INTERNATIONAL JOURNAL OF RESEARCH IN COMPUTER APPLICATIONS AND ROBOTICS
ISSN 2320-7345

INCREASING PERFORMANCE OF CLUSTER USING HPC

P. AshwiniReddy [1], V.Mounika[2], A.RojaRamani[3]

[1] Associate Professor; Department of Computer Science& Engineering, Teegala Krishna Reddy Engineering College, Hyderabad, T.S-500 097, India
ashwinireddy90@gmail.com
[2]Asst.Professor; Department of Computer Science &Engineering, Teegala Krishna Reddy Engineering College, Hyderabad, T.S-500 097, India
mounikatkrcse@gmail.com
[3]Asst.Professor; Department of Computer Science &Engineering, Teegala Krishna Reddy Engineering College, Hyderabad, T.S-500 097, India
rojadabbulu@gmail.com

Abstract: - Interest in HPC in the cloud has been growing over the past few years. High-performance computing begins with understanding what you are trying to achieve, the assumptions you make to get there, and the resulting boundaries and limitations imposed on you and your HPC system. Cloud is not HPC. Broader use of HPC in the cloud also presents some key challenges. Primary among them is the lack of high-speed interconnects and noise-free operating systems to enable tightly coupled HPC applications to scale. High performance computer is so that the individual nodes can work together to solve a problem larger than any one computer can easily solve. The systems that function above a teraflop or 10^{12} floating-point operations per second. High performance computing is using a supercomputer (or a tight cluster of computers working as one) to process information.

Keywords: Generic cluster layout, HPCC system architecture, HPC cluster, Two-node cluster design.

Generic cluster layout:

The architecture of a cluster is pretty straightforward. You have some servers (nodes) that serve various roles in a cluster and that are connected by some sort of network. That's all. It's that simple. Typically the nodes are as similar as possible, but they don't have to be; however, I highly recommend that they be as similar as possible because it will make your life much easier. Fig 1 is a simple illustration of the basic architecture.

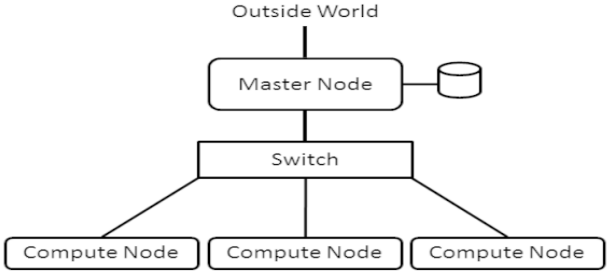

Fig1: **Generic cluster layout.**

Almost always you have a node that serves the role of a "master node" (sometimes also called a "head node"). The master node is the "controller" node or "management" node for the cluster. It controls and performs the housekeeping for the cluster and many times is the login node for users to run applications. For smaller clusters, the master node can be used for computation as well as management, but as the cluster grows larger, the master node becomes specialized and is not used for computation. Other nodes in the cluster fill the role of compute nodes, which describes their function. Typically compute nodes don't do any cluster management functions; they just compute. Compute nodes are usually systems that run the bare minimum OS – meaning that unneeded daemons are turned off and unneeded packages are not installed – and have the bare minimum hardware. As the cluster grows, other roles typically arise, requiring that nodes be added. For example, data servers can be added to the cluster. These nodes don't run applications; rather, they store and serve data to the rest of the cluster. Additional nodes can provide data visualization capabilities within the cluster (usually remote visualization), or very large clusters might need nodes dedicated to monitoring the cluster or to logging in users to the cluster and running applications.

HPCC system 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, extract, transform, load processing of the raw data, record linking and entity resolution, large-scale ad-hoc complex analytics, and creation of keyed data and indexes to support high-performance structured queries and data warehouse applications. The data refinery is also referred to as Thor, a reference to the mythical Norse god of thunder with the large hammer symbolic of crushing large amounts of raw data into useful information. A Thor cluster is similar in its function, execution environment, file system, and capabilities to the Google and Hadoop Map Reduce platforms.

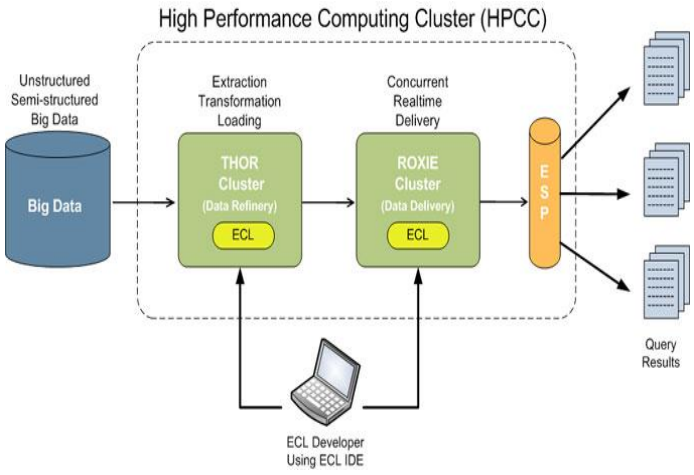

Fig-2

HPC system.in parallel computing using multiple nodes, you need at least two separate systems (nodes), each with its own operating system (OS). To keep things running smoothly, the OS on both nodes should be identical. (Strictly speaking, it doesn't have to be this way, but otherwise, it is very difficult to run and maintain.) If you install a package on node 1, then it needs to be installed on node 2 as well. This is a source of possible problems when you have to debug the system. The second thing your cluster needs is a network to connect the nodes so they can communicate to share data, the state of the solution to the problem, and possibly even the instructions that need to be executed. The network can theoretically be anything that allows communication between nodes, but the easiest solution is Ethernet. In this article, I am initially going to consider a single network, but later I will consider more than one.

