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

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 10, October 2015

Study of Mosix Software Tool for Cluster Computing In Linux

Vinayak D. Shinde[1], Priyanka Tarmale[2], Shahista shaikh[3]

H.O.D., Department of Computer Engineering, Shree L.R. Tiwari College of Engineering, Mumbai, India[1]

Student of M.E, Department of computer Engineering, Shree L.R. Tiwari College of Engineering, Mumbai, India[2, 3]

Abstract: Mosix conveniently supports a multi-user time-sharing environment for the execution of both sequential and parallel task. it is designed to respond to variations in the resource usage among the nodes by migrating processes from one node to another. Mosix use for load-balancing and to prevent memory depletion from any node .The core of the Mosix technology is the potential of multiple servers (nodes) to work cooperatively. Mosix is a system for sustaining cluster computing. It consists of kernel-level, adaptive resource sharing algorithms that are setup for superiority, overhead-free scalability and ease-of-use of a scalable computing cluster of a single system. It is consider as a leader in the field of high performance computing community.

Keywords: Cluster computing, Mosix Technology, scalability in Mosix.

1. INTRODUCTION

The mosix technology provide more than two nodes work combine as they were part of single system. In order to understand what Mosix does, let us compare a Shared Memory (SMP) multicomputer and a CC. In an SMP system, several processors share the memory. The main advantages are increased processing volume and fast communication between the processes (via the shared memory). SMP's can handle many simultaneously running processes, with efficient resource allocation and sharing. Mosix used to originate distributed operating system for cluster computing. One of the advantage is that is low cost device. Any time a process is started, finished, or changes its computational profiler, the system adapt instantaneously to the resulting execution environment. Mosix is a set of algorithms that support adaptive resource sharing in a scalable CC by dynamic process migration. It can be viewed as a appliance that takes CC platforms one step closer towards SMP environments. By being able to allocate resources globally, and distribute the workload dynamically and efficiently, it simplifies the use of CC's by taking users burden from managing the cluster-wide resources. This is particularly evident in more users, timesharing environments and in asymmetric CC' .In CC systems the user is responsible to allocate the processes to the nodes and to manage cluster resources. In many CC systems, even though all the nodes run the same operating system, coordination between the nodes is limited because most of the operating system's services are locally confined to each node. It count the time required by CPU to process.

2 WHAT IS MOSIX?

It is the tool for the operating system such as linux uses adaptive resource sharing algorithm .it provides the uni process to work in cooperation with other process to faster the speed of execution. The resource sharing algorithms are designed to respond on line variation in resource the goal is to improve overall (cluster -wide) performance and to create a convenient lot of different users, townhouse environment for the execution of both sequential and parallel applications. In resource sharing algorithm it passes the process to one node or other node primitively for load balancing .CC is
The standard run time environment of Mosix, in which the cluster-wide resources are available to each node. Clusters of X86 workstations, SMP's that are connected by standard LANs is the cureent implementation designed by mosix [2].

2.1 THE TECHNOLOGY

The technology consist of two parts a set of algorithms for adaptive resource sharing. And primitive process migration .they are implemented at kernel level and they can be easily demonstrated by application layer. The PPM can migrate any process to any node at any time the migration process is done by the resource sharing algorithm. Super-user). Manual process migration can be useful to implement a particular policy or to test different scheduling algorithms. We observed that the super-user has spare privileges regarding the PPM, such as defining general policies as well as which nodes are available for migration .Home node denoted the location where process was created .this node denotes that the user had log in . The PPM is the main tool for the resource management algorithms. As long as the essentials for assets, such as the CPU or main memory are below certain threshold, the user's processes are constricted to the UHN. When the needs for resources exceed some threshold levels, then some processes may be migrated to distinct nodes, to take convenience of ready to use remote resources. this provide efficient utilization of network wide resource . If during the execution of the processes new resources become available, then the resource sharing algorithms are designed to use these new facilities by possible reassignment of the processes among the nodes.

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

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 10, October 2015

Mosix don't have mean control or master-slave relationship between nodes: each node can operate as an autonomous system, and it makes all its control decisions independently. Algorithms for scalability ensure that the system runs well on large configurations as it does on small configurations [1].

