159.735

Final Report

Cluster Scheduling

Submitted by:

Priti Lohani
04244354

Table of contents:

1. Introduction:

1.1 Clusters:

When a group of loosely coupled computers work together so that they can be viewed as if they are one computer, it is called cluster. Clusters are generally arranged so that the speed and reliability provided by single computer can be improved. They are quite cost effective compared to the single computer with similar speed and reliability. In the parallel architecture, SMP (symmetric multiprocessing) denotes system architecture in which multiple processors are connected via bus or crossbar to the single shared main memory. This solution scales up to a dozen processors because of the limited available memory bandwidth. Clusters are composed of interconnected SMP nodes in order to provide increasing processing power.

1.2 Scheduling:

The method of matching tasks to the resources at particular times is referred to as scheduling. Scheduling usually is a static problem in most of the environments where once the solution is achieved; it can be used multiple times. There is however some dynamic environments where all the tasks are not really known before hand allowing only a subset of tasks to be scheduled.

2. Classification of clusters:

Clusters can be classified into 4 different types:

- High availability clusters
- Load balancing clusters
- High performance clusters
- Grid clusters

High availability clusters are employed to improvise on the availability of services provided by clusters. They have redundant nodes that are used in order to provide the services every time the system component stops working. In order to provide redundancy, it is vital that the size of the high availability clusters must be at least 2 nodes. High availability cluster implementations focuses on managing the redundancy inherent of the cluster to ensure that the single point of failure is successfully removed.

Load balancing clusters are employed in order to improvise on the performance. Additionally, it has a feature of HA clusters described above. Load balancing clusters works by getting all workload to come through single or more load balancing front ends, which are then shared to a group of back end servers.

High performance clusters are most commonly used in scientific computing. They are employed to offer an increased performance by dividing a computational task across lots of different nodes in the cluster. The most common HPC implementations is cluster with the nodes running Linux as OS and the free software to implement parallelism. This configuration is usually known as Beowulf cluster.

Grid clusters are employed for workloads that includes lots of packets of jobs or independent jobs which don't require the sharing of data between the jobs in the computation process. Grids focus on managing the distribution of jobs to computer that will do the job in parallel with the rest of the grid cluster.

Clusters can also be classified on the basis of node ownership. It is as follows:

- Dedicated cluster
- Non- dedicated cluster

Whenever the workstation does not belong to any particular individual or the resources are allocated so that the jobs can be parallelized across the entire cluster, it is said to be **dedicated cluster**.

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.

3. Requirements for scheduling jobs in a cluster:

There are many clustering algorithms. Some of them are listed below:

Scalability: The scheduler must be able to scale to thousands of nodes and processing thousands of tasks at the same time.

Broad scope: The scheduler must be able to sustain a various range of tasks with comparable efficiency.

Sensitivity to compute nodes and interconnect architecture: The scheduler must match compute nodes and interconnect architecture with the job profile.

Fair- share capability: The scheduler must be able to share the resources in a fair manner under heavy situations and at diverse times.

Capability to integrate with standard resource managers: The scheduler must be able to interface with the resource manager that is in use plus the general resource managers, e.g. open PBS, Torque etc.

Fault tolerance: The algorithm must not be stopped by the break down of one or several nodes and must persist functioning for the nodes that are up at that point in time.

4. Resource management systems:

Resource management systems focus on managing the processing of load by ensuring that the jobs do not compete with one another for the limited resources allowing effective and efficient use of the available resources.

Resource managers are responsible for basic node state check, receiving job requests and processing the request on the compute node. Resource managers are able to increase the utilization of a system from 20% to 70% in complex cluster environments.

Job scheduler make scheduling decisions by gathering the information about queues, loads on compute nodes and available resources by communicating with the resource managers. They are mainly employed because they are able to give better throughput of user applications on the systems they deal with.

5. Job scheduling algorithms:

The main focus of scheduling algorithm is to get the most out of the user experience as well as the system utilization while protecting scalability of the algorithm.

