Systems Administration of Scientific Computing on NT

Rémy Evard & Michail Gomberg
Mathematics and Computer Science Division, Argonne National Laboratory

Abstract

Traditionally, scientific computing has been performed on large and expensive UNIX-based supercomputers. The advent of high quality commodity hardware and software has the potential to decrease the cost of scientific computing and change the way in which it is performed.

We describe the Chicago Pile 6 – a network of computers running Windows NT, built in order to prototype scientific computing methods in this type of environment. We discuss the administrative requirements for the system and our approaches to administering it.

Introduction

The term "scientific computing" refers to an approach to doing scientific experiments by modeling them on a computer. These models take a long time to execute, but they can usually be sped up by writing them using parallel programming techniques and running them on parallel computers. Thus, most large-scale scientific computing is done on large supercomputers at various supercomputing facilities around the world.

Even running in parallel, a scientific computation will typically execute for a long period of time, during which it will read and write very large datasets sometimes reaching into the terabyte range. It is also becoming common for scientists to use advanced graphical techniques to analyze their data and to distribute their computations across multiple computers in order to reduce the execution time.

The Mathematics and Computer Science Division (MCS) of Argonne National Laboratory has been involved in parallel computing and scientific computing for well over 10 years. We currently operate a large IBM SP supercomputer, which is used by scientists from around the world. The SP runs AIX, and the majority of users are UNIX programming experts.

Scientific Computing on Commodity Platforms

The growing significance of NT is interesting to us in a number of ways.

First, we have the same questions that most predominantly UNIX-based sites have. How do we integrate the two operating systems into a consistent and cohesive environment? Is it possible to manage NT desktops as easily as we can manage our UNIX workstations? What security implications does the NT world bring to us? These questions are a part of the day-to-day management of our environment and are related to scientific computing only in that many of our users are starting to have NT machines on their desktops.

A more interesting aspect of NT to us is its role in the commodity computing marketplace. Standard PC hardware has reached a sweet spot in the price/performance curve, and everyone has noticed. By measuring performance of processors and comparing prices, it appears that a PC-processor based supercomputer could be constructed for about a third of the cost of an equivalent supercomputer. This is obviously compelling, but leaves open questions such as the operating system, the network interconnect, and mass storage issues.

The majority of people working in this area are building supercomputer style clusters of PCs running Linux, built around a standard 100baseT Ethernet switch. This approach makes a great deal of sense: most supercomputing applications and tools are built on UNIX, so getting them to run on a Linux-based machine is quite easy. Some fascinating work is going on in this area, perhaps best exemplified by the Beowulf project [1]. Researchers active in commodity supercomputing met at the Pentium Pro Cluster workshop [2] in the spring of 1997.

However, we believe there is another part of the commodity marketplace that will soon influence the world of scientific computing (among others): the incredibly large set of software that runs on the Win32 API. This could have a profound effect on the world of scientific software, in much the same way that Intel processors are affecting the workstation market. To understand what NT could mean to us and to take advantage of it, we decided to build a prototype of an NT-based supercomputing cluster.

The Argonne Chicago Pile 6

First, a word on terminology: while we often refer to this set of machines as an "NT cluster" out of convenience, we are not using any clustering technology in the strict sense of the term. The machines are loosely coupled over an Ethernet, and don't yet do any kind of failover, process migration, or peripheral sharing. A more accurate description of the machines is a "pile". So, the project has taken on the name "Chicago Pile 6" or CP-6. (The previous five Chicago “piles” were nuclear reactors located around Argonne.)

We had a number of goals in mind when we built the CP-6. Among them were the following:

1. To use component parts and off-the-shelf software to try to build a system that could be used for scientific applications.
2. To use the system as a porting testbed for research projects in our division.
3. To test and compare applications performance.
4. To understand the systems administration issues for a network of NT machines.
5. To study the scalability issues that would be involved to build a TeraFLOP-class machine.
6. To explore the potential for integrating scientific computing with desktop computing.

