OpenMP 1

Victor Eijkhout & Cyrus Proctor

PCSE 2015

Parallel region

```
#pragma omp parallel [clauses....]
```

- `private` and `shared` control thread data
- `num_threads` use value different from default
- `if` conditional use of parallelism

Parallel region and worksharing

```
#pragma omp parallel
{
  // parallel code here
}
```

- Parallel region: create team
- Worksharing construct: distribute work

Work sharing

Work sharing constructs

construct	description
do/for	loop iterations
sections	discrete code sections
single	only one thread
workshare	(Fortran only) unit of work

Parallel loop

```
#pragma omp parallel
#pragma omp for
  for (i=0; i<N; i++) {
    ....
  }
```

- Loop iterations are divided over the thread team
- Loop variable is automatically private
- Many ways of dividing the iterations.
- Some restrictions on loop variable.
- Special case: reductions.

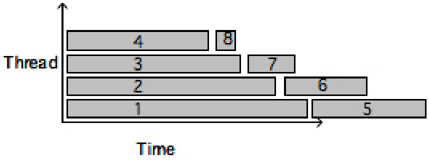
Often abbreviated: `omp parallel for`

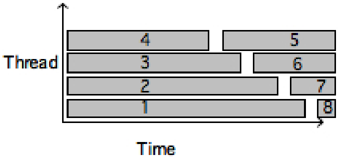
Iteration scheduling

Add a scheduling clause:

- `schedule(static)` n iterations divided in blocks of n/p.
- `schedule(static,m )` iterations divided in blocks of m ('chunk size'), assigned cyclically.
- `schedule(dynamic)` single iterations, assigned whenever a thread is idle
- `schedule(dynamic,m)` blocks of m iterations, assigned whenever a thread is idle
- `schedule(guided)` decreasing size blocks
- `schedule(auto)` leave it up to compiler/runtime
- `schedule(runtime)` using environment variable `OMP_SCHEDULE`

static vs dynamic





Barriers

- No barrier at the start
- Implicit barrier at the end
- No barrier with `nowait`

```
#pragma omp parallel
{
  x = local_computation()
#pragma omp for nowait
  for (i=0; i<N; i++) { .... }
#pragma omp for
  for (i=0; i<N; i++) { .... }
}
```

Data scope

- Data can be shared: from the master thread
- Data can be private: every thread its own copy
- How are private variables initialized?
- Private variables disappear after a parallel region

Data scope

- Loop variable is private
- Data allocated in parallel region is private
- `private` make private copy of shared variable
- `firstprivate` like private, but initialized to shared value
- `lastprivate` private copy of shared variable, copied out
- `default(none)` requires explicit private/shared declarations

lastprivate

```
#pragma omp parallel for \
        lastprivate(tmp)
for (i=0; i<N; i+) {
  tmp = ......
  x[i] = .... tmp ....
}
..... tmp ....
```

- `tmp` is temporary, should be private
- final value is used after the loop: use `lastprivate`
- this can also be used for the loop variable.

Reduction

```
#pragma omp parallel for \
        reduction(+:s)
  for (i=0; i<N; i++)
    s += f(i);
```

- Reductions are atomic operations
- Can be solved by private variable per thread
- `reduction` clause is shorthand for all that