Using Mosix for Wide-Area Compuational Resources

By Brian G. Maddox[1]

Open-File Report 2004-1091

[1] USGS Mid-Continent Mapping Center, Rolla, MO, 65401

U.S. Department of the Interior
U.S. Geological Survey

Contents

Key Words

Mosix, Distributed Processing, Beowulf

Abstract

One of the problems with using traditional Beowulf-type distributed processing clusters is that they require an investment in dedicated computer resources. These resources are usually needed in addition to pre-existing ones such as desktop computers and file servers. Mosix is a series of modifications to the Linux kernel that creates a virtual computer, featuring automatic load balancing by migrating processes from heavily loaded nodes to less used ones. An extension of the Beowulf concept is to run a Mosixenabled Linux kernel on a large number of computer resources in an organization. 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.

Introduction

Distributed processing has seen increased use in recent years as a way to use multiple machines to speed the processing of computationally intensive applications. The most common form of distributed processing, Beowulf clusters, have become famous for providing near supercomputer-class processing performance for a price that can be several orders of magnitude lower than an actual supercomputer. Built with commodity components, they are typically dedicated to compute tasks.

The problem with such a setup is that machines usually need to be dedicated to a cluster, requiring either additional computers to be purchased or redirecting machines to be dedicated to the cluster. For some organizations, purchasing additional hardware is not a problem. Others, however, may not be able to make use of this technology due to dwindling information technology budgets.

Emerging technologies, such as Mosix, might pave the way for organizations to use their existing computer infrastructure in a distributed processing environment. These technologies would lead to a wide-area processing cluster with all the advantages of distributed processing while lowering the overall implementation cost. A large percentage of an organization's computers could provide a much greater amount of processing power than would be available with a typical Beowulf cluster. Also, existing emulation and translation technologies could allow non-distributed and non-Linux applications to take advantage of the increased processing power.

This paper describes several emerging technologies and how they might be used to build a new type of system that can provide new methods of performing distributed processing. It also details research that led to using several technologies to create organization-wide processing clusters with existing desktop computers. Each technology, along with its actual research and experimentation, is briefly discussed before the wide-area cluster concept is presented.

Background

Beowulf

The first major technology that can enable organization-wide distributed processing is a Beowulf cluster. Beowulf clusters have brought distributed processing to the masses by using commercial off-the-shelf components in their construction. The idea behind a Beowulf originated in 1993 when Donald Becker and Thomas Sterling began to sketch the outline of a commodity-based cluster system that was designed as a cost-effective alternative to large supercomputers (Merkey). Supercomputer time was, and still is, very expensive to rent if an organization lacks one of its own. Distributed processing was not a new concept at this time, but usually was relegated to high-end UNIX®[1] workstations.

1 UNIX is a registered trademark of The Open Group in the United States and other countries.

A Beowulf is characterized by a dedicated high-speed network for communication with nodes that are dedicated to processing tasks for the cluster. Typically the system runs on a personal computer architecture with a Linux kernel and distribution. Systems are normally homogeneous with the types of equipment in each compute node. These configurations simplify administration and cost as configurations are the same and hardware can be purchased in bulk at a discount. Homogeneous hardware is also important for certain algorithms and load balancing, which will be discussed below.

Distributed processing on these clusters involves running subsets of a problem on different computing nodes. A common form of processing involves the master-slave model, where the master node controls what work is done while the slave nodes are purely computational. Processes are spawned remotely (usually due to some manual configuration) and communication is accomplished by message passing between systems.

The homogeneity of the compute nodes comes into play because issues such as load balancing are typically done by hand on these systems. If certain nodes finish processing out of order or faster than others, some algorithms might fail. Because of this, care is taken to ensure that the small pieces of processing can be done independently of others or that the work can run sequentially on the slave nodes.

Although working together, it is important to note that Beowulf-class systems are still treated as a collection of separate computers. Memory and process spaces are separate and non-visible between systems. Software must be specially developed for these systems. Existing software cannot simply be dropped in and run on them; it must be modified or completely redesigned to work in the message-passing environment. The modifications also include porting the software to run on Linux instead of some other operating system such as Microsoft Windows. In other words, one cannot simply take an existing application such as a Geographic Information System (GIS) package and have it take advantage of the power of a Beowulf without a lot of additional work.

Mosix

Another method for distributed processing involves treating the various processing nodes as part of a larger virtual computer by providing common memory and file system access. This technique goes one step beyond a Beowulf where each computer is still independent of the rest of the system. Mosix takes this approach and is the second enabling technology to organization-wide clustering. Mosix is a series of patches to a Unix-like kernel (currently Linux) that allows resource sharing between the various nodes in a cluster. This is done by

> "migrating processes from one node to another, pre-emptively and transparently, for load-balancing and to prevent thrashing due to memory swapping. The goal is to improve the overall (cluster-wide) performance and to create a convenient multi-user, time-sharing environment for the execution of both sequential and parallel applications."(Barak).

