IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 7, Issue 11, November 2018

Beowulf Cluster- PXSMAlg Parallel Exact String-Matching Algorithm

Ninad Jagadish Jadhav

Graduate Student, Vidyavardhini's College of Engineering and Technology, University of Mumbai, Mumbai, India

Abstract: The needs and expectations of modern-day applications are changing in the sense that they not only need computing resources (processing power, memory or disk space), but also the ability to remain available to service user requests almost constantly 24 hours a day and 365 days a year. These needs and expectations of today's applications result in challenging research and development efforts in both the areas of computer hardware and software. Parallel supercomputers have been in the mainstream of high-performance computing for the last ten years. The decline of the dedicated parallel supercomputer has been compounded by the emergence of commodity-off-the-shelf clusters of PCs and workstations. Hence, the idea of the cluster came into front with certain recent technical capabilities, particularly in the area of networking, have brought this class of machine to the vanguard as a platform to run all types of parallel and distributed applications. This literature provides all the technical details required to implement the Beowulf Cluster in order to achieve high performance as well as high availability. The literature gives an overview of the features the system provides with.

Keywords: Beowulf Cluster, Building a Beowulf Cluster, Pattern Matching, Handling Node Border Issues

I. INTRODUCTION

A computer cluster is a group of linked computers, working together closely thus in many respects forming a single computer. The components of a cluster are commonly, but not always, connected to each other through fast Local Area Networks. "Beowulf is a multi-computer architecture which can be used for parallel computations. It is a system which usually consists of one server node, and one or more client nodes connected together via Ethernet or some other network". The concept of Beowulf clusters was originated at the Center of Excellence in Space Data and Information Sciences [CESDIS] located at NASA Goddard Space Flight Center in Maryland by Thomas Sterling and Donald Becker. The first Beowulf cluster was built from 16 Intel DX4 processors connected by a channel-bonded 10Mbps Ethernet ran on LINUX. The Cluster was named after an epic poem, Beowulf.

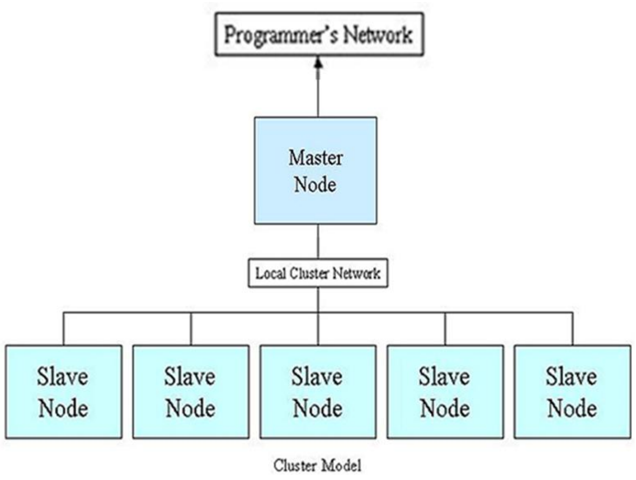


Fig.1 Beowulf Cluster

II. IMPORTANT CONSIDERATIONS FOR BUILDING A BEOWULF CLUSTER

A. PROCESSING SPEED: The master node can be kept busy doing out work to the other nodes, a slowdown here can have a huge negative impact on the entire cluster as the slave nodes waste time waiting for their next instruction.

B. NETWORK SPEED: The faster the network, the better the performance of the cluster.

C. RAM: is crucial in the master node for two reasons. First, the more RAM, the more processes can be run without accessing the disk. Second, the Linux kernel can and will cache its disk writes to memory and keep them there until they must be written to disk. For installing OS When installing Linux on the master node, it is recommended to use separate partitions. This is not necessary, but it can allow for easier administration in the long run.

For the slave nodes, we just partition the root (/) and the swap partition, since the applications will all be installed on the master node's usr/local partition, and since all user files will be stored on the master nodes /home partition. The /home and /usr/local of the slave nodes are mounted to those on the master node, so they can be any size on the slave node. If you type "cd /home or cd /usr/local" on the slave node, we are actually going to the directory on the master node and if the firewall is installed, make sure that it allows connections via SSH and RSH.

