Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 4, April 2014, pg.590 – 596

RESEARCH ARTICLE

Mosix the Operating System that Support Multiple Cluster Environment with its Advancements & Features

Rahul Rajkumar Pahlajani
Student of Master of Engineering in (CSE)
PRMIT college of Engineering and Technology
Amravati, India
rahulpahlajani@gmail.com

Dr. G. R. Bamnote
Head of the Department of (CSE)
PRMIT College of Engineering and Technology
Amravati, India
grbamnote@rediffmail.com

Abstract - Mosix is a series of modifications to the Linux kernel. MOSIX Design Objectives turn a network of Linux computers into a High Performance Cluster computer. MOSIX is the brainchild of Amnon Barak. MOSIX is a cluster operating system that provides users and applications with the impression of running on a single computer with multiple processors (single-system image) and Hide cluster complexity to users. This paper describes the enhancement of MOSIX to openMosix and its cloud environment. There are many advance features of MOSIX by which large number of application work fastly and properly, they also mentioned in this paper.

I. INTRODUCTION

The MOS for UNIX (MOSIX) is a multi-computer Operating System with decentralized management. Mosix is based on Unix and provides a single-systems image as if using one computer with multiple CPUs. It geared to reduce the management complexity of users. The user does not have to login all the time. Also user's do not need to "login" or copy files to remote nodes. Also there is no need to link applications with special libraries. Mosix has limited support for shared-memory

Mosix creates a virtual computer, featuring automatic load balancing by migrating processes from heavily loaded nodes to less used node. An extension of the Beowulf concept is to run a Mosix enabled kernal. This configuration would provide a very large amount of computational resources based on pre-existing equipment. The advantage of this method is that it provides much more processing power than a traditional Beowulf cluster without the added costs of dedicating resources.

MOSIX is a software solution to minimize OS level bottlenecks - More technically MOSIX is a Single System Image (SSI) cluster that allows Automated Load Balancing across nodes through preemptive process migrations. Why Mosix is most
Accurate for this? Because you are not sure about the load on each node in the cluster and you are not the single user of the facility. If you know how to stop unnecessary services and reduce overheads on the cluster, but you have no control over OS limitations. Mosix makes a network of machines behave like a single machine with many processors and lots of memory. In a MOSIX cluster there is no need to modify or to link applications with any library, to copy files or login to remote nodes, or even to assign processes to different nodes – it is all done automatically, like in an SMP.

The latest version of MOSIX is MOSIX2. It is compatible with Linux-2.6 and 3.0 kernels. MOSIX2 is implemented as an OS virtualization layer that provides users and applications with a single system image with the Linux run-time environment. It allows applications to run in remote nodes as if they run locally. Users run their regular both sequential and parallel applications while MOSIX transparently and automatically seek resources and migrate processes among nodes to improve the overall performance. MOSIX2 can manage a cluster and a multicluster as well as workstations and other shared resources. It can run in native mode or in a virtual machine (VM). In native mode, performance is better, but it requires modifications to the base Linux kernel, whereas a VM can run on top of any unmodified operating system that supports virtualization, including Microsoft Windows, Linux and Mac OS X.

The algorithms in MOSIX support load-balancing [6], memory ushering [7], parallel I/O [8] and cluster-wide file operations [8] these algorithms monitor uneven resource usage among the nodes and if necessary, assign and reassign processes automatically among the nodes in order to continuously take advantage of the best available resources. The MOSIX algorithms are geared for maximal overall performance, overhead-free scalability and easy to use.

II. MOSIX CLUSTER APPROACH

A MOSIX cluster is a set of connected computers; it is not isolated machines + OS + middleware. The Cluster is a single machine running Cluster OS on different nodes. It allows automatic work distribution among cluster nodes. It has Granularity at process level, and work on principle like: "Fork and Forget".