Mosix has been in various forms of development for over two decades now, and is currently used in many different applications and organizations.

The central idea behind Mosix is that it can dynamically balance the virtual computer's load by migrating processes between machines. Process migration works by taking the process's entire memory space and moving it across the network to another machine whenever the local machine's processing load reaches a certain level. On the originating machine, a small “stub”process is created by the operating system that the now-remote process uses to communicate with the originating machine. This process migration is transparent to not only the process itself, but also to anything communicating with that process, as the stub handles local communication on the originating machine. Mosix also allows access to every machine in the cluster through a common file system so that remote running processes can easily access storage across the system.

Because process migration and load balancing occurs at the operating system kernel level with Mosix, it has some interesting characteristics that Beowulf clusters do not possess. Any process under Mosix is a candidate for migration and does not need any special modifications to run in this fashion. The remote process is still able to access local resources on the originating machine by going through the stub process. It can also directly access resources on the remote machine. Mosix allows traditional batch-processing applications to benefit from distributed processing by starting them on a single node then migrating them across the cluster nodes to balance the load so that they run in parallel instead of serially.

Mosix also provides some flexibility that a traditional Beowulf system lacks. Machines can enter and leave a Mosix cluster on demand without interrupting processing. When a Mosix node goes through a normal shutdown process, it will offload remote tasks to another node. A node can be added to a Mosix cluster at any time to augment processing. Traditional message-passing systems typically are static in that machines have to be allocated ahead of time and stay as part of the currentjob until processing is complete.

Wine Is Not an Emulator

The third enabling technology is the WINE project. WINE, which stands for WINE Is Not an Emulator, is an Open Source project that translates Win32 calls to native Unix calls at runtime, thus allowing some Windows-based software to run under Linux (WINE Headquarters). It is important to note that WINE will NOT run all Windows software under Linux. Some software contains system calls that have not been implemented in WINE. Others are specifically tied to a Windows-based kernel and therefore cannot run. WINE is implemented in several parts. The first is a front end that loads the Win32 application and then runs it through the translation system. The second part is a series of libraries that simulate their Win32 counterparts but run directly on the local system. These libraries are loaded by the application just as they would be under Windows, but skip the translation step as they are implemented natively. Additionally, actual Windows libraries can be used in place of their WINE counterparts for situations where different functionality is necessary.

Emulators

The final technology that is necessary for organization-wide clusters deals with the issue of how applications that absolutely must run under Windows are handled. Other Unix applications can in many cases be easily ported to run under Linux. Some Windows applications can be run under WINE. However, sometimes an application has a tight dependency on an actual Windows kernel. In these cases, an emulator such as VMWare can run Windows on top of the Linux system.

Emulators are different from WINE in that they emulate computer hardware at the lowest level instead of translating operating system calls at the application level. In the VMWare case, operating systems such as Windows are installed on top of the virtual hardware. The operating system inside the emulator sees no difference between the virtual machine (VM) and a real computer. With Windows actually running, applications that depend on the Win32 kernel can function, whereas they might fail with WINE as it is translating calls at run time.

If one can actually run Windows on Linux using an emulator, why not use emulation entirely instead of a translator such as WINE? In the WINE case, applications are actually running on Linux. Even though WINE is translating system calls in real time, each Windows program through WINE is viewed as a separate process. In the emulator case, the emulator is the only process visible to the base operating system. Multiple processes may be “running”in the virtual machine, but the entire system is still a single process. When running on Mosix, this makes a significant difference. Multiple WINE applications can be migrated around the cluster. Multiple applications within an emulator cannot, as they are not separate processes.

Methods and Testing

As part of a Geography Discipline Prospectus project, in 2002 the USGS Mid-Continent Mapping Center (MCMC) conducted research in combining Mosix technology and traditional Beowulf clusters. The idea was that this combination could provide a “best of both worlds”approach to distributed processing. While Beowulf uses commodity components in a distributed processing cluster, Mosix provides automatic kernel-level process migration and load balancing to turn the cluster of individual machines into a single virtual machine. The combined technologies worked very well together, as slave processes could simply be started on a single machine and Mosix could then automatically migrate them around the cluster to balance the processing load. This method was much simpler than the manual configuration and tuning required from traditional message-passing protocols. *Extending Beowulf Clusters,* USGS Open File 03-203 describes more information on these experiments.