Two-node cluster design:

For a simple two-node cluster with HPC, design one master node and one compute node. However, because we have only have two nodes, applications would most likely run on both – because why waste 50% of your nodes? The network connecting the cluster nodes could be any networking technology, but the place to start is with wired Ethernet, which ranges from 100Mbps to 56Gbps; however, I'll stick to the more common Fast Ethernet (100Mbps) or Gigabit Ethernet (1,000Mbps).The network topology you use for clusters is important because it can have an effect on application performance. A simple network layout has a single switch with all nodes plugged in to that switch. This setup, called a fat tree topology, has only one level and is simple and effective, particularly when building smaller systems. As systems get larger, you can still stick to the fat tree topology, but we will likely have to have more levels of switches. To re-use existing switches, design the topology carefully to avoid bottlenecks. For smaller systems, Ethernet switches are pretty inexpensive, costing just a few dollars per port. Switches are going to be better than an Ethernet hub. Although hubs will limit performance, it won't stop the cluster from working. Because our interest in "high performance," and want to do everything possible to keep the cluster network from reducing performance. A common approach is to put the cluster on a private Ethernet network. The address space is uncountable, so the compute nodes will effectively be "hidden" from a routable network, allowing you to separate your cluster logically from a public network. However, a good idea would be to log in to the cluster from a public network, and the way to do that when the cluster is on a private network is to add a second network interface controller (NIC) to the master node. This NIC will have a public IP address that allows you to log in to the cluster. Only the master node should have the public IP address, because there is no reason for compute nodes to have two addresses. (You want them to be private.) For example, you can make the public address for the master node something like 72.x.x.x and the private address something like 10.x.x.x. The order of the network interfaces doesn't make a huge difference, but you have to pay attention to them when installing the OS. You can give the master node two private addresses if you are behind a network address translator (NAT). This configuration is very common in home routers, which are also NAT devices. For example, in my home network, I have an Internet router that is really a NAT. It converts packets from a private network, such as 192.168.x.x, to the address of the router (the Internet) and vice versa. My simple clusters have a master node with a public IP of 192.168.x.x, and they have a second NIC with an address of 10.x.x.x, which is the cluster's private network. Another key feature of a basic cluster architecture is a shared directory across the nodes. Strictly speaking this isn't required, but without it, some MPI applications would not run. Therefore, it is a good idea simply to use a shared file system in the cluster. NFS is the easiest to use because both server and client are in the kernel, and the distribution should have the tools for configuring and monitoring NFS. The classic NFS approach to a shared directory is to export a directory from the master node to the compute nodes. You can pick any directory you want to export, but many times, people just share /home from the master node, although sometimes they will also export a new directory, such as /shared. The compute nodes also mount the shared directory as /home. Therefore, if anything in /home is local to each node, it won't be accessible. Of course, you can get much fancier and more complicated, and you might have good reasons to do so, but in general you should adopt the KISS (Keep It Simple Silly) approach. Simple means it is easier to debug problems. Simple also means it's easier to reconfigure the cluster if you want (or need). With the architecture established, I'll turn to the software you'll need.

Conclusion:

The first opportunity is in optical networking, which will improve hardware manageability and interconnect performance while also reducing power consumption, and software-defined networks, which will further improve the manageability of cloud networking. Furthermore, developments in non-volatile memory will improve check pointing performance and, in the long term, address the data deluge and new programming models leveraging non-volatility. Both technologies will create disruptive improvements and increase HPC's adoption in the cloud. For improved computational performance or improved storage performance, though, you might want to contemplate separating the traffic into specific networks. For example, you might consider a separate network just for administration and storage traffic, so that each node has two private networks: one for computation and one for administration and storage. In this case, the master node might have three network interfaces.

REFERENCES

[1] Handbook of Cloud Computing, "Data-Intensive Technologies for Cloud Computing," by A.M. Middleton. Handbook of Cloud Computing. Springer, 2010.
[2] "HPCC Systems: Introduction to HPCC (High-Performance Computing Cluster)". CiteSeerX. 24 May 2011. Retrieved 29 October 2015.
[3] Handbook of Data Intensive Computing, "ECL/HPCC: A Unified Approach to Big Data," by A.M. Middleton. Handbook of Data Intensive Computing. Springer, 2011.
[4] "LexisNexis Will Open-Source Its Hadoop Alternative for Handling Big Data". Read Write. 15 June 2011. Retrieved 20 November 2014.
[5] "9 Useful Open Source Big Data Tools". Enterprise AppsToday. 11 Nov 2015. Retrieved 18 November 2015.
[6] "A Performance and Cost Analysis of the Amazon Elastic Computer Cloud (EC2) Cluster Compute Instance" by Fenn, M., Holmes, J., Nucclarone, J., and Research Computing and Cyber infrastructure Group, Penn State University.

AUTHOR DETAILS:

P. AshwiniReddy is Associate Professor; Department of Computer Science& Engineering, Teegala Krishna Reddy Engineering College, Hyderabad, T.S-500 097, India.
ashwinireddy90@gmail.com

V. Mounika is Asst.Professor; Department of Computer Science &Engineering, Teegala Krishna Reddy Engineering College, Hyderabad, T.S-500 097, India.
mounikatkrcse@gmail.com

A.RojaRamani is Asst.Professor; Department of Computer Science &Engineering, Teegala Krishna Reddy Engineering College, Hyderabad, T.S-500 097, India.
rojadabbulu@gmail.com