Although there are several mechanisms to schedule parallel jobs only some are used in practice. There are two approaches that have dominated others long time. They are:

- Backfilling
- Gang scheduling

Before I talk any further about the above mentioned approaches, it is vital I talk about classification of cluster algorithms as well as types of schedulers. Let us start with the classifications of cluster algorithms.

Cluster algorithms can be classified into four different parts. They are listed as follows:

- Exclusive clustering
- Overlapping clustering
- Hierarchical clustering
- Probabilistic clustering

In **exclusive clustering**, data are grouped in such a way that if a datum belongs to a particular cluster it cannot be added in any other clusters.

In case of the **overlapping clustering**, data are clustered by using a fuzzy set such that each point may be owned by two or more clusters that have different degrees of membership.

Hierarchical clustering is supported on the basis of amalgamation between the two closest clusters.

Last but not least, **probabilistic clustering** is based completely on a probabilistic approach.

Now, let us look at the type of schedulers. They are two types. They are listed as follows:

- Time sharing
- Space sharing

5.1 Time sharing:

Time sharing techniques are 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 the time slots depends on the cost of context switching.

Time sharing algorithms:
- Local scheduling
- Gang scheduling
- Communication driven co-scheduling

In case of **local scheduling**, the scheduler shares a global run queue. The threads that need to be executed are placed in a queue. As soon as the processor gets free, it removes the next thread from the queue, executes it for certain time and then is returned to the back of the queue. Fairness is easily ensured as each of the thread gets an equal share of machine and the priority mechanisms are pretty simple to enable. Fine grained communications between threads do not perform very well with local scheduling.

Gang scheduling is also known as explicit co-scheduling. As each parallel job is composed of threads and processes it is a very good idea to run them on several CPUs in the same time. This type of threads or process group is referred as gang and the scheduling approach they use is called gang scheduling.

Fine grained communications between threads perform very well under gang scheduling. One of the most important features of gang scheduling is that context switching is coordinated across the nodes. Therefore, all processes can be scheduled and de-scheduled simultaneously. Since the processes of the jobs are scheduled together, gang scheduling has a huge advantage of faster completion time. However, its need of global synchronization overhead in order to co-ordinate the set of processes is its disadvantage.

Since the scheduling for entire cluster cannot be possible with respect to the task rate, partitioning the resources is vital. Gang scheduling uses either a dynamic or a fixed scheduling for CPU’s.

In case of dynamic partitioning, partitions are allowed to change sizes but the drawback would be the requirement of complex synchronization of context switching across the whole parallel system. Therefore, it is a good idea not to do repartitioning with every context switch.

In case of the fixed partitioning, CPU’s are divided into disjoint subsets and the jobs are sent to be scheduled within the subset. Fragmentation can be introduced if the gang sizes are not same.

When each node in a cluster have its own scheduler that co-ordinates the communicating processes of parallel job, it is said to be communication driven co-scheduling. In order to determine when and which process to schedule, all these algorithms depends on one of the two events i.e. arrival of message or waiting for message.

5.2 Space sharing:

Space sharing techniques is the most common scheduler. It provides the requested resources to one particular job until the job is completely executed. The main advantage of space sharing algorithm are low overheads and high parallel efficiencies. Disadvantage of space sharing algorithm is poor average turn around times as it sometimes allows short jobs to sit in queue waiting for the long jobs.

Space sharing algorithm:

- Batch scheduling

Batch scheduling is employed for the networked set of computers that fit in a single administrative domain. Batch scheduling is most common in managing dedicated clusters for running non interactive job.

Types of batch scheduling:

- FCFS (First come first serve)
- SJF (Shortest job first)
- LJF(Longest job first)
- Advance reservation
- Backfilling
- Preemptive backfilling

FCFS:

- FCFS is the simplest space sharing technique where jobs are considered in order of arrival. It ensures the fairness with respect to job ordering but wastes resources that can actually be used by other jobs. Once the enough processor is vacant, the processors are allocated and the jobs are started. However, if the processors are not vacant, job must wait for presently running job to stop and free the processors.

