A Modification of OpenMosix's Process Migration Algorithm to Improve Cluster Performance and Scalability

Travis Frisinger
Computer Science Department
University of Wisconsin-Eau Claire
Eau Claire, WI 54702
frisintm@uwec.edu

Abstract

By modifying OpenMosix's process migration algorithm from its current state, one based on an economic algorithm, to allow processes generated on the same node to have a higher chance to migrate to the same host node to improve an OpenMosix cluster's performance and scalability. The reason for this is that once a process is migrated to a new node a communication channel is opened between the Unique Home Node (UHN) and the host node. This channel is very resource intensive and is required for each process that is migrated. By migrating processes generated on the same node to a sub set of the collection one can devote more of the clusters resources to processing the individual processes. This would be done by reusing the communication channels for processes that have been migrated from the UHN to the same host node, hence increasing the performance of an OpenMosix cluster.

1 Introduction

There are two types of High Performance Computing (HPC) clusters a Beowulf cluster and an OpenMosix cluster. In a Beowulf cluster a software layer either MPI or PVM is required to allow the cluster to function. This software layer is a parallel message passing interface that allows application written in C/C++ or FORTRAN to run on the cluster. This also requires the current applications written in C/C++ or FORTRAN be re-written to use this special software layer.

With an OpenMosix cluster a software patch is applied to the Linux kernel that allows information about each node of the cluster to be exchanged. This information is used to evaluate nodes for process. An OpenMosix cluster does not require that application be rewritten to run. OpenMosix does suffer from scalability issues because of the overhead involved with process migration and remote execution. By selective modification of the process migration algorithm it is possible to increase OpenMosix's scalability and process execution time.

2 Evaluation Environment

The testing of modifications to the OpenMosix kernel is done thought the use of virtual machines. One virtual machine is placed on each physical machine running Windows XP. Each virtual machine is assigned its own IP address, instead of using NAT to translate between the virtual machine and the physical machine's IP. This is done to reduce the amount of network overhead between the virtual machine and physical machine. The virtual machine does add a bit of overhead and might slightly affect the outcome of the analysis, but since all under lying windows machines have the same configuration, it is safe to assume that the affects of overhead is constant or very close to constant and may be ignored.

The tests to the modifications are done using Free Losses Audio Codec (FLAC) encoding. This test was chosen because it is easy to scale the number of parent processes and is known to fork child processes; both of which are requirements for evaluating the efficiency of the modification to the OpenMosix kernel.

3 OpenMosix Background

“openMosix is the GPLv2, Open Source, project to extend the outstanding MOSIX project. New releases of MOSIX became proprietary software in late 2001 and openMosix was begun February 10, 2002 by Moshe Bar to keep this highly regarded Linux Clustering solution available as open source.” [1]

Each process has a Unique Home Node (UHN); this is usually the node to which the user logged into. Since not all data related to the process is migrated along with the process, a communication channel is maintained to the UHN to allow the process access to data left behind. This communication channel is a very resource intensive and is a short coming of OpenMosix clusters. Since each migrated process requires a UHN communication channel, as the cluster takes on more processes, more and more resources are required to perform remote executions of the processes.

4 Modification of Process Migration Algorithm

There where two major steps in modifying the process migration algorithm. First off was to develop a way for migrated process information to spread around the cluster. This was to be used to evaluate possible nodes for migration candidates, with emphasis to be place on nodes that already contained processes from the UHN. Secondly, a method was needed to determine when to add a new node to the pool of possible migration candidates along with restricting the number of communication channels used to access resources on the UHN.

This modification resulted in a need to develop a message packing strategy when accessing resources on the UHN. It was believed that the reduction in the number of communication channels would free up more cluster resources allowing processes to execute faster and improve the scalability of the cluster.

