2006:NM02

Resource handling in a cluster environment

Morten Dalhaug

MAGISTER EXAMENSARBETE
Nätverksteknisk Magisterår
Institutionen för teknik, matematik och datavetenskap

MAGISTER EXAMENSARBETE

Resurshantering i en datorklustermiljö

Sammanfattning

Ett stort datorkluster kan genomföra komplexa beräkningar på betydligt kortare tid än det skulle ta för en ensam dator att genomföra samma beräkning. För att utnyttja alla resurser i ett datorkluster måste man använda en resurshanterare. Resurshanteraren har till uppgift att schemalägga beräkningar, detta för att uppnå högsta verkningsgrad. För att resurshanteraren och schemaläggaren skall fungera korrekt måste man veta hur de olika beräkningsprogrammen fungerar, samt vad de olika beräkningsprogrammen har för hårdvarukrav. Beräkningprogrammens funktionalitet och hårdvarukraven ligger sedan till grund för var schemaläggaren skall placera beräkningen. Genom att samla information från beräkningsingenjörerna har vi skapat en lista av parametrar och krav som de olika beräkningsprogrammen har. Detta papper visa hur man kan använda denna information för att optimera beräkningarnas verkningsgrad.

Författare: Morten Dalhaug
Examinator: Stefan Christiernin
Handledare: Andreas Boklund, Högskolan Väst
Program: Nätverksteknisk Magisterår
Ämne: Nätverksteknik **Examensnivå**: Magister
Datum: 2006-06-07 **Rapportnummer**: 2006:NM02
Nyckelord: Kluster, Resurshantering, Applikationsparametrar
Utgivare: Högskolan Väst, Institutionen för teknik, matematik och datavetenskap,
461 86 Trollhättan
Tel: 0520-22 30 00 Fax: 0520-22 32 99 Web: www.hv.se

MASTER'S THESIS

Resource handling in a cluster environment

Summary

Large computing clusters are used to do work that is to complex for a single computer to do on its own within reasonable time. To maximize the resources of a computer cluster a resource manager is used to schedule jobs. To be able to tailor the different jobs and the applications used in a heterogeneous cluster, there needs to be an understanding of the different characteristics of what each job needs and how this affects the placement on the different nodes. To get the information about the applications and the parameters that are important, the engineers that use these applications were consulted. Using information gathered from the engineers, parameters were chosen and an overview of the requirements of the different applications was constructed. This paper will show how to use this information to optimize the use of resources.

Author:	Morten Dalhaug		
Examiner:	Stefan Christiernin		
Advisor:	Andreas Boklund, University West		
Programme:	Master Year in Network Technology		
Subject:	Network Technology	**Level**:	Master
Date:	07 June 2006	**Report Number**:	2006:NM02
Keywords	Cluster, Resource Management, Application Parameters		
Publisher:	University West, Department of Technology, Mathematics and Computer Science, S-461 86 Trollhättan, SWEDEN Phone: + 46 520 22 30 00 Fax: + 46 520 22 32 99 Web: www.hv.se		

Innehållsförteckning

Resource Handling in a Cluster Environment

Morten Dalhaug
University West, Department of Computer Science, Trollhättan, Sweden
Communicating author: morten.dalhaug@student.hv.se

Abstract

Large computing clusters are used to do work that is to complex for a single computer to do on its own within reasonable time. To maximize the resources of a computer cluster a resource manager is used to schedule jobs. To be able to tailor the different jobs and the applications used in a heterogeneous cluster, there needs to be an understanding of the different characteristics of what each job needs and how this affects the placement on the different nodes. To get the information about the applications and the parameters that are important, the engineers that use these applications were consulted. Using information gathered from the engineers, parameters were chosen and an overview of the requirements of the different applications was constructed. This paper will show how to use this information to optimize the use of resources.

Keywords:
Cluster, resource management, application parameters

1. Introduction

When constructing a new skyscraper today the engineers use computer modeling techniques to calculate the strength of winds that affect the skyscraper and what kind of turbulences are generated around the building. This have a large impact on the placement and design of the skyscraper since the forces working on and around the building can be quite severe.

To be able to calculate this wind flow, computer models are amongst the methods used. Different applications use advanced algorithms to calculate how the wind behaves. This can be very computationally intensive because of the "random" movement of the wind. Because of the complexity using only a single computer would be prohibitive because of the time taken to complete the computations. Instead a cluster of computers [1] are used to generate results within a reasonable time.

Today the use of clusters to compute large computations have become commonplace in the research and development efforts of nations and large corporations. By connecting a large number of separate computers, also called nodes, together in one large cluster it is possible to run larger computations than what are possible on a single computer.

These clusters can be either a homogenous system where all the machines are identical or a heterogeneous system that consist of computers with different hardware.

To be able to manage the jobs that engineers want to be able to run on the cluster a resource manager [1, 2, 17] should be implemented. This is an application that takes job requests and put them in a queue and then assigns them to nodes on the cluster depending on where there are free nodes available.

