International Journal of Innovative Research in Advanced Engineering (IJIRAE)
Volume 1 Issue 2 (April 2014)

A Parallel Computing-a Paradigm to achieve High Performance

Prof.Ashwini M. Bhugul
Asst. prof. MMCOE, Pune
ashu_b53@rediffmail.com

Abstract-Over last few years there has been rapid changes found in computing field.today, we are using the latest upgrade system which provides the faster output and high performance. User view towards computing is only to get the correct and fast result. There are many techniques which improves the system performance. Today's widely use computing method is parallel computing. Parallel computing, including foundational and theoretical aspects, systems, languages, architectures, tools, and applications. It will address all classes of parallel-processing platforms including concurrent, multithreaded, multicore, accelerated, multiprocessor, clusters, and supercomputers. This paper reviews the overview of parallel processing to show how parallel computing can improve the system performance.

1. INTRODUCTION

Parallel computer architecture provides a proportional increase in performance with increase in performance with increasing system resources. System resources are scaled by the number of processors used, memory capacity enlarged, access latency tolerated, I/O bandwidth required, etc.parallel architecture has a higher potential to deliver scalable performance Parallelism appears in various forms such as look ahead, pipelining, vectorization, concurrency, data parallelism, etc.

Parallel computers are those that execute programs in MIMD code. There are two classes, namely-shared-memory multiprocessors and message passing multicomputers.In parallel computing a large task is divided into subtask and all the sub tasks will execute in parallel. This will help to reduce the waiting or stalled stages in pipeline of the computer system. Lower the number of clock cycles required higher performance will be achieve by the system. The degree of parallelism depends upon the parallel computer architecture. A parallel computer (or multiple processor system) is a collection of communicating processing elements (processors) that cooperate to solve large computational problems fast by dividing such problems into parallel tasks.

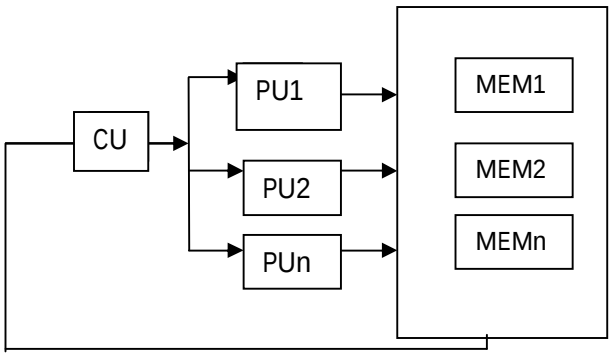
There are some classification made for parallel computing, one of is Flynn's classification. Flynn's classification is based on multiplicity of instruction streams and data streams in computer system. Its machine organization includes SISD (single instruction single data), SIMD (single instruction multiple data), MISD (multiple instruction single data), and MIMD (multiple instruction multiple data).

Some Broad issues involved:

1. The concurrency and communication characteristics of parallel algorithms for a given computational problem.
2. What portions of the computation and data are allocated or mapped to each PE.
3. How the processing elements cooperate and communicate.
3. How data is shared/transmitted between processors.
4. Abstractions and primitives for cooperation/communication and synchronization.

Parallel computing improves the system performance, as the complete task is divided into stages, which will take less time to execute the stream of data. Both data and instructions are fetched by memory module. Instructions are decoded by the control unit, which sends to processor units for execution. Each instruction stream is generated by an independent control unit. Multiple data stream originates from shared memory module. This improves the system speedup and performance.

1.1General overview of parallel processing system:



CU-control unit
PU-processing unit
MEM-memory

Figure 1.1: SIMD architecture showing parallel computing

1.2 Need of Parallel Computing

Whenever there is need to work on large amount of data, system need a fast performance, more computing cycles/memory needed, there will be overload on machine when it performs a whole task, parallel computing make this task efficient by dividing a task into stages that will reduce the computing clock cycles and memory too. Some of the applications of parallel computing are: Scientific/Engineering computing: CFD, Biology, Chemistry, Physics, General-purpose computing: Video, Graphics, CAD, Databases, Transaction Processing, Gaming, etc.

Number of transistors on chip growing rapidly. Clock rates expected to continue to go up but only slowly. Actual performance returns diminishing due to deeper pipelines. Increased transistor density allows integrating multiple processor cores per creating Chip-Multiprocessors (CMPs) even for mainstream computing applications that is also one of application of parallel processing The increased utilization of commodity of-the-shelf (COTS) components in high performance parallel computing systems instead of costly custom components used in traditional Super computers leading to much lower parallel system cost. Today's microprocessors offer high-performance and have multiprocessor support with parallel processing.

