Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 6.017

IJCSMC, Vol. 6, Issue. 2, February 2017, pg.90 – 94

Parallel Computing using OpenMP

Ms. Ashwini M. Bhugul
MMCOE, Pune-52

Abstract- Parallel Computing is most widely used paradigm today to improve the system performance. System performance can be increase by executing application on multiple processors. Today's Computers are complex they are capable of executing application program on multiple processors. OpenMP is API for running application parallelly to improve the speedup. This paper reviews about OpenMP API and focuses on improving the system performance.

Keywords- OpenMP, Parallel Computing

1. Introduction to parallel Computing

Parallel Computing is a part of computation which allows to execute a program on a single computer by multiple processors simultaneously. A program is split into parts and run simultaneously on multiple processors. Parallel processing has been developed as an effective technology in modern computers to meet the demand for higher performance, lower cost and accurate results in real-life applications. It utilizes all resources to solve one problem. There are many platforms available for parallel computing such as OpenCL, MPI, OpenMP, etc, with which we can improve system performance by dividing task to processors. Parallel programming introduces additional sources of complexity.

2. Introduction to OpenMP

OpenMP “Open multiprocessing” is API for application that uses shared memory. It is portable, user friendly and efficient. It was first released in 1957. It is an API for writing multithreaded applications. It is a set of compiler directives and library routines for parallel application programmers. Using OpenMP, the programmer can write code that will be able to use all cores of a multicore computer and will be run faster if the number of cores are increased.it is used in shared memory architecture. This is a block -structured approach introduced for concurrency.

It operates on fork and join model of parallel execution as shown in figure 2.1. All OpenMP programs started as a single process called as master thread master thread executes sequentially until a parallel region encountered. At this point the master thread 'forks' into a number of parallel worker threads. The instructions in the parallel region are then executed by this team of worker threads. At the end of the parallel region, the threads synchronise and join to become the single master thread again. Parallelisation with OpenMP is specified through compiler directives which are embedded in the source code.

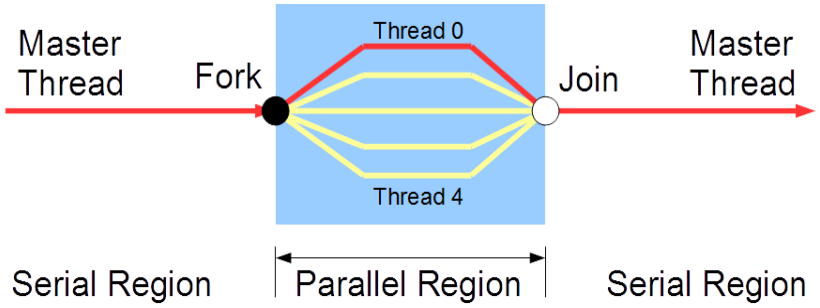


Figure 2.1 fork and join model

OpenMP programs can be written in C,C++ and FORTRAN.it consists of directives, functions and environment variables. It requires supportive compilers such as GNU,MS,Intel,HP,etc.

1. **Directives**

- It initializes parallel region.
- Divides the work.
- Synchronizes Data.
- Sharing Attributes.

2. **Functions**

- Defines number of threads.
- Gets thread ID.
- Supports Nested Parallelism.

3. **Environment Variables**

- Scheduling Type.
- Number of Threads.
- Dynamic Adjustment of Threads.
- Maximum Number of Threads.

