High Performance Computing with Linux Clusters

Panduranga Rao MV Mtech, MISTE
Lecturer, Dept of IS and Engg.,
JNNCE, Shimoga, Karnataka
INDIA
email: raomvp@yahoo.com
URL: http://www.raomvp.bravepages.com/

Roopa K. BE.,
Lecturer, Dept of CS and Engg.,
JNNCE, Shimoga, Karnataka
INDIA
email: roopasindhe@lycos.com

Building a supercomputing cluster on the Linux platform is a matter of interconnecting existing systems, communications, architecture, programming, applications, and algorithms.

1. Abstract:

The design objective of the cluster under Linux is to deliver the highest parallel processing performance for the lowest price. We discuss the advantages of using Linux environment for building the cluster, Logical view of a Linux cluster, methods of programming parallel computers andConcept of sharing storage space in a cluster ensuring the consistency of data across various nodes. The suitable approaches are Shared disk, Shared nothing and Mirrored disk. The techniques of Algorithm design and tuning strategies, also Hardware components necessary for Linux cluster are briefed.

2. Introduction:

Many PCs today have more processing power than the record-breaking supercomputers of yesterday. Incidentally, the IT industry does not call these machines supercomputers. They are better known as high performance machines and the area is known as HPC (High-performance Computing). In fact, most of the supercomputers are based on comparatively slower processors. The earliest is vector processing, made famous by Cray supercomputer. Then came parallel processing, which is today the most widely used method for creating supercomputers. Beowulf, using commodity machines to build high-performance clusters, emerged out of this. Clusters with over a thousand processors are generally called MPPs (Massively Parallel Processors). The first Beowulf cluster was built in 1994, by Donald Becker and Thomas Sterling from 16 Intel DX4 processors connected by a channel-bonded 10 Mbps Ethernet, and it ran Linux. After the success of the first Beowulf cluster, several more were built by others using several generations and families of processors and network interconnects. Today, Beowulf indicates an arena of high-performance clusters built out of commonly available parts, running Linux, Solaris or Windows. One of the plus points of a Beowulf cluster is that almost anyone can build a high-performance cluster. The main drawback of a standard cluster architecture is the poor performance of the standard support to inter-process communication over any LAN hardware. Current implementations of industry-standard communication primitives (RPC), APIs (sockets), and protocols (TCP, UDP) usually show high communication latencies and low communication throughput.

2.1 Analysis:

A cluster is a type of parallel or distributed processing system that consists of a collection of interconnected stand-alone computers working together as a single, integrated computing resource. A node of the cluster can be a single or multiprocessor computer. Each node has its own memory, I/O devices and operating system. Typically, a cluster will appear as a single system to users and applications.

A cluster consists of two or more interconnected computers sharing memory, storage space or processing power, designed to operate as a logically unified system. In other words, a server cluster is a group of small, independent servers managed as a single system. These smaller servers, individually called nodes, appear as a single logical unit to the user and share a common service or responsibility. The key benefits of using server clustering are in availability, performance and management. Servers in a cluster are available to back each other up in case one of them fails. In advanced systems this changeover can take place almost instantly. The advantage in performance arises from the ability to load balance the cluster as a whole. A cluster is also easier to manage than individual servers as it appears as a single logical entity.

