Building the UVIC Grid Testbed

by:

Graeme Smecher
gsmecher@uvic.ca

supervised by:

Randall Sobie
Department of Physics and Astronomy
University of Victoria

April 25, 2002
ENSC 295

Abstract

For the last year or so, a new technology — the Grid — has been garnering press and funding in the field of computational science. If it lives up to its (considerable) promises, the Grid could transform the Internet from a passive collection of computers which interact only as directed by users, to an active collection of resources which organize and optimize their own interactions. This transformation has been compared to the emergence of the electrical power grids from independantly powered sites, and is touted by Grid pundits as the next logical evolution of the Internet.

As part of the Grid Canada effort, a group of research physicists at UVIC has constructed a small Grid-enabled cluster as part of an effort to create a Canadian Grid testbed.

This report describes the requirements of the UVIC Grid testbed, and introduces the technologies used to achieve them. It explains the topology and construction of the testbed, and makes some observations about the degree success which has been achieved. Finally, this report makes some recommendations for the extension or replacement of the UVIC Grid testbed.

Contents

List of Figures

List of Tables

1 Introduction

What follows is an introduction to the situation motivating my co-op semester, and a brief description of my work environment and colleagues at the University of Victoria. Due to some obscure technical content, I have included an introduction to the technologies which we used in the testbed. Finally, I have provided a description of the finished product and some of the details of the work that went into its creation.

I felt that without adequate documentation, my work at UVIC would languish after the end of my term. However, there is immense potential for the expansion and use of the Grid testbed, among other things: therefore, during my stay here, I produced some technical documentation that is related to, though beyond the scope of, this report. It has been attached at the end as an appendix.

First, however, a description of the Grid and its promises is required.

1.1 The Grid

As explained in the abstract, there are two different faces to the Grid. The most obvious is the marketing hype and promises that accompany every press release or presentation; the second, which occasionally gets drowned out in the noise, shows the real effort and talent that has gone into proving the Grid's potential.

There is the optimistic version: The Grid is a glorious new paradigm[1] which results in a good portion of the Internet behaving like one giant computer. Computing resources (be they number-crunching, storage, or bandwidth) are allocated off of the Grid at whim, as required, much as electricity is taken from the power grid. Automagically, your jobs propogate over the planet, finding the most efficient allocations and routing pathways that meet their requirements. New, unexplored avenues of distributed computing become possible. The Grid cures cancer and ends the terrorist menace.

I am a little less sceptical about the other conception of the Grid. Typefied by the Globus Toolkit, this Grid presents a set of tools that allow developers to create applications which may run throughout the Internet. Distributed software is extremely difficult to write well, and I am doubtful that its essence will *ever* be captured in a single piece of finished software that solves all problems and fits every application. However, by providing developers with a reasonably high-level set of tools, much of the repetetive drudgery can be taken out of the development of a distributed application.

"The Grid" doesn't yet tangibly exist — at best, it is not a single entity but a

[1] ...never mind that distributed computing has been around for decades...

large number of fragmented testbeds and prototype installations. The political framework for the Grid is also in its infancy. A number of organizations exist, such as the Globus Project, the Global Grid Forum, and Grid Canada, but most are still recruiting or trying to settle on a mandate.

1.2 The UVIC Grid Team

The University of Victoria team, too, is trying to come to grips with their role within Grid computing. Our Grid efforts are led by staff of the Physics and Astronomy department, with connections to the ATLAS, TRIUMF, and BaBar groups. It is noteworthy that these guys spend most of their time working out more violent ways to smash Greek letters into one another. Then, they write papers arguing about just how fast their letters were going, and exactly how many and what sort of bits they flew into.

Part of the aftermath of a jolly day of smashing involves a lot of analysis. Sometimes, when they can't get their hands on smashing equipment, they simulate smashing instead. Both simulation and analysis require a massive amount of computing power, and some see the Grid as an effective way of using other people's surplus computer time instead of their own.[2]

Table 1 lists the primary people involved with this project.