We started out building a small prototype system in order to test the concept. This is an exploration; it's not an attempt to actually turn these machines into a user facility that is supported the same way our supercomputer is. If this project is a success, that will eventually be the goal, but we're not there yet.

We've started with a modest set of machines – 11 Pentium Pro machines, with varying numbers of CPUs. They are connected over a 100baseT Ethernet switch. We will soon be expanding the pile to add substantially more nodes. We plan to grow to around a hundred machines in the near future, so that we will be able to test scalability issues. The current configuration is detailed in Figure 1.

The machines are operated as their own NT domain, which trusts the primary NT domain of our division. Most of the nodes are compute nodes, running Windows NT Workstation 4.0; another node is the nominal front-end (see below), which is running NTRIGUE [3].

Requirements and Solutions

Traditional supercomputers are built around a model that consists of the following pieces:

- One or more front-ends, which a user logs into remotely in order to start their job. In some systems, the user compiles and debugs on the front end as well. The front-end serves as a remote access mechanism, a way to invoke jobs, and a compilation site.

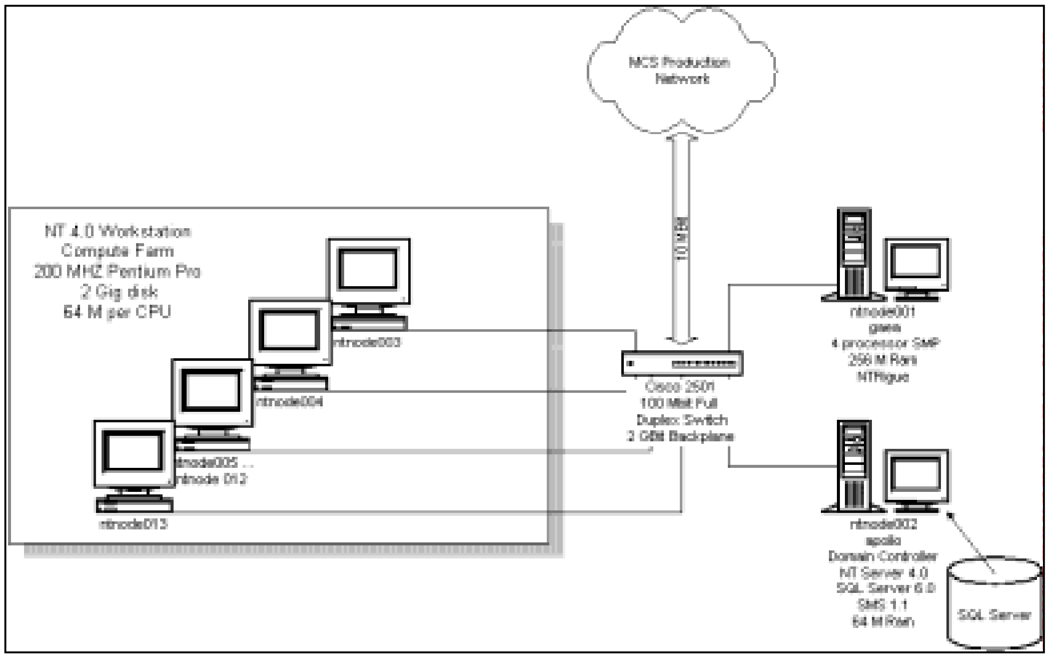
MCS Production Network
10 Mbit
NT 4.0 Workstation Compute Farm 200 MHZ Pentium Pro 2 Gig disk 64 M per CPU
ntnode003
ntnode004
ntnode005 ... ntnode 012
ntnode013
Cisco 2501 100 Mbit Full Duplex Switch 2 GBit Backplane
ntnode001 gaea 4 processor SMP 256 M Ram NTRigue
ntnode002 apollo Domain Controller NT Server 4.0 SQL Server 6.0 SMS 1.1 64 M Ram
SQL Server

