OpenMP

Intro

- A collection of compiler directives, library routines, and environment variables for specifying shared-memory parallelism.
- There are versions of the standard for Fortran and C/C++.
- See www.openmp.org.
- The directives are ignored by the compiler unless a command-line option is specified.
- Directives allow work sharing, synchronization, and sharing and privatizing of data.
- Other aspects of the model:
  - User-defined parallelism
  - SPMD
  - Fork/join model. Only a master thread is executing when outside a parallel region.
  - parallel directive causes multiple threads to be started (or continued), each executing all or a part of the specified block.

Work-Sharing Constructs

- The parallel construct.

```c
#pragma omp parallel [clause[ clause ...]] new-line
structured-block
```

where clause is one of the following:

```c
if(scalar-expression)
private (list)
firstprivate (list)
default (shared | none)
shared (list)
copyin (list)
reduction (operator: list)
```

- The for construct.

```c
#pragma omp for [clause[ clause ...]] new-line
for-loop
```

where clause is one of the following:
private (list)
firstprivate (list)
lastprivate (list)
reduction (operator: list)
ordered
schedule(kind [, chunk_size])
nowait

• The sections construct.

#pragma omp sections [clause[ clause ...]] new-line
{
    [#pragma omp section new-line]
    structured-block
    [#pragma omp section new-line]
    structured-block

    .
    .
    .}
}

where clause is one of the following:

private (list)
firstprivate (list)
lastprivate (list)
reduction (operator: list)
nowait

• The single construct.

#pragma omp single [clause[ clause ...]] new-line
structured-block

where clause is one of the following:

private (list)
firstprivate (list)
nowait

Synchronization Constructs

#pragma omp master
#pragma omp critical

#pragma omp barrier

#pragma omp atomic

#pragma omp flush

#pragma omp ordered

Runtime Library Functions

omp_set_num_threads()
omp_get_num_threads()
omp_get_max_threads()
omp_get_thread_num()
omp_get_num_procs()

omp_init_lock()
omp_destroy_lock()
omp_set_lock()
omp_unset_lock()
omp_test_lock()

Environment Variables

OMP_SCHEDULE

OMP_NUM_THREADS

OMP_DYNAMIC

OMP_NESTED

Scheduling Parallel Loops

- static
  
  Each thread gets chunk_size contiguous iterations.
#pragma omp parallel for schedule(static)
for (i=0; i<n; i++)
  invariant_amount_of_work(i);
}

- **dynamic**

Each thread gets the next chunk_size contiguous iterations, assigned dynamically.

```c
#pragma omp parallel for schedule(dynamic)
for (i=0; i<n; i++)
  unpredictable_amount_of_work(i);
```

- **guided**

First thread gets n/p iterations, then reduce size of assignment from there, until get down to chunk_size.

```c
#pragma omp parallel
{
  #pragma omp sections nowait
  {
    // ...
  }
  
  #pragma omp parallel for schedule(guided)
  for (i=0; i<n; i++) {
    invariant_amount_of_work(i);
  }
}
```

- **runtime**

Check environment variables.