Often the jobs are put into the queue with no regards to where it would be best to place the job. This is mostly a problem in a heterogeneous system where there is different hardware throughout the system. Since the different applications might have different resource demands placing them on the most fitting hardware is very important for optimal resource utilization.

An important factor to consider with regards to homogenous clusters is that when a new job is created on the cluster, it ceases to be homogenous. This is due to the fact that since the new job consumes resources the resources available will no longer be the same and the cluster will therefore no longer be homogenous. This makes it even more important to effectively place jobs to maximize resource use.

To effectively place the jobs amongst the nodes there need to be possible to use characteristics of the different simulation applications to choose where to put the job. This can come both from the configuration files generated by the application when creating a job, or from the engineers themselves.

The goal of this thesis is how to extract this information from the applications used and how to use this information in the scheduler to optimize the use of resources.

2. Background

Clusters have become common in the high tech industry because of their ability to create affordable high performance computing that are able to execute complex simulations in different fields. The

amount of computing power compared to price in clusters has improved 80% each year compared to supercomputers which have only improved 20-30% each year [3]. This makes it much more desirable to create clusters when needing advanced calculations.

The reason for this efficiency is because clusters can be made up of general computers connected through standard Ethernet networks. It is also possible to create clusters based on more specialized hardware and network connections all depending on the level of performance needed for the simulations.

The clusters that are used in this project are separated into three different parts where the main difference that separates them is hardware. The largest part is a 160 node low end cluster that consists of general Intel processors. The second type of cluster is a mid range cluster that consists of machines with AMD Opteron processors. These machines have higher performance and larger RAM. This gives them better performance when it comes to applications that have need for these resources.

The third type of cluster is a high range clusters that consists of four high end Itanium nodes with four processors and eight GB of RAM per node. This gives them the ability to run high end jobs that would need a large amount of memory. Also jobs that have requirement for a high end machine for the master node is possible to run on this cluster. Because this is the smallest of the clusters it is only used for the most demanding of applications.

When running calculations on clusters much of the computation is done in parallel because of the ability to share the job over several nodes. This lends itself well to technical simulation where wind or fluid flow is calculated.

There can be several users that each runs different applications that are using cluster resources at the same time. To be able to efficiently handle the resources of the cluster so that they are utilized as much as possible at all times, it is important to implement a resource manager.

There exist several different resource managers that can be used such as LSF [3, 4, 17], PBS [5, 19], IBM LoadLeveler [6, 18] and SGE [7, 20]. These applications accept jobs submitted from the engineers and place them into a queue. Placement in this queue is based on a priority value that is derived from different policies. Based on this queue the job is placed on the nodes according to priority.

To be able to handle the calculation of priority the resource manager uses a job scheduler. This job scheduler communicates with the resource manager to gather information about queues, node load and available resources.

It is also possible to use an external job scheduler that manages the queue instead of the internal job scheduler. An example of such an external scheduler would be Maui [8, 17] which is a successful extension to the IBM LoadLeveler which has been ported to several other resource managers.

To be able to effectively manage a cluster queue it is necessary to have information regarding the resources available on the cluster. There can for example be a limited number of licenses in use and the scheduler and resource manager need to be able to keep account of how many licenses are available, so it would be possible to schedule the different jobs according to when a license would be free to use.

The important part about a scheduler is to extract parameters from the cluster about what type of resources that are available and where they are located. This can be accomplished either through the scheduler or using external applications. For example data from a parallel project [9] can be used to this end.

When these parameters are gathered they can be coupled up against jobs that are submitted to the scheduler. The importance of the different parameters depends on what type of application that the job is using to do the calculations. Some applications have the need for a fast disk, while others have the need for low latency. What type of parameters that are important for each application will have to be gathered by talking to the users and finding out what they see is important, or tests will have to be run on each application to get statistics that can give a picture of what is important.

Using these parameters the job is put into the queue based on which nodes are available or will be available. Jobs that need a larger number of nodes will usually have to wait longer than those that need fewer nodes. This is because the fewer the nodes the easier it is to find enough free nodes to start the job. The placement of jobs is also very much influenced by what kind of policy that is applied to the queue.

When the necessary nodes are available the scheduled job will be placed on the reserved nodes. It will then have to start the job and notify the user that the job has started on the given nodes. The user that submitted the job will also need to get access to the nodes in case logging or manual configurations need to be made.

When the job is finished the data from the job will have to be transferred to a file server and the user need to be informed that the job is finished and where to find the generated data. Parameters to be used in a self-learning job scheduler will have to be extracted from the job logs after the job has been completed.

3. Method

When looking at how to better tailor the different applications to the queues there are two types of information that are needed. On one hand there

needs to be gathered parameters from the cluster about the status and setup of the nodes. How to gather some of these parameters will be looked at in a parallel project [9].

The other type of information to be gathered is information about the applications themselves. Each application might have different needs depending on how they divide their work. Some might be very sensitive to the network because of frequent communication between the different nodes used in the job. Others might be very dependant on disk input/output (I/O), which would lead to large delays where the processor waits for data from the disk. This would lead to the application needing fast disk on the nodes.