4.1 Modification One

The first attempt to disseminate UHN and migrated child process information was a peer-2-peer protocol, but this was quickly abandoned as too resource intensive. It was discovered that a slight modification to the current OpenMosix framework would allow the information to reach all nodes with out adding extra processing and network overhead to the cluster. Once this information was allowed to reach the nodes just like all other load information it could be used to evaluate migration candidates. The current migration algorithm required all nodes to be evaluated before choosing the best node to migrate the process to.

By simply short circuiting the evaluation to grab the first node with migrated processes there was a .2% efficiency increase when the cluster contained more then four nodes. When it contained less then four nodes there was almost no difference at all. This is attributed to the fact that less CPU cycles where used in kernel space to evaluate the migration of a process. Since the evaluation process is uninterruptible and must finish before any other CPU cycles may be used to process the cluster's user space processes.

4.2 Modification Two

During the second phase of the project the goals where how to restrict the amount of available nodes for migration from the UHN and to implement a message packing strategy to restrict the number of communication channels to the UHN.

Currently there is a limit of only allowing up to one-half of the nodes as migration candidates for processes leaving their UHN. There is also a restriction of only allowing a two-to-one ratio of processes to communication channels. This is placed as an arbitrary restriction and will be replaced with a more dynamic algorithm for limiting the number of nodes a UNH may send processes to. This also raises another issue encountered during the course of this portion of the research, which is that currently once a process has been migrated it may only stay at the first host node until it finishes execution.

This is because there is no longer a one-to-one mapping of communication channels to migrated processes. There is the possibility of orphaning a process in the cluster; this is currently being dealt with. There is also a need to modify migration candidacy, possibly requiring two processes from the same UHN to migrate instead of one. The cluster should not have to incur the overhead of the communication channel for just one process.

Currently, there is a .9% percent performance increase when the number of nodes is greater then eight. There is a .3% increase in performance when the number of nodes is less then or equal to four. The increase in efficiency is due to the fact that the cluster is using more resources for user space processes; even though it is limiting the number of nodes a process may migrate to. Once the restriction of single migration is removed the cluster should realize a universal improvement in performance of 2-3%.

5 Future Modifications of Process Migration Algorithm

The next step is to allow migration of processes beyond a single node to see the impact of this on the cluster's performance. This may result in either a dynamic approach to selecting the number of nodes a process may migrate to, or a statically restricted amount. The statically restricted method would be preferred because it would require less overhead. The down side to the static restriction is that it may impact performance slightly.

Then there is the process of evaluating the impact of restricting the communication channels to dynamically evaluated value or to set it statically.

Once again static restriction is the preferred choice due to the lack of overhead, but it may impact performance even more then a dynamically chosen value. The reason being it is harder to predict the communication needs of a processes set, hence the possible need to allow a dynamically chosen number of communication channels.

6 Summary

The amount of overhead required to migrate and execute a processes remotely is about 3.9% more then that required to execute that same processes locally. With the modifications the overhead is 3.3% percent of local execution, a .6% increase in efficiency. Since this is per processes incurred overhead there are greater gains in performance as the amount of processes in the cluster increases.

Even though the performance gains where not at the level expected, about 2%, there where many reason for this. One is the lack of diversity in processes sets, it would be nice to run more simulations on a wide range of processes sets to evaluate the processes sets impact on the clusters performance. There is also the possibility that overhead incurred from the use of virtual machines may have played a greater role then expected.

Overall the experiment was a success, on other future goal would be to scale the cluster well beyond sixteen nodes to maybe two or three hundred nodes on physical hardware and evaluate the modifications impact.

7 Resources

[1] Knox, Bruce. openMosix, an Open Source Linux Cluster Project. 10 Feb. 2002. The openMosix Community. <http://openmosix.sourceforge.net/>.

[2] Bar, Dr. Moshe. Home Page. 2003. <http://openmosix.sourceforge.net/linux-kongress_2003_openMosix.pdf>.

[3] Hanquez, Vincent. om_internals. 28 Dec. 2004. <http://tab.snarc.org/article/om_internals.xhtml>.