A. SINGLE CLUSTER APPROACH

A MOSIX cluster is a set of connected servers and workstations, called “nodes”, which are administrated by a single owner and run the same version of MOSIX. In a MOSIX cluster, each node maintains information about availability and the state of the resources of all the nodes. Nevertheless, MOSIX processes can run in remote clusters while still using the environment provided by their respective private home clusters. MOSIX2 is most suitable for running compute intensive applications with low to moderate amount of input/output (I/O). Tests of MOSIX2 show that the performance of several such applications over a 1 Gbit/s campus grid is nearly identical to that of a single cluster

B. *MULTI- CLUSTER APPROACH*

A MOSIX multi-cluster (also called “an intra-organizational multi-cluster”) is a collection of private MOSIX clusters that run the same version of MOSIX and are configured to work together. A MOSIX multi-cluster usually belongs to the same organization, but each cluster may be administrated by a different owner or belongs to a different group. The cluster-owners are willing to share their computing resources at least some of the time, but are still allowed to disconnect their clusters from the multi-cluster at any time.

In a MOSIX multi-cluster, each node maintains information about availability and the state of the resources of all the nodes in all the connected clusters. Different clusters may (or may not) have a shared environment such as a common NFS file system. From the user’s perspective, MOSIX transforms such a multi-cluster into a single cluster by preserving the user’s local run-time environment. In MOSIX multi-clusters there is usually a high degree of trust, i.e., a guarantee that applications are not viewed or tampered with when running in remote clusters. Other possible safety requirements are a secure network and that only authorized nodes, with identifiable IP addresses, are included.

C. MOSIX CLOUD APPROACH

A MOSIX cloud is a collection of entities such as MOSIX clusters; MOSIX multi-clusters; Linux clusters (such as a group of Linux servers); individual workstations and Virtual Machines (VM). Each entity may possibly run a different version of Linux or MOSIX. In a MOSIX cloud, different entities are usually administrated by different owners and rarely share any file systems (such as NFS). In this cloud, nodes in each entity are aware of one or more nodes in other entities, including their IP addresses and services they are willing to provide, but there is no on-going automatic flow of information between entities.

In a MOSIX cloud, users can launch applications from their workstations or a private home-cluster, on target nodes of other entities. These applications have access to files on nodes of these entities, while still allowing the applications to access files on their launching node. This is accomplished by the MOSIX Reach Clouds (MOSRC), described in Sec. IV, which allows applications to run in remote nodes, without the need to copy files to/from remote clusters.

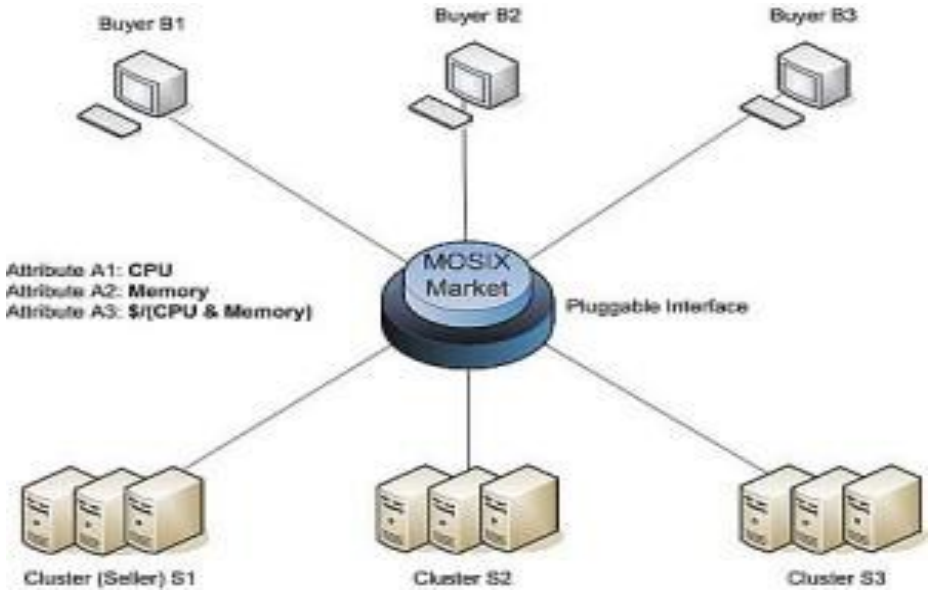

Fig. 1: User's and server's connected in MOSIX cluster

III. WORKING OF MOSIX CLUSTER

Processes are the building blocks of MOSIX.