Figure 1: CP-6 Configuration

- The compute engine, which may consist of a large number of distinct nodes or may be a set of processors all sharing a large memory space. In an individual node system (like the NT pile), jobs are typically invoked on the compute nodes by a form of remote shell.
- Some kind of scheduling mechanism, which makes policy decisions about which jobs have priority in a multiuser system.
- Data storage mechanisms, which range from standard file systems such as a user's home directory where code is stored, to large, high-performance file systems for large data sets.
- Visualization components, where computations display graphical output, which in some cases (such as Argonne's CAVE [4]), can be used to interact with the computation.

In addition, users have a number of implicit requirements, such as having the same password on all systems, a consistent and uniform file system interface, and ubiquitous email.

In this section, we explain how we have tried to fulfill these requirements on an NT-based system.

Interestingly, nearly every decision has included some aspect of this question: Should we try to make NT look like UNIX, or should we adapt ourselves and our methods to NT's model? On one hand, we want NT to act like the UNIX systems that we understand because we have an enormous base of knowledge, experience, code, and tools that already work. On the other hand, NT doesn't fit into the UNIX mold very well, and the potential advantages of using NT as a commodity system disappear if one layers piles of UNIXisms on top of it.

We don't have a set answer to this; there probably isn't one. Our approach was to set out to build an environment as much like the one as we are accustomed to, and to modify that environment where it made sense or we had to.

Remote Access

Our UNIX users take remote access for granted; rlogin and telnet are things that they do constantly, and this is how they expected to use the NT computing cluster. Remote access, in that sense, is a completely foreign concept in the NT world. It's amazing how often this comes up in discussions, and how little understood this issue is.

To get a handle on this problem (which we certainly needed to understand if people were to use the NT computing cluster without visiting the machine room), we tried to identify what our users mean by "remote access":

- They want to start a process on another computer.
- That process should be running with their permissions and have the same environment (e.g., file systems and environment variables) that it would have if they were sitting at the console of the computer.
- They should be able to do this regardless of whether or not someone else is logged into the console, and their process shouldn't be impacted by the presence of that user or that user's processes, other than possible restrictions on CPU or disk usage.
- They want to be able to run exactly the same programs over this remote connection that they could from the console of the computer.

This entire set of capabilities doesn't exist as a whole under NT, and some of them are foreign to its design. In fact, when we discussed these needs with some NT users, they didn't even understand the desire for remote access.

Here's what we've learned about remote access under NT:

- Starting a process on another NT computer is not hard. This can be done with the NT rsh utility that comes with the NT 4.0 Resource Kit [5], or any of several products.
- The rsh utility that comes with the resource kit has to be installed as a service and can be set to start automatically at startup. We discovered quite early, though, that this rsh doesn't establish the correct security context for the user process on the remote machine. This is a real problem, as the remote process can't access network resources (such as filesystems) that would be available to the user.
- Ataman Software [6] sells a version of rsh that gets around this problem by storing the user's password and then essentially logging in as that user. We're currently using this product as the way to spread processes across the CP-6. This works well for users, but password maintenance has become an administrative burden. Ataman's rsh stores the user's encrypted password in a local registry hive. We have had to write a set of programs and scripts to take that password and propagate it to the other nodes in the cluster. Also, one gets a vague sense of unease with yet

one more place in the registry where passwords are kept with yet another encryption scheme.

- Even under Ataman's rsh, it's not clear how much of the user's environment is available on the remote machine. For example, most of the user registry hive doesn't get loaded during rsh, meaning that user-mapped drives and environment settings won't be present for the process. There's also a strong chance for drive letter collisions, if two different user processes try to map different network drives to the Z: drive, for example.
- Multiuser access of the type described above simply isn't possible under pure NT 4.0. While modified, multiuser versions of NT are available from vendors other than Microsoft, at the time of this writing, they're not a standard part of NT. This is a serious drawback.

