**THIS CONTAINS WEEK 4 AND 5**

Compiler directive- guides compilers to do what you want e.g. pragma in C/C++

Runtime library functions- does stuff only at runtime, some of them play the role of interface between openMP and OS supported threading

Environment variables-control openMP execution environment

**If there are more threads then the number of cores, then multiple threads will be assigned to a single core.**

UMA- uniform memory access system. Also known as SMP symmetric multi-processer

NUMA- non-uniform memory access system

Clauses- additional optional component of a pragma. Used to specify conditional parralelisation, number of threads and data scope handling.

Construction of openMP:

1. Shared memory multiprocessor hardware level
2. Multithreading support os level
3. Openmp api
4. Application

Default num of threads= num of cores on machine

Four threads: numbered 0-3

Num\_threads(number of threads created in the parallel region)

Clauses used for data scoping:

Private- indicates the set of variables specified in list are local to each thread. Each thread has its own copy of each variable in this list., only exists in scope of parallel region

Shared- all variables in list are shared among all the threads. There will only be one copy for each of the variables in the list

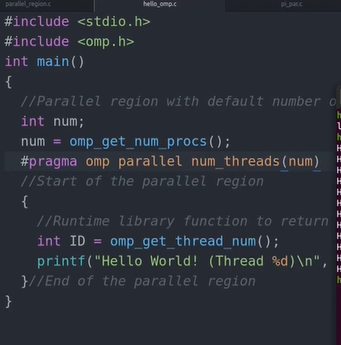
Firstprivate- similar to private except initial values of listed variables are initialised to corresponding values just before entering the parallel region

Default- can take either shared or value none. If this clause is used, then the variables in the data scoping are not specified explicitly either using private, first private shared or others, will take the default setting as specified in clause default. None implies state of each variable used in a thread must be explicitly specified E.g. if we take the value of shared, the data scoping of those variables that are not explicitly specified will take shared scoping

First private- value becomes initialised before entering parallel space

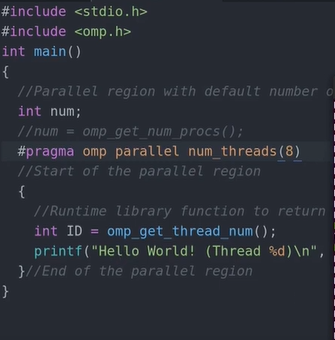
Private- value is undetermined despite having been initialised at the start

Get number of cores: omp\_get\_num\_procs()

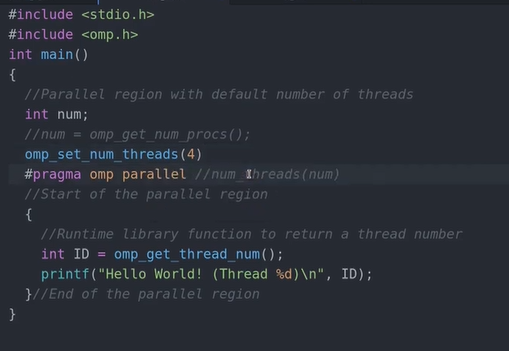


Three ways to set number of threads:

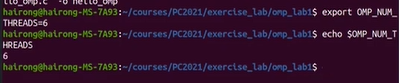
Use clause num\_threads



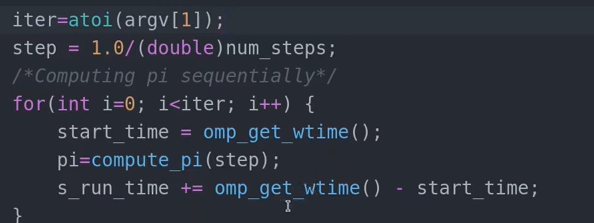
Omp\_set\_num\_threads() can set the number of threads as well.



Can set env variable to change number of threads as well:

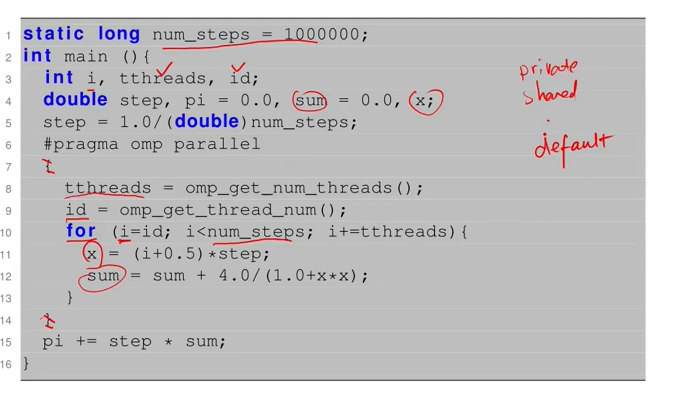


Getting elapsed time

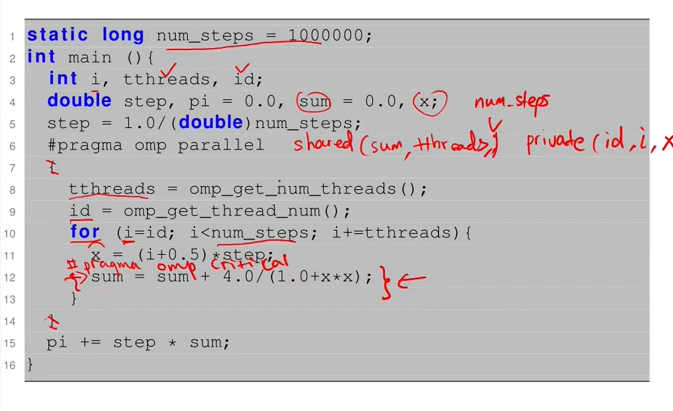


Cache value on one processor must be updated on other processors- cache coherence

