HIGH PERFORMANCE COMPUTING CLUSTER

An Efficient Global Network

[1]Deepak Chaturvedi
[2]Kashish Gupta

[1]B.Tech-Computer Science Engineering with specialization in Oil & Gas (III Year)
[2]B.Tech-Computer Science Engineering with specialization in Oil & Gas (III Year)

University of Petroleum and Energy Studies, Bidholi via Prem Nagar, Dehradun, Uttarakhand, India

[1]deepakchaturvedi728@gmail.com
[2]kashish.gupta45@yahoo.com

Abstract

What is the limit of technology? It is impossible to judge where the technology will extend its feet in next span of time. In present times most of the technological utilities are functioning due to the IT inputs; which reduces the human efforts and is the backbone of the most of the modern technological systems. As the technology is getting upgraded so computerized systems must also be upgraded by the time and one way to achieve this, is by improving the networks in computing system and this can be done by bringing in use the HPCC networks.

HPCC is the use of parallel processing for running advanced application programs efficiently, reliably and quickly. And the purpose of HPC Cluster is to provide a group of computers which can handle massive jobs easily and are able to process jobs parallely, so that one can save time and money.

A HPC Cluster can be defined as an interconnected network system of information, communication technologies and control systems used to interact with automation and business processes across the entire IT sector encompassing fast data generation, transmission, distribution and consumption with the help of networking.

Hence, our paper lays an objective that after installing a High Performance Computing Cluster we are able to run heavy jobs by reducing the time consumption of its complete process. As it makes system more efficient in point of time and space. HPC system in cluster will overcome all the existing problems in processing, transferring, storing and functioning of data in any interconnected network, as it makes the system and its network more efficient to process the data parallely.

Keywords: HPCC, network system, automation, supercomputing, node, parallel processing.

Introduction

High-performance computing (HPC) is the use of parallel processing for running advanced application programs efficiently, reliably and quickly. The term HPC is occasionally used as a synonym for supercomputing, although technically a supercomputer is a system that performs at or near the currently highest operational rate for computers.

The most common users of HPC systems are scientific researchers, engineers and academic institutions. Some government agencies, particularly the military, also rely on HPC for complex applications. As demand for processing power and speed grows, HPC will likely interest businesses of all sizes, particularly for transaction processing and data warehouses.

High-Performance Computing (HPC) is used to describe computing environments which utilize supercomputers and computer clusters to address complex computational requirements, support applications with significant processing time requirements, or require processing of significant amounts of data. Supercomputers have generally been associated with scientific research and compute-intensive types of problems, but more and more supercomputer technology is appropriate for both compute-intensive and data-intensive applications.

Mechanism: Many computing problems are suitable for parallelization; often problems can be divided in a manner so that each independent processing node can work on a portion of the problem in parallel by simply dividing the data to be processed, and then combining the final processing results for each portion. The most important reason for developing data-parallel applications is the potential for scalable performance in high-performance computing, and may result in several orders of magnitude performance improvement.

Architecture: The HPCC system architecture includes two distinct cluster processing environments, each of which can be optimized independently for its parallel data processing purpose. The first of these platforms is called a Data Refinery whose overall purpose is the general processing of massive volumes of raw data of any type for any purpose but typically used for data cleansing and hygiene, ETL processing of the raw data, record linking and entity resolution.

The second of the parallel data processing platforms is called Roxie and functions as a rapid data delivery engine. This platform is designed as an online high-performance structured query and analysis platform or data warehouse delivering the parallel data access processing requirements of online applications through Web services interfaces supporting thousands of simultaneous queries and users with sub-second response times queries and users with sub-second response times. Roxie utilizes a distributed indexed file system to provide parallel processing of queries using an optimized execution environment and file system for high-performance online processing.

Requirement Specification

Hardware Requirement:

Installer Node/Head Node Minimum Requirements	2 GB of physical memory (RAM) 80 GB of free disk space Two Ethernet interfaces.(public network and compute node connections) DVD drive
Compute Node Minimum Requirements for Package-based Installation	1 GB of physical memory (RAM) 40 GB of free disk space One Ethernet interface
Compute Node Minimum Requirements for Image-based Installation	3 GB of physical memory (RAM) 40 GB of free disk space One Ethernet interface
Compute Node Minimum Requirements for Diskless Installation	4 GB of physical memory (RAM) One Ethernet interface
Compute Node Minimum Requirements without using Platform HPC provisioning	512 MB of physical memory (RAM) 10 GB of free disk space One Ethernet interface
Max Number of Compute Nodes	Platform HPC Work group: 32 nodes Platform HPC Enterprise: Unlimited

Software Requirement:

