Parallel Computing with GPUs

Advanced OpenMP
Part 3 – Nesting and Summary

The University Of Sheffield.

Dr Paul Richmond

http://paulrichmond.shef.ac.uk/teaching/COM4521/

This Lecture (learning objectives)

- Nesting
  - Operate on nested loops using OpenMP
  - Compare the performance implications of different approaches for nesting
- Summary
  - Classify permitted use of the various OpenMP clauses

Nesting

- Consider the following example…
  - How should we parallelise this example?

```
for (i = 0; i < OUTER_LOOPS; i++){
    for (j = 0; j < INNER_LOOPS; j++){
        printf("Hello World (Thread %d)\n", omp_get_thread_num());
    }
}
```

Nesting

- Consider the following example…
  - How should we parallelise this example?

```
#pragma omp parallel for
for (i = 0; i < OUTER_LOOPS; i++){
    for (j = 0; j < INNER_LOOPS; j++){
        printf("Hello World (Thread %d)\n", omp_get_thread_num());
    }
}
```

- What if OUTER_LOOPS << number of threads
  - E.g. `OUTER_LOOPS = 2`

Nesting

- We can use parallel nesting
  - Nesting is turned off by default so we must use `omp_set_nested()`
  - When inner loop is met each outer thread creates a new team of threads
  - Allows us to expose higher levels of parallelism
    - *Only useful when outer loop does not expose enough parallelism*

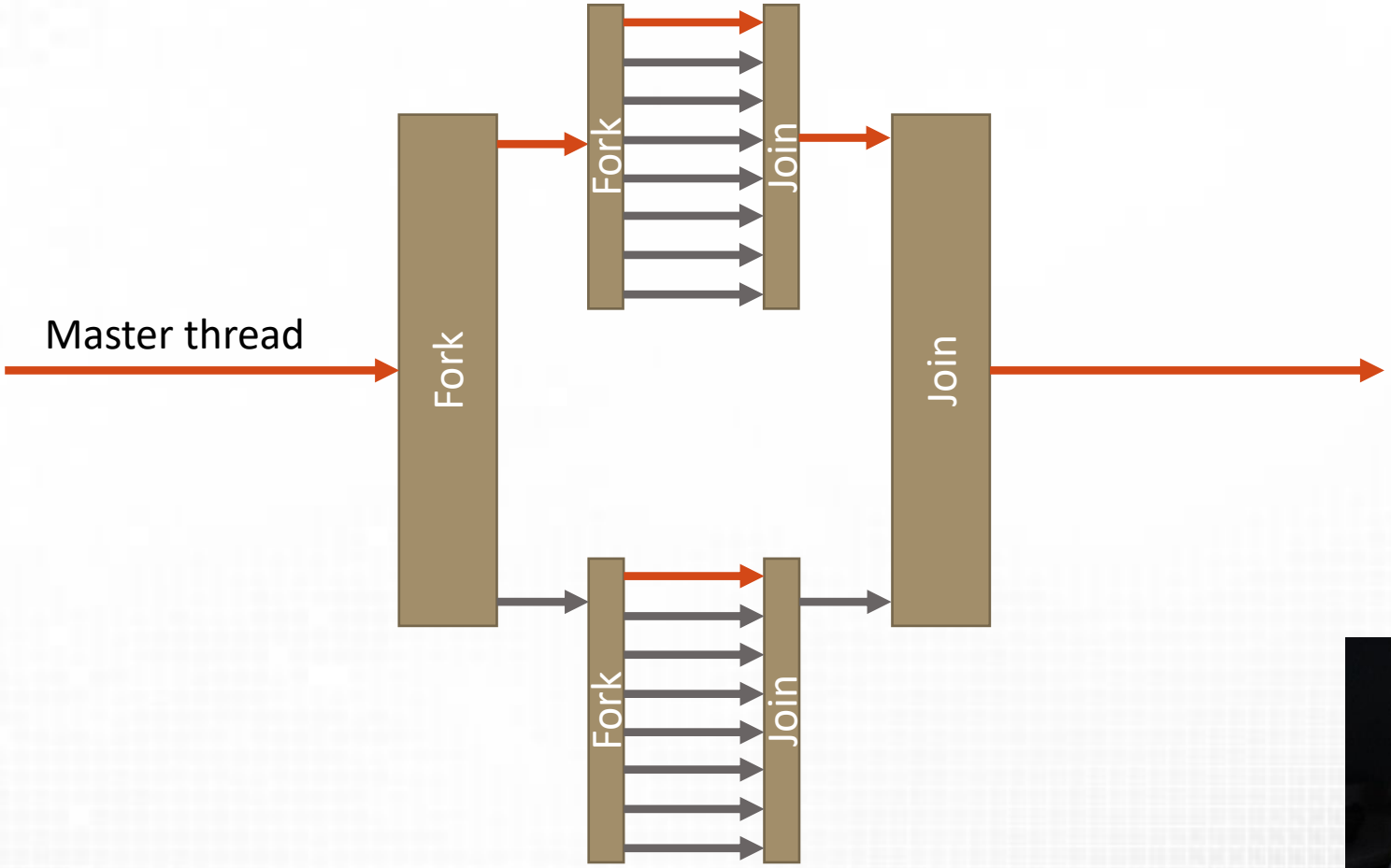
```
omp_set_nested(1);

#pragma omp parallel for
  for (i = 0; i < OUTER_LOOPS; i++){
    int outer_thread = omp_get_thread_num();
      #pragma omp parallel for
        for (j = 0; j < INNER_LOOPS; j++){
          int inner_thread = omp_get_thread_num();
          printf("Hello World (i T=%d j T=%d)\n", outer_thread, inner_thread);
        }
  }
```

```
Hello World (i T=0 j T=0)
Hello World (i T=0 j T=1)
Hello World (i T=0 j T=3)
Hello World (i T=1 j T=2)
Hello World (i T=1 j T=1)
Hello World (i T=1 j T=0)
Hello World (i T=0 j T=2)
Hello World (i T=1 j T=3)
```