Table 1: The UVIC Grid Team

- Ashok Agarwal, Research Associate. Ashok supervized much of the construction of the testbed.
- Greg King, co-op student. Greg and I have traded advice throughout the term.
- Dr. Randall Sobie, Adjunct Associate Professor Randy is my co-op supervisor and leads the UVIC Grid effort.
- Graeme Smecher, co-op student extraordinaire.
- Jan Van Ujtven, Research Associate. Jan administers the Physics network and has been an invaluable source of help.

[2]One of the most compelling arguments for the Grid is that by moving onto the Grid, computing centers must only provide for their average computing resource requirements. Unconnected, non-Gridded centers must provide for peak usage. By offloading peak requirements onto the Grid, more efficient use of resources becomes possible.

1.3 Project Focus

Because the UVIC Grid Team is led by physicists, its focus is mostly in getting real applications to do their work more quickly, and more efficiently, by using the Grid. This means, of course, using other people's computers in addition to our own resources. While UVIC has a large computing center, the appropriate equipment here is overused at some times, and underused at others.

With this in mind, Ashok, Randy, and I went to the Global Grid Forum conference in Toronto in February. Grid Canada, an organization intended to foster Grid activities across the country, met and concluded that a national Grid testbed needed to be created. The three of us decided that the Grid team could benefit greatly by participating, and that the creation of a separate, secure, experimental testbed was in order.

At the same time, the Physics cluster (called `muse`) was being upgraded. The newer nodes outperformed older ones by a factor of at least 4, so we were able to liberate 8 of the old ones for our cluster.[3]

2 Technical Introduction

With these pieces, we started to plan the cluster. I was given free rein: the only requirements were Red Hat Linux, Globus, and Condor. First, I worked up a brief list of attributes which I thought were desirable. I did not expect to attain all of them: A testbed is only a testbed. It was intended as a low-risk experiment to guide later projects and identify some of the challenges that they could expect to undergo.

2.1 Goals

The following are some of the considerations that went into the design of the testbed:

2.1.1 Security

Under ideal circumstances, people from all over the world would be able to use our testbed. That means two things:

- Users should be reasonably confident that their data is safe from other users

[3] ...one of which turned out to have set itself on fire, destroying the power supply and motherboard...

- Malicious users should not be able to learn any sensitive information about other users and their work
- UVIC work not related to the testbed must be kept separate and safe at all costs: no resources that are not allocated for the testbed should be used by it

2.1.2 Ease of Administration

With a potentially global user base, the Grid testbed could experience a highly variable number and cross-section of users, with highly different resource requirements.

- It should not be difficult to add, remove, or modify users.
- Monitoring should be automated where possible, so that problems such as full hard drives should not go unnoticed.
- Auditing and ongoing maintainance should not require a significant amount of time.
- Services should be centralized.
- It should be easy to add and remove nodes without disrupting service.

2.1.3 Ease of Use

Users should not be faced with any surprises when using the Grid testbed. Most access will occur through Globus software, rather than interactive user sessions; therefore, “surprises” (APIs that do not work as expected, etc.) could have serious consequences for Grid software not designed to handle them.

Wherever it is appropriate, the testbed should behave as any user would expect a handful of generic Linuxes box to behave.

2.1.4 Performance

The performance of the testbed cluster is not of terribly high importance. Currently, we are using old, castoff equipment. A properly-constructed Grid cluster would be built using, e.g., gigabit ethernet and non-obsolete computers. Evaluating peak performance on outdated nodes and with old network is not terribly meaningful.

Ideal distributed computing applications for the Grid are highly parallel and CPU-bound; that is, they spend most of their time simply crunching numbers on as many computers as are available. On a given node, CPU speed and amount of system memory are the primary constraints; local network speed is potentially a close third. Barring a serious problem, performance is more dependant on the equipment purchased than it is on anything I am able to change.

2.2 Linux

The physics simulation and analysis software being used at UVIC is exclusively written for UNIX machines. Almost all of it is run on Linux.