The best way to gather information about the applications is to talk to the engineers about how the applications behave in their work environment. Since the engineers work with these applications on a daily basis they would know how each application works on the cluster and where the different bottlenecks are located.

After talking to the different engineers each application will be mapped up against its strengths and weaknesses and special needs when it comes to how many nodes usually used and disk. Using this information the best placement on the cluster would be decided. Some applications would be better suited to a certain type of hardware, while others would fit more generally or use different hardware depending on special needs for master nodes.

This information would also be used to determine what type of parameters that are needed to help determine placement both on the cluster and in the queue. Depending on the special needs of the applications some jobs might have to wait for certain nodes to become free before starting a job, while others might be put on nodes that might not give the most optimal resource usage, but is needed because of time constraints.

4. Parameters

There are different parameters that the scheduler has to take into account when assigning jobs to nodes. These parameters influence how fast the job will run and which nodes the jobs should use on the cluster. Since different applications differ in where the bottleneck for the performance may manifest itself, the applications should be placed on the cluster where they may utilize the resources as much as possible.

The type of parameters that need to be determined about the applications will have to be how they behave on the cluster and parameters that affect the computation time.

The parameters themselves should be divided into two parts. Pre-job Parameters would be parameters that are necessary to know before the start of the job and which influences the placement in the queue.

Post-job Parameters would be parameters that are extracted after the job is finished and stored to be used in deciding where later jobs would be placed in the queue. This will be important because it would be interesting to create a database of information on finished jobs to better determine the run time of new jobs. The type of the different parameters is detailed in table 1.
Some of these parameters will both need to be used when starting a new job and when extracting information after the job has finished. The use of parameters extracted from finished jobs will not be available in the beginning because of the need for a large number of jobs before any statistical data of significance can be extracted.

4.1. Pre-job Parameters

The first parameter that is important to know of is what kind of application is used to do the job. This is important because each application can have different needs when it comes to usage of hardware such as disk space and RAM. To be able to know what the preferences are for each application the ability to choose the correct nodes to use is possible.

When a job is put in the queue it is important to get a certain time estimate for when the job would be finished. Since this can vary depending on the kind of job that is started and the difficulty of the job, one way to get the time estimate would be for the submitter of the job to give a time estimate of how long the job might take. In the future it would be interesting to use a database over earlier jobs to calculate a close estimate of the total time.

When starting a job it is also important to know how large a job is. This has implications for where the job might be placed on the cluster. If it is a very large job it might have to be placed on nodes that have the capacity to handle such large jobs.

There are also a certain number of parameters that have to come from the applications themselves. To be able to calculate an estimate of run time at a later date, certain parameters that define the length of the job have to be used. There are two types of parameters that are used in most of the jobs. The first one is the number of mesh nodes that the model is built up from. There is usually a direct connection between the number of mesh nodes and the length of the calculation.

The second one is the number of iterations used. Most of the applications iterate through a number of algorithms for each mesh node. How many iterations are needed to go through to get the desired results have an implication on the total length of the job. Other parameters that can be used to calculate total time is time steps which is how long each step should be iterated. The applications

differ in exactly what parameters are necessary to gather.

Another parameter that is necessary to know is how many nodes the job is to be placed on. This depends very much on the application that is to be used. Some applications scale very well, while others do not get any benefit through adding more nodes. It would be important to know how many nodes that are needed for a specific job so that optimal placement on the cluster can be achieved. If for example an application do not get much benefit from adding new nodes it could be scaled down to fit in an earlier but smaller time slot.

When using commercial simulation applications the use of licenses is integral to the use of these applications. Some will use a token system, where the type of job to be done takes a set amount of tokens, and that varies depending on what type of job is run. Other applications use a simpler system where one token is needed per job or per node. To be able to know how many tokens are available is an important parameter to know. If for example a job need to use a certain number of licenses, but not enough is available at the moment, it would be possible to modify the job so that it uses fewer licenses or to start another type of job until the necessary licenses become available.

4.2. Post-job Parameters

The most important of the post-parameters is the total time taken for the job. This is not important to the placement of the job in the short run, but would be important to know to be able to estimate a run time when enough statistics are gathered. This is because to be able to decide total time from the parameters gathered before they are started they would have to know how long time a job of similar length would have taken at an earlier time.

A number of parameters need to be gathered to help decide where the job is going to be placed on the cluster. Due to differences in how each application behaves in relation to hardware demands means that some applications need more of a certain type of resource than other applications.

The parameters that are looked at here are CPU, RAM, Disk I/O, Scratch space and network. CPU is the speed of the nodes that the job is placed on. RAM is how much memory is available. Disk I/O is how fast the application can read and write to disk. Scratch is a temporary file that the applications write to. Network is the latency and bandwidth of the network.

Each of these can affect the run time of jobs depending on whether the part of the cluster that the job is placed on have adequate hardware to support the job. Using information gathered from the applications the resource manager can be instructed to different limits on the jobs, depending on which hardware is needed. It would also be possible to set the best set of resources but make the job able to choose a less optimal placement if not enough free nodes were available on the cluster where it would be preferential to place the job.

