International Journal of Computer Science and Network (IJCSN)
Volume 1, Issue 4, August 2012 www.ijcsn.org ISSN 2277-5420

A Review on New Paradigm's of Parallel Programming Models in High Performance Computing

[1]Mr.Amitkumar S Manekar, [2] Prof.Pankaj Kawadkar, [3]Prof. Malati Nagle

[1] Research Scholar Computer Science and Engineering, PIES, RGPV, Bhopal, M.P, India.

[2]Computer Science and Engineering, PIES, RGPV, Bhopal, M.P India.

[3]Computer Science and Engineering, PIES, RGPV, Bhopal, M.P, India.

Abstract
High Performance Computing (HPC) is use of multiple computer resources to solve large critical problems. Multiprocessor and Multicore is two broad parallel computers which support parallelism. Clustered Symmetric Multiprocessors (SMP) is the most fruitful way out for large scale applications. Enhancing the performance of computer application is the main role of parallel processing. Single processor performance on high-end systems often enjoys a noteworthy outlay advantage when implemented in parallel on systems utilizing multiple, lower-cost, and commodity microprocessors. Parallel computers are going main stream because clusters of SMP (Symmetric Multiprocessors) nodes provide support for an ample collection of parallel programming paradigms. MPI and OpenMP are the trendy flavors in a parallel programming. In this paper we have taken a review on parallel paradigm's available in multiprocessor and multicore system.

Keywords: ***Parallelism, MPI (Message Passing Interface), OpenMP, Heterogeneous (hybrid) systems, SMP (Symmetric Multiprocessor).***

1. Introduction

Even as MPI has a distributed memory conceptual view, OpenMP is directed to the shared memory systems. In parallel environment with the existing merits and demerits of MPI and OpenMP both are co-exists with hybrid (OpenMP+MPI). In this work our aim is to explore the performance of the OpenMP/MPI and hybrid programming model and analysis the shared and distributed memory approaches, as well as the present heterogeneous parallel programming model.
Microprocessor based single processing unit are facing heat dissipation and energy consumption issues with limited clock frequency and number of jobs conducted in each clock period. Multi-core architectures put forward enhanced the performance and energy efficiency for the same processing unit [2]. Furthermore nowadays we have classified approaches one which is capable to integrate more than one core in to a single microprocessor (probably two to ten) called as multi-core approach which use sequential programming. Another one is many cores approach having built with large number of cores (as many as possible) basically used for parallel programming. Clearly, this change of paradigm has had (and will have) a huge impact on the software developing community. Parallel computers are taking over the world of computing. The computer industry is ready to submerge the market with hardware that will only run at full speed with parallel programs [3]. This can be largely attributed to the inherent complexity of specifying and coordinating concurrent tasks, a lack of portable algorithms, standardized environments, and software development toolkits [17]. Sequential programming is over headed due to stalling of clock frequency.
Concurrency using several cores can overcome these issues in attendance are nothing but many and multi core processor using parallel ones this is also called as

concurrency revolution [4]. Scalability in terms of application can scale seamless automatically with number of processors. In this regards there are two approaches for doing parallelism (1) Auto Parallelism (2) Parallel Programming [5].

• **Auto Parallelism:-**Using instruction level parallelism (ILP) or parallel compilers sequential programs are automatically paralleled. Actual programs without doing modification is recompiled using these ILP or parallel compilers it has a limitation that amount of parallelism is very less due to complexity of automatic transformation of code .
• **Parallel programming approach:-**Application are turned to exploit parallelism by partitioning the total work into small task, this task then mapped on the cores. It provides high parallelism.
Besides of these two some typical parallelism also present into the computer programs i.e. data, recursive, pipelined. The main four phases for the parallelism is finding concurrency, algorithm structure, supporting structure and implementation mechanism. Depending on these four patterns or phases ***SPMD*** (single Program Multiple Data) - different data is used several times with respect to single program. Fig 1 will describe the overview of the stated outline model for language.
• ***Master/Worker*** – Master process setup a poll of worker process and bag of task.
• ***Loop Parallelism***- Concurrent execution of different iteration of one or more loops.
• ***FORK/JOIN-*** Main process forks off in different processes that execute concurrently until they finally join in single process [10].

There are two classical categories of parallel system (1) Shared Memory (2) Distributed Memory [8].

