void omp_set_dynamic (int dynamic_threads);
int omp_get_dynamic ( );
void omp_set_nested (int nested);
int omp_get_nested ( );
```
Lock Routines

• Mutual exclusion

```c
#include <omp.h>

void omp_init_lock (omp_lock_t *lock);
void omp_destroy_lock (omp_lock_t *lock);
void omp_set_lock (omp_lock_t *lock);
void omp_unset_lock (omp_lock_t *lock);
int omp_test_lock (omp_lock_t *lock);
```
Timing Routines

double omp_get_wtime(void)  
returns elapsed wall clock time in seconds.

double omp_get_wtick(void)  
returns the precision of the timer used by omp_get_wtime.
How to use the clock?

......

double start;
double end;

start = omp_get_wtime();

... work to be timed ...

end = omp_get_wtime();

printf("Work took %f seconds\n", end - start);

......
Environmental Variables

- OMP_NUM_THREADS
- OMP_DYNAMIC
- OMP_NESTED
- OMP_SCHEDULE
- OMP_STACKSIZE
- OMP_WAIT_POLICY
- OMP_MAX_ACTIVE_LEVELS
- OMP_THREAD_LIMIT
How to set environment variable?

- Using Shell command:

  $ export OMP_NUM_THREADS=4
Which one has higher priority?

- `export OMP_NUM_THREAD=8`
- `omp_set_num_threads(4)`
- `#pragma omp parallel num_threads(2)`

Get answer from OpenMP spec.
Example: omp_env.c

• How to set environment variable
  $ export OMP_NUM_THREADS=4

• Use functions
  omp_set_num_threads(8)
  Nthreads = omp_get_num_threads()

• Understand the effectiveness of them
  – ”set” not equal to “create”
  – What is “max”?
  – What is the scope of variable?
Outline

• Fundamentals for parallel programming
• Overview of OpenMP

• **EXAMPLE OF HPC USING OPENMP**
Example 1: Matrix Multiplication:

- Sequential algorithms
- Parallel algorithms
Sequential: which is faster?

1. for (i=0;i<N;i++)
2.     for (j=0;j<N;j++)
3.         for (k=0;k<N;k++)
4.             product[i][j] += a[i][k]*b[k][j];
5.     -----------------------------------------------------
6. for (i=0;i<N;i++)
7.     for (k=0;k<N;k++)
8.         for (j=0;j<N;j++)
9.             product[i][j] += a[i][k]*b[k][j];
10. ------------------------------------------------------
11. for (k=0;k<N;k++)
12.     for (i=0;i<N;i++)
13.         for (j=0;j<N;j++)
14.             product[i][j] += a[i][k]*b[k][j];

(i, j, k)

(i, k, j)

(k, i, j)
Parallel: which way is right?

1. start=omp_get_wtime();
2. // #pragma omp parallel for
3. for (i=0;i<N;i++)
4.     // #pragma omp parallel for
5.         for (k=0;k<N;k++)
6.             // #pragma omp parallel for
7.                for (j=0;j<N;j++)
8.                   product[i][j] += a[i][k]*b[k][j];
9. end=omp_get_wtime();
Example 2: “Hello World”

• This example will show how load is distributed
  • node=?:ppn=?
Example 3: making Pi

• Recipe: Monte Carlo method
• Code:
• Test it:
Questions for Discussion

1. In the examples
   – What is parallel level/granularity
   – What is the coordination enforcement?
   – What is the relation between environment variables and node=?:ppn=? In PBS script?

2. Any more questions/comments?
References

- **OpenMP standard:**
- **OpenMP books:**
  [http://openmp.org/wp/resources/#Books](http://openmp.org/wp/resources/#Books)
Thanks!

Turn in your feedback sheet please.
Next seminar:

PC to HPC: Parallel Computing with MPI