SJF:

- It works by sorting and executing the shortest job first. It provides extremely good turn around time for short jobs but delays the long jobs.

LJF:

- It works by sorting and executing the longest job first. It maximizes the system utilization at the cost of turnaround time.

Advance reservation:

- In some cases, applications have huge resource requirements and thus needs a concurrent right to use to resources from more than one parallel computer. This is when the advance reservation of resources is employed.

- It stores resources and produces schedule by using execution time predictions provided by the users. Hence, it ensures that the resources are available when needed.

Backfilling:

- The main goal of backfill algorithm is to try and fit small jobs into scheduling gaps.
- It improvises the system utilization by running low priority jobs in between the high priority jobs.
- Runtime estimate of small jobs provided by the users are required by the scheduler in order to use backfill.
- For example, let us take FCFS. When this algorithm is applied, all jobs are considered according to the arrival time. First job will get served first. If there is enough free processors, then they are simply allocated and the job is initiated. However, if there are not enough free processors then the first job will have to stay in queue and wait until there will be enough processor that are free. This will hold up the job as other small jobs will also wait in the queue despite of having few free processor so that the FCFS order is not distorted. This is where backfilling must be employed as it allows the small jobs to move ahead and use the idle processors to run the job without violating the FCFS order.

Preemptive backfilling:

- If the resources are not available for high priority jobs then they can preempt the lower priority jobs.
- However, if non of the high priority jobs are in the queue, then the resources that are reserved for them can be used for running low priority jobs. But, if the high priority job is received then those resources can be reclaimed for high priority jobs by preempting the lower priority jobs.

6. Types of cluster schedulers available:

There are many cluster schedulers available today. Some of them I studied are listed below:

- Maui
- Moab workload manager
- Sun grid engine

I am going to talk about Maui in detail as I have chosen it as my case study. However, I will give basic ideas about the later two for better picture and understanding.

6.1 Sun grid engine:

It is an open source batch queuing system that is maintained by sun Microsystems that basically focuses on managing and scheduling the sharing of distributed resources such as processors, disk space, memory etc. In other words, sun grid engine schedules batch jobs across the nodes.

6.1.1 Enabling sun grid engine to run in a cluster:

There are 3 steps that must be taken into account:

Step 1: Be a super user of the node in the cluster that is responsible for hosting the sun grid engine.

Step 2: Generate a failover resource group to include the sun cluster for sun grid engine resources.

Step 3: Insert a resource for a sun grid engine logical host name to the failover resource group generated in the previous step.

6.1.2 Some features of sun grid engine:

Advance reservation, Array job interdependencies, Enhanced remote execution, Multi-clustering, Daemons managed by the Service Management Facility on Solaris, Pseudo TTY support for interactive jobs, client-side and server-side job verification, GUI Installer

6.1.3 Platform:

- AIX, BSD - FreeBSD, Net BSD, Open BSD, HP-UX ,IRIX, Linux, Mac OS X, Tru64, Windows via SFU or SUA,

6.2 Moab workload manager:

It is a next generation cluster scheduler. Moab workload manager basically have all the functionality that is present in Maui scheduler but has a new architecture that allows for dynamic management of resources.

6.2.1 Some features of Moab workload manager:

Scheduling optimization, Advance reservation, Malleable jobs, Job preemption, Fairness policies, Event policies, Local area grid integration, Multi resource manager support, Generic resource support.

6.2.2 Resource managers:

TORQUE Resource Manager, Open PBS, PBS Pro, Sun Grid engine (SGE), SGE Enterprise Edition (SGEE), Load Leveler, LSF, BProc/Scyld, Scalable System Software (SSS-RM), or Quadrics RMS

6.2.3 Platform:

Linux, Mac OS X, Windows, AIX, OSF/Tru-64, Solaris, HP-UX, IRIX, FreeBSD & other UNIX platforms

6.2.4 Advantages of Moab workload manager:

- Employs intelligent techniques for allocating resources and workload ordering. Thus, it increases the overall cluster performance.
- It reserves the set of resources to use by any group for any period of time.
- It increases the system utilization and decreases the end user support time by modifying already existing jobs.
- It has ability of preempting to perform a higher priority jobs.