The BaBar software is an unweildly behemoth of C++, Fortran, Perl, TCL and a handful of other packages. It is sufficiently picky that it will only run on precariously balanced combinations of kernels, compilers, interpreters, libc releases, and databases; even then, it only behaves when the moon is full.

Rather than tempt fate by choosing untested combinations of software, we installed Red Hat Linux, version 7.2.[4]

2.3 NFS

NFS should be familiar to anyone who has more than a passing familiarity with UNIX. It allows a filesystem on one machine to be mounted on another machine. For example, most campus networks (e.g. SFU) mount user's home directories via NFS so that they are identical on many machines within the network.

2.4 Kerberos

Kerberos is an authentication standard developed by MIT, which neccessitates that it is large, complex, and difficult to master. It is a secure way of verifying that "principals" (...agents on the network, generally users or services) are who they claim to be. A well-installed Kerberos "realm" makes it less likely that passwords are sent unprotected across the network where they may be stolen. Additionally, a central Kerberos server obviates the need for each machine to have its own password database, reducing the load on system administrators.

For each server and each user to individually agree on a method of password exchange either requires the use of many separate passwords, or reduces the security of a password to that of the least secure server. Kerberos allows network services and users to delegate all authentications to a single Kerberos

[4] Indeed, even choosing RH 7.2 is a departure. BaBar requires version 6.2. However, many developers have had luck using newer releases.

server. It's quite complex, but the upshot is that Kerberos allows a degree of security on a network in which *nothing* other than Kerberos — including clients, servers, and network connections — is trusted.

A thorough explanation of how Kerberos works is beyond the scope of this document. An excellent (if self-indulgent) dialectic on the design of Kerberos may be found at (Bry88). The official Kerberos home page is at (Var00a), which includes links to many good resources.

2.5 LDAP

LDAP stands for Lightweight Directory Access Protocol. It defines a protocol for accessing a directory server, which has much in common with object-oriented databases.

An LDAP directory is structured like a tree, in which each node is an object. Nodes are uniquely identified by a DN (Distinguished Name), which traces its lineage back to the root of the tree. Objects are instantiations of one or more object classes, and consist of attributes, which may be enforced against a schema. Sources for more information include the OpenLDAP administration guide ((Var00b)) and a good introductory article ((Don00)).

Figure 1 shows a small part of the LDAP tree used for the Grid testbed.[5] Three core UNIX configuration files on each machine in the cluster have been replaced by a single, common LDAP directory. The familiar `/etc/passwd` file, which containing login information, has been replaced by the subtree under `ou=People`. The `/etc/group` file containing UNIX group membership information has been replaced by the subtree under `ou=Group`. Finally, the `/etc/services` file, containing a mapping between TCP or UDP ports and service names, has been replaced by the subtree under `ou=Services`.

The LDAP standard contains a novel authentication scheme. When users authenticate to the LDAP database, they do so by "binding" themselves to an object within it. Notice the `userPassword` field within the user entries: it tells LDAP to authenticate users by verifying their password against that of their principal in the Kerberos database. Furthermore, ACLs (Access Control Lists) attached to each object can determine which users may perform which actions (including reading, writing, comparing, and authentication.)

At this point, a brief explanation of how LDAP and Kerberos work in concert to manage Linux logins would be beneficial.

[5] A few slight simplifications have been made, and a *lot* has been left out.