II. Different granularities of parallelism

In parallel computing units of work can be execute concurrently if sufficient recourses exists. There are two granularities:

2.1 Thread level parallelism
2.2Instruction level parallelism

2.1Thread level parallelism (TLP)

A thread is separate process having its own instructions and data. Thread may be a separate process or single program executing in parallel. In parallel computing, the thread offers different instructions and data so that the processor can switch to another thread and can continue the instructions, this will include the thing that, processor will continuously execute instructions without stalls, this will take less time to execute instructions.TLP is a software that enables a program often a high end programs such as web application, etc.

There are two approaches to TLP:
1. Fine-grained multithreading
2. Coarse-grained multithreading

Fine grained multithreading switches threads on each instruction, that will execute multiple instructions simultaneously, CPU must be able to switch threads every clock cycles. The advantage of using this technique is that it can hide both short and long stalls. Coarse-grained multithreading switches threads only when current thread is likely to stall for some time.it is more natural for any given thread and consume less time to switch threads. It is also easier to implement switching process.

2.2 Instruction Level Parallelism

Instruction-level Parallelism CILP) is a family of processor and compiler design techniques that speed up execution by causing individual machine operations to execute in parallel. Pipelining can overlap the execution of instructions when they are independent on each other, this potential overlap among instructions is called as Instruction level

Parallelism. Processing instructions in parallel requires some tasks:

1. Checking independencies between instructions, if two instructions are independent on each other, they can be group together to execute in parallel.
2. Assigning instructions to the functional units on hardware.
3. Determining when instructions are initiated. This include the technique called score boarding, it allows instructions to execute out of order when there is sufficient resource available.

ILP architectures can be classified as follows.

- **Sequential architectures:** architectures for which the program is not expected to convey any explicit information regarding parallelism. Superscalar processors are representative of ILP processor implementations for sequential architectures.
- **Dependence architectures:** This architecture provides information of those programs in which instructions which are dependent on each other. Dataflow processors are one of the examples of this architecture.
- **Independence architectures:** architectures for which the program provides information as to which operations are independent of one another. Very Long Instruction Word (VLIW) processors are examples of the class of independence architectures.

III. Some Parallel Programming Techniques

Sometimes in large scale industries, there is a multiprocessor system, which requires number of instructions to be executed simultaneously, as the system consists of number of processor connected together there may happens some memory conflicts while performing tasks parallel. Processor needs to communicate with each or want to share some data with each other; hence there are some parallel programming techniques, three of them are explained below:

3.1. Message passing parallel programming

In massage passing programming programs view their programs as a collection of cooperating processes with private variables. This style is suitable for message passing parallel computers. This technique needs only two primitives that is send and receive, that will send or receive instructions from one processor to another. Figure given below showing Message-Passing example of Intel Paragon.

Figure 31: Sandia's Intel Paragon XP/S-based Supercomputer

There are most widely used Message Passing Programming tools are:

1. Message Passing Interface (MPI):

It provides a standard for writing concurrent message-passing programs.MPI offers rich set of functions for doing point to point communicaton.it will divide a task into stages and that separate tasks can communicate with each other.MPI implementations include parallel libraries used by existing programming languages (C, C++).

2. Parallel Virtual Machine (PVM):

PVM is portable message passing programming system, designed to form separate host machines to from "virtual machine". Virtual machine is collection of processors that may work simultaneously or at different time. It enables a collection of heterogeneous computers to use as a coherent and flexible concurrent computational resource. PVM support software executes on each machine in a user configurable pool, and provides computational environment of concurrent applications. User programs written for example in C, FORTRAN or Java are provided access to PVM through the use of calls to PVM library routines.

3.2. Shared Memory Programming

In shared memory programming, programmers view their programs as a collection of cooperating processes that will access a common pool of shared variables. In such computers, processors may not have private program or data memory. a common data or program is stored in shared memory that will use by all processors. The main program creates separate processes and assigns a tasks to every processor with a common shared memory, each processor computes independently and after all, result of each processor stored in common memory. Two statements are use for this purpose: 1. Fork 2. Join

Example: Process X; Process Y;
fork y;
join y;
end y;

Process X when it encounters fork y invokes another process y, after invoking y, it continues doing its work. Process y starts executing concurrently in another processor. When X reaches join y statement has to wait till y finish its work. In shared memory programming when multiple processors are in use and sharing a common memory so care has to be taken that no two dependent instructions should execute simultaneously. For shared memory programming three models of multiprocessors are:

1. The Uniform Memory Access (UMA) Model:

In this model all physical memory is shared by all processors. All processors have equal access to all memory addresses.