MOSIX recognizes two types of processes: Linux processes and MOSIX processes. Linux processes run in native Linux mode and cannot be migrated. On the other hand MOSIX processes can be migrated. Linux processes usually include administrative tasks and processes that are not suitable for migration. Another class of Linux processes is those created by the "mosrun -E" command. These processes can be assigned by the "-b" option of "mosrun" to the least loaded nodes in the cluster (but not to nodes in other clusters, for which the MOSRC tool can be used, see Sec. IV for details).

MOSIX processes are usually user applications that are suitable and can benefit from migration. All MOSIX processes are created by the "mosrun" command. MOSIX processes are started from standard Linux executable, but run in an environment that allows each process to migrate from one node to another. Each MOSIX process has a unique home node, which is usually the node in which the process was created [3]. Child processes of MOSIX processes remain under the MOSIX discipline (with the exception of the **native** utility, that allows programs, mainly shells, already running under mosrun, to spawn children in native Linux mode).

Two tier technology is used for Information gathering and dissemination by using probabilistic dissemination algorithms. This helps each node to have sufficient knowledge about available resources in other nodes, without polling. It supports Preemptive process migration that can migrate any process, anywhere, anytime – transparently. This is supervised by adaptive algorithms that respond to global resource availability.

Because of the decentralized control and autonomy each node makes its own control decisions independently. Each node is capable of operating as an independent system. As it does not required centralized control Nodes may join or leave the farm with minimal disruption. Each node work efficiently and correctly as in every unit of time (1 second) each node gathers and disseminates information about:
CPU(s) speed, load and utilization, free memory, Process tables, etc.

IV. ADVANCEMENTS IN MOSIX

A. THE OPENMOSIX

The openMosix is open source implementation of MOSIX. The project leader of openMosix is Dr. Moshe Bar [10]. Cluster Configurations of openMosix is created as Singlepool. In Singlepool all the servers and workstations are used as a single cluster: each machine is a part of the cluster and can migrate processes to each other. The software like PVM and MPI run more efficiently on the openMosix kernel than on Cluster unaware OS kernels. This is used for comparison during cluster configuration.

1) Openmosix Features: In comparison with the Beowulf Cluster openMosix Cluster has many features:

1. It allows dynamic addition & removal of nodes.
2. Preemptive process migration offer optimal load balancing across the nodes.
3. Adaptive resource allocation scheme allows use of heterogeneous nodes in the cluster.
4. No need to modify, compile or link user code with any special software library
5. Offers optimal performance for CPU bound code
6. Ideal for multi user, time shared systems
7. Optimal use of computing facility and it monitored Load in each node.

2) Openmosix Attractions: Adding new nodes to a running openMosix cluster can be made dynamic with the auto discovery daemon. Nodes can join in and withdraw gracefully without disturbing the processes running on the Cluster. The hardware in openMosix can be used as best as possible because it Recycle all your old hardware. Applications that improve performance of openMosix include:
Matlab 5, Octave, MJPEG tools, flac POVRAY, MPI, Postfix, CISILIA, etc.

Kernal sourse code and other releated information about openMosix is available at http://openmosix.sourceforge.net/
Diskless nodes to form a cluster is also available for free download.
E.g.: cluster knoppix, CHAOS, plumpOS ….

B. MOSIX REACH THE CLOUDS

MOSIX Reach the Clouds (MOSRC) is a tool that allows applications to run in remote computers in any MOSIX cloud entity (see Sec. II-A). rMOSRC users launch applications from their workstation (or private home-cluster) on target nodes of other entities.
MOSRC can run on both Linux computers and MOSIX clusters. The hybrid environment on target nodes can be used for remote file access; file-sharing among different computers and users All "mosrun" features can be used on clouds running MOSIX [11].
As such, MOSRC can be useful to users that need to run applications but do not wish to store data on commercial clouds. MOSRC provides consistent access to files, even among multiple MOSRC jobs that run on different targets.

MOSRC consists of two parts: a launching program that can send jobs from a head-node to designated target nodes, and a run-time environment that provides file services to running jobs on target computers. The head-node could be the user's workstation and any computer that has access to the user's files. The head-node and the target nodes can run Linux or MOSIX. If the target node is part of a MOSIX cluster, then MOSRC jobs can benefit from all the MOSIX features. In particular, MOSIX processes generated by MOSRC jobs can be automatically dispersed among the nodes of that cluster or even among other MOSIX clusters in a multi-cluster. Launching a job (from the head-node) is done by the "mosrc" command; see the mosrc manual for details.