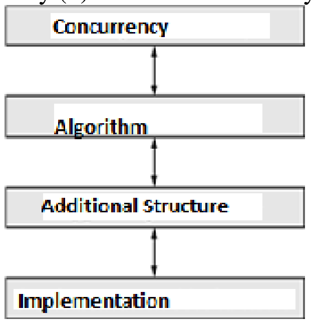

Fig 1:- Overview of the outline language
• **Shared Memory: -** A single Memory address space is used by all processors. Basically used in servers and high end workstations, today multi core processors used shared memory space. Fig 2 shows the shared memory architecture.

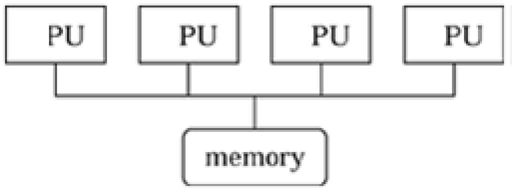

Fig:-2 The shared memory architecture

• **Distributed Memory:** - Each processor with its own serving memory blocks are the distributed memory model. These models work in network or a grid of computers. Fig 3 is the distributed memory architecture.

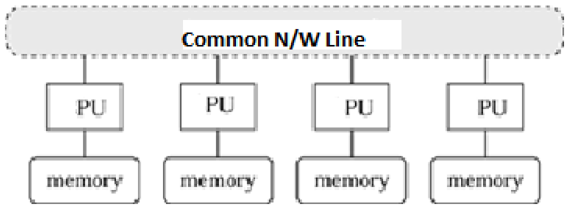

Fig 3:- The distributed-memory architecture

2. Related Work

Beside of this Hybrid shared with distributed memory system can be used. The conventional parallel programming practice involves a pure Shared Memory Model [8]. Usually using the OpenMP API [11], in shared memory architecture, or a pure message passing model [8] using MPI API [12], on distributed memory system [1].old approach of doing parallelism involves pure shared memory models [8]. Usually in shared memory architecture uses of OpenMP API [11]. For distributed MPI API once [12]. In this paper we revive the parallel programming models in high performance computing (HPC) with classification of parallel programming models used today.

2.1 Classification of Parallel Programming Model

Doing Parallelism is not specific for any hardware boundaries, the reason behind this is today many processors can put together to achieve parallelism. This provide flexibility for generate parallel programs with maximum efficiency and appropriate balance in communication and computational model. General purpose computation GPUs in multicore system lead to

heterogonous parallel programming (HPP) models. Depending on all these multicore architecture different parallel programming models lead to hybrid model called as hybrid parallel programming model.
In conventional programming approach OpenMP [6] for shared memory and MPI for distributed memory i.e. classical or pure parallel models are available. With availability of using new processor architecture multicore CPU and many core GPUs gives us heterogeneous parallel programming models, also partitioned global address space (PGAs) model in distributed environment using global memory space is available. So architecture available prompts us for hybrid, shared distributed memory with GPUs model. One more thing should be in consideration with these about available programming language. Let's have a look of all these.

3. Pure Parallel Programming Language Models

Classification of parallel programming models using a pure shared or distributed memory approach., shared memory OpenMP, and distributed memory Message Passing models(MPI) is a specification for message passing operations [7], [14], [15], [16]. Table 1 collects the characteristics of the usual implementations of these models

TABLE-1 PURE PARALLEL PROGRAMMING MODELS IMPLEMENTATIONS [9].

Implementation	**OpenMP**	**MPI**
Programming Model	Shared Memory	Message Passing
System Architecture	Shared memory	Distributed and shared Memory
Communication Model	Shared memory	Massage passing or shared address
Granularity	Fine	Course-fine
Synchronization	Implicit	Implicit or Explicit
Implementation	Complier	Library

3.1 Shared Memory OpenMP

Based on the compiler directives ,library routines and environment variables it is used to form parallelism on shared memory machines, it's an a industry standard directives guide the compiler which region is execute in parallel together with some instruction. This model use fork and join.

Characteristics:-

• OpenMP codes will only run on shared memory machines
• Not Portable
• Permits both courses gain and fine gain parallelism.
• User directives which help the compiler parallelized the code.
• Each thread sees the same global memory.
• Implicit messaging.
• Use fork-join model for parallel computation.

Limitation:-

• OpenMP works only for shared memory.
• Limited scalability, not much speed up.
• Threads are executed in a non deterministic order.
• OpenMP requires explicit synchronization [18]

3.2 MPI (Message Passing Interface)

In distributed memory model with explicit control MPI gives parallelism. Every process are associated with read and write operation with respective their local memory. Appropriate subroutine call is used to copy data for each process from their local memory.MPI is define as a set of function and procedures.

Characteristics:-

• MPI runs on both distributed and shared memory model.
• Portable.
• Particular adaptable to coarse grain parallelism.
• Each process has its own memory.
• Explicit Messaging.