These parameters will both be used to help decide the placement on the cluster before the start of the job, but will also need to be gathered after the job is finished to help in the creation of a self-learning database. Using this information the information gathered from the engineers about the applications can be validated or discarded.

Table 1 List of important parameters

Parameter	Use
Application name	Pre
Time estimate	Pre
File Size	Pre
Mesh nodes	Pre
Iterations	Pre
Cluster nodes	Pre
Licenses	Pre
Total time	Post
CPU	Post
RAM	Post
Disk I/O	Post
Scratch	Post
Network	Post

5. Applications

There are several different applications used to calculate different parts. The names and use is detailed in table 2. The reason for choosing these applications was because these were the main applications being used on the clusters today. Since these were in active use, the different characteristics of each application could be gathered from the engineers that used them. This would give a good estimation of the needs of each application in a working environment. The different applications are divided into two types of simulation. These are Finite Element Analysis (FEA) and Computational Fluid Dynamics (CFD) simulations.

Table 2 Simulation applications in use on cluster

Name	Usage
MSC.Marc[11]	FEA
MSC.Nastran[12]	FEA
Ansys[13]	FEA
Fluent[14]	CFD Modeling
CFX[15]	CFD Modeling
Internally Developed	CFD Modeling

Each of the applications has differing needs when it comes to hardware. This affects the placement on the cluster. Some of the applications have high need for RAM while others might need lots of CPU. The information described here is gathered from talking to the engineers that use these applications on a daily basis. There seems to be few papers that have previously tried to gather this information.

5.1. MSC.Marc

MSC.Marc [11] is a nonlinear Final Element Analysis application that enables the simulation of structural integrity and performance of parts in different stress situations from different types of load. It is able to simulate a large number of different materials using an extensive library of material models.

Marc is currently run on the low-end cluster although certain performance characteristics would make it perform better on the mid-level cluster. This is because of the need for large amount of RAM. The nodes that are used to today have not enough RAM to contain the whole dataset used when running a job. This leads to high levels of swap use which can have an impact on the lifetime of the hard drive used in the node.

The input data is divided into one file for each node, and each node does the calculation and data handling on the node itself. There is a master node that controls the execution of the job, but all the output data is saved on the respective nodes so that it is usually not extra taxed in workload, related to being a master node.

While there is no input data being transferred between the nodes there are a lot of network traffic used to coordinate the execution of the job between the slave nodes and the master node. This means that network latency will have an implication on the communication between the nodes and whether the nodes will have to wait for input. Because of this it would be beneficial to put the Marc jobs on a single switch if possible.

Marc gives good performance when running in parallel on several nodes. The usual number of processors that is used to run a job is between 3 and 7. Any more than that and the gains from running in parallel will start to dwindle.

The important parameters to extract from a Marc job would be the number of mesh nodes used, the number of elements used, the number of time steps used and the number of welds in the model. Because the information is stored in normal text files this information is quite easy to extract.

5.2. MSC.Nastran

MSC.Nastran [12] is an implicit nonlinear FEA application that is used in the structural analysis of strengths and fatigue of full body structures. Placement of Nastran jobs is very dependant on which solver that is used. A solver is a number that determine what kind of calculation that is going to be performed on the simulation model.

Some of the solvers for Nastran have no gain of being run in parallel on the cluster, and would therefore be run on the local workstation of the engineer. Other solvers are better suited to be used on the cluster and would therefore have to be submitted to the queue system.

The main bottleneck of Nastran is the disk I/O. Nastran writes heavily to a scratch space on the disk during execution of the job. This makes fast disk a necessity to be able to get the most out of the hardware. Often the processor has to wait for the disk before continuing processing. Because of this RAID-disk with striping would be very advantageous.

Another aspect of this is the need for an optimized scratch space. The best solution would be to have the scratch space on a separate partition not only because it could be optimized in terms of speed but also because the computation would not be affected if the “tmp” directory on the main partition would be filled up.

The important parameters to gather from Nastran would be the number of mesh nodes and the number of iterations. These parameters combined gives an estimate of the amount of time a computation might take. Nastran like Marc use normal text files for the configurations. This makes it easy to extract the needed parameters from the job.

5.3. Ansys

Ansys [13] is a Finite Element Analysis application used to calculate structure and temperature, which is the basis for life time estimates for different structures. It is able to use a large number of different materials using a library of material models. It solves both linear and non-linear solutions problems.

Ansys is quite demanding when it comes to larger jobs. The main bottleneck is the memory usage, but it has also need for much disk space. Because of these characteristics, Ansys jobs are mainly placed on the high end cluster. It is also in need of good disk I/O depending on the size of the simulation.

