High Performance Computing Systems

- OpenMP

Doug Shook

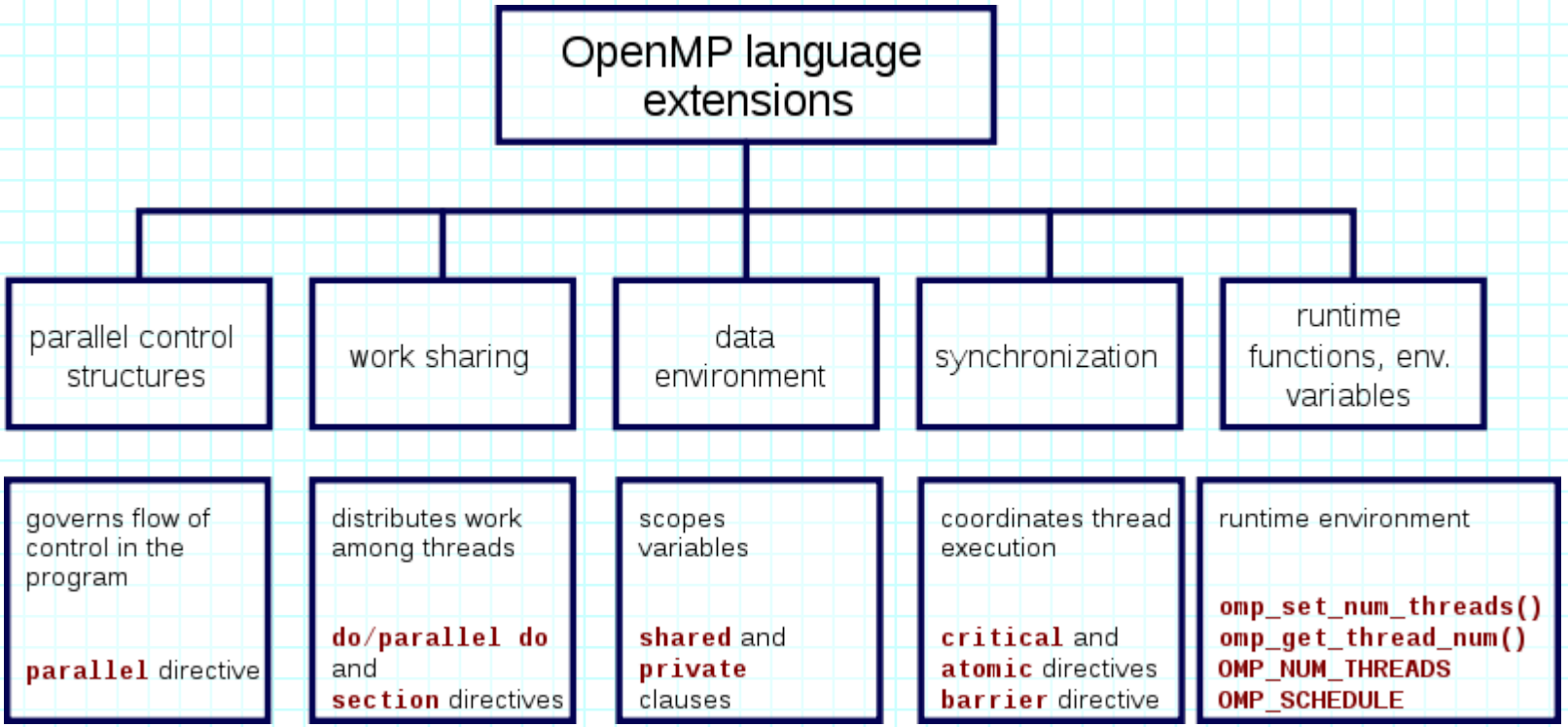
Automatic Parallelization

- Attempts to parallelize serial code
- Problems:
  - Dependence analysis
  - Loops
  - Coordination of global resources

OpenMP

- First developed in 1997
  - Intel, IBM, Cray, AMD, Microsoft, Nvidia, etc.
- Provides a simple API for developing parallel programs
- C, C++, Fortran

OpenMP



- Utilizes master – team execution model

OpenMP - Installation

- Linux / Mac:
  - libopenmp, libopenmp-devel, libgomp, libgomp-devel
- Windows:
  - Can use gcc (mingw) or Visual Studio

OpenMP – Setup / Compilation

- #include <omp.h>
- gcc -fopenmp
- Set appropriate environment variables

OpenMP – Environment Variables

- OMP_NUM_THREADS
- OMP_THREAD_LIMIT
- OMP_PROC_BIND
- OMP_NESTED
- OMP_MAX_ACTIVE_LEVELS
- OMP_WAIT_POLICY

OpenMP - directives

- Prefixed by #pragma omp
  - Parallel
  - For
  - Section

OpenMP – Hello world

- Grab the code from the website
  - Compile and execute it
  - Change the number of threads

OpenMP – For

- Used for loops
  - Schedule
  - Shared
  - Private
  - Reduction
- Vector addition example

OpenMP – Sections

- Used to parallelize non-iterative code
  - Private
  - Reduction
- Sections example
- What happens if number of threads != number of sections?
- Which thread executes which section?

OpenMP – Exercises

- Compile and run workshare1.c
  - Run it a few more times. What do you notice?
  - Change the scheduling to be static and run it again. What do you notice?
  - Rerun it again, what do you notice?
  - Play around with the chunk size
    - What do you notice?
- Compile and run workshare2.c
  - Run this code several times. What do you notice?

OpenMP – Synchronization

- Several ways to synchronize threads:
  - Master
  - Critical
  - Barrier

- Critical example

OpenMP – Reduction

- Compiles all of the values from each thread into a single result while performing a given operation
- Reduction example

Nested Parallelism

- Parallel section within a parallel section
  - Each can spawn up to OMP_NUM_THREADS threads
- Be careful with this!
  - Useful in some situations, easy to go overboard

Random Number Generation

- Problem: rand() is not thread-safe
  - It stores a state
  - Effects?
- Solutions
  - Put RNG in critical section
  - Generate RNG for each thread with its own seed
  - Other thread-safe versions of rand()
    - drand48_r()
    - rand_r()

Scalasca

- Relatively new profiler
  - Specifically for parallel applications
- Use scalasca to compile/instrument your code
  scalasca -instrument
- Run program with scalasca
  OMP_NUM_THREADS=4 scalasca -analyze
- Examine the result
  - scalasca -examine
- Demo

OpenMP - Exercises

- Assume the chunksize is 1 in this example. Why is this a bad idea? What would be a better chunksize?
  - Hint: consider the behavior of the cache

```
for (i=0; i<N; i++) {
    a[i] = // some calculation
}
```

OpenMP – Exercises

- Revisit hw1 and parallelize it using OpenMP
  - Record execution time with OMP_GET_WTIME
  - Try to find the fastest implementation possible by altering the following:
    - Number of threads
    - Chunksize
    - Scheduling
- What is the speedup of your solution? The efficiency?
- Try out Scalasca