ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 2, pp: (362-365), Month: April - June 2015, Available at: www.researchpublish.com

# Parallel Processing

Anju Kundal

***Abstract:*** **The requirement of high computational tasks which require large jobs sizes has increased, due to this demand of supercomputer increases which could handle large computational requirements. Computational multi-clusters are an important emerging class of supercomputing architectures. Multi-cluster schedulers must address not only node resource allocation but also inter-cluster network utilization. In this work a new adaptive scheduling policy for heterogeneous multi-cluster systems will be developed. In this work Firstly, it identifies several of the parameters necessary to make intelligent scheduling decisions with respect to job co-allocation in a multi-cluster, to study the effect of multi-site parallel execution parameters, calculate waiting time and co-allocation time parameters, calculate the response parameters and then compare the waiting time and co-allocation parameters to increase the performance of multi-cluster systems.**

***Keywords:*** **requirement, computational, supercomputer, Multi-cluster, scheduling, allocation, parameters.**

## I. INTRODUCTION

### 1. Parallel Processing:

Parallel Processing is a term used to denote large class of technique that is used to provide simultaneous data processing task for increasing the computational speed of a computer system. Parallel computers can be roughly classified according to the level at which the hardware supports parallelism, with core and multi processor computers having multiple processing elements within a single machine, while clusters, MPPs, and grids use multiple computers to work on the same task. Specialized parallel computer architectures are sometimes used alongside traditional processors, for accelerating specific tasks.

Parallel computer programs are more difficult to write than sequential ones, because concurrency introduces several new classes of potential software bugs, of which race conditions are the most common. Communication and synchronization between the different subtasks are typically some of the greatest obstacles to getting good parallel program performance. The maximum possible speed-up of a single program as a result of parallelization is known as Amdahl's law.

Parallel computing is the simultaneous use of multiple compute resources to solve a computational problem:

- A problem is broken into discrete parts that can be solved concurrently
- Each part is further broken down to a series of instructions
- Instructions from each part execute simultaneously on different processors
- An overall control/coordination mechanism is employed

The performance of a computer system is defined by three factors.

- Time to execute a program is number of instructions to execute.
- The average number of clock cycles required.
- Clock Cycle Time.

The computational problem should be able to broken apart into discrete pieces of work that can be solved simultaneously; Execute multiple program instructions at any moment in time; Be solved in less time with multiple compute resources than with a single compute resource. The computer resources are typically either a single computer with multiple processors/cores or An arbitrary number of such computers connected by a network

**2. Need Of Parallel Processing:**

***A. Save Time And/or Money***

In theory, throwing more resources at a task will shorten it's time to completion, with potential cost savings. Parallel computers can be built from cheap, commodity components.

***B. Solve Larger / More Complex Problems***

Many problems are so large and/or complex that it is impractical or impossible to solve them on a single computer, especially given limited computer memory.

***C. Provide Concurrency***

A single compute resource can only do one thing at a time. Multiple compute resources can do many things simultaneously.

***D. Take Advantage Of Non-Local Resources***

Using computer resources on a wide area network, or even the Internet when local compute resources are scarce or insufficient.

***E. Make Better Use Of Underlying Parallel Hardware***

Modern computers, even laptops, are parallel in architecture with multiple processors/cores. Parallel software is specifically intended for parallel hardware with multiple cores, threads, etc. In most cases, serial programs run on modern computers "waste" potential computing power.

## II. JOB SCHEDULING

Job scheduling determines the sequence of starting execution of the jobs waiting in the queue. It is possible that multiple job requests are submitted at relatively same time. It means there will be some jobs running while others must wait in a job queue to be processed. Therefore, there is a need of scheduler to manage multiple jobs in the queue.

***1. First Come First Served (FCFS)***

In this approach first incoming job request get the highest priority. The first job is checked whether it fits the available resources. If the first job fits the available resources it is executed immediately but if the number of available processors cannot meet the requirement of the first job in queue, all the other jobs would be blocked until enough processors for first job become available and job waits until the next re-scheduling action takes place.

The FCFS policy allows jobs to run only in the order of queue. This situation force some jobs with enough processors available are to wait while the processors remain idle, this results in fragmentation problem which in turn degrades system utilization and performance.

The following are three non-FCFS policies which do not require users' estimates of runtime before job submission and need much less calculation upon making scheduling decisions than backfilling policies.

***A. First Available***

In this policy each job is put into the queue according to the arrival order and at each time making a scheduling decision, the scheduler scans the queue to find a job which can run with current available processors. Therefore, the first job in the queue would not block the jobs after it from execution even when it itself does not has enough processors for immediate execution and must wait.

***B. Smallest First***

In this policy, each job is put into the queue according to its requirement of processors. Jobs with smaller requirements of processors will be put before jobs with larger requirements of processors.

***C. Largest First***

In this policy, each job is put into the queue according to requirement of processors by the job. However, jobs requiring larger number of processors will be put before jobs with smaller requirements of processors.

## III. NEED OF SCHEDULING

Efficient scheduling across nodes is necessary to maximize application performance regardless of the efficiency of parallel algorithms. The addition of newer and more powerful machines due to rapid computer industry growth lead to heterogeneous clusters.

