OpenMP Loop Parallelism (2A)

- Loop
-

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice and Octave.

Distributing the work over the threads

In a **team** of **threads**,
initially there will be replicated execution;

a **work sharing construct** divides
available parallelism over the threads.

OpenMP uses **teams** of **threads**,
and inside a **parallel region**
the **work** is distributed over the **threads**
with a **work sharing construct**.

threads can access **shared data**,
and they have some **private data**.

https://pages.tacc.utexas.edu/~eijkhout/pcse/html/omp-parallel.html

Work Sharing Constructs

sections Construct – structured block

single Construct – only one of the threads

workshare Construct – a separate units of work

https://pages.tacc.utexas.edu/~eijkhout/pcse/html/omp-parallel.html

workshare construct

- divides the execution of
  the enclosed structured block
  into separate units of work

- causes the threads of the team
  to share the work

- each unit is executed
  only once by one thread,
  in the context of its implicit task.

https://pages.tacc.utexas.edu/~eijkhout/pcse/html/omp-parallel.html

workshare construct

A worksharing directive (!) which allows parallelisation of Fortran 90 array operations, WHERE and FORALL constructs.

```
!$omp workshare
    structured-block
!$omp end workshare [nowait]
```

https://www.openmp.org/spec-html/5.0/openmpsu39.html#x61-1170002.8.3

Clauses (6)

nowait

Use this clause to avoid the **implied barrier**
at the end of the **sections** directive.

This is useful if you have multiple independent
work-sharing sections within a given parallel region.

Only one **nowait** clause can appear
on a given **sections** directive.

https://www.ibm.com/docs/en/xl-c-aix/13.1.2?topic=processing-pragma-omp-section-pragma-omp-sections

Clauses (13)

Ordered

During execution of an iteration of a loop or a loop nest within a loop region, the executing thread must not execute more than one ordered region which binds to the same loop region. As a consequence, if multiple loops are associated to the loop construct by a collapse clause, the ordered construct has to be located inside all associated loops.

Specify this clause if an ordered construct is present within the dynamic extent of the omp for directive.

https://www.ibm.com/docs/en/xl-c-aix/13.1.2?topic=processing-pragma-omp-section-pragma-omp-sections

Implicit task (1)

In addition to **explicit tasks** specified using the **task** directive,
the OpenMP specification version 3.0 introduces
the notion of **implicit tasks**.

An **implicit task** is a task generated
- by the **implicit parallel region**,
- when a **parallel construct** is encountered during execution.

The **code** for each **implicit task** is
the code inside the **parallel construct**.

Each **implicit task** is
- assigned to a different **thread** in the **team** and is **tied**;
- always executed from beginning to end
by the **thread** to which it is initially assigned.

https://docs.oracle.com/cd/E19205-01/820-7883/6nj43o69j/index.html

Implicit task (2)

All **implicit tasks** generated
when a **parallel construct** is encountered
are guaranteed to be complete
when the **master thread** exits the **implicit barrier**
at the end of the **parallel region**.

all **explicit tasks** generated within a **parallel region**
are guaranteed to be complete
on exit from the next **implicit** or **explicit barrier**
within the **parallel region**.

https://docs.oracle.com/cd/E19205-01/820-7883/6nj43o69j/index.html

Implicit task (3)

When an **if clause** is present on a **task construct**
and the value of the scalar-expression evaluates to false,
the thread that encounters the task must immediately execute the task.

The **if clause** can be used to avoid the **overhead** of
generating many **finely grained tasks** and
placing them in the **conceptual pool**.

https://docs.oracle.com/cd/E19205-01/820-7883/6nj43o69j/index.html

Implicit barrier

Implicit BarriersSeveral OpenMP* constructs have implicit barriers

- parallel
- for
- single

Unnecessary barriers hurt performance

- Waiting threads accomplish no work!

Waiting threads accomplish no work!
Suppress implicit barriers, when safe, with the nowait

https://www.intel.com/content/dam/www/public/apac/xa/en/pdfs/ssg/Programming_with_OpenMP-Linux.pdf

References

[1] en.wikipedia.org
[2] M Harris, http://beowulf.lcs.mit.edu/18.337-2008/lectslides/scan.pdf