Job Invocation and Scheduling

Many supercomputers have some kind of front-end machine, which is used as a way to access the facility, to invoke jobs on it, and sometimes as a place to build programs for it. We initially felt that we would need some kind of front-end for the NT pile, and we thought that this would perhaps help solve the remote access problem described above.

We purchased NTRIGUE [3], a multi-user version of NT 3.51, from Insignia Software. We installed it on a 4-cpu machine and made it available to anyone wishing to use the NT pile. This works well for people who don't have NT machines of their own but are instead using X Windows-based systems; it gives them a location where they can access NT resources and build programs. This machine, however, doesn't perform any particularly important role in the cluster. Jobs can be invoked from any NT machine trusted by the NT machines, not just the NTRIGUE box.

As a way of giving a remote user access to an NT desktop, NTRIGUE works well. The only serious problem we've had with it is that NTRIGUE is based on older versions of NT. Since NTRIGUE requires major modification to the kernel, it is always going to be behind the current OS release. Service packs are also delayed until they are certified to work with NTRIGUE, and some software simply doesn't work because many packages are not capable of functioning in multiuser environments. We expect to see a 4.0 version of NTRIGUE shortly, but this type of relationship between vendors always implies a release lag. (In addition, at the time of this writing, there is rampant speculation about the future of NTRIGUE, as Microsoft appears to be moving toward a similar, or perhaps identical, solution in the near future.)

Some people thought that we could install NTRIGUE on every machine in the cluster, thus solving the remote invocation problem. It's a nice idea, but it doesn't scale. When scientists start a job, they want it to automatically run on a hundred nodes. They don't want to bring up a desktop on each node in order to start the program.

As it turns out, with using Ataman's rsh, a job can be started from anywhere. One simply needs to have the correct files available in the file system, which is possible to do from anywhere in our NT environment.

File Systems Capabilities

We need to use file systems for local storage on each of the nodes, some shared directory for the user's code, and shared applications. We've found the NT file access model to be flexible enough for our initial needs.

We created an identical share (C:\startup) on each node, which all domain users have write access to. Each is shared as \\<node-name>\startup. We will typically put the user's application and data into the share remotely, and then during execution, the program will find its data in C:\startup, regardless of which node it's running on.

Alternatively, users can pick one share and use it for their primary file location, and then access it over the network from all the other nodes. Or, in the same way, they could use their network-based home directory.

These file systems don't address the future need of either very fast file access across multiple nodes, or hierarchical storage, but they do demonstrate that the NT network file system model can be used for our needs.

Future Requirements

As we're currently using the NT cluster as a proof of concept, there are a number of requirements that we've been able to delay. We will need to address these shortly:

- Advanced visualization systems: this should be simple, as the NT world is making incredible headway into the graphics arena.
- Process scheduler: We will probably have to write our own queuing and job scheduling

system if we plan to use the cluster in batch processing mode.

- Parallel and hierarchical file systems: We hope that these will show up as supported products from vendors, possibly as an offshoot of Microsoft's recent scalability push.
- Performance and scalability: These are things we have not yet tested substantially. Eventually we believe the network will be the bottleneck, but by that point, gigabit Ethernet will be an option.

Using the Cluster

The following are the steps that we currently use to run a job on the cluster:

1. The first step is to enable the user to launch executables remotely. Since our startup method is Ataman rsh, we first have to make sure that the user is enabled for rsh on all the cluster nodes. To accomplish this, we have the user type in their Domain password in the Ataman setup on any one of the machines where Ataman is already installed. We then run a set of scripts and Win32 binaries that populate the rest of the cluster with the user/password information via direct registry edits.

2. Next we decide what resources the user will need on each of the nodes. A common requirement is to have local disk where temporary files and output can be stored. We have created a directory on each node called \\<node>\startup. Each node shares this directory to the network. Another requirement may be that each node have a particular DLL in the system directory, usually to reduce the program load time on the node. In that case we'll copy the DLL to \\<node>\c$\winnt\system32 to all the nodes.