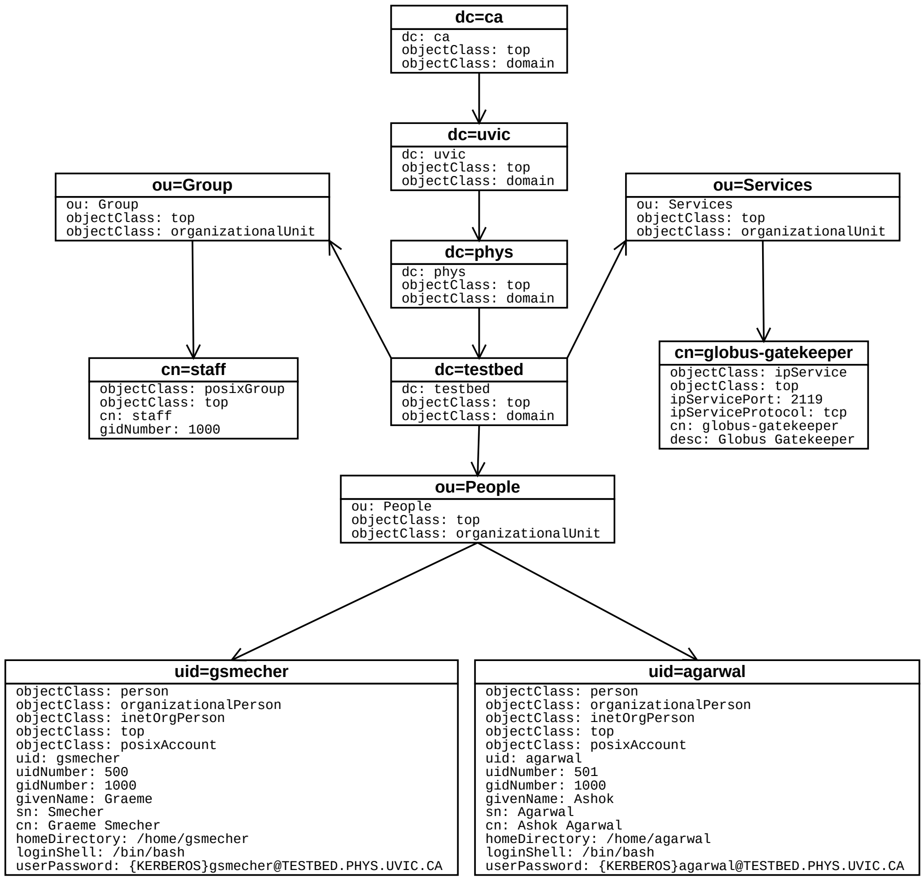

Figure 1: Part of the testbed LDAP tree

2.6 Linux Authentication with Kerberos and LDAP

When users log in to a Linux box, two services are invoked. The first is PAM (Pluggable Authentication Modules), which takes care of user authentication. The second is NSS (Name Service Switch), which provides some core UNIX information services. Both of these services are pluggable: they simply allow Linux to connect to any one of a number of back-end service providers. On the testbed, PAM authenticates against Kerberos, and NSS reads information via LDAP, (...which, is also authenticated through Kerberos.)

The standard combination of Kerberos and LDAP is officially blessed as the LDAPv3 specification. While this combination seems complex, it means that all authentication and login tasks are centralized. A system administrator only needs to change user information within the Kerberos database and LDAP directory, rather than changing several files on each individual testbed node.

Resources written by Turbo Fredriksson were miserable as tutorials, but invaluable as reference materials. They are available at (Fre02b) and (Fre02a). (Note: These documents are constantly being updated.)

2.7 Globus

Globus is the middleware that its adherents believe will form the backbone of the Grid. While it contains a number of complete programs and utilities, most of the effort into Globus' development has gone into the formulation of the APIs (Application Programming Interfaces) that allow other developers to use its libraries in their own work.

Globus provides interfaces and frameworks for common services that most Grid applications will require. It includes tools for resource allocation, security management, directory and information services, and storage and file transfer services.

Much of the software UVIC wishes to run — BaBar included — has not been written to the Globus Toolkit APIs. These packages are large enough that a comprehensive conversion to Globus services would be a major retrofit, and it is unlikely that the BaBar or ATLAS developers are interested. However, Globus tools may still be used to run these tasks as they currently exist on the Grid. In this context, Globus becomes a secure interface between users and computing resources, which may run their own clustering software. At UVIC, we use Condor for clustering.

Installed as such, Globus provides several accessible client/server pairs. `globus-job-run` and the Globus Gatekeeper provide remote job submission services. A specially modified LDAP server and standard LDAP clients provide Grid information services. GridFTP and friends provide extended file transfer capabilities[6].