At the current time Ansys is not run in parallel because of the lack of a parallel solver of good enough efficiency. Instead the jobs are run on a single node using from one to four of the processors on the node to do the calculation using SMP. Ansys would still have to use the queue system since each of the processors on the nodes can be scheduled individually

The parameters that is important to extract from the configuration files of an Ansys job would be mesh nodes and elements, time steps and sub steps. The configuration file of Ansys can be created in several different ways depending upon the choice of the engineer. To select the number of iterations a loop can be used to run through a certain number of steps. Another way is to write the iterations that are to be used. Each step can be divided into sub steps divided using either a set number of steps or using the size of the step.

5.4. Fluent

Fluent [14] is a Computational Fluid Dynamics application that can solve the simulation in both an implicit and explicit way. In this cluster it is solved in an implicit way, because of improved performance in low air speeds compared to the explicit solver.

Fluent jobs can be run in parallel on up to 6-7 nodes, but if the size of the job demands a large amount of RAM, more nodes can be used. The memory on the master node is very important for the job. The CPU is also very important for the master node because it runs more serial jobs than the other nodes.

The jobs can be run in two different ways. In the unsteady way the iterations are divided into time steps and each step can run all its iterations or reach a break point where it has reached a sufficient level of accuracy in the simulation and it will start the next time step. Running the simulation in steady state there are a set number of iterations. The job need not go through all the iterations of the simulation to reach a sufficient level of accuracy. If this happens the simulation will end. This break point is set before the start of the job.

The extraction of parameters can vary depending on how the simulation files are stored. They can either be in standard text format or it can be stored in compressed format using gzip. If they are compressed it has to be uncompressed before they can be read. The files vary between text and binary. In most cases the text part are in the first part of the file and the second part is in binary. It is possible to only extract the information from the first part and use it there. The parameters that would be important are iterations, grid nodes and time-steps.

5.5. CFX

CFX [15] is a Computational Fluid Dynamics application that is implicit and solves the equations in a linear manner. It is based on a mesh built from mesh nodes. The mesh can be very advanced but need to be at least one cell row deep. This means that if a two dimensional area is to be simulated it has to at have least a depth of one row. The application scales quite well in a linear fashion. A general rule is that there should be about 200k mesh nodes on each node.

Depending on how well the job is partitioned it can make good use of CPU power on the nodes. It is also demanding on the memory. If very large jobs are to be undertaking the master node need to be more powerful than the slave nodes, both because of the need for a lot of memory on the master node, and also because when data is written to disk it is the master that is the only node in use. The network is also important to CFX because there are a lot of communications between the nodes during the run of the job. Also since all data is written on the master node, information from the slave nodes need to be transferred over the network to the master node.

The important parameters for CFX are mesh nodes, iterations and time steps. The more granular the mesh is the shorter time steps are necessary. The mesh and configuration files are in a binary format which means there is no easy way to extract information from CFX before the start of the job. At the end of the job there is created a .out file that contain most of the information gathered from the job. A way to get the necessary parameters before the start of the job is to have the engineer enter the parameters that are interesting to the application.

Table 3 Characteristics of applications where "I" is important and "U" is unimportant

	MSC.Marc	MSC.Nastran	Ansys	Fluent	CFX	Internally Developed
CPU	I	U	I	I	(I)	I
RAM	I	U	I	I	I	I
Disk I/O	U	I	I/U	I	U	U
Scratch disk	U	I	U	U	U	U
Network	I	U	U	U	I	I
Nodes	1-7	1-4	1-4	1-8	1-10 (> 250)	1-20 (> 250)
Disk space	< 1 GB	> 100 GB	> 126 GB	> 100 MB	> 1 GB	>100 MB
Comments	Needs lots of RAM, if there are no RAM free uses a lot of SWAP. This has a tendency to wear out the disks quite fast	Needs lots of scratch space, preferably on its own partition.	Very demanding, needs high performance hardware	Can be difficult to estimate time to completion	200000 mesh nodes per CPU node. Master node needs more RAM than slave nodes	Internally developed. Master node needs more RAM than slave nodes

5.6. Internally Developed

The internally developed application is a Computational Fluid Dynamics application that is implicit and solves the jobs in a nonlinear fashion. It has been developed in house at the company where this project was done. This application depends much on the memory and processors of the nodes. It is important to distribute the application evenly amongst the nodes that are to be used to get a good scaling of the application. It is especially important that the master node have a lot of memory.

The master is the one that controls all the slaves and also the one that write information to disk. Network is also important because of communication between the nodes and transferring of results to the master node. The application sends the information seldom, but with large amounts of data each time. This means that it is more the available bandwidth and not the latency that is important to the application.

The important parameters for this application would be time steps and the number of mesh nodes. The time steps are defined from the number of solutions that are to be used in the calculations and the number of steps per solution. This gives the number of total steps for the job. All the information about the job exists in text files which makes it very easy to extract the necessary information from the files.

6. Gathering of parameters

It is important to determine where each of the parameters can be gathered. Some can come from the engineers while others can be gathered from the applications themselves.

There are three different time periods where it is possible to gather parameters. The first time period to gather the parameters is before the job is committed to the job scheduler. This is necessary because there would be a need to find out performance parameters from the configuration files of the application before submitting the job into the queue. Using these parameters it would be possible in the future to calculate an estimate of the run time using historical statistics gathered from earlier jobs. Because these parameters are needed to influence placement they should be gathered before the job is submitted.

