OpenMP - Worksharing directives

Matic Brank

LECAD laboratory, FS UL

OpenMP – **Exercise shared vs. private**

- Goal: To see difference between private and shared variable.
- Instructions:
  - Open file *exercise_shared_vs_private.c*
  - Follow the instruction on the comments:
    - After variable `int thr_num`, set number of used threads to 4
    - Define a new variable named `list_of_thread_ids`
    - Initialize all 4 elements in the array to 0
    - For parallel pragma, define private variables `thr_num` and `list_of_thread_ids`
    - Inside parallel pragma, get ID number of each thread and assign it to `thr_num`
  - Run the code:
    - What is the output?
    - What are the elements inside `list_of_thread_ids` after execution of parallel pragma?
  - Modify line `#pragma omp parallel…` and change the type of `list_of_thread_ids` from private to shared

OpenMP – **Exercise shared vs. private**

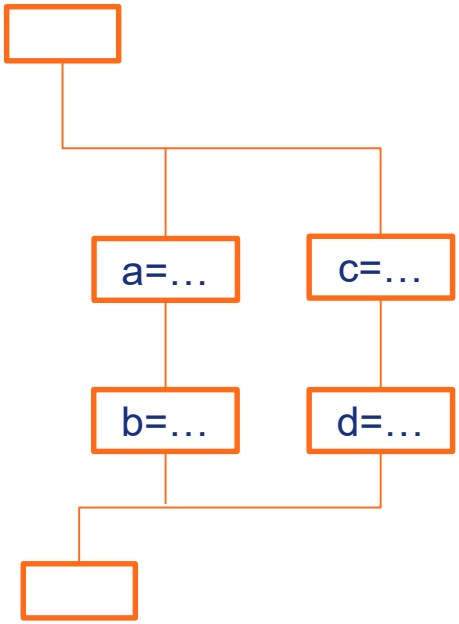
- What is the new output?
- What is the difference between `list_of_thread_ids` being private or shared?

OpenMP – Worksharing constructs

- They divide the execution of code region among the members of the team
- Constructs do not launch new threads
- They are enclosed dynamically within the parallel region
- Examples:
  - `sections`
  - `for`
  - `task`
  - `single`
- No barrier defined on entry

OpenMP – Sections construct

- When using `sections` construct, multiple blocks of code are executed in parallel



Sections in parallel

- C/C++:

```
#pragma omp parallel
{
#pragma omp sections
{{a=…; b=…;}
#pragma omp section
{
        c=…;
        d=…;
}
}// end of sections
}// end of parallel
```

OpenMP – For construct

- C/C++:

```
#pragma omp for [clause[[,]clause]…]
```

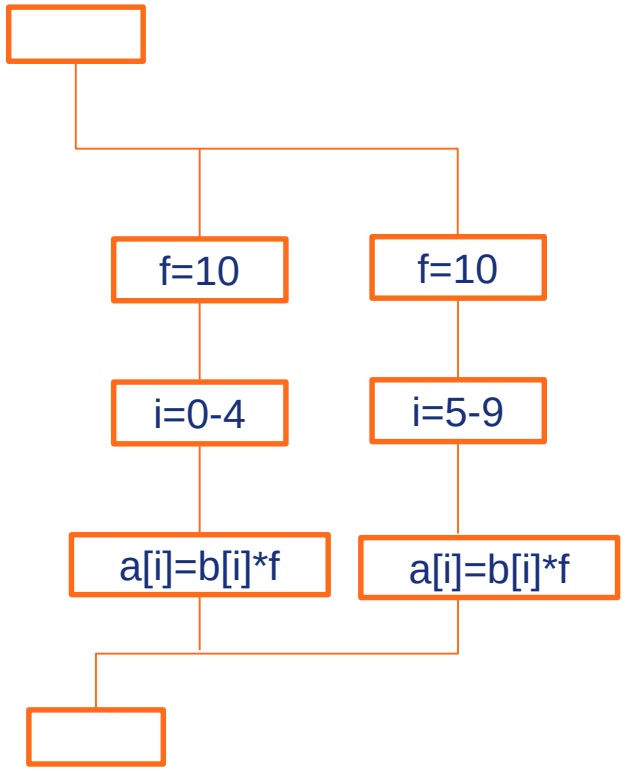
- Corresponding for loop must have a *canonical* shape, where an iterator (`i`) must not be modified in the loop body:

```
for (int i=it; i<M; i++)
```

- Clauses:
  - `private (list)`
  - `schedule` -> classifies how the iterations of loops are divided among the threads
  - `collapse (n)` -> the iterations of n loops are collapsed into one larger iteration space

OpenMP – For construct

- Private variable f is fixed in each thread, list a is updated in parallel.