V. FEATURES OF MOSIX

A. AUTOMATIC RESOURCE DISCOVERY

Resource discovery is performed by an on-line information dissemination gossip algorithm, providing each node in all the clusters with the latest information about availability and the state of system-wide resources [1]. The algorithm is based on a randomized gossip dissemination, in which each node regularly monitors the state of its resources, including the CPU speed, current load, free and used memory, etc. In [1] we presented bounds for the age properties and the rates of propagation of the above algorithm.

B. LOAD BALANCING

In MOSIX, load balancing is carried out by dynamic process migration[6]. Process migration is allowed only among processors with same instruction set, because application task may be allowed only among the nodes of the cluster. In MOSIX, each processor sends its local and other load estimate which is known to the processor. For balancing load processes are migrating from slower to faster nodes and from nodes that run out of free memory. Migration is done by dividing process into two pieces:

A deputy component:

that is kept in the UHN (Unique Home Node) & contains the kernel context (description of used resources + kernel stack + task structure).

A remote component:

that contains the user context
(code+stack+data+memory map + registers) that gets migrated.

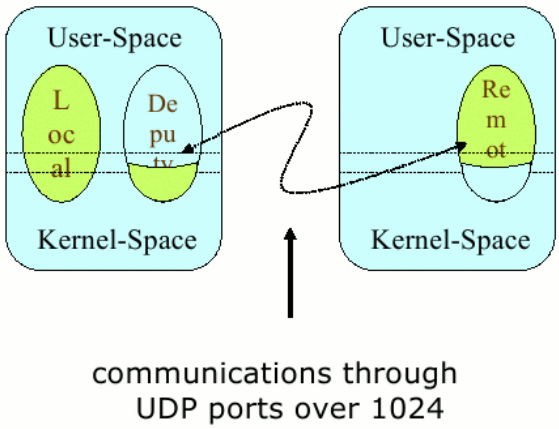

Fig. 1: How process migration works by dividing into two pieces

Process partition created for load balancing purpose preserves the user's run-time environment. Users need not care where their processes are running. Load balancing reduce variance between pairs of nodes to improve the overall performance.

The algorithm used for load balancing is probabilistic. These algorithms are intended to provide each node with the latest, up-to-date information about the load of other node. As provide in [8] this is achieved in O(log N) units of time for an N-processor system. The decision to migrate a process is based on different parameters, including past profile, the amount of local versus remote I/O, size of the process, etc. Greedy, a popular online algorithm for load-balancing, assigns a new job to a machine in order to minimize the resulting resource utilization.

C. INTERPROCESS COMMUNICATION OVERHEAD IS REDUCE IN COMPUTING CLUSTER

Computing Clusters (CC) consisting of several connected machines, could provide a high-performance, multiuser, timesharing environment for executing parallel and sequential jobs. In order to achieve good performance

in such an environment, it is necessary to assign processes to machines in a manner that ensures efficient allocation of resources among the jobs.

The opportunity cost algorithms for online assignment of jobs to machines in a Cluster is used. These algorithms are designed to improve the overall CPU utilization of the cluster and to reduce the I/O and the Interprocess Communication (IPC) overhead. This approach is based on known theoretical results on competitive algorithms. In our computing model, there are also parallel job which consists of several communicating processes, with an arbitrary communication topology. Processes are assigned online to machines in order to minimize the maximal load and IPC overhead among the machines.

D. JOB MANAGEMENT IN CLUSTERS

EnFuzion is an application level package that provides a high level environment for the creation, distribution and management of large parameter sweep applications. MOSIX is a software package that enhances Linux with cluster computing capabilities. The core of MOSIX includes adaptive management algorithms and a preemptive process migration mechanism that transforms the cluster into a single system parallel computing environment, almost like an SMP. This opportunity cost method converts the usage of several heterogeneous resources in a machine to a single homogeneous cost. Assignment and reassignment of jobs are then performed based on that cost.

Combining EnFuzion and MOSIX yields a powerful platform, where EnFuzion generates, allocates and queues jobs to the cluster and MOSIX manages and optimizes the load distribution between nodes within the cluster. In particular, EnFuzion benefits from MOSIX's ability to perform preemptive process migration, and MOSIX benefits from a queue management system.