In 2003, Beowulf and Mosix were combined to test an organization-wide cluster using desktop machines on the existing MCMC cluster. The first part of this testing examined ways to enable Win32 (Microsoft Windows) programs to run under Linux using translation or emulation. This configuration would open the possibility of using Mosix to distribute the processing of some Windows applications so they would not only run under Linux, but also be migrated. Many critical production systems, such as orthophoto production for the 133 Urban Areas project, are Windows applications. Running such programs in a distributed manner can greatly speed processing run times.

WINE was installed on the head node of the cluster. The head node was the only node running the X Window System that WINE needs for graphical applications. It was only installed on the head node to check that Mosix could migrate the entire process space without needing additional software installed on the other nodes. This capability would verify that large amounts of software would not need to be installed on each node in an organization-wide cluster, thus reducing installation complexities. After installation, the author ran some sample applications such as Windows notepad and regedit to test initial functionality.

With WINE installed and running on the head node, the author tested several applications in common use at MCMC. Process migration was turned off during the initial compatibility testing. Functional testing was performed to check that the applications loaded and were able to perform their major tasks. ArcView version 3.3 was tested and found to function fairly normally with some sample datasets. A font problem occurred and several TrueType fonts had to be installed so that the application text was readable. Add-on packages such as Spatial Analyst failed to install. However, the extensions that come with the main package functioned normally. ERDAS Imagine was tested next, but would not install due to problems with the license manager. As a side note, this is an example of how some applications that need low-level Windows functionality might not run under WINE. Global Mapper was tested and found to run under WINE. The author tested Lotus Notes for functionality, which also depends on TrueType fonts for readability.

Once initial testing was completed, the project focused on applications that were used on the 133 Urban Areas project. Much of this processing is done in batch, yet it is performed sequentially. The process at MCMC involves a combination of command-line and graphical applications from ZI Imaging. The graphical applications had installation problems that were similar to Imagine and were therefore not tested. Some of the command-line applications, however, ran perfectly under WINE. These applications included mkov.exe for making overviews of the images to be processed and mr_file.exe for tiling the images.

These two applications were initially tested without Mosix in an environment that mimicked the production one. Data was staged on both local and remote storage and output was written locally. The author wrote a shell script that would run the applications in parallel under WINE and tested both remote and local data input. Testing in this case showed a 25 to 45-percent speed increase under WINE and Linux. It should be noted, however, that the Linux kernel used on this test machine included several low-latency and pre-emptible kernel patches to increase system throughput.

A similar test was then performed on the Beowulf/Mosix cluster with process migration enabled. The data was staged on the head node and on a remote node. The author then wrote a script to run the applications through several iterations using both remote and local storage. Instead of being run sequentially, however, the script was set to run the command-line programs in background mode so that Mosix could migrate them across the cluster.

To a point, a similar speedup to the previous testing was observed. However, the data-bound processes suffered from minor starvation problems as they were all trying to read data from a single file-server node. This was observed by monitoring the file-server node and tracking how successfully it was completing requests by checking the process list and Ethernet timeouts and resets. This problem was somewhat alleviated by using multiple file-server nodes and changing the batch script to point different processes to the different servers.

The next testing involved the performance of emulators while running on a cluster. The author chose VMWare for this test as it has actually been in use on his desktop machine for over a year. During this time, the following Windows applications have been tested and run successfully: Microsoft Visual C++, Microsoft Office 2000, Rational Rose, ArcView, Global Mapper, Adobe Acrobat, Mozilla, OpenOffice, Lotus Notes, and several others.

Testing involved checking that these applications work under normal usage patterns. For example, Visual C++was tested to see that it could generate Win32 executables that run on a native Windows system. Files created from office and desktop publishing applications were loaded into software running natively under Windows and checked to see that they were valid. The GIS applications were checked by loading several GeoTIFF files and performing common operations such as data overlays and merges. During this time, all applications performed normally. The only problems the author observed were periodic lags in the movement of the mouse due to interaction between VMWare and the native operating system. VMWare can run in full screen mode so that it appears to the user that Windows is actually the native operating system. This appears to sometimes help with the mouse lag.

During 2003 testing, the author installed VMWare on several MCMC Beowulf machines to test how it would run in combination with Mosix. Three dual-processor Dell workstations were used as they had sufficient computational capacity when compared to the older machines that make up the MCMC cluster. The workstations also all have more modern and high-end video cards as compared to the older ISA text-only cards that are in the original sixteen Beowulf machines at the center. The virtual machines on each install were created identically. After this was done, the author installed Windows 2000 Professional under one of the VMWare systems. This installation was then cloned to the remaining VMWare test nodes.