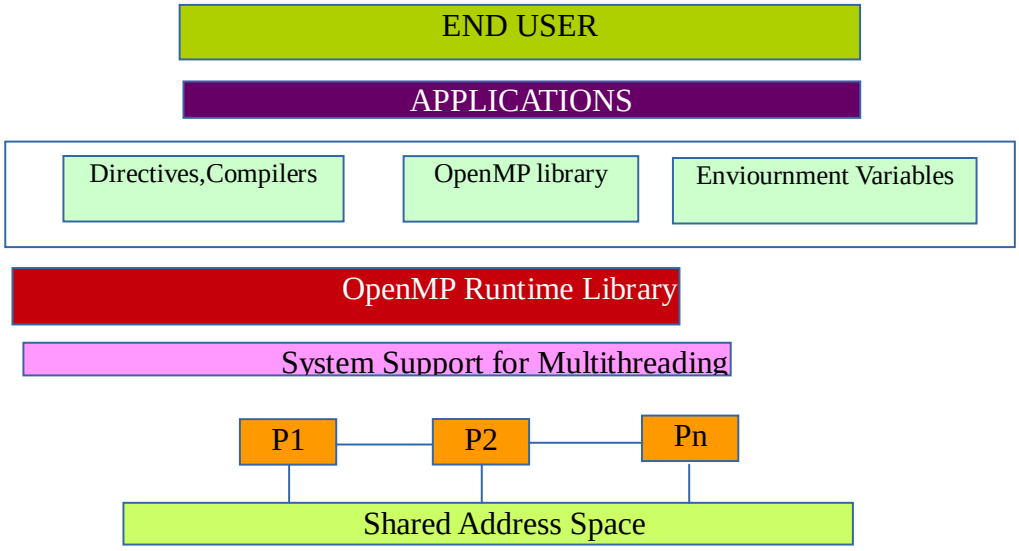


Figure 2.2 Basic Structure of OpenMP

3. Creating a program using OpenMP

Example: Hello World program using OpenMP & C

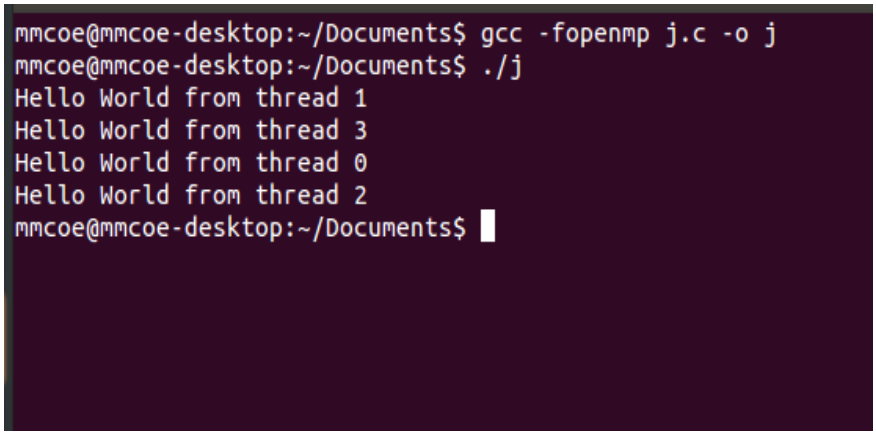
```
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>

  int main (int argc, char *argv[]) {
  int th_id, nthreads;
 #pragma omp parallel private(th_id)
 {
   th_id = omp_get_thread_num();
   printf("Hello World from thread %d\n", th_id);
   }
   return 0;
}
```

To create a OpenMP program, we can use header file named as <omp.h>.Above program creates multiple threads and run them in parallel by using #pragma omp parallel.

To Compile and run : gcc -fopenmp programname.c -o programname

./programname

O/P :



Execution begins with a single thread A team of threads is created at each parallel region. Thread executions are distributed among available processors. Execution is continued after parallel region by the Master Thread.

To get total number of threads and thread ID, OpenMP provides method omp_get_thread_num( ) which returns the ID of a thread, where the ID ranges from 0 to the number of threads minus 1. The thread with the ID of 0 is the master thread and omp_get_num_thread( ) returns the actual number of threads in the current team of threads. Every thread has access to shared and private memory. Threads can communicate with each other using shared memory. We can declare variables for shared and private memory using shared( ) private( ).

eg. #pragma omp parallel private(threadid) shared(totalthreads)

private will keep threadid hidden from all other threads and shared will keep all threads in shared state.

4. Advantages of OpenMP:

1. Shared memory parallelism is easier to learn.
2. It uses both fine grain and coarse grained parallelism.
3. It is widely available and portable.
4. serial code statements usually don't need modification
5. Directives can be added incrementally

5. Applications

1. Matlab
2. Mathematica

6. Conclusion

OpenMP is beneficial for multi core programming. We can improve system performance by lots of parallelization.it is very good parallel platform to achieve high performance.

References

[1]. Sheela Kathavate, N.K.Srikanth, "Efficient of Parallel Algorithms on Multi Core Systems Using OpenMP", International Journal of Advanced Research in Computer and Communication Engineering, Vol. 3, Issue 10, October 2014.

[2]. Pranav Kulkarni, Sumith Pathare, "Performance Analysis of Parallel Algorithms over Sequential using OpenMP", IOSOR Journal of Computer Science, Volume6, March-April 2014.

[3]. Sanjay Kumar Sharma, Dr. Kusum Gupta, "Performance Analysis of Parallel Algorithms on Multi-core System using OpenMP Programming Approaches", International Journal of Computer Science Engineering and Information Technology(IJCSEIT), Vol.2, No.5, October 2012.

[4]. http://www.openmp.org/resources/tutorials-articles/Tim Mattson's (Intel) "Introduction to OpenMP" (2013) on YouTube.

[5]. An Overview of OpenMPRuud van der PasSenior Staff Engineer Technical Developer Tools Sun Microsystems, Menlo Park, CA, USA.

[6]. An OpenMP Runtime API for Profiling Marty Itzkowitz, Oleg Mazurov, Nawal Copty, and Yuan LinSun Microsystems, Inc. 16 Network Circle Menlo Park, CA 94025.