E. SUPPORT DFSA FOR SCALABLE CLUSTER FILE SYSTEM

As stated earlier MOSIX is a cluster management system that supports preemptive process migration. The MOSIX Direct File System Access (DFSA), is a provision that can improve the performance of cluster file system by allowing a migrated process to directly access file in the current location. By combining this capability, could substantially increase the I/O performance and reduce the network congestion by migrating an I/O intensive process to a file server. DFSA is suitable for cluster that manages a pool of shared disks among the multiple machines. With DFSA, it is possible to migrate parallel processes from a client node to file servers for parallel access to different files. Any consistent file system can be adjusted to work with DFSA.

To test the performance, we developed the MOSIX File-System (MFS) which allows consistent parallel operations on different files.

VI. CONCLUSIONS

MOSIX is an operating system-like management system that consists of a comprehensive set of tools for sharing computational resources in Linux clusters, multi-clusters and clouds. Its main features are geared for ease of use by providing the impression of running on a single computer with multiple processors. This is accomplished by preserving the interface and the run-time environment of the login (home) node for applications that run in other nodes. As a result, users need not modify or link applications with any library, they need not login or copy files to remote nodes or even know where their programs run.

The unique features of MOSIX include automatic resource discovery, dynamic workload distribution by process migration, a priority method that allows processes to migrate among nodes in a multi-cluster, to take advantage of available resources

beyond the allocated nodes in any private cluster. This is particularly useful in shared clusters or when it is necessary to allocate a large number of nodes to one group, e.g., to meet a deadline. The disruptive configuration provisions allow an orderly migration of processes from disconnecting clusters, including long running processes when remote resources are no longer available. Other unique features include process migration, queuing and a tool to run applications on clouds, without the need to pre-copy files to these clusters.

REFERENCES

[1] Barak, A., La'adan, O. and Shiloh, A., " Scalable cluster computing with MOSIX for Linux", Proc. 5-th Annual Linux Expo, Raleigh, NC, pp. 95-100, May 1999.
[2] Barak and Braverman, "Memory ushering in a scalable computing cluster", Journal of Microprocessors and Microsystems, 22 (3-4), pp. 175-182, 1998.
[3] Charles Bookman, "Linux Clustering: Building and Maintaining Linux Clusters" (Paperback - June 29, 2002).
[4] A. Barak, S. Guday, and R. G. Wheeler, The MOSIXDistributed Operating System: Load Balancing for UNIX,Secaucus, Ed. New York, USA: Springer, 1993
[5] http://www.MOSIX.org.
[6] Ahmed, B.S. ; Samsudin, K. ; Ramli, A.R. ; Basri, "A Descriptive Performance Model of a Load Balancing Single System Image" S. Modeling & Simulation, 2008. AICMS 08. Second Asia IEEE International Conference
[7] Keren A., and Barak A., "Opportunity Cost Algorithms for Reduction of I/O and Interprocess Communication Overhead in a Computing Cluster," *IEEE Tran. Parallel and Dist. Systems*, 14(1), pp. 39–50, 2003.
[8] Maoz T., Barak A. and Amar L., "Combining Virtual Machine Migration with Process Migration for HPC on Multi-Clusters and Grids," *Proc. IEEE Cluster 2008*, Tsukuba, 2008.
[9] Barak, A. ; Shiloh, A. ; Amar, L. **,"** An organizational grid of federated MOSIX clusters**"** Cluster Computing and the Grid, 2005. CCGrid 2005. IEEE International Symposium on
[10] http://www.imsc.res.in/~kabru/parapp/IMSCTalk.pdf
[11] http://www.mosix.cs.huji.ac.il/pub/MOSIX_wp.pdf
[12]http://os.inf.tu-dresden.de/Studium/DOS/SS2013/05- MOSIX.pdf
[13] http://en.wikipedia.org/wiki/MOSIX
[14]http://www.csse.monash.edu.au/~davida/papers/MosixPDCS03.pdf
[15]http://mulix.wordpress.com/2003/10/29/prof-amnon-barak-of-mosix-fame-to-give-a-seminar-at-hrl/