- ➔ Centos 6
- ➔ Condor
- ➔ Dependencies for condor
- ➔ MPI Libraries

Configuration

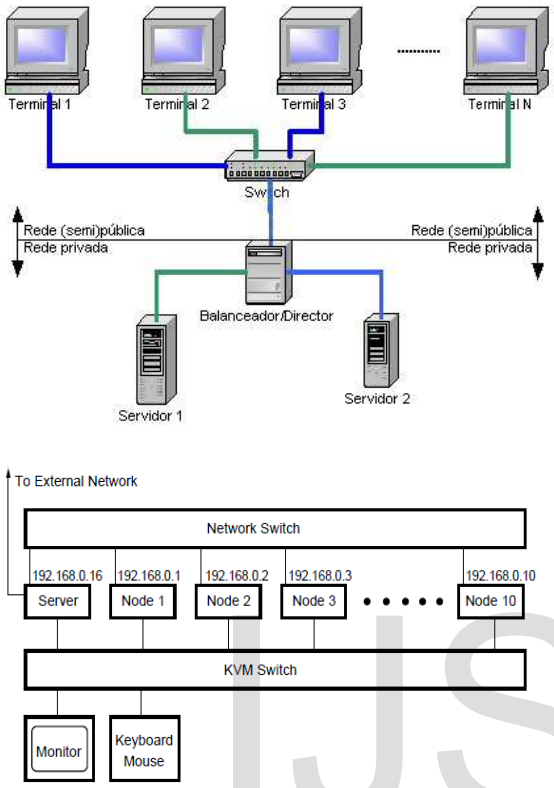

FIG. 1: Illustration of the cluster hardware configuration.

Block Diagram

Steps to Follow

- Install head node and compute node with proper Operating System with all required packages.
- Install and configure condor, MPI (Message Passing Interface) libraries and other required dependencies on each node.
- Check the connectivity between nodes and run the jobs.

Job Submission

In order to submit a job on cluster you have to follow these steps:

- Compile the source code to obtain object file (gcc -c hello.c -o hello.o).
- Link in the Condor libraries (condor_compile gcc hello.o -o hello).
- Create a submit file and this submit description file is placed in the same directory as the executable.
- Submit the program for execution under Condor (condor_submit submit.hello).

Common Commands

- condor_q.
- condor_submit.
- condor_status.
- condor_rm.
- condor_compile.

Features of proposed system

Computer clusters that utilize numerous commodity servers have replaced expensive parallel supercomputers. However, the concerns related to data and its complexities seem to have increased in number. This makes companies invest huge capital on fixed-sized clusters that has their own conventional challenges of increasing the utilization, lessening time-to-result for users and bringing down operational expenses.

HPC clusters or Cloud HPC clusters can create clusters without worrying about having different operating systems, applications, encryption, security and other allied software application. Scientists are able to create clusters that can add servers automatically and turn the servers off when the work is done. This allows the science researchers to carry on calculations only when they require computing power.

In order to have a clear understanding of expenses, let us have a look at the HPC Clusters prior to cloud computing. When purchasing a cluster, researchers would size it to finish their largest calculation suite efficiently in a time they have fixed. Along with

storage, the same sizing and planning would take place to assure that there is sufficient space to support the working data and the outcome of the calculations. Buying the cluster needs heavy capital investments for machines and filers are needed to carry on calculations. Hence, when a cluster is being operated for the first time, it is not in its full capacity.

In the recent past, eminent IT and other companies have come up with built-to-order HPC cluster solutions that offer the customers with speed access to data, manage growth efficiently and seamlessly and bring down risk factors. The design will benefit a wide array of application environments comprising the ones optimized for financial services, manufacturing, industrial design, government, life sciences and education.

Whether your workloads requires a huge complex cluster or an office cluster, eminent service providers allows you in establishing well-integrated and dynamic HPC cluster infrastructure that is simple to manage and implement. Furthermore, the market players in HPC cluster solutions have an alliance with the hardware providers and HPC middleware technologies. This assists the users to have a contact point for a complete cluster application that assures less risk.

Implementation

Condor: Condor is an open source high-throughput computing software framework for coarse-grained distributed parallelization of computationally intensive tasks. It can be used to manage workload on a dedicated cluster of computers, and/or to farm out work to idle desktop computers — so-called cycle scavenging. Condor runs on Linux, UNIX, Mac OS X, FreeBSD, and contemporary Windows operating systems. Condor can seamlessly integrate both dedicated resources (rack-mounted clusters) and non-dedicated desktop machines (cycle scavenging) into one computing environment.

After installing condor, we will run the jobs on our head node and will divide the load on compute nodes of cluster which makes a cluster to run their jobs parallel on each node. This division of load or load balancing is done by condor itself.

Before installing condor we have to make a file system which is accessible from each node in the cluster network, for that we will create a Network File System (NFS). After creating this file system we can easily share data across the cluster, after creating NFS we will install condor and then run the application.

After running the application we have to monitor the load and CPU utilization on each node of cluster, for this we have to install webmin.

Webmin is a web-based system configuration tool for Unix-like systems, although recent versions can also be installed and run on Windows. With it, it is possible to configure operating system internals, such as users, disk quotas, services or configuration files, as well as modify and control open source apps, such as the Apache HTTP Server, PHP or MySQL.

→Install head node and compute node with proper Operating System.

→Install Condor, MPI libraries and other required dependencies.

→Run the applications on head node and parallely divide the load on each compute node.