One of the major facilities that the Globus toolkit provides is a so-called "single-sign-on" environment: a mechanism by which users must "log in" once, using a password. Then, the sign-on program (`grid-proxy-init` in Globus) creates some credentials from that password and some user-specific configuration information[7] which may be used to authorize further actions on behalf of the user without requiring that the user's password be entered again.

The Globus Project may be found at (Var02b). An administrator's guide for the Globus Toolkit version 2.0 provides a reasonable introduction, and may be found at (Bac02).

[6]GridFTP is a superset of the standard FTP protocol, which has been extended for extremely high loads.

[7]...namely, a public-key infrastructure (PKI) implementation using X.509 certificates.

2.8 BaBar

The BaBar software framework is a fairly general family of APIs and programs dealing with particle interactions which I know very little about. It deals with simulation, analysis, and probably many more obscure details besides. The BaBar project is based in Stanford, but it has active users and developers all over the world.

The BaBar software is appropriately named: it is a massive, ungainly beast of a project. It requires many third-party software packages (AFS and Objectivity are particularily nasty offenders), and is picky enough about versions that many utilities included with RedHat 7.2 (such as TCL, Motif, GNU Make, and GCC/G++) won't work without extensive retrofitting.

Because I had no particular wish to go through the nightmare of installing the BaBar release on testbed, I looked for ways to leverage the BaBar installation UVIC already has on its `muse` cluster. After some investigation, I developed a script which extracted a list of script dependencies, shared libraries, and other goodies, and built a remote, minimal BaBar environment. Once Ashok (a local BaBar collaborator) overcomes his shock and suspicion that this works reliably, he will likely publish instructions at SLAC.[8]

The home for BaBar software seems to be at (Var02a).

2.9 Condor

Condor is a cluster solution originally intended for use as a cycle-stealer — that is, it may be configured to run jobs only on computers that would otherwise be idle. While Globus is intended to span the Internet as a whole, Condor seems to behave best on a single site. It was not constructed with security in mind, but provides levels of service Globus cannot.

Using Condor, users may submit jobs complete with a specification of requirements (i.e. available memory) which *must* be met, and preferences (such as machine speed), which bias Condor towards matching certain machines with certain jobs.

Unfortunately, when placed underneath the current Globus release, users lose the ability to specify `rank` and `requirements` for a job. Globus attempts to homogenize access to schedulers and remote resources; occasionally, however, that means presenting the user with the lowest common denominator.

[8]The solution works reliably and is simple to reproduce, but it seems twisted enough to seriously offend those who appreciate elegance in software. In particular, the BaBar software appears to be a hotbed of the beauty-in-design crowd, which might explain the current fragility and complexity of their software.

2.10 Others

In addition to the major services provided above, it became evident that some additional services would be extremely useful to system administration. Because the user (ideally) never directly interacts with these services, I have not given them their own section: however, they are just as important to the overall smooth functioning of the Grid testbed. (Indeed, some are absolutely essential for the sanity of users and administration.)

- Firewalling / NAT Software is required because all of the Grid nodes are only connected to the Internet through a single console machine. In order for these nodes to have Internet access, the console node must provide an IP masquerading service and act as a bridge between the Internet and the internal network. Additionally, firewalling provides better security for the console and filters access to services that should not be accessible from the outside.
- A DNS (Domain Name System) server is required for hosts within the testbed to consistently resolve each other's names without driving administrators insane.
- The DHCP (Dynamic Host Configuration Protocol) allows for hosts to be added or removed from the testbed with almost no manual configuration. Details such as network addresses and hostnames are set by the DHCP server every time a node boots up.
- A web server has been installed to provide some rudimentary documentation. Additionally, we installed a webmail server, which I opted to disable until we determined that it was necessary.
- The SSH (Secure SHell) daemon provides secure remote login services.
- The Name Service Cache Daemon keeps LDAP information cached on each machine. Not only does this ease the load on the LDAP server, but it prevents programs from crashing in certain circumstances.[9]
- The NTP (Network Time Protocol) daemon synchronizes system clocks on all of the Grid nodes. While this may seem trivial, many security infrastructures, including Kerberos and Globus's own GSI (Grid Security Infrastructure) require strict time synchronization to maintain security. They will not operate correctly if the clock difference between two hosts is too great.

