Work-sharing constructs

- #pragma omp for [clause …]
- #pragma omp section [clause …]
- #pragma omp single [clause …]

- The work is distributed over the threads
- Must be enclosed in parallel region
- No implied barrier on entry, implied barrier on exit (unless specified otherwise)

Work Sharing Constructs: for Directive

Purpose:

- It specifies that the iterations of the loop immedietely following it must be executed in parallel by the team.
- This assumes a parallel region has already been initiated, otherwise it executes in serial on a single processor.

Work Sharing Constructs: for Directive

```
#include <omp.h>
#define N 1000
#define CHUNKSIZE 100

main(int argc, char *argv[])
{
    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;
```

Work Sharing Constructs: for Directive

```
    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 region */
}
```

nowait: If specified, then threads do not synchronize at the end of the parallel loop.

Work Sharing Constructs: for Directive

- Simple vector-add program
  - Arrays a,b,c and variable N will be shared by all threads
  - Variable I will be private to each thread; each thread will have its own unique copy.
  - The iterations of the loop will be distributed dynamically in CHUNK sized pieces.
  - Threads will not synchronize upon completing their individual pieces of work(nowait).

Work Sharing Constructs: Sections Directive

Purpose:

- It is a non-iterative work-sharing construct. It specifies that the enclosed section(s) of code are to be divided among the threads in the team.
- Independent SECTION directives are nested within a SECTIONS directive. Each SECTION is executed once by a thread in the team.
- Different sections may be executed by different threads. It is possible for a thread to execute more than one section if it is quick enough and the implementation permits such

Work Sharing Constructs: Sections Directive

```
#include <omp.h>
#define N 1000
main(int argc, char *argv[])
{
    int i, chunk;
    float a[N], b[N], c[N],d[N];

    /* Some initializations */
    for (i=0; i<N; i++)
    {
        a[i] =  i * 1.5;
        b[i] = i+22.35;
    }
```

Work Sharing Constructs: Sections Directive

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

}  /* end of parallel region */
}
```

Work Sharing Constructs: Single Directive

Purpose:

- The SINGLE directive specifies that the enclosed code is to be executed by only one thread in the team.
- May be useful when dealing with sections of code that are not thread safe(such as I/O).

Work Sharing Constructs: Single Directive

#pragma omp single [clause ...] newline

private (list)

irstprivate(list)

nowait

structured_block

Clauses: Threads in the team that do not execute the SINGLE directive, wait at the end of the enclosed code block, unless a nowait clause is specified

Restrictions: It is illegal to branch into or out of a SINGLE block.

References

1.www.cs.fsu.edu/~xyuan/cda5125/lect9_openmp.ppt
2.http://eclass.uoa.gr/modules/document/file.php/D186/%CE%8E%CE%BB%CE%B7%202011-12/PachecoChapter_5.pdf
3. www.cs.utah.edu/~mhall/cs4961f11/CS4961-L5.ppt
4.sddsd