Some of the applications bundled with Windows 2000 were tested on VMWare running on the cluster to check for functionality. The image viewers were run to check that graphically intensive applications would perform normally. Files were loaded into

Wordpad to check that it could read and write correctly. Full-screen mode was turned on and off under VMWare to verify that it was working properly with the Linux nVidia graphics drivers. Also, the VMWare tools were installed inside the virtual machine to check functions such as copy and paste between the X Window System and VMWare itself.

Initially, the author discovered that VMWare should not be allowed to migrate to other nodes of the cluster. Any graphical application in general should not be allowed to migrate due to the extreme slowdown it would suffer. When migrated, the graphical application can still communicate with its home machine through the stub process. However, graphical applications still display to the originating screen, which requires an extreme amount of bandwidth. Remote processes would become bottlenecked by trying to transmit all of this data through the slower network instead of the much faster system bus.

Once configured, VMWare was left running on the workstations in the cluster. Some applications were started that would consume processor time. VMWare on the head node of the cluster was used interactively during testing. The author used this test to check how much processor load VMWare would put on the computers and determine how Mosix would interact with it (for example, would Mosix migrate applications to a node that was running VMWare?).

Testing then began by running several applications on the cluster while VMWare was active. Data was loaded on several nodes so that the test applications could be directed to one or more computers for file access. The above-mentioned 133 Urban Areas software and doq2geotiff were then started in parallel batch mode on different nodes while VMWare was running. Mosix began process migration, and would actually migrate processes to the nodes running VMWare. When the VMWare task itself began to consume large amounts of processor time, Mosix would attempt to migrate the remote processes to another machine in an attempt to balance the system load. The Windows applications running under WINE appeared to migrate without any problems, even to those machines that did not have WINE installed.

There are a few things worth noting from the tests regarding performance. First, Windows inside VMWare would experience occasional small lag when the base Linux system was heavily loaded. This lag was noticeable when mouse movements inside of VMWare would appear to “freeze.” When a node was dynamically added to Mosix, it would not immediately begin processing due to synchronization with the rest of the system. *Extending Beowulf Clusters* details previous Mosix experiences, especially situations where there are more processes than processing nodes.

Discussion

Up to this point, this research appears to answer many questions. Traditional distributed processing applications can run under Mosix and can indeed benefit from the added flexibility that Mosix provides. The author tested nodes by adding and removing them dynamically while processes were running. When nodes leave, the only effect is on any processes that are running on the node that is exiting the cluster as they migrate to another node. Added nodes will start processing after a short delay to synchronize with the rest of the Mosix cluster. Some Windows applications will also run under WINE and then be able to take advantage of distributed processing, especially those that are designed to run in batch.

The testing demonstrated that it is possible to combine all of these technologies so they co-exist on a processing cluster. During runtime, Mosix proved that it could indeed migrate applications, thus showing that some Windows applications could benefit from distributed processing using this method. While speedup was noticeable, it could be improved by studying different data access patterns. Migrating I/O intensive applications can create the situation that multiple machines are trying to access a single file server. This situation can introduce data starvation problems when the file server temporarily becomes flooded with requests. More information on prior research into data-bound processing can be found in *Processing Large Remote Sensing Image Data Sets on Beowulf Clusters*, USGS Open-File 03-216.

With all of these technologies in place, we can finally begin to combine them to make a wide-area computing cluster a reality. The first step would be to install a Linux distribution as the base operating system on the desktop computers that will be part of the wide-area cluster. These systems would then be outfitted with a Mosix-enabled kernel and set up with a list of all the other Mosix machines in the organization. WINE could be installed for people who need to run some Windows applications under Linux. For others, VMWare could be installed for those who need to keep a full installation of Windows. VMWare could be run in full-screen mode so that the desktop user does not need to know or care that Linux is the real operating system on the machine. This insulates them from any knowledge that their computer is a part of the overall Mosix cluster, allowing them to continue their work as usual.

Once this configuration is in place, processes could be run on the cluster to take advantage of the Mosix process distribution. Existing batch processes could be ported to Linux and modified so that they run in parallel instead of sequentially. Batch jobs could be started on any machine that is a part of the Mosix cluster and migrated to machines throughout the cluster. Windows applications through WINE could also be started on any Mosix node and migrated to the other nodes. This method would provide current software systems with distributed processing capabilities without modifying those systems.

If applications such as ERDAS Imagine and Arc/INFO could be made to run under WINE, then this could provide a great benefit to Geographic Information System (GIS) work in the Discipline. These applications can use subtasks that run separately from the main application. These subtasks would be visible to the Mosix kernel and could therefore take advantage of process migration. GIS applications could then use distributed processing capabilities to work on larger datasets and work on other datasets more quickly.