There are a wide range of high performance computing applications, including:

- ➔ Geological analysis for oil and gas exploration
- ➔ Bioscience and genome mapping
- ➔ Nanotechnology research and development
- ➔ Financial analysis
- ➔ Fluid dynamics
- ➔ DoD testing and simulation
- ➔ Weather forecasting
- ➔ Data mining/predictive optimization

Testing

First Job → The Vanilla Universe:

A first example submits a run of the program mathematica as a vanilla universe job. The vanilla universe does not require a re-compilation of the program using condor_compile. Where the source and/or object code to a program is not available, as

well as for job execution on a Windows machine (where the standard universe is not available), execution under the vanilla universe is the simplest choice.

A very simple submit description file goes with this example. It queues up program mathematica to run one time under Condor.

```
########################
# submit description file for mathematica
########################
executable    = mathematica
universe      = vanilla
input         = test.data
output        = test.out
error         = test.error
log           =  test.log

queue
```

The contents of the submit description file identify the executable, the universe, and they specify files for input, output, error, and logging. This submits description file is placed in the same directory as the executable (mathematica). The input file, test.data, is utilized as standard input for job execution. The output file, test.out, is similar in function, but is used for standard output. Standard error goes to test.error. A log file,test.log, will be produced that contains events the job had during its lifetime inside of Condor. When the job finishes, its exit conditions will be noted in the log file.

Use of a log file is always recommended, as it permits the job submitter to keep track of what happened to the job. No platform (architecture and operating system) is specified, so Condor will use its default assumption: that the executable is meant to run on a machine with the same platform as the machine from which it was submitted. The queue command tells Condor to submit the job for execution one time. Note that the capitalization within the submit description file is unimportant for the commands, but it is preserved for file names.

Second Job → The Standard Universe

This second example job uses the standard universe. Therefore, the program must be linked with the Condor libraries to provide the remote system call technology. The standard universe is supported on a subset of Unix platforms. Preparing this simple job starts with writing, compiling, and submitting the "hello, world" program. The C language source code is written:

```
#include <stdio.h>
int main(void)
{
printf("hello, Condor\n");
return 0;
}
```

This code is compiled to produce object files:

```
gcc -c hello.c -o hello.o
```

Use of the standard universe requires that the program be linked with the Condor libraries to provide the remote system call technology. This I/O support will be needed to send the program's output to a file. Link in the Condor libraries (again using gcc):

```
condor_compilegcchello.o -o hello
```

A very simple submit description file goes with this example. It queues up program hello to run one time under Condor. The submit description file is called submit.hello for this example.

```
########################
# submit description file for hello program
########################
Executable    = hello
Universe      = standard
Output        = hello.out
Log           = hello.log
Queue
```

This submits description file is placed in the same directory as the executable, hello. The executable is specified, as it must be for every job submission. A file where output it to be sent is specified for this program. No input or error commands are given in the submit description file, so there will be no input, and any error messages will be thrown away. (Files stdin, and stderr refer to /dev/null on Unix by default). A log file, hello.log, will be produced that contains events the job had during its lifetime inside of Condor. When the job finishes, its exit conditions

will be noted in the log file. It is recommended to always have a log file to keep track of what happened to the jobs. No platform (architecture and operating system) is specified, so Condor will use its default, which is to run the job on a machine with the same platform as the machine from which it was submitted. The Queue command tells Condor to submit the job for execution one time.

The program is submitted for execution under Condor using the program condor_submit.

condor_submitsubmit.hello

Node failure management

When a node in a cluster fails, strategies such as "fencing" may be employed to keep the rest of the system operational. Fencing is the process of isolating a node or protecting shared resources when a node appears to be malfunctioning. There are two classes of fencing methods; one disables a node itself, and the other disallows access to resources such as shared disks.

The STONITH method stands for "Shoot the Other Node in the Head", meaning that the suspected node is disabled or powered off. For instance, power fencing uses a power controller to turn off an inoperable node.

The resources fencing approach disallows access to resources without powering off the node. This may include persistent reservation fencing via the SCSI3, fibre Channel fencing to disable the fibre channel port or global network block device (GNBD) fencing to disable access to the GNBD server.

Conclusion

After installing a High Performance Computing Cluster we are able to run a heavy jobs on a HPC Cluster over which we will notice the change in time consumed after whole execution, the time consumed in running job on a HPC Cluster is less than a normal single computer.

HPC Cluster provides flexibility and reliability to users, saves time and/or money, and provides concurrency.

Applications of HPCC

- WRF (Weather Research Forecasting).
- SCADA (Supervisory Control and Data Acquisition).
- Other heavy applications.

Bibliography

- http://research.cs.wisc.edu/condor/yum/
- http://www.cis.upenn.edu/~amir/cis501-F04/simplescalar/condor.html
- http://research.cs.wisc.edu/condor/quick-start.html
- http://research.cs.wisc.edu/condor/
- http://spinningmatt.wordpress.com/2011/06/12/getting-started-creating-a-multiple-node-condor-pool/
- http://research.cs.wisc.edu/condor/manual/v6.7/2_11Parallel_Applications.html
- http://research.cs.wisc.edu/condor/manual/v6.7/2_11Parallel_Applications.html#SECTION00311300000000000000