Limitation:-

• In MPI communication can often create a large overhead, which needs to be minimized.
• Global operations can be very expensive.
• Significant change to the code is often required, making.
• Transfer between the serial and parallel code difficult.
• In MPI dynamic load balancing is often difficult [13].

3.3 Hybrid (OpenMP+MPI)

Hybrid rational model takes both advantages from the MPI/OpenMP. It achieves simple and fine-grain parallelism with explicate decomposition of task placement. Both MPI and OpenMP are industry standard so it takes advantages of its portability on SMP Clusters.

Characteristics:-

• Match Current hardware trend.

• Support two levels of parallelism for application both coarse-grained (MPI) and fine-grained (OpenMP).
• Limitation of MPI (scalability) is overcome by adding OpenMP.
• Assign different no threads by OpenMP for load balancing to achieve synchronization.

Limitation:-

• Programming overhead as mixed mode implementation.
• Not a solution to all parallel programs but quite suitable for certain algorithms [13].

4. Conclusions

This survey work is based on modern parallel programming model, from this study it is clear that available multi core and many core models with efficient parallelism provide arena to trend computer science curricular.

In our study we observed that MPI i.e. distributed memory model is a main sharing partner that's why distributed memory parallel programming approach is huge demanding in last decade with MPI library standards. Also it has been observer that OpenMP have continual progress in HPC with shared memory model. Among all different approaches MPI for distributed memory and OpenMP for shared memory are useful.

References

[1] J. Diaz, C. Munoz-Caro and A. Nino: A Survey of Parallel Programming Models and Tools in the Multi and Many-core Era, IEEE, Parallel and distributed system, Vol- 23 no 8 pp 1369-1386. Aug, 2012.

[2] D. Kirk and W. Hwu. Programming Massively Parallel Processors: A Hands-on Approach, Morgan Kaufmann, San Francisco, 2010.

[3] H. Sutter, J. Larus. "Software and the Concurrency Revolution",ACM Queue, vol. 3, no. 7, pp. 54-62, 2005

[4] H. Kasim, V. March, R. Zhang and S. See. "Survey on Parallel Programming Model", Proc. of the IFIP Int. Conf. on Network and Parallel Computing, vol. 5245, pp. 266-275, Oct. 2008.

[5] B. Chapman, G. Jost, R. van der Pas, Using, OpenMP: Portable Shared Memory Parallel Programming. MIT Press, 2007.

[6] J. Dongarra, I. Foster, G. Fox, W. Gropp, K. Kennedy, L. Torczon and A. White. The Sourcebook of Parallel Computing, Morgan Kaufmann Publishers, San Francisco, 2003.

[7] M. J. Sottile, T. G. Mattson and C. E. Rasmussen, Introduction to Concurrency in Programming Languages. CRC Press, 2010.

[8] OpenMP. "API Specification for Parallel Programming", http://openmp.org/wp/openmp-specifications. Oct. 2011.

[9] T. G. Mattson, B. A. Sanders and B. Massingill. Patterns for Parallel Programming. Addison-Wesley Professional, 2005.

[10] OpenMP. "API Specification for Parallel Programming",http://openmp.org/wp/openmp-specifications. Oct. 2011.

[11] W. Gropp, S. Huss-Lederman, A. Lumsdaine, E. Lusk, B. Nitzberg, W. Saphir and M. Snir, MPI: The Complete Reference, 2nd Edition,Volume 2 - The MPI-2 Extensions. The MIT Press, Sep. 1998.

[12] Sandip V.Kendre, Dr.D.B.Kulkarni: Optimized Convex Hull With Mixed (MPI and OpenMP) Programming On HPC, IJCA (0975 –8887), Volume 1 – No. 5,2010

[13] P. S. Pacheco, Parallel Programming with MPI, Morgan Kaufmann, San Francisco, 1996.

[14] W. Gropp, E. Lusk, and A. Skjellum, Using MPI: Portable ParallelProgramming with the Message-Passing Interface, 2nd ed. MIT Press,Cambridge, MA, 1999

[15] W. Gropp, E. Lusk, and R. Thakur, Using MPI-2: Advanced Features of the Message-Passing Interface. MIT Press, Cambridge, MA, 1999.

[16] A Grama,A Gupta: An Introduction to Parallel Computing: Design and Analysis of Algorithms 2nd Edition. Pearson Publication 2007.

[18] Nadia Ameer,EPCC, A Microbenchmark Suite for Hybrid Programming,sep,2008