- C/C++:

```
#pragma omp parallel private(f)
{
f=10;
#pragma omp for
for (i=0; i<10; i++)
a[i] = b[i]*f;
} /* omp end parallel */
```

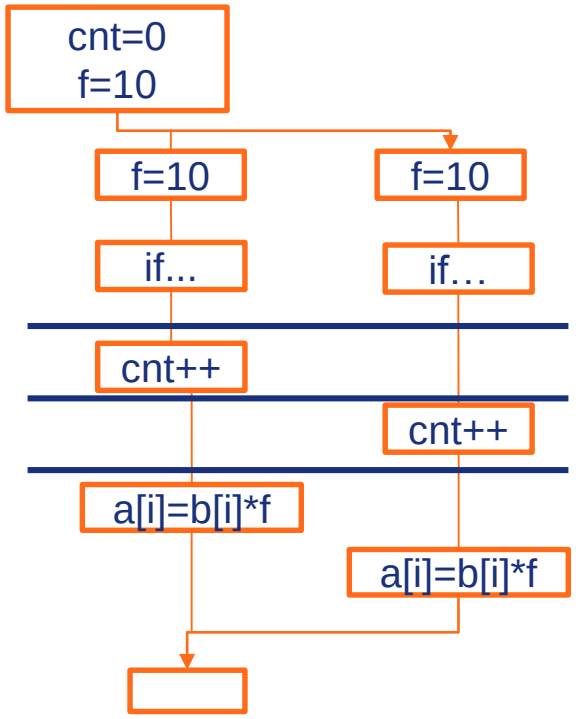
OpenMP – Synchronisation

- Implicit barrier
  - It represents a barrier for beginning and end of parallel constructs, as well as all other control constructs
  - Implicit synchronization can be removed with `nowait` clause
- Explicit barrier
  - Can be achieved by using `critical` clause
  - Code, enclosed in `critical` clause is executed by all threads, but is restricted to only one thread at the time
  - `critical` clause in C/C++:

```
#pragma omp critical [(name)]
```

OpenMP – Synchronisation - critical clause

- Private variable f is fixed in each thread, list a is updated in parallel.



Due to `critical` clause only one thread at a time is executed for `cnt` variable

- C/C++:

```
cnt = 0;
f=10;
#pragma omp parallel
{
#pragma omp for
      for (i=0; i<10; i++) {
         if (b[i] == 0) {
            #pragma omp critical
            cnt ++;
         } /* endif */
         a[i] = b[i]*f;
     } /* end for */
} /*omp end parallel */
```

OpenMP – **Exercise 3**

- Goal: To use a worksharing construct `for` and `critical` directive. To use runtime library calls, conditional compilations, environment variables and parallel regions by calculating pi constant.
- Explanation of algorithm in script *e3.c*
  - Value π is calculated by solving integral
    - $\pi = \int_0^1 \frac{4}{1+x^2} dx$
  - Integral is solved by Riemann sum
    - $\pi = 4 \sum_{i=0}^{n-1} f\left(x_i + \frac{w}{2}\right) w$

OpenMP – **Exercise 3**

- Instructions
  - Compile file *e3.c* and run it. What is the output. Is the value of pi correct?
  - Open file *e3.c* and move to `/* Begin of SPACE for third exercise */`.
  - Add parallel region and parallel `for` directive in *e3.c* and compile it
  - Set environment variable OMP_NUM_THREADS to 2 and run the file.
    - The result is wrong
  - Set environment variable OMP_NUM_THREADS to 12 and run again
    - The result is wrong
  - Add private (x) clause in *e3.c* and compile
  - Set environment variable OMP_NUM_THREADS to 2 and run
    - Should be still incorrect

OpenMP – **Exercise 3**

- Set environment variable `OMP_NUM_THREADS` to 12 and run
  - Still incorrect
- Set environment variable `OMP_NUM_THREADS` to 2 and add `critical` directive in *e2.c* before the `sum` variable in `for` loop and compile again
  - What is the value of pi? (should be correct)
  - What is the CPU time?
- Set environment variable `OMP_NUM_THREADS` to 12 and run
  - What is the CPU time? (it should take longer)
- How to optimize the code? Try to modify the code and move the `critical` directive outside `for` loop to decrease computational time.

THANK YOU FOR YOUR ATTENTION

www.prace-ri.eu