The second time period to gather parameters is to extract them from the job scheduler after they have been accepted into the queue and while running. The most important parameter to get here would be the job number assigned when the job is accepted. This would enable the extraction of parameters from the information saved when the job is finished.

The third time period to gather parameters is after the job is finished. In the job scheduler information about each job is stored and can be accessed using commands that are part of the job scheduler. These parameters would give information about what happened during the simulation.

7. Extracting information

Sun Grid Engine (SGE) is an advanced job scheduler that uses policies [16] to schedule different jobs onto a cluster. It has the ability to request different resources either as “hard” resources where they are required to be present for the job to start, or as “soft” resources which are nice to have, but not necessary for the job to start.

SGE uses different types of hosts depending on what role the host should have in the cluster. The hosts that are used to control the resource manager is called the master hosts. This node controls the job scheduler and handles the prioritization of the different jobs. The hosts that have the capability to submit jobs to the resource manager are called submit hosts. These hosts should be the engineers that have the means to start new jobs on the cluster. The third type of hosts is called execution hosts. These are the nodes on the cluster that are used to execute the jobs.

There is also the possibility to create hosts groups that have a certain number of execution hosts belonging to them. This can be used to group similar hosts into a common environment and make it easier to select where to place the job on the queue depending on location. For example can all the nodes under a switch be grouped into a host group so that if a job is sensitive to network latency it can be placed into a single host group and only be run under that switch.

The resource manager also has the ability to change priority on a job based on parameters added when submitting the job. This can be used to for example change the jobs priority based on whether the job has been given a deadline. The ability to change priority should be restricted to only certain personnel so that not everyone will be able to go in and change the priority.

The resource manager also handles the use of licenses through the use of a load sensor that can be created. This makes it possible to schedule jobs based on how many licenses are available. It also protects against jobs submitted to the cluster that is unable to start because there are not enough licenses available.

The test environment as depicted in figure 1 consisted of two nodes identical in hardware to the ones used in the low-end production cluster. One of the nodes served both in the role of a master host and an execution host. The other node was a normal execution node. For submitting jobs to the resource manager there were created scripts for each

application that needed to submit jobs. This was done because there are some differences between the different applications when it comes to starting new jobs.

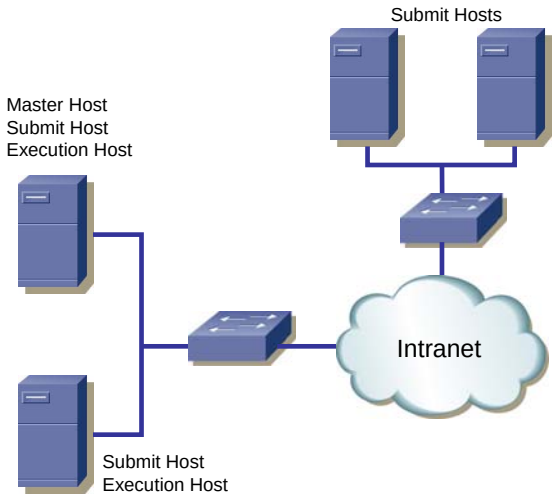

Figure 1 Test Environment

7.1. Variable extraction

To be able to extract the necessary parameters one of the scripts were modified to implement the commands needed to get the parameters. The original file accepted input of different parameters and of the input file that was to be used in the calculation. When the necessary parameters were present the script would use the `qsub` command from SGE to submit the script to the resource manager.

A part of the script extracts the name of the input file. Using this it is possible to open the file and get parameters that would influence the length of the job. This is information that needs to be acquired before the file is submitted to the resource manager because of the possibility at a later date to use this information to estimate the runtime of the job and therefore use this information to better place the job in the queue.

Since there was a script for each application the name of the application to start could be statically entered. Using this application name and the information gathered about the different applications, where to place the job on the cluster could be decided. From a list of different resources available the needs of the application would be matched up and the placement decided.

After these parameters have been extracted the job is submitted to the job system with any modifications needed to decide placement. The best way to affect the placement would be to use the soft and hard resource limits. For example if the job needed a lot of RAM there could be set a hard limit on the need for that amount of RAM. Another example would be if a job were sensitive to the network conditions, the script could check if there were enough free nodes under a single switch, and if so would set the host group to use to be the switch that was found to be acceptable.

When the job is submitted to the resource manager it is assigned a job number. This job number is important to extract from the job system since it will make it possible to extract information about the job once it has finished running. Through the use of the `qstat` command it is possible to get information about running and queued jobs. To get information about the job that was just submitted, the `qstat` command is used to print all jobs that belong to the current user and with the name of the job submitted. If the user has submitted more than one job with the same name, the last job is selected. From this output the job number can be acquired. The script to get the information using `qstat` would be run as a prolog to the job execution. The script that is to be run is possible to be selected in the resource manager.