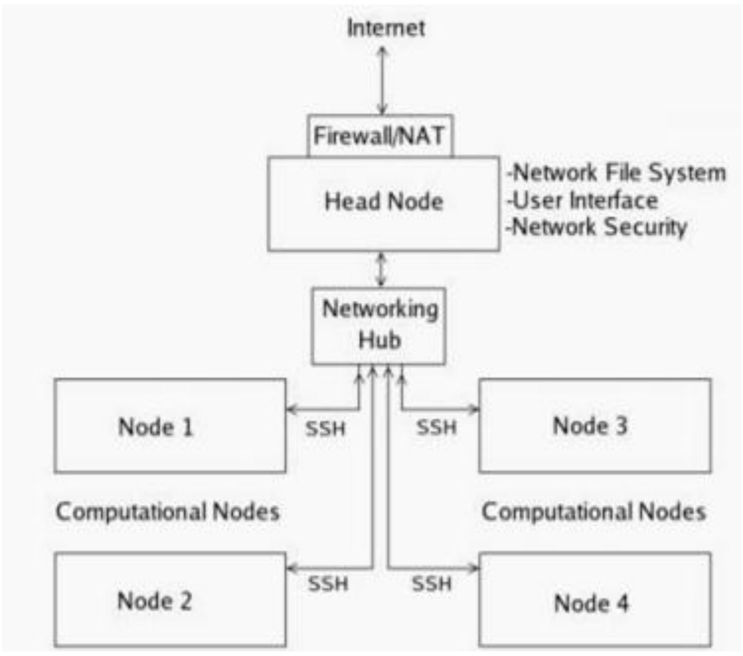


Fig.2 Building Beowulf Cluster

III. TECHNICALITIES TO BE CONSIDERED

Message Passing Interface (MPI) is a library standard for writing message passing programs, which has the advantage of portability and ease of use. MPI is a widely-available communications library that enables parallel programs to be written in C, Fortran, Python and many other programming languages. mpiexec -l -n 2 (no. of process) /cpi (program name), command used to run a program on Master node. GCC Compiler is implemented on Server node for compilation of C programming.

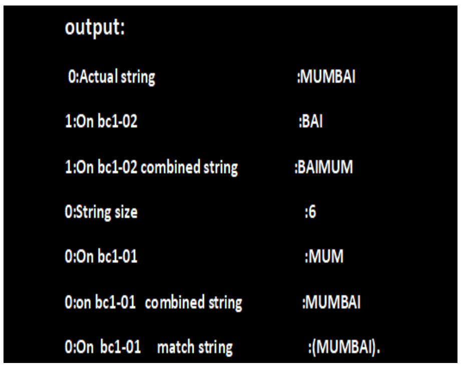
The Network File System (NFS) allows remote hosts to mount file systems over a network and can export directories that can be mounted on a remote host. This enables to share common directories between Master node and Slave node to get desired output. Possible topologies are Star topology, Star-Ring Hybrid topology, Star-Hypercube Hybrid topology cluster; depending on need and flexibility choose any one of this topology cluster.

IV. APPLICATIONS AND IMPLIMENTATIONS

Though the concept of Beowulf Cluster was originated at NASA for Image Processing, it is also used at various other places for Matrix multiplication, pattern matching and also for performance improvement; but most of them are not foolproof either because of lack of efficient algorithms and methods or because of lack of resources. Here, we have implemented Parallel-Exact-Strings-Matching-algorithm-(PXSMAlg).
The parallelization process of the PXSMAlg platform is accomplished through a set of steps. First, the Master node reads the Pattern and the Text (Source-File). Second, the Master node calculates the Source-File size and divides it into multiple parts, according to the determined nodes number. Then, the Master node distributes each part to a specific node. After that, the searching function starts in each node to find the Pattern, with each node searching in its source file

part. Finally, the number of matches is collected from all nodes. The Master node finally calculates all the collected results and then prints the total result.
Let, consider a Source-File is "MUMBAI”, the Pattern is "MUMBAI," and the number of nodes is two. First, divide the Source-File into two parts according to the number of nodes, Part1 is "MUM" and Part2 is "BAI." As we can notice, node1 searched for the Pattern "MUM" and node2 searched for the Pattern "BAI" in Part2. Finally, string “MUMBAI” matches and collected result at Master node.The PXSMAlg process Code is as follows.

```
Algo PXSMAlg()
{
 // Declare String "MUMBAI";
 // initialize rank and size;
 // character declaration for processor_name[MPI_MAX_PROCESSOR_NAME];
 // Declared Processor Name, String Size and Rank of the Processor    using MPI for Interprocess communication and
desired output.
 if(rank==1)
    {
      //For first half of the string
  Loop: Starts from first later and compare till strsize/2
                {
                 if((strsize/2)-1)
End of string
}//For remaining string
  Loop: Starts from strsize/2 and compare till strsize
                {
                 if(strsize/2)
                  End of string
                 }
else if(rank==0)
{
//Print “Actual String” and "Actual String Size”
Loop: Starts from strsize/2 and compare till strsize
              {
               if(strsize/2)
               End of string
              }
Loop: start from first later and compare till strsize/2
              {
               if((strsize/2)-1)
                End of string
              }
 //print processor_name and Combined String
 if(rank==0)
 print String Matched and processor_name
 }
```



```
output:
 0:Actual string                    :MUMBAI
1:On bc1-02                         :BAI
1:On bc1-02 combined string        :BAIMUM
0:String size                        :6
0:On bc1-01                          :MUM
0:on bc1-01  combined string        :MUMBAI
0:On  bc1-01    match string          :(MUMBAI).
```

Fig.3 Output for String Matching

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

Vol. 7, Issue 11, November 2018

Processor efficiency: The result showed high efficiency in the PXSMAlg platform. In comparison with the sequential mode, the Quick Search executing time and speedup were highly improved.