7. Case study: Maui cluster scheduler:

7.1 Introduction of Maui:

- Maui is an advanced job scheduler for clusters or super computers that are particularly designed for optimizing system utilization in policy-driven, heterogeneous HPC environments.
- Maui scheduler is considered to be extremely suitable for HPC due to its feature of focusing on fast turn around time of big parallel jobs.
- It is said to be the best open source scheduler.
- Maui is capable of improvising on the manageability and efficiency of machines that range from clusters of few processors to multi- teraflops super computers.
- Two phase of scheduling algorithm are introduced by Maui.

First phase: Advance reservation is employed to schedule high priority jobs.
Second phase: Backfill algorithm is employed to schedule a low priority jobs in between the earlier scheduled jobs.

7.2 Jobs state in Maui scheduler:

There are 3 job states associated with Maui scheduler:

- Running
- Queued
- Non- queued

Running jobs are basically those jobs that have chosen its required resources and have the computation process in progress[10].

Queued job refers to the jobs that are waiting in the queue and are qualified to run. Numbers of jobs a group or users can have in this state are limited [10].

Non queued job refers to the jobs that are not qualified to run [10].

7.3 Job priority calculation with Maui scheduler:

There are few factors that are responsible for calculating Job priority presented by Maui scheduler. They are:

- Resource
- Queue time
- Expansion
- Target
- Fair share
- QoS

Resource factors basically focus on describing the required resources to run the job such as number of processors, amount of memory, size of empty disk space and swap size [10].

Queue time is based on the time the jobs have been in a queued state [10].

Expansion is based on the following equation**:**
XFactor = (Queue Time + Job Time Limit) / Job Time Limit [10]
A job with low time limit will increase its priority more quickly than a long job, pushing it to the front of the queue [10].

If the expansion factor is not enough to meet the scheduling goals, there is a **Target** factor that is increased exponentially as the actual queue time approach the target queue time [10].

The **QoS** factor is a set number used to make up for jobs with high quality-of-service[10] .

7.4 Resource management system in Maui cluster scheduler:

Resource managers:
TORQUE, PBS, Load leveler,SGE, BProc, SSS XML, LSF or Wiki FlatText Scheduling API's

7.4.1 PBS:

PBS stands for portable batch system. It batch jobs by allocating a network resources. It is capable of scheduling jobs to perform on complex multi platform UNIX environment.

There are 3 versions of PBS available:

- Open PBS
- PBS pro
- Torque

Open PBS:

- Original version of PBS.
- It was basically developed for NASA in 1990s.
- Works on complex multiplatform UNIX environments.
- Provides simple FIFO scheduler along with custom scheduler to take full benefit of system specific feature.

PBS Pro:

- Allows preemptive job scheduling
- Has improved fault tolerance
- Highly compatible with open PBS
- Scheduler backfilling
- Improved fault tolerance

Torque:

- It joins with advance schedulers to improvise on scheduling, overall utilization as well as administration on the cluster.

Open PBS VS Torque:

Torque is much better than open PBS in following areas:

- Fault tolerance
- Scheduling interface
- Scalability
- Usability

7.5 Platform:

Linux, AIX, OSF/Tru-64, Solaris, HP-UX, IRIX, FreeBSD, and other UNIX platforms

7.6 Scheduling policies for Maui:

- Advance reservation
- Backfill
- Job Prioritization

Advance reservation:

Maui uses **advance reservation** to ensure the availability of resources at any time. Each reservation consists of following three things:

- Time frame

- List of resources
- Access control list

Maui will check all possible resources and place the required resources in groups that are specified by the task explanation. Access control list is responsible for determining which jobs can use reservation and finally the timeframe specifies when the resources will be reserved or dedicated to jobs that meets the reservation's ACL.