After the necessary information is gathered with the `qstat` command, the job is executed on the cluster. After it is finished an epilog script can be run. This script would use the `qacct` command to extract information about parameters that are needed after the job is run. This is information that can be saved and the later used to determine how long a given job might take to finish. It will also give stats that can give indication of any bottlenecks that can show up when running a job. This information can also be used later to determine a better placement on the cluster.

When all the information is extracted the information gathered are moved from the local machine to a common network directory where information from all completed jobs is stored. The flow of the extraction is depicted in figure 2.

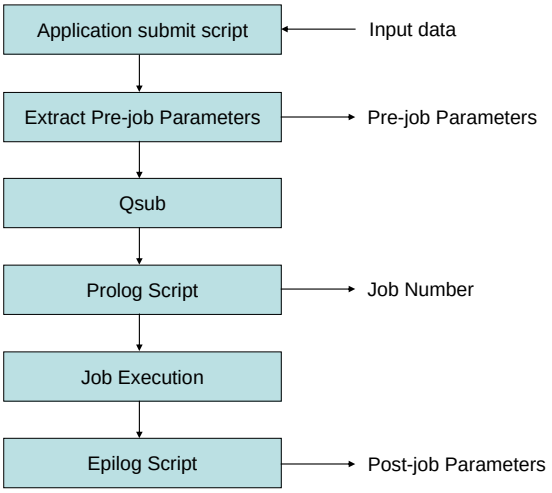

Figure 2 Flowchart of the extraction of parameters

8. Result and discussion

The use of a resource manager can greatly help the planning of resource use. It will also help to utilize the cluster to a much greater degree. Current usage of the cluster is at an average load of 52% and an average node utilization of 63% measured over a month. Variation in usage patterns means that these numbers can vary much from day to day, but to make the most out of the cluster the usage should try to reach upwards to 100% both on load and node utilization. This is of course almost impossible to do because of the natural delays when it comes to transferring data to the different nodes and to start the job, but it would still be possible to reach quite high levels of utilization.

The paper on capacity handling [13] has shown that it is possible to extract information about the cluster and the resources available there. Using this information makes it easier to determine where to place the jobs on the cluster. This paper shows that it is possible to extract information about the usage of the cluster through the use of a resource manager.

Together the information that we gather from this would make it possible to tailor the different jobs placement on the cluster and thus reach a higher level of resource utilization.

An example on how to reach this higher utilization would be to see how a job would be placed on the cluster using these parameters. For example a job that need a lot of RAM, would need to be placed on either the mid range or high range cluster. First the name of the application would be extracted. Using this it would be possible to find out what the needs of the specific application to be used in the job would be.

Then gathering data about the cluster from the capacity handling project [9] it would be seen that the low-range cluster would have 1 GB of RAM while the mid-range cluster would have 4 GB and the high end 16 GB of RAM. Since the high end would only be used if it was necessary, and the low end would not have enough RAM, the job could add an extra parameter used when submitting the job that would restrict the job only to the node group that consisted of the mid range cluster

Another example would be where three different jobs would be placed into the queue system. If an MSC.Marc job, an Ansys job and a Nastran job would enter the job system at the same time, they would each have differing needs and requirements. The Ansys job would need very powerful machines and would be placed on the high end cluster while both MSC.Marc and Nastran job would have need for both CPU and RAM.

The difference would be that Nastran would have need for Disk I/O but would not be sensitive to the network, while the MSC.Marc job would have the opposite priority. This would enable the job scheduler using information gathered from the capacity handling project [9] to place the Nastran job on a part of the cluster that had fast disk, while the MSC.Marc job would be placed under it own switch as to minimize the effect of network traffic. The scheduler would also find out which switch would have the least load on it at the moment and try to find enough free nodes to place the MSC.Marc job there.

9. Conclusion

This paper has showed that the use of a resource manager in a cluster can be an effective method to increase the utilization of the cluster. To be able to do this efficiently it would be desirable to place the different applications that are used on the cluster on the nodes in such a way that the nodes that were most fitting would be used. This can be achieved first by extracting parameters that would be of importance to the different applications. The parameters that were chosen is described in chapter 4 and listed in table 1.

These parameters were decided upon after talks with the engineers that used the cluster in daily use. Using this information it was decided it would be best to split the parameters into two parts. This would be parameters that would be extracted before the job started and parameters that would be extracted after the job was finished. Using this information extracted from the parameters it would be possible to create a database of earlier job runs that could in the future be the basis for a self-learning database that could predict run time of new jobs more accurately.

To be able to place the jobs on the cluster, in a way that would optimize the usage based on what type of need the applications might have, the engineers were also asked about characteristics of the applications. These characteristics were detailed in chapter 5 and an overview of the different applications was shown in table 3. This showed that the applications differed in hardware needs. This would also affect the placement on the cluster.

A practical test was also done to see how parameters might be extracted from the resource manager. This showed that it is possible to extract the needed parameters both before and after the job had run. This would be needed to affect the placement and resource needs of the different applications when they were submitted to the resource manager.