3. The advantages of using Linux environment for building the cluster:

- Linux software is usually free of charge and available in source form. We are able to find bugs and write extensions ourselves, which eliminates the long wait for the vendor of a commercial package to respond. There exists Linux software for nearly any application. While written documentation is sometimes scarce the support through mailing lists and other Internet resources is usually better than for most commercial software.
- By cross mounting /home and /usr/local and installing the same packages on all machines the users get the exact same environment, regardless which machines they are sitting on. This provides for much better utilization of resources. Also, backup of one central /home is easy. Because users have complete control over their own home directory they can still customize their environment to their hearts content.
- The Linux kernel provides a filesystem-like interface into the virtual memory system. That makes it simpler to create page-based shared memory systems that allow the entire memory of a cluster to be accessed transparently from any node.
- Integrating */proc* filesystems across a cluster would allow standard Unix tools for process management and job control to work transparently on the virtual machine without the need for any modifications, because most of these utilities get all of their information from the */proc* filesystem.
- By cross mounting /var/spool/mail each user has her mail delivered "locally". This opens the mailsystem up for customization and filtering that would be much more difficult with POP or IMAP.
- All processes running on a Linux / Unix machine have a unique Process ID. In a parallel distributed processing system, you want to have unique process IDs across an entire cluster, spanning several kernels.
- In a cluster, every machine is responsible for running its own copy of the kernel, and nodes are generally autonomous at the kernel level. However all the kernels participate in a number of global namespaces, such as the Global Process ID space.

- Configuration and updating has to be done only once. With our setup, here it will automatically spread to other machines.
- ASCII config files, text processing tools and scripting languages make custom configuration easy. Repetitive tasks can be done automatically on each computer because no GUI is involved.
- Remote execution and control of computers via the SSH (Secure Shell) is seamless and can be done automatic (password less) and secure.
- Much better security and performance. Stability of the Linux OS is definitively better than Windows, but slightly worse than other free systems like FreeBSD and OpenBSD.

4. Logical view of a Linux cluster:

A Linux cluster uses a multicomputer architecture. It features a parallel computing system that usually consists of one or more master nodes and one or more compute nodes, or cluster nodes, interconnected via widely available network interconnects. All of the nodes in a typical Linux cluster are commodity systems—PCs , workstations, or servers—running commodity software such as PVM or MPI.

The master node acts as a server for Network File System (NFS) and as a gateway to the outside world. As an NFS server, the master node provides user file space and other common system software to the compute nodes via NFS. As a gateway, the master node allows users to gain access through it to the compute nodes. Usually, the master node is the only machine that is also connected to the outside world using a second network interface card (NIC).

The sole task of the compute nodes is to execute parallel jobs. In most cases, therefore, the compute nodes do not have keyboards, mice, video cards, or monitors. All access to the client nodes is provided via remote connections from the master node. Because compute nodes do not need to access machines outside the cluster, nor do machines outside the cluster need to access compute nodes directly, compute nodes commonly use private IP addresses, such as the 10.0.0.0./8 or 192. 168.0.0/16 address ranges.