3. At this point the user is free to use the cluster to execute jobs. In most cases the user will copy the data and executables to the nodes and then use rsh to launch the executable. An outline of an NT (CMD) batch script can be as follows.

```
for i in (nodes) do copy \\<%i>\startup\data.file
for i in (nodes) do copy \\<%i>\startup\image.exe
for i in (nodes) do rsh %i c:\startup\image.exe.
```

Once the jobs complete, the user can copy the output files from \\<node>\startup. An alternative is to copy a single batch file to all \\<node>\startup directories. The batch file can be started via rsh and should copy the data and image files from \\<home server>\<user> directory. The batch file can also start the executable and copy the output files back to a single share. There is no particular advantage that we're aware of in either method.

4. To monitor the performance of the nodes, we use the perfmon.exe application that is supplied with Windows NT. We monitor such variables as processor usage, or number of IO transactions.

5. In some cases, we need to manually kill runaway processes on remote nodes. To accomplish this task, we have installed the remote kill service that is available with the NT 4.0 resource kit. This allows us to view and kill remote processes on all the nodes from a single command line.

This works for all of the general cases. Some of the users are working on writing their code using NT-style client/server mechanisms, in which case we will experiment with registering their code as an NT service and then try to invoke it remotely.

Administering the Cluster

One of our main goals in building CP-6 was to use it to experiment with different ways to administer a network of NT machines and to really understand the issues involved. For example, our UNIX-based supercomputers are closely integrated with our workstation environment, so we hope to have the CP-6 integrated just as well. At the same time, we want to run the NT cluster the "Microsoft way", assuming we can figure out what that is.

Configuration and Integration

The CP-6 is its own NT domain. It trusts the primary NT domain in MCS. The primary NT domain in MCS is largely independent of our UNIX environment, except that we are using samba on the UNIX servers to provide file and print service to the NT machines. Most users have accounts in both environments, with passwords being maintained separately.

Building Nodes

One of our first tasks was installing the base OS on all of the machines. We wanted to be able to reinstall the OS easily, since these are experimental machines which we intend to break often. While NT comes with some tools to do this, we found that some of the tools were not flexible enough for our installation. The major problem was the lack of support for new network hardware. To counter this problem, we developed our own base installation floppy that contains just enough of LanManager for DOS to start the system in DOS mode and mount a remote drive via TPC/IP. For the rest of the installation we use the recommended NT rollout procedure that is well-described in Microsoft documentation.

Installing Applications in NT

We used Microsoft's Systems Management Server as way to install and keep track of applications. Initially, this had a very serious drawback: it required someone to login in order to invoke the SMS jobs, thereby completely eliminating any advantages of trying to administer the CP-6 from a central machine. However, during the last several months, Microsoft has come out with two additions to SMS called the SMS Installer and the Package Command Manager. These were exactly what we needed, and we are now using SMS to install software onto the nodes of the cluster.

Remote Tweaks

Sometimes users want to do something directly on the machine. We've tried this a number of ways, all of which work reasonably well:

- Ataman's rsh to invoke perl scripts.
- NT Resource Kit remote command line tools
- Win32 API remote registry access

Installing services

Another one of the useful utilities that comes with the NT 4.0 Resource kit is the remote service install tool (SRVINSTW.EXE). This tool can be used to install services on any machine where one has Administrator access. While the tool is useful, it appears that it is limited to installing services that don't require additional registry entries other than those needed to start the service. However, we were able to use this tool to install the remote kill service (wrkillsrv.exe) on all the nodes in the cluster.

Home Directories

Users on the cluster currently have home directories on the primary domain server (apollo). The UNC name for their home directory is:

    \\apollo\users\<username>

When a user logs into an NT console, we map this to Z:. (This is more convenient for use on a workstation than for computing use.)

