

Introduction to Parallel Programming in OpenMP
Dr. Yogish Sabharwal
Department of Computer Science & Engineering
Indian Institute of Technology, Delhi

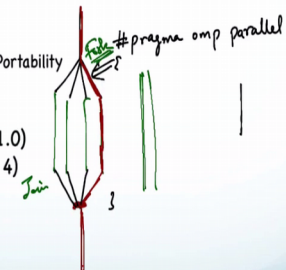
Lecture – 16
OpenMP: About OpenMP

So what is openMP?

(Refer Slide Time: 00:02)

OpenMP (Open Multiprocessing)

- API for shared Memory Model programming
 - Compiler directives (e.g. `#pragma omp parallel`)
 - Runtime Library routines (e.g. `omp_get_num_threads()`)
 - Environment variables (e.g. `OMP_NUM_THREADS`)
- Advantages
 - Standardization and Portability
 - Ease of use
- Since 1997 (Version 1.0)
- Latest 2013 (Version 4)
- Thread Based
- Fork-Join Model



It is an application programmer interface for shared memory model programming. It consists of compiler directives, we saw compiler directives which one did we see hash `pragma omp parallel`, when we wanted to define a parallel region, right. These are directives for the compiler these are pigged up by the compiler when you compile the code and replaced with some other code the compiler figures out what to put over there. Then there are runtime library routines, these are functions which can be invoked within your code. So, there is a library that is a linked along with your code right the openMP library and it provides support for all these functions.

What are the examples we saw `OMP get num threads`, `OMP get thread num` and so on right. So, these are functions, these are also a part of openMP and then there are environment variables. So, we saw an example of an environment variable, `OMP num threads` right that is sets the number of threads that you want to execute in the parallel region all right. So, what are the advantages of using openMP? Standardization and

portability is the most important one. So, as you carry your code from one platform to another platform, you do not have to change your code again right when you move from a GCC compiler to Lithuania x compiler you do not have to change your code again.

OpenMP is the standard which if a compiler says it supports. So, it supports the entire set right that different versions of openMP. So, you just have to be aware of which version is being supported and what are the features being provided by that version and no matter which compiler you used with supports that version of openMP, it your code will compile and run properly. So, that is the biggest advantage and of course, is ease of use. So, openMP is a very very simple in the sense that we can start off with the code that you had the sequential code, that you had and you can this compare it with a openMP and run you just have to in hash include OMP dot h it will run just tried it to learn sequentially.

And now if you want to paralyze, figure out which is the most time consuming part go to that part and just figure out how to paralyze that part you do not have to touch the remaining code. That is a huge advantage that makes a paralyzing codes very very simple you can start with a sequential code. If you go to something like MPI message passing interface, you have to rewrite the entire code before you can run it in parallel that is a challenging message passing programs. So, that is the advantage of shared memory programming and openMP of course, that ease of use you can you know incrementally paralyze your code, start with a sequential code and then incrementally pick up one part then another part and so on and parallel is it. OpenMP has been around since 1997 that was version one the latest one is version 4.

It is a thread based model. So, there are threads which share the same address space right. So, everything is shared by the threads except for their own stacks right and the thread local space. So, all the global variables the code everything is shared, right.

So, finally, openMP is based on the fork join model. So, what is the fork join model, when you encounter the hash pragma OMP parallel region right when you encounter this compiler directive, at that time what happens is the compiler replaces this with the code that actually creates threads right. So, up to this point up to the OMP parallel point code is executing sequentially, write there is a single thread that is executing.

So, let us say OMP num threads is set to 4, right. So, what will it do it will actually at this point of time it will launch three threads it will launch three new threads, and one of

the threads is going to be the thread which was already executing right. So, this thread which was already executing this continuous on as one thread and three additional threads get launched at this point in time and then the compiler sets them up so that they execute this code which is there in the parallel region, right.

So, all of them execute this parallel region and when you encounter this parenthesis close, when the parallel region ends at that point of time a join operation is performed. So, this is called forking right and at the end a join operation is performed, where essentially the master thread. So, this was the master thread remember it will at this point of time wait for all the other threads to complete. So, all the other threads when they complete they will come and join over here, and when all the other threads have completed they have joined back at that time it will proceed ahead again with the remaining code.

And now if it encounter the another hash pragma OMP parallel directive, another parallel region it will again do the same thing over right. Again it will launch three additional threads, they will execute and they will join back again. So, that is called the fork join model.