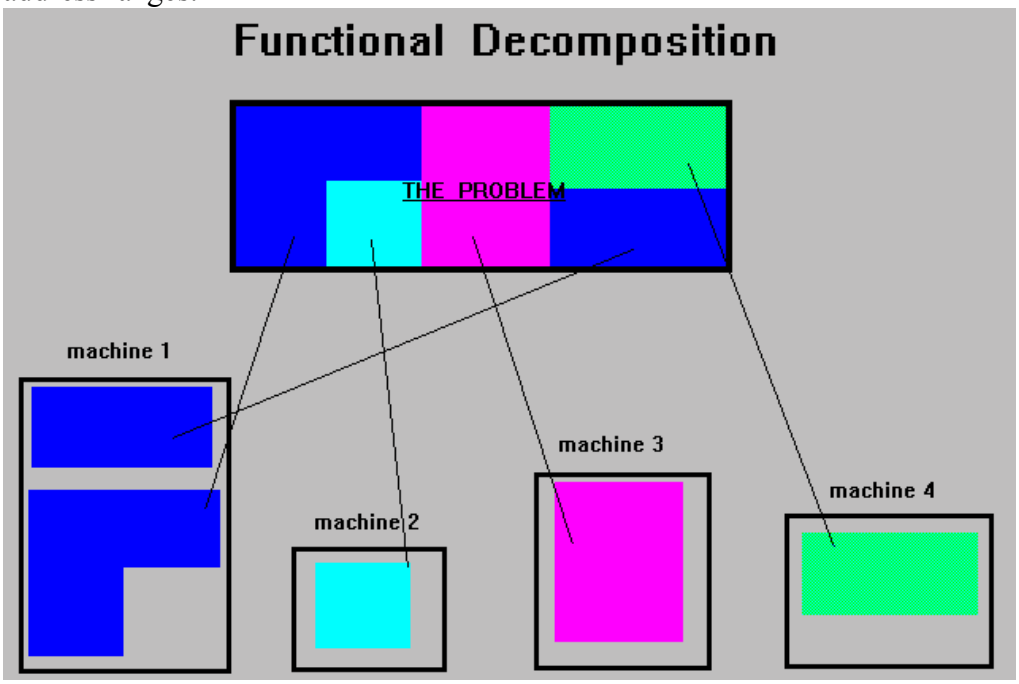

Figure: Execution of a scientific program through the cluster of many nodes.

Applications must be written in parallel style and use the message-passing programming model. Jobs of a parallel application are spawned on compute nodes, which work collaboratively until finishing the application. During the execution, compute nodes use standard message-passing middle-ware, such as Message Passing Interface (MPI) and Parallel Virtual Machine (PVM), to exchange information. There are many methods of programming parallel computers. Two of the most common are message passing and data parallel.

- Message Passing - the user makes calls to libraries to explicitly share information between processors.
- Data Parallel - data partitioning determines parallelism.
- Shared Memory - multiple processes sharing common memory space.
- Remote Memory Operation - set of processes in which a process can access the memory of another process without its participation.
- Threads - a single process having multiple (concurrent) execution paths
- Combined Models - composed of two or more of the above.

5. Sharing storage space:

The most important *component* of clusters is the storage space. How do the individual nodes share storage space? How is consistency of data ensured across various nodes, especially in case of a failure of a node? There are three main ways in which a cluster can be set up.

- Shared disk
- Shared nothing
- Mirrored disk.

5.1 Shared disk:

In the shared disk configurations all nodes share a common storage space, but use their own memory and processing power. This removes the problem of data inconsistency, as all changes are made available to other nodes instantly, removing the time lag which exists before changes are reflected in the storage areas of individual nodes in case of shared nothing nodes. However, there are two main problems with this technique. Firstly, that of a single point of failure—if the storage space goes down, then so does the whole cluster. Secondly, such a setup often requires special arrangements to ensure that the simultaneous reads and writes of various nodes are synchronized properly.

5.2 Shared nothing:

Here the clusters are made up of nodes that have independent storage systems in addition to independent memory, processing power and other resources. These nodes exchange information using standard messages. Changes made by a node in its storage area are mirrored across other nodes. This provides for automatic backup of data in case of the failure of one or more nodes. Also, there is no need of any special techniques to co-ordinate the reads and writes of various nodes as these are now independent processes. But, this technique too has its drawbacks. Firstly it suffers from a time lag before the changes are reflected across all nodes, as discussed above. Also, it involves higher costs as it requires large amounts of storage space.

5.3 Mirrored disk:

Mirroring involves replicating all the data from a primary storage to a secondary storage device for availability and reliability purpose. Replication occurs while the primary system is online. If a failure occurs, the fail-over process transfers control to the secondary system. However, some applications can lose some data during the fail-over process. One of the advantages of using mirroring is that your network doesn't crash due to disk failure, nor is there any data loss. However, it may not be economical due to the redundant disks.

6. Dynamic Applications and Algorithms:

Clusters are used in many scientific disciplines, including biology (genome mapping, protein folding), engineering (turbo-fan design, automobile design), highenergy physics (nuclear-weapons simulation), astrophysics (galaxy simulation) and meteorology (climate simulation, earth/ocean modeling).