In chapter 8 we presented examples on how the use of these parameters and characteristics of the applications could be used to schedule jobs on the cluster. Using the resource manager it would be possible to place the jobs to utilize the different resources available and raise the utilization of the cluster.

10. Miscellaneous

This paper is best read and understood if you have access to the other two parts of this three way split project. The three projects were conducted by three different persons handling each of their parts. Part one, by Björn Lindberg, is about Availability [10] and the second part, by Thomas Lyshaugen, is about Capacity Handling [9].

11. Future work

Future work that could be done in this field would be to implement a self-learning database that would be able to accurately determine total run time of jobs that are submitted to the resource manager. This can be very important because of the possibility to increase the utility rate of the cluster which will come from better planning of jobs.

Other topics of interest would be to implement more advanced forms of scheduler management, which would be able to use the parameters gathered here to more accurately place jobs on the cluster so as to increase the performance of the running jobs.

One should also implement a graphic view of the cluster queue so that the engineers would be able to place the jobs where they would find it most fitting. To be able to do this in an accurate way the self-learning database should probably also be implemented so that a estimate of how long time a given job would be given, and the choice of where to place the job could be selected from this information.

12. Acknowledgements

We want to thank for all the help in this project our supervisor Andreas Boklund and our examinator Stefan Christiernin at the Division of Computer Science at University West, Trollhättan Sweden.

We also want to thank Susanne A., Göran T. and Christian S. for the help to make this master thesis possible.

Also thanks to all the engineers that helped with the gathering of data from the different applications and parameters.

13. References

[1] Buyya, R. (1999) High Performance Cluster Computing : Volyme 1 Architectures and systems. USA, Upper Saddle River, N. J. ; London: Prentice Hall PTR

[2] Ian Foster, Carl Kesselman, (2004) The Grid 2. Elsevier. St. Louis, MO

[3] Fumie Costen, John Brooke, Moke Pettipher. (1999). Investigation to Make Best Use of LSF with High Efficiency. Cluster Computing, 1999. Proceedings. 1st IEEE Computer Society International Workshop on

[4] Platform Computing (2006). LFS Resource Manager. [Electronic]. Available at: http://www.platform.com [2006-05-17]

[5] Altair Engineering (2006). PBS Professional. [Electronic]. Available at: <http://www.altair.com/> [2006-05-17]

[6] IBM (2006). LoadLeveler. [Electronic]. Available at: <http://www-03.ibm.com/servers/eserver/clusters/software/loadleveler.html> [2006-05-17]

[7] SUN (2006). Sun Grid Engine. [Electronic]. Available at: <http://www.sun.com/software/gridware/> [2006-05-17]

[8] Cluster Resources (2006). Maui Cluster Scheduler. [Electronic]. Available at: <http://www.clusterresources.com/> [2006-05-17]

[9] Lyshaugen, T. (2006). Capacity handling, a necessity in Linux Cluster. Sweden: (Not yet published) . [Electronic]. Available at: <http://www.bibliotek.hv.se/extra/pod/?lang=english>

[10] Lindberg, B. (2006). Computer Availability Within a Computer Cluster. Sweden: (Not yet published). [Electronic]. Available at: <http://www.bibliotek.hv.se/extra/pod/?lang=english >

[11] MSC Software (2006). MSC.Marc. [Electronic]. Available at: <http://www.mscsoftware.com/products/marc.cfm?Q=396&Z=400> [2006-05-17]

[12] MSC Software (2006). MSC.Nastran. [Electronic]. Available at: <http://www.mscsoftware.com/products/nastran.cfm?Q=131&Z=401> [2006-05-17]

[13] Ansys (2006). Ansys Professional. [Electronic]. Available at: <http://www.ansys.com/products/professional.asp> [2006-05-17]

[14] Fluent (2006). Fluent. [Electronic]. Available at: <http://www.fluent.com/> [2006-05-17]

[15] Ansys (2006). Ansys CFX. [Electronic]. Available at: <http://www.ansys.com/products/ cfx.asp> [2006-05-17]

[16] J. H. Abawajy, S. P. Dandamudi. (2003) Parallel Job Scheduling on Multicluster Computing Systems. September 2003, IEEE Conference on Cluster Computing 2003

[17] Brett Bode, David M. Halstead, Ricky Kendall, Zhou Lei. (2000). The Portable Batch Scheduler and the Maui Scheduler on Linux Clusters. Proceeding of the 4th Annual Linux Showcase & Conference, Atlanta.

[18] Agnihotri, P. Agarwala, V.K. Nucciarone, J.J. Morooney, K.M. Das, C. (1998). The Penn State Computing Condominium Scheduling System. Proceedings of the 1998 ACM/IEEE SC98 Conference (SC'98)

[19] Jones, J.P. Nutzberg, B. Henderson, B. (2001). Workload management more than just job scheduling. Proceedings of the 2001 IEEE International Conference on Cluster Computing (CLUSTER'01)

[20] Gentzsch, W. (2001). Sun Grid Engine Towards Creating a Compute Power Grid. Proceedings of the 1st International Symposium on Cluster Computing and the Grid (CCGRID '01)