2.2 THE RESOURCE SHARING ALGORITHM

It based on the concept of load-balancing and the memory ushering. It reduces load difference by migrating the process from higher loaded to lesser loaded node. The entire node uses the rescorce share algorithm and pairs of node performed independently the reduction of load difference. The algorithm is triggered in this case the when a node starts excessive paging due to shortage of free memory. In this case an attempt is made to migrate the process to free memory space and override the load balancing algorithm even if this migration would result in uneven load balance [3].

2.3 PROCESS MIGRATION

Mosix support PPM and process continue to interact with environment regarding the of its location .ppm can be divided into two parts the user context – that can be migrated, and the system context – that is UH- N dependent, and not be allowed to migrated. That context of user, called the remote, contains the program code, stack, data, memory-maps and registers of the process. The remote enclose the process when it is running in the user level. The system context, called the representing, contains declaration of the resources which the process is attached to, and a kernel-stack for the execution of system code on the side of the process. The deputy encloses the process when it is running in the kernel. It holds the site subordinate part of the system context of the process; hence it must remain in the UHN of the process. The user-context and the system context is well defined interface. This is implemented at link layer with special communication channel for interaction [1].

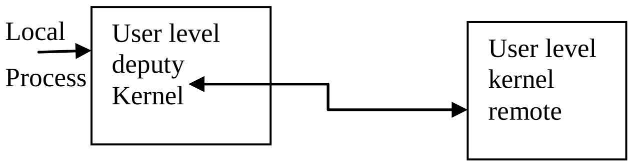

Fig. local process and a migrated process

In the figure, the left process is a normal Linux process while the right process is break in remote part migrated to another node. The migration time has a fixed component, for establishing a new process in the new remote site, and a linear component, proportional to the number of memory pages to be transferred. To decrease the migration om high, only the page tables and the process' dirty pages are transferred. In the execution of a process in Mosix, location translucence is achieved by forwarding site dependent system calls to the deputy at the UHN. System calls are collectively form of interaction between the two process contexts.
The system calls that are executed by the process are interposes by the remote site's link layer.
If the system call is site independent it is executed locally the at the remote site. Otherwise, the system call is forwarded to the deputy, which executes the system the UHN. The deputy recompenses the result(s) back to the remote site, which then proceed with the execution of the user's code. We note that this approach is robust, and is not affected even by major moderations of the kernel. It depends on almost no machine dependent features of the kernel, and thus doesn't obstruct to different architectures.

One draw bag of deputy approach is extra overhead in the execution of system calls. In the UHN, all network links (sockets) are created. Thus imposing communication overhead if the processes migrate away from the UHN. To overcome this problem we are elaborating "migratable sockets", which will move with the process, A direct link is allowed in between migrated processes. Currently, this overhead can significantly be reduced by initial distribution of communicating processes to different nodes. Should the system become imbalanced, the Mosix algorithms will reallocate the processes to improve the performance [3].

Statistics about a process' behaviour are collected regularly, such as at every system call and every time the process accesses user data. This information is used to assess whether the process should be migrated from the UHN. These statistics delay in time, to adjust for processes that change their execution profiler. They are also cleared completely on the "execve ()" system call, since the process is likely to change its nature.

3. CONCLUSION

Mosix provide the new idea of scaling to cluster computing with linux. High-performance, scalable CC from commodity components can be easily constructed by mosix. The one of the advantage over cc system is to respond to many users for irregular resources requirement. Other featured are symmetry, flexibility of its configuration .parallel application can be executed by mosix to assign and reassign the process to best possible node The Mosix R&D project is expanding in various directions. It shows good utilization of resource and good speed up too in cc. The general concept of this optimization is to migrate more resources with the process, to reduce remote access overhead.

REFERENCES

[1] Y. Amir, B. Averbuch, A. Barak, R.S. Borgstrom, and A. Keren. An Opportunity Cost Approach for Job Assignment and Reassignment in a Scalable Computing Cluster. In Proc. PDCS ' 98, Oct. 1998.
[2] A. Barak, A. Braverman, I. Gilderman, and O. Laden. Performance of PVM with the MOSIX Preemptive Process Migration. In Proc. Seventh Israeli Conf. on Computer Systems and Software Engineering, pages 38–45, June 1996.
[3] A. Barak, S. Guday, and R.G. Wheeler. The MOSIX Distributed Operating System, Load Balancing for UNIX. In Lecture Notes in Computer Science, Vol. 672. Springer-Verlag, 1993.
[4] Platform Computing Corp. LSF Suite 3.2.1998.