[9] For obscure technical reasons, statically linked programs segfault when NSS performs an LDAP lookup. This is because LDAP libraries are dynamically loaded, and chaining from a static binary, through a dynamic library, to *another* dynamic library seriously confuses the runtime linker. This is not a bug so much as a design flaw, and it will not change anytime soon.

- Two mail daemons (Sendmail/SMTP for sending mail between servers, and IMAP for mail reception by remote clients) provide e-mail capabilities for users.
- A Certificate Authority[10] (CA) provides certificate signing services for the Grid testbed. Globus, and a number of SSL-enabled servers, require the use of certificates, and running our own Certificate Authority (CA) makes using and administering these services simpler.
- A CVS Repository has been provided, largely for my own use. CVS is intended to help many developers work on a single project without driving themselves insane by merging different modifications into the source code. It makes a dandy repository for my documentation, scripts, and configuration files.

3 Building the Testbed

The previous section described the technologies which I chose to integrate into the testbed. The following sections describe exactly where each component was installed on the Testbed, and explains my reasons for choosing such an installation.

3.1 Topology

As listed above, there are an extremely large number of pieces involved in the Grid testbed. Figure 2 attempts to show where each of these pieces is installed.

Note that *all* machines run a number of common services. These have been left out of the diagram, except in special circumstances. The following details are left out for simplification:

- On the firewall host (named `grid.phys.uvic.ca` or `firewall.testbed.phys.uvic.ca`, depending on the network interface), Sendmail is configured to accept mail for the entire domain.
- On the other hosts, Sendmail forwards all mail to `firewall`.
- Condor runs on all hosts, but is configured only to submit or execute jobs on the `grid__` nodes.
- NTP and SSH run on all hosts.

[10] Certificate Authorities are part of the X.509 public-key infrastructure that is used, e.g. by Netscape or Internet Explorer to provide encrypted sessions. Basically, a CA can vouch for the identity of other entities, i.e. users and servers, by signing certificates. I'm not needlessly spewing acronyms: there are many, many more that could be introduced. It's complicated, but very interesting stuff.