- *Internet applications*: Systems like Linux Virtual Server directs clients network connection requests to multiple servers that share their workload.
- C*ompression*: Systems like Evoke Communications speech-to-email service uses clusters to perform transcoding that meets real-time requirements.
- *Data mining*: Efforts like Terabyte Challenge uses clusters at multiple sites to manage, mine, and model large distributed data sets for high-energy physics, health care or web crawlers.
- P*arameter study*: Tools like Nimrod uses a cluster to execute task farming and parameter study applications (the same program repeatedly executed with different initial conditions) as a means of exploring the behavior of complex systems like aircrafts or ad-hoc networks.
- *Image rendering*: A ray tracer can distribute the rendering among different nodes.
- *Generic:* Numerical algorithms, Graph algorithms, Genetic algorithms, Computational geometry, Searching and optimization, Parallel simulation, Climate ocean modeling, Molecular modeling.

7. Techniques of Algorithm Design and Tuning strategies:

It is important to show by example how to design and implement programs that make use of the computational power provided by clusters.

- Process-level parallelism
- Partitioning and divide-and-conquer
- Communication and synchronization
- Agglomeration and mapping
- Load balancing and termination detection

Parallel algorithms can be categorized according to a number of criteria, including regular or irregular, synchronous or asynchronous, coarse or fine grained, bandwidth greedy or frugal, latency tolerant or intolerant, distributed or shared address space.

8. Hardware Components of Linux Cluster:

We used the following hardware for setting up our cluster. You should have at least a 2 GB or bigger hard disk on the server. It should have a graphics card that is

supported by PCQLinux 7.1 or newer version and a floppy drive. You also need to plug in two network cards—preferably the faster PCI cards instead of ISA—supported by PCQLinux. The physical networking medium can be anything from a 10/100 Mbit Ethernet or Gigabit Ethernet or even a high-speed fiber-optic channel. We used a 100/10 Mbps switch. A 100 Mbps switch is recommended because the faster the speed of the network, the faster is the message passing. All cluster nodes will also be connected to this switch.

The domain controller was an IBM Netfinity PIII Xeon 550 MHz with 256 MB RAM. The clients used were an assembled P-II 233 with 128 MB RAM, a P II 300 MHz with 196 MB RAM, and a HP Brio 400 (Celeron 400 with 128 MB RAM). All the three client machines are used for regular desktop work. The network inter-connect was 100 Mbps Ethernet running over a 3Com LinkBuilder switch.

9. Conclusions:

Cluster under Linux allow parallelism only within a given "step" of a solution. Many scientific problems can benefit from domain decomposition: a splitting up of space or time into relatively independent chunks, which can be processed on separate nodes in a cluster. Running the Linux operating system, becoming more and more attractive as cheap and efficient platforms for parallel and distributed applications. It is very much scalable, No need for Special Hardware, Easy Backups, Recovery, Data Consistency and Integrity.

10. References and Acknowledgements:

- G. Pfister, “An Introduction to the InfiniBand Architecture”, *High Performance Mass Storage and Parallel I/O*, IEEE Press, 2001. www.infinibandta.org
- G. Ciaccio, G. Chiola, Low-cost Gigabit Ethernet at work, (poster) in proceedings of CLUSTER 2000,Chemnitz, Germany, November 28 - December 1, 2000. IEEE
- Ragsdale, Susan, ed., "Parallel Programming", McGraw-Hill, Inc., New York.
- D. Spector, *Building Linux Clusters*, O’Reilly, 2000.
- S. Roosta, *Parallel Processing and Parallel Algorithms: Theory and Computation*, Springer Verlag, 2000.
- B. Wilkinson and M. Allen, *Parallel Programming,* Prentice Hall, 1999.
- G. Ciaccio, M. Ehlert, B. Schnor, Exploiting Gigabit Ethernet Capacity for Cluster Applications, in proceedings of LCN 2002, workshop on High-Speed Local Networks (HSLN), Tampa, Florida, November 6 - 8, 2002 IEEE.