Nesting Fork and Join

- Every parallel directive creates a fork (new team)
  - In this case `omp parallel` is used to fork a new parallel region



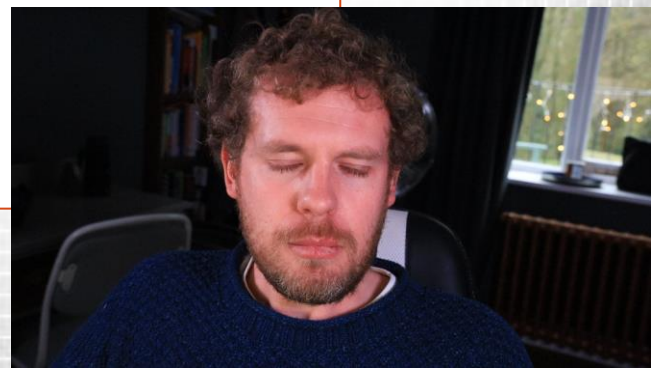
Collapse

- Only available in OpenMP 3.0 and later (not VS2017)
  - Can automatically collapse multiple loops
  - Loops must not have statements or expressions between them

```
#pragma omp parallel for collapse(2)
  for (i = 0; i < OUTER_LOOPS; i++){
    for (j = 0; j < INNER_LOOPS; j++){
      int thread = omp_get_thread_num();
      printf("Hello World (T=%d)\n", thread);
    }
  }
```

Work around…

```
#pragma omp parallel for
  for (i = 0; i < OUTER_LOOPS* INNER_LOOPS; i++){
    int thread = omp_get_thread_num();
    printf("Hello World (T=%d)\n", thread);
  }
```



Clauses usage summary

Clause	Directive: #pragma omp …					
	parallel	for	sections	single	parallel for	parallel sections
if						
private						
shared						
default						
firstprivate						
lastprivate						
reduction						
schedule						
nowait						

Performance

- Remember ideas for general C performance
  - Have good data locality (good cache usage)
  - Combine loops where possible
- Additional performance criteria
  - Minimise the use of barriers
    - Use `nowait` but only if it is safe to do so!
  - Especially minimise critical sections
    - High overhead. Can you use reduction or atomics?
    - Benchmark to find best solution
  - Experimentally try out different scheduling approaches and chunk sizes

Summary

- Nesting
  - Operate on nested loops using OpenMP
  - Compare the performance implications of different approaches for nesting
- Summary
  - Classify permitted use of the various OpenMP clauses

- Further Reading: https://software.intel.com/en-us/articles/32-openmp-traps-for-c-developers