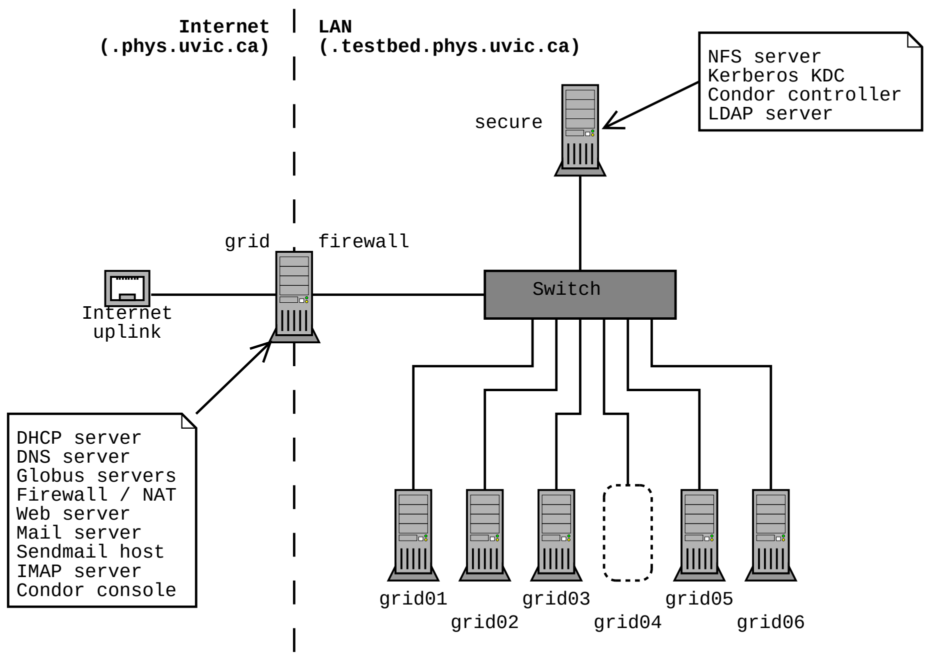

Figure 2: Network topology of the Grid Testbed

Both Globus and Condor are designed to run portions of their servers as the users (`globus` and `condor`, respectively.) Therefore, it was convenient to install both Globus and Condor into their respective home directories (`/home/globus` and `/home/condor`). Furthermore, since home directories are shared via NFS, *all* of the nodes share common installations of both Condor and Globus. The steps involved in setting up new nodes are minimal, and most have been taken care of by modifications I have made to RedHat's remote install (KickStart) scripts.

Some of the technical details of the installation procedures are available as part of the documentation included as appendices; however, comprehensive documentation would be too long and time-consuming to be practical. I will not go into further details about installation unless required.

4 Evaluating the Testbed

As mentioned above, the performance of the testbed is almost entirely dependant on the money spent on individual nodes. Apart from splitting the server duties of the cluster between two nodes (`secure` and `firewall`) to avoid a heavy localized load on the console, I do not feel that measuring the performance of the console provides any indication of success.

Instead, the criteria should be based on practicality and ease of use and administration. However, the testbed is still too new to have acquired a user base, and many details have yet to be polished. Until real users are active and productive on the testbed, it has not truly succeeded. In the meantime, some points can be made about its practicality. Those I could identify are summarized in Figure 2.

Table 2: Successes and difficulties using the Grid testbed

- Pro: Administration is standardized on Kerberos and LDAP. With a little scripting, system management is extremely simple.
- Con: The "perfect" scripts for management have not yet been created. Simple management tasks are a pain to perform.
- Pro: Important tasks "just work". Condor and Globus are rock-solid. Kerberos and LDAP operate very stably.
- Pro: Adding and removing nodes is extremely simple. Customized installation scripts reduce administrative workload to near zero.
- Con: Currently, the number of available execute nodes is quite small (5), and those that are present are reasonably slow (400 MHz).
- Con: The security infrastructure is not perfect. For example, NFS is a weak point for security. (Perfect security is impossible in practice. However, this is a 'con' nonetheless.)

None of the problems listed above are show-stoppers: with better hardware and more effort, the Grid testbed could be perfected. As a testbed, however, it functions extremely well.

I have been able to run BaBar analysis jobs using the minimization procedure described above without problems. The job throughput of the testbed compares to that of the Muse cluster when relative CPU speeds and numbers are taken into account.

One result of the construction of the testbed about which I am particularily pleased is the planned adoption of LDAP for the Physics department's upcoming network upgrades. The purpose of a testbed is to evaluate different technologies without incurring any risk: for the techniques used in the testbed to be adopted "in the real world" is an excellent indication of success.

More useful than a general, wide-scope explanation of Testbed performance, is a brief illustration of how a user might interact with the testbed. Such an illustration follows.

4.1 User interaction with the Testbed

The following section attempts to provide a brief description of user interaction with the UVIC Grid Testbed. No familiarity with Condor or Globus is assumed.

This demonstration requires that the user has obtained all of the neccessary certificates, etc., to use the Grid testbed. The process of configuring Condor-G, Condor, and Globus is not part of everyday use, and detailing these procedures would not further the purpose of this section.

4.1.1 Submitting a Condor job

Recall that Condor is used across the Grid testbed as a local scheduler; that is, it is responsible for providing queue functionality and for spreading jobs around the cluster resource.

When Globus receives a job submission, it re-submits the job using the `condor_submit` command on `grid.phys.uvic.ca`. The following description does not use Globus at all, but simply performs the same task and submits a job directly to Condor.

Since Condor services are not available from the Internet (...because Condor is used as a local scheduler only), the following commands must be performed from one of the nodes within the testbed network.

To submit commands to a Condor queue, we must create a Condor "submit description," consisting of details about the job to be run. An example is shown in Table 3.