This setup differs from methods such as grid computing and Internet-distributed models like those used by SETI@home. These schemes have more in common with Beowulf-type systems, where applications have to be specifically written for the system. They also take advantage of idle time, where the user manually joins the processing system or it runs through some application such as a screen saver when the computer has been idle for a preset time. Mosix allows the computer to be used simultaneously with user applications as it has a more advanced load-balancing scheme. Many times, even if a computer is in use, the processor is not fully loaded. This means that there are still spare processor cycles that could be used for other purposes. Mosix would allow a better utilization of these resources, as the load-balancing scheme would work to keep each processor running at its full capacity.

Once deployed, the wide-area Mosix cluster would provide distributed processing capabilities to anyone, anywhere, in the organization. Anyone who is on a machine that is a part of the Mosix cluster could start processes on the system. Traditional applications could run without modification. This system could also be used to provide processing capabilities for applications such as web map services.

Future Work

There are several areas that should be examined in the future for creating organization-wide processing clusters. The first issue should be how normal network traffic can affect process migration. Some internal networks are incredibly busy. For example, a one-minute sample on the author's computer at MCMC recorded 722 individual network packets while the computer was not sending any network traffic itself. Testing should be done to determine whether a separate private network should be installed to provide maximum processing performance.

Work should also be done in combining low-latency and kernel pre-emption patches with the Mosix kernel. The Mosix patches touch many parts of the kernel and change the source code fairly drastically in some places. The low latency and pre-emptible kernel patches help to increase system throughput and response by lowering the time it takes for the operating system to respond to certain events. In the Mosix case, these patches could benefit process migration and disk access by increasing the system throughput. These patches are currently incompatible with the Mosix kernel due to the large number of changes that Mosix makes to the standard Linux kernel.

Finally, the author suggests work to improve the interoperability of certain Windows applications and WINE to run them under Mosix. For example, Codeweavers is a commercial company that sponsors WINE and also offers a commercial version that is designed to easily run Windows applications on Linux. They also can be contracted to get specific Windows applications running through WINE. We could use such a mechanism to make sure some applications would run and that they would be able to be distributed as individual processes instead of running through an emulator.

Conclusion

While Beowulf clustering does provide high-end performance for a relatively low cost, it comes with some disadvantages. Applications must be specifically written to work in the message-passing environment on clusters. Each machine is still treated as a separate entity, thus requiring some resource duplication between machines. Load balancing and other issues must be done manually.

Mosix, when combined with Beowulf technologies, allows a much more flexible solution to distributed processing. By transparently migrating processes at the operating system kernel level, Mosix performs automatic load balancing across all of the machines in a cluster, treating them as one large virtual machine. When run on large numbers of machines in an organization, Mosix can then provide a wide-area distributed processing cluster.

Windows applications could also use this distributed processing power. Translation such as that done by the WINE project could allow them to natively run under Linux as separate processes that could be migrated. For other users, an emulator such as VMWare would run Windows and Windows-dependentapplications with the emulator's process space. However, applications within an emulator would not migrate as they are not separate applications and instead run inside the emulator process.

REFERENCES

Barak, Amnon, *Mosix*: The Hebrew University of Jerusalem. <http://www.mosix.org>.

Barak, Amnon, Orlen La' adan, and Amnon Shiloh. *Scalable Cluster Computing with MOSIX for LINUX*: The Hebrew University of Jerusalem. <http://www.mosix.org/ftps/mosix4linux.ps.gz>.

Codeweavers, LLC, *WINE Headquarters*, <http://www.winehq.org>.

Crane, M., Steinwand, S., Beckmann, T., Krpan, G., Liu, S., Nichols, E., Haga, J., Maddox, B., Bilderback, C., Feller, M., Hamer, G., 2001, *A Parallel-Processing Approach to Computing for the Geographic Sciences: Applications and System Enhancements*, USGS Open-File Report 01-465.

Grid Computing Info Centre, *Grid Computing Info Centre (GRID Infoware)*, <http://www.gridcomputing.com>.

Merkey, Phil, *Beowulf History*: Scyld Computing Corporation. <http://www.beowulf.org/beowulf/history.html>.

Steinwand, D., Maddox, B., Beckmann, T., and Hamer, G., 2002, *Extending Beowulf Clusters*: USGS Open-File Report 03-208.

Steinwand, D., Maddox, B., Beckmann, T., Schmidt, G., 2003, *Processing Large Remote Sensing Image Data Sets on Beowulf Clusters.* USGS Open-File Report 03-216.