Job prioritization:

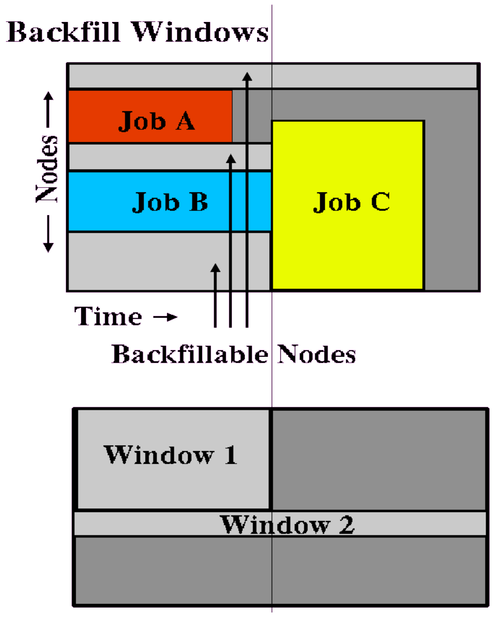
Maui uses a job prioritization approach in order to assign loads to various objectives in order to link overall priority with each possible scheduling algorithm.

Backfill:

It is important to note that each job have a start time as well as a wall clock limit. Therefore, Maui is capable of finding out the finishing time of all the jobs that are in the queue along with the earliest time the resources are going to be free in order to start the high priority jobs. Maui basically uses backfill to be able to make the most out of the available resources.

Backfill algorithm [8]:

- Maui will make two backfill scheduling passes.
- On the first pass, jobs satisfying soft fairness throttling policy constraints are considered and scheduled. And on the second pass, those jobs that meet the less limiting hard fairness throttling policies are considered.



- The above figure shows a batch environment that basically has two running jobs and a reservation for third job.
- The left most end of the box refers to the current time and the future is represented towards right side.
- Idle nodes qualified for backfilling is represent by light grey boxes.
- Now, assuming that the space provided covers 8 nodes with 2 hour time frame, Maui studies the idle nodes in order to determine the backfill window. Maui does this by finding out the largest node- time rectangle.
- Maui determines window 1 and window 2.
- Window one has 4 nodes and only 1 hour of time as some of the nodes are blocked by the reservation for job C. Window 2 has 1 node but since it is not blocked by reservation for job c it has no time limit.
- Once the backfill window is determined, they are traversed by Maui via widest window first or longest window first.[8]

8. Conclusion:

Clusters are quite cost effective compared to the single computer with similar speed and reliability. Scheduling jobs on cluster are required to match task to resources in particular times.This report basically explained about requirements, approaches and algorithms to schedule job on a cluster.

This was a very good topic as it helped me understand the need of scheduling jobs on a cluster and a method to improve the overall job performance with great efficiency.I got to give seminar on the topic that maximized my presentation skill that is vital in my professional life as an engineer.

9. References:

[1]http://searchdatacenter.techtarget.com/sDefinition/0,,sid80_gci762034,00.html
[2] http://www.srce.hr/crogrid/infrastructure/jms.html
[3] http://www.clusterbuilder.org/pages/encyclopedia/alphabetized/m/maui-cluster-scheduler.php
[4] http://www.clusterbuilder.org/pages/cluster-resources/moab-cluster-suite.php
[5] http://www.clusterresources.com/products/maui/
[6] http://en.wikipedia.org/wiki/Computer_cluster
[7] http://en.wikipedia.org/wiki/High_performance_computing
[8] http://www.clusterresources.com/products/mwm/docs/8.2backfill.shtml
[9] http://www.clusterbuilder.org/pages/encyclopedia/alphabetized/s/sun-grid-engine-sge.php
[10] http://www.nsc.liu.se/systems/retiredsystems/grendel/maui.html
[11] http://dcwww.camp.dtu.dk//pbs.html
[12] http://www.sao.nrc.ca/~gabriel/pbs/pbs_user.html
[13] http://www-unix.mcs.anl.gov/openpbs/
[14]http://www.clusterresources.com/pages/products/torque-resource-manager.php