If no clauses are specified in code, by default, the variables are shared amongst all the threads. The i in the loop will be private (all loop indices), but all others will be shared



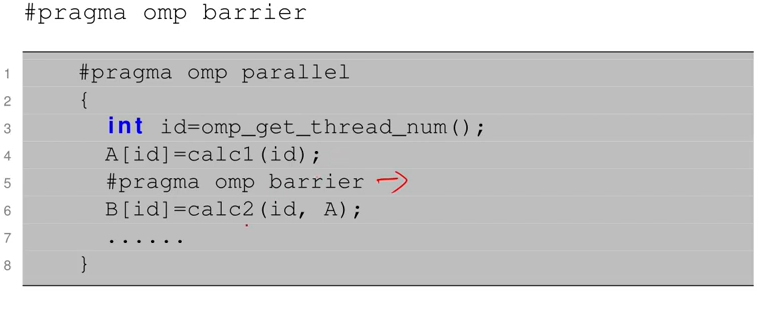
By using the critical code block, it allows only one thread to update the variable at a time. Prevents race condition



If you have a critical code block without braces following and enclosing the code below it, the code line below it will be what is executed by the critical section. If you have multiple lines of code that you wish to run in the critical section, it must be enclosed by braces.



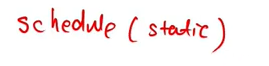
Has less overhead than critical but statements inside are limited



Threads wait for all to arrive before barrier construct.



Getting speedup in omp

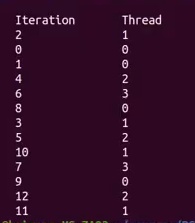


If chunk size isn’t specified with schedule clause, the chunk sizes are equally distributed amongst the threads. E.g. 4 threads, 1000 iterations-> 1000/4= 250 each thread’s num iterations

Dynamic- when a thread has finished its chunk iterations, it will take another chunk, should the amount of chunks exceed num threads. Therefore some threads may execute more chunks than others

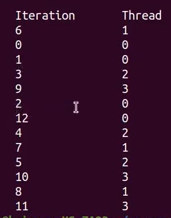
When compiler type isn’t specified, open mp will mostly use static as default





Static compiler, chunk size 2





Dynamic, chunk size 3

Ordered needs to go in pairs, need to be as a construct and as a clause, could be in the parallel construct, or for construct

REDUCTION

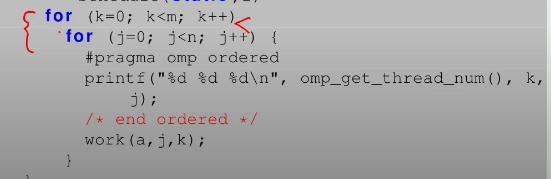
Graphical user interface, application

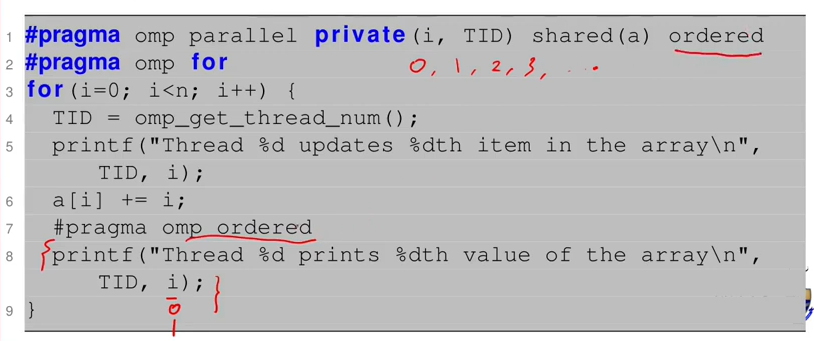
Description automatically generated

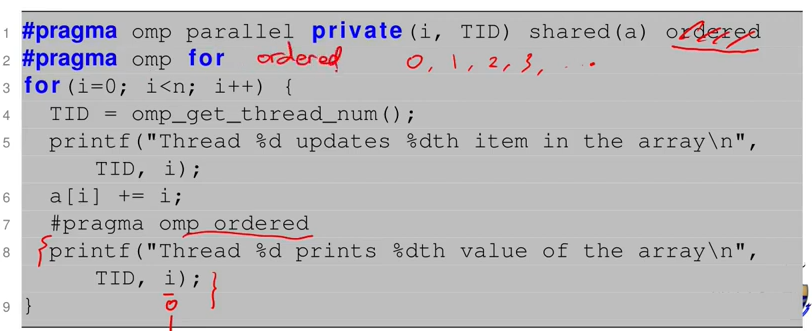
Collapse(2)-> merge 2 nested for loops

Collapse(3)-> merge 3 nested for loops

There must be nothing in between the loops in order for it to work







For loops cannot always be parallelised. Sometimes we can remove the dependencies, but sometimes we cannot.

Lastprivate- last sequential iteration will be passed to variable outside parallel region.

Nowait- allows threads to move on to next clause when they are done rather than staying at the barrier at the end of the loop

Can only add nowait clause if statements are not dependent on each other entirely.

FALSE SHARING

Does not produce incorrect results but ruins program performance due to more accesses to memory to update stale cache lines.

Examples are multiple threads changing an array. Each thread will have to fetch the changed array to update its cache line.

Even though the threads write to separate variables, those variables are likely to be on the same cache line. This means that the cores will be wasting a lot of time and bandwidth updating each other's copy of this cache line.

e.g.

Text, letter

Description automatically generated

RACE CONDITION

Multiple threads try to update the same shared variable. Can cause incorrect results.