We started using this naming convention before we had installed Microsoft's Dfs. We have just recently begun to move towards a Dfs-based file system, which provides the ability to create a single logical tree structure for multiple shared volumes anywhere on the network. We will soon rename the UNC volumes such that a user's home directory will become:

    \\dfs\homes\<user>

This allows us to abstract out the name of the server – a much needed feature in UNC names.

Remote Management

We use several utilities for managing the nodes:

- To monitor and log performance, we use perfmon.
- To look at processes on remote nodes, we use pviewer from the NT 4.0 Resource Kit.
- To kill processes, we use the wrkillsrv service and the rkill utility, also from the resource kit.
- To reboot nodes remotely, we use shutgui.
- To rebuild a node from scratch, we have to put a floppy in the drive and use the console of the machine to initiate the rebuild.

Consoles

The consoles of the nodes are managed with a simple, keyboard controllable console switch. We have four consoles available, but could get by with one. We generally only use the consoles for rebuilding, although we had to bootstrap the system (and do initial SMS installs) by logging into each machine.

Integration with the NT Environment

Integrating the CP-6 with the rest of our NT environment is completely painless. The NT domain trust relationships, the UNC naming, and the utilities described above work as they are supposed to.

Integration with the UNIX Environment

Integrating our NT environment with our UNIX environment is not quite so simple.

We are using samba, which provides file sharing and print services from the UNIX machines to the NT machines. We are investigating NFS server products for our NT machines in order to test the performance of using an NT server with UNIX clients.

Our main desire right now is a ubiquitous account creation and management system. We would like to create an account once and then have that user be able to login to both the UNIX machines and the NT domain using the same password. Then, when a user changes the password anywhere, that change should get propagated to the entire environment. We believe this can be accomplished with a series of hacks, but we're hoping that someone will develop a more elegant solution. (NTRIGUE comes quite close to providing this capability now, acting as a gateway between the two. However, one must set it up as the PDC for the NT domain, which we don't want to do.)

Directions and Plans

At this point, scientists in the division are porting their code to NT, and several are using the CP-6 as a test environment. We have done some early performance tests, and those imply that the initial tuning needs to take place in user level code rather than in system configuration.

In the near future, we will be working on serious performance analysis and comparisons, expanding the number of nodes in the pile and looking into high-performance storage and networking solutions.

Summary

The CP-6 has now been operational for several months. We are able to manage the system and users are able to run jobs on it without too much trouble. This is most certainly not a complete system, but it proves the concept, which was one of our initial goals. We are quite certain that a network of NT machines can be used for scientific computing, and that we will be able to manage the environment in a reasonable way. How the system will compare with currently existing supercomputers remains to be seen.

Author and Project Information

Rémy Evard is the manager of Advanced Computing Technologies and Networks in the Mathematics and Computer Science Division at Argonne National Laboratory. He holds an M.S. in computer science from the University of Oregon. His research interests include systems administration and on-line collaboration. His email address is evard@mcs.anl.gov.

Michail Gomberg is a systems administrator in the Mathematics and Computer Science Division at Argonne National Laboratory. He is the principal engineer of the CP-6. His email address is gomberg@mcs.anl.gov.

More information about the CP-6 project can be found at http://www.mcs.anl.gov/ntcluster/.

This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Computational and Technology Research, U.S. Department of Energy, under Contract W-31-109-Eng-38.

References

[1]: The Beowulf Project:
http://cesdis.gsfc.nasa.gov/beowulf/

[2] The Pentium Pro Cluster Workshop:
http://www.scl.ameslab.gov/workshops/

[3] Insignia Systems NTRIGUE product:
http://www.insignia.com/

[4] The Argonne National Lab CAVE:
http://www.mcs.anl.gov/FUTURES_LAB

[5] Microsoft Windows NT Workstation 4.0
Resource Kit, Microsoft Corporation,
Microsoft Press, 1996.

[6] Ataman Software Incorporated:
http://www.ataman.com/