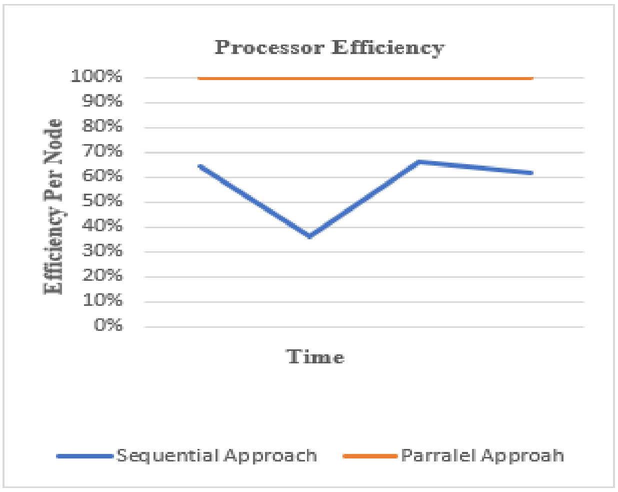


Fig.3 Processor Efficiency

V. HANDLING THE NODES BORDERS ISSUE

Border issue is key element that the PXSMAlg platform faces. This issue happened when the Pattern located between the Source-File parts borders led to a mismatched (i.e., not found) Pattern. To resolve this issue, the PXSMAlg platform allows node "n" to check the border between node "n" and node "n+1," node "n+1" to check the border between node "n+1" and node "n+2," and so on. For illustrative purposes, suppose the Source-File is "EXACT STRINGS MATCHING," the Pattern is "INGS," and the number of nodes is two. First, divide the Source-File into two parts according to the number of nodes, Part1 is "EXACT STRIN" and Part2 is "GS MATCHING." As we can notice, if node1 searched for the Pattern "INGS" in the border of the two parts, it will find it; otherwise, node1 will not be able to find the Pattern "INGS" in Part1 and node2 will not be able to find the Pattern "INGS" in Part2.

VI. BENEFITS

Beowulf clusters offer a number of specific benefits:

A. Cost-effective: One of the main benefits of a Beowulf cluster is its cost-effectiveness. Beowulf clusters are built from relatively inexpensive commodity components that are widely available.

B. Keeps pace with technologies: Since Beowulf clusters only use mass-market components, it is easy to employ the latest technologies to maintain the cluster as a state-of-the-art system.

C. Flexible configuration: Users can tailor a configuration that is feasible to them and allocate the budget wisely to meet the performance requirements of their applications. For example, fine-grain parallel applications (which exchange small messages frequently among processors) may motivate users to allocate a larger portion of their budget to high-speed interconnects.

D. Scalability: When the processing power requirement increases, the performance and size of a Beowulf cluster can be easily scaled up by adding more compute nodes.

E. High availability: Each compute node of a Beowulf cluster is an individual machine. The failure of a compute node will not affect other nodes or the availability of the entire cluster.

F. Compatibility and portability: Thanks to the standardization and wide availability of message passing interface, such as MPI and PVM, the majority of parallel applications use these standard middleware's.

CONCLUSION

Today, the overwhelming price/performance advantage of this type of platform over other proprietary ones, as well as the other key benefits mentioned earlier, means that clusters have infiltrated not only the traditional science and engineering marketplaces for research and development, but also the huge commercial marketplaces of commerce and industry. It should be noted that this class of machine is not only being used as for high-performance computation, but increasingly as a platform to provide highly available services, for applications such Web and database servers. Hence, we can conclude that this approach is highly feasible and has extremely wide scope.

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

Vol. 7, Issue 11, November 2018

REFERENCES

[1]. Small College SuperComputing:Building A Beowulf Cluster At A Comprehensive College by Joel Adams David Vos
[2]. The Beowulf HOWTO by Kurt Swendson.
[3]. AN INNOVATIVE PLATFORM TO IMPROVE THE PERFORMANCE OF EXACT-STRINGMATCHING ALGORITHMS by Mosleh M. Abu-Alhaj, M. Halaiyqah, Muhannad A. Abu-Hashem, Adnan A. Hnaif1, O. Abouabdalla1, and Ahmed M. Manasrah
[4]. D.J. Becker, T. Sterling, D. Savarese, J.E. Dorband, U.A. Ranawak, C.V. Parker Beowulf: A Parallel Workstation for Scientific Computation
[5]. Frinkle, K., & Morris, M. (2015). Frinkle, K., & Morris, M. (2015). Developing a Hands-On Course around Building and Testing High Performance Computing Clusters. Procedia Computer Science, 51, pp. 1907-1916.
[6]. A. Georgi, S. Höhlig, R. Geyer, W.E. Nagel Linux cluster in theory and practice: A novel approach in teaching cluster computing based on the Intel atom platform Procedia Computer Science (2011), pp. 1917-1926.
[7]. Stavrakas, I.K. (2005). Beowulf Clusters for Parallel Programming Courses. The International Conference on Computer as a Tool. 1, pp. 791-794. IEEE.