2. Distributed memory or Non-uniform Memory Access (NUMA) Model:

In this model, Shared memory is physically distributed locally among processors. Access latency to remote memory is higher.

3. The Cache-Only Memory Architecture (COMA) Model:

In this model, A special case of a NUMA machine where all distributed main memory is converted to caches. No memory hierarchy at each processor.

3.3 Data parallel Programming

In Data Parallel Programming many programs can apply same operation to each element of composite data.sometimes, looping statements performed repetitive tasks, using data parallelism this repetition carried out automatically by having multiprocessors working concurrently. Logically, single thread of control, performs sequential or parallel steps conceptually, a processor is associated with each data element.

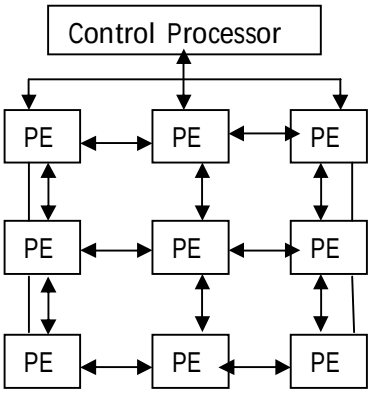


Figure 3.3: Data parallel system (All PE's are synchronized)

Example machines, Thinking Machines CM-1, Maspar MP-1 and MP-2, etc.Data parallel programming makes programs easy to read and write. Same program will be executed over different data sets simultaneously on multiple processors. a popular data parallel programming language is high performance Fortran (HPF). figure 3.3 given below shows example of data parallel systems. Array of many simple processors each with little memory. Processors don't sequence through instructions. Attached to a control processor that issues instructions Specialized and general communication, global synchronization

IV. CONCLUSION

At the end of this review paper, we may observe some current some current trends. High performance computers are increasingly in demand, Parallel computing architecture is today's widley used architecture which provides speed and high performance. It is used in many scientific, engineering, medical, military and research areas to achieve high system performance.

V. REFERENCES

1. Rafiqul Zaman Khan, Md Firoj A Comparative Study on Parallel Programming Tools in Parallel Distributed Computing System: MPI and PVM, Department of Computer Science, Aligarh Muslim University, Aligarh-202002,(U.P), India.
2. Ramakrishna Rau, Joseph A. Fisher, Instruction-Level Parallel Processing: History, Overview and Perspective, Computer Systems Laboratory,HPL-92-132,October, 1992.
3. M. D. Smith, M. Horowitz and M. Lam. Efficient superscalar performance through boosting. Proc. Fifth International Conference on Architectural Support for Programming& Laneua&es and Operating Systems (Boston, Massachussetts, October 1992), 248-259.
4. R. Chaiken, B. Jenkins, P.-_A. Larson,B. Ramsey, D. Shakib, S. Weaver, and J. Zhou. SCOPE: Easy and e_cient parallel processing of massive data sets. International Conference of Very Large Data Bases (VLDB), August 2008.
5. D. DeWitt and J. Gray. Parallel database systems: The future of high performance database processing. Communications of the ACM, 36(6), 1992.
6. R. Pike, S. Dorward, R. Griesemer, and S. Quinlan. Interpreting the data: Parallel analysis with Sawzall. Scienti_c Programming,13(4):277{298, 2005.
7. F. Rabhi and S. Gorlatch. Patterns and Skeletons for Parallel and Distributed Computing. Springer, 2003.
8. Y. Yu, M. Isard, D. Fetterly, M. Budiu,U. Erlingsson, P. K. Gunda, and J. Currey.DryadLINQ: A system for general-purpose distributed data-parallel computing using a high-level language. In Proceedings of the 8th Symposium on Operating Systems Design and Implementation (OSDI), December 8-10 2008.
9. R. Z. Khan, A. Q. Ansari and Kalim Qureshi—Performance Prediction for Parallel Scientific Application‖, Malaysian Journal of Computer Science,Vol. 17 No. 1, June 2004, pp 65-73.
10. Kai Hwang, Faye A. Briggs, "Computer Architecture and Parallel Processing" McGraw-Hill international Edition
11. Kai Hwang, "Advanced Computer Architecture", Tata McGraw-Hill
12. V.Rajaraman, L Sivaram Murthy, "Parallel Computers", PHI.