**Lecture 6\_1**

Sections- allows different threads to carry out different work. In contrast to for where threads perform same tasks because they are running the same code block but on different data, which is often referred to as data parallelism. Allows users to specify different code blocks for each thread explicitly.

Each section will be given to a thread.

No shared clause with section construct

If there are more threads than the number of section code blocks, only one thread will be working whilst the other threads will be idle. If there are more sections than threads, then some threads will execute more than one code block

This construct must be enclosed in a parallel region.

Graphical user interface, text, application

Description automatically generated

Quicksort parallel coded eg has nested parallelism which degrades parallelism very badly. To deal with it, need to stop parallelism at some stage to not have parallel creation overhead.

Nested parallelism is off by default. We can enable it with:



Or use library function to enable it.



1 means it is true

0 will disable the nested parallelism

By having more number of threads than needed used, it increases overhead

Task construct achieves similar goal as section construct but uses a different approach

Tasks allow us to parallelise irregular problems including unbounded loops, recursive algorithms etc.

Clauses are not exhausted for the tasks in the list

Untied- tells runtime system if task is deferred/ undeferred

No last private in task. Not valid.

Graphical user interface, text, application, email

Description automatically generated

Single construct guarantees that only 1 thread does the job. All threads do the task unless you specify how many threads. There is always an implicit barrier at the end of single constructs. For the tasks one though you need to make an explicit barrier.

**Lecture 6\_2**

Tied means task is deferred- not necessarily executed immediately

Graphical user interface, text, application

Description automatically generated

With the tasks the order in which they run are undeterministic, therefore the output could be a soccer match or a match soccer

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Task wait barrier ensures that the tasks are completed before the things after it

Task construct clauses:

If clause: create from generating small tasks

Final clause- prevent from generating too many tasks than necessary

Text

Description automatically generated

Using this if clause makes certain task happens immediately, but it creates overhead as is no better than using a single thread.

For task constructs, the data scoping is default **firstprivate**