

# Advanced OpenMP

## Nested Parallelism



# Nested parallelism

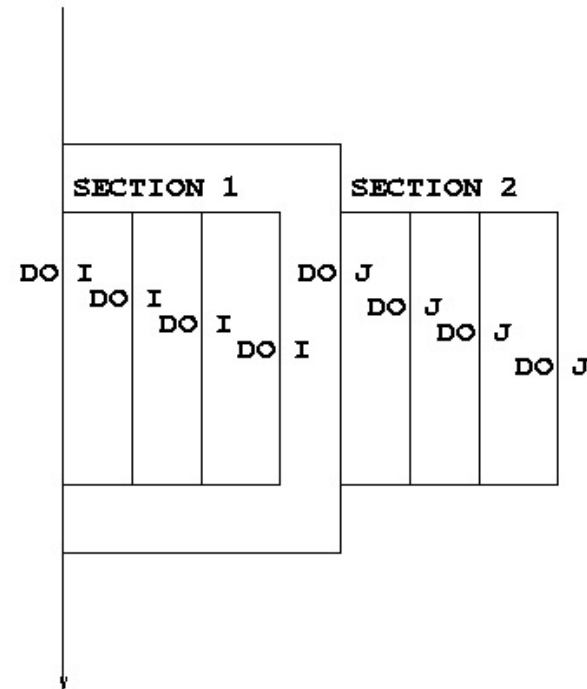


- Nested parallelism is supported in OpenMP.
- If a `PARALLEL` directive is encountered within another `PARALLEL` directive, a new team of threads will be created.
- This is enabled with the **`OMP_NESTED`** environment variable or the **`OMP_SET_NESTED`** routine.
- If nested parallelism is disabled, the code will still execute, but the inner teams will contain only one thread.

# Nested parallelism (cont)

Example:

```
!$OMP PARALLEL PRIVATE(myid)
myid = omp_get_thread_num()
if (myid .eq. 0) then
!$OMP PARALLEL DO
    do i = 1,n
        x(i) = 1.0
    end do
elseif (myid .eq.1) then
!$OMP PARALLEL DO
    do j = 1,n
        y(j) = 2.0
    end do
endif
!$OMP END PARALLEL
```



## Nested parallelism (cont)



- Not often needed, but can be useful if the outer level does not contain enough parallelism
- Note: nested parallelism turns out to be hard to implement correctly without impacting performance.
  - usually disabled by default
  - don't enable nested parallelism unless you are using it!

## Controlling the number of threads

- Can use the environment variable

`export OMP_NUM_THREADS=2,4`

- Will use 2 threads at the outer level and 4 threads for each of the inner teams.
- Can use `omp_set_num_threads()` or the `num_threads` clause on the parallel region.

## omp\_set\_num\_threads()



- Useful if you want inner regions to use different numbers of threads:

```
CALL OMP_SET_NUM_THREADS(2)
!$OMP PARALLEL DO
    DO I = 1,4
CALL OMP_SET_NUM_THREADS(innerthreads(i))
!$OMP PARALLEL DO
    DO J = 1,N
        A(I,J) = B(I,J)
    END DO
END DO
```

- The value set overrides the value(s) in the environment variable OMP\_NUM\_THREADS

## NUM\_THREADS clause

- Another way to control the number of threads used at each level is with the NUM\_THREADS clause:

```
!$OMP PARALLEL DO NUM_THREADS(2)
    DO I = 1,4
!$OMP PARALLEL DO NUM_THREADS(innerthreads(i))
        DO J = 1,N
            A(I,J) = B(I,J)
        END DO
    END DO
```

- The value set in the clause overrides the value in the environment variable OMP\_NUM\_THREADS and that set by **omp\_set\_num\_threads()**

## More control....

- Can also control the maximum number of threads running at any one time.

```
export OMP_THREAD_LIMIT=64
```

- ...and the maximum depth of nesting

```
export OMP_MAX_ACTIVE_LEVELS=2
```

or call

```
omp_set_max_active_levels()
```



# Utility routines for nested parallelism



- `omp_get_level()`
  - returns the level of parallelism of the calling thread
  - returns 0 in the sequential part
- `omp_get_active_level()`
  - returns the level of parallelism of the calling thread, ignoring levels which are inactive (teams only contain one thread)
- `omp_get_ancestor_thread_num(level)`
  - returns the thread ID of this thread's ancestor at a given level
  - ID of my parent:  
`omp_get_ancestor_thread_num(omp_get_level()-1)`
- `omp_get_team_size(level)`
  - returns the number of threads in this thread's ancestor team at a given level

# Nested loops

- For perfectly nested rectangular loops we can parallelise multiple loops in the nest with the **collapse** clause:

```
#pragma omp parallel for collapse(2)
for (int i=0; i<N; i++) {
    for (int j=0; j<M; j++) {
        . . . . .
    }
}
```

- Argument is number of loops to collapse starting from the outside
- Will form a single loop of length NxM and then parallelise and schedule that.
- Useful if N is O(no. of threads) so parallelising the outer loop may not have good load balance
- More efficient than using nested teams

## Synchronisation in nested parallelism

- Note that barriers (explicit or implicit) only affect the innermost enclosing parallel region.
- No way to have a barrier across multiple teams
- In contrast, critical regions, atomics and locks affect all the threads in the program
- If you want mutual exclusion within teams but not between them, need to use locks (or atomics).

# Reusing this material



This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

<https://creativecommons.org/licenses/by-nc-sa/4.0/>

This means you are free to copy and redistribute the material and adapt and build on the material under the following terms: You must give appropriate credit, provide a link to the license and indicate if changes were made. If you adapt or build on the material you must distribute your work under the same license as the original.

Note that this presentation contains images owned by others. Please seek their permission before reusing these images.