Heterogeneous clusters, whether dedicated or non-dedicated, have individual computing characters. In dedicated cluster whenever the workstation does not belong to any particular individual or resources allocated so that the jobs can be parallelized across the entire cluster, whereas, In case of non-dedicated cluster, workstations belong to a particular individual and the jobs are performed by taking idle CPU cycles. The ideas with this type of clusters are based on the fact that even in the busy hours there will still be some idle CPU cycles in the workstation.

Since heterogeneous node computing characteristics are unique, node finish times will differ. As a result, faster nodes will have to wait in idle states for slower nodes, which lead to inefficient use of cluster resources. Increasing resource utilization is an important issue for resolution. Scheduling in heterogeneous clusters is hard and the dynamic scheduling in a heterogeneous environment is significantly more complicated. Hence, we need effective scheduling policy like adaptive scheduling which fulfill the requirements of the job [1].

## IV. PARALLEL JOB SCHEDULING POLICIES

These can be categorized broadly into two parts Space Sharing and Time Sharing.

### *1. Space Sharing*

This policy provides the requested resources to the job until the job is completely executed. It has low overhead and high parallel efficiencies. It is further of three types Static scheduling, dynamic scheduling and Adaptive scheduling.

### *2. Static scheduling*

Scheduling is performed at compile time, and then jobs are allocated to individual processors before execution. The allocation remains unchanged during the execution. Most static scheduling methods needs to gather information needed for the scheduler to make further decisions, according to what kind of resources are necessary for job.

### *3. Dynamic scheduling*

Scheduling is performed at run time and it can support dynamic load balancing and fault tolerance. Load balancing could share the load between nodes even when the job is in its execution state.

### *4. Adaptive Scheduling*

It combines static and dynamic scheduling. The part of static scheduling is to gather information needed for the scheduler to make decisions on what kinds of resources are necessary. And the other part of dynamic scheduling is load balancing of job's loop.

### *5. Time Sharing*

This technique is employed to ensure that the time on a processor is divided into many discrete intervals or time slots which are assigned to unique jobs. The size of time slots depends on the cost of scheduling.

## V. LITERATURE SURVEY

D.G. Feitelson, L. Rudolph (2007), have concluded that Parallel supercomputers are anexpensive resource. Parallel computers are more difficult to use. The issue of job scheduling has suffered due to common lack of distinction between job scheduling by operating system and static and dynamic scheduling.

H.Y. Chang, K.C. Huang (2007), in their paper the focus is on throughput computing. It concerned with improving grid computing performance through appropriate workload management. In multiprocessor systems workload is managed in two steps. The single pool centralized queue approach has great potential for delivering good performance on grid computing. The single site centralized queue approach regards all processors at different sites in computing grid as a

single large processor pool and allows a job to be allocated across site. The cross site parallel computation can avoid the fragmentation problem.

K.C. Huang, P.C. Shih, Y.C. Chung (2009), have develops adaptive processor allocation based on the moldable property of parallel jobs both for homogeneous parallel computers and heterogeneous computational grid environments. The policies are required us to provide estimations of job execution time upon job submission. The policies are evaluated and the inexact run time estimations on system problems are also investigated. There are several methods for selecting which site to allocating a job. There is global scheduler which handles all job scheduling and resource allocation activities. For simplifications and efficient load sharing all computing nodes are assumed to be binary compatible. The major difference between the adaptive processors allocation procedures for a homogeneous parallel computer and for heterogeneous grid environment is the site selection process regarding to the computation and comparison of computing power of different sites. The proposed adaptive processor allocation policies are effective as well as stable under different system configurations and can tolerate a wide range of estimation errors.

William M. Jones (2005), have concluded that the effective job scheduling plays a crucial role in the efficient usage of multi-clusters. As this computational platform becomes more prevalent, intelligent scheduling algorithms that can exploit specific job and grid attributes become increasingly significant. These schedulers must make decisions that are sensitive not only to node resource allocation, but also shared inter-cluster network bandwidth utilization.

Additionally, these schedulers must be designed so that the optimizations they make to increase job throughput can be integrated with mechanisms to provide fairness among jobs arriving to individual participating clusters.

## REFERENCES

[1] HarpreetKaur, GursimranjeetKaur, AmitChhabra, "Adaptive Job Scheduling inHeterogeneous Multicluster Systems", International Journal of Advanced Trends in Computer Science and Engineering (IJATCSE), Vol.2 , No.3, Pages : 22-25, 2013.

[2] K.C. Huang, P.C. Shih, Y.C. Chung, "Using Moldability to Improve Scheduling Performance of Parallel Jobs", Springer-Verlag Berlin Heidelberg, pp. 116–127, 2008.

[3] K.C. Huang, P.C. Shih, Y.C. Chung, "Adaptive Processor Allocation for Moldable Jobs in Computational Grid", 10 International Journal of Grid and High Performance Computing, 10-21, January-March 2009.

[4] K.C. Huang, H.Y. Chang," An Integrated Processor Allocation and Job Scheduling Approach to Workload Management on Computing Grid*",This research was partly supported by the National Science Council under contracted no. NSC 94-2213-E-432-001.

[5] William m. Jones," improving parallel job scheduling performance In multi-clusters throughSelective job co-allocation"december 2005.