Figure 3: A simple Condor submit description

```
executable = /bin/hostname
arguments = -f
universe = vanilla
output = hostname.out
error = hostname.err

queue
```

The contents of this file are largely self-explanatory: the user must specify which executable is to be run (...on the local machine: it will be automatically transferred to the executing machine). The `universe` entry is Condor-specific: for almost all circumstances, the `universe = vanilla` entry should remain unchanged. `output` and `error` specify where Condor should place `stdout` and `stderr` (respectively) on the local machine. Finally, the `queue` command tells Condor to add this job to the Condor queue.

On one of the testbed nodes, the user must simply run the command shown in 4, where `<submit file>` is the filename of the file we created above.

Figure 4: Submitting a Condor job

```
[gsmecher@firewall]$ condor_submit <submit file>
```

The queue status may be monitored using the `condor_q` command. The status of each of the individual testbed nodes may be monitored using the `condor_status` command.

Eventually, the expected output of the `hostname` command should be found in the `hostname.out` file. It should contain the full hostname of whichever testbed node actually executed the job — which is *not* usually the machine from which the job was submitted!

4.1.2 Submitting a Globus job

Globus jobs may be submitted from any host on the Internet which has a proper Globs 2.0 installation. The following steps detail how to run the same command as above (...`hostname -f`) using Globus instead of Condor.

The first step is to obtain a Globus certificate. [11] Simply run the command shown in Table 5.

Figure 5: "Logging in" to Globus

```
[gsmecher@chimera gsmecher]$ grid-proxy-init
Your identity: /C=CA/ST=British Columbia/<...>/CN=Graeme Smecher<...>
Enter GRID pass phrase for this identity:
Creating proxy ..................................... Done
Your proxy is valid until Tue May  7 23:19:28 2002
[gsmecher@chimera gsmecher]$
```

This command performs the single-sign-on step mentioned in the above section on Globus. Now, Globus jobs may be submitted. The command shown in Figure 6, performed on a machine on a separate network from the testbed, results in the `hostname -f` command being executed on the Grid testbed.

[11] For those familiar with AFS or Kerberos, this step is analogous to obtaining a Kerberos ticket or AFS token.

Figure 6: Running a job on Globus

```
$ globus-job-run grid.phys.uvic.ca /bin/hostname -f
firewall.testbed.phys.uvic.ca
```

The machine `firewall.testbed.phys.uvic.ca` is the same as `grid.phys.uvic.ca`, and is the console node bridging the Internet with the rest of the testbed. No matter how many times this command is executed, the job will still run on `firewall.testbed.phys.uvic.ca` (...which is the same as `grid.phys.uvic.ca`). However, this is supposed to be a cluster: why aren't jobs being sent to whichever cluster node is available to execute them?

The answer is simple: Globus introduces the concept of *job managers*, which provide different job execution facilities for different jobs depending on how they are submitted. By default, Globus supplies only the so-called "fork" job manager, which simply executes the job on whichever machine Globus is running on. Different job managers may be chosen by adding a `/jobmanager-name` immediately after the hostname is specified in `globus-job-run`. So, for example, one might run `hostname -f` on a Globus resource with different job managers as shown in Figure 7.

Figure 7: Submitting Globus jobs to different job managers

```
$ globus-job-run grid.phys.uvic.ca /bin/hostname -f
firewall.testbed.phys.uvic.ca
$ globus-job-run grid.phys.uvic.ca/jobmanager /bin/hostname -f
firewall.testbed.phys.uvic.ca
$ globus-job-run grid.phys.uvic.ca/condor /bin/hostname -f
grid1.testbed.phys.uvic.ca
```

The first two use the fork jobmanager listed above. This job manager is invoked explicitly as "hostname/jobmanager", and implicitly if no job manager is specified (e.g. by using only the hostname.)

The third might seem like a bit of a surprise — after all, we submitted the Globus job directly to `grid.phys.uvic.ca`, which is the same machine as `firewall.testbed.phys.uvic.ca`. However, the `hostname -f` command claims that the job was executed on `grid1.testbed.phys.uvic.ca` — why?

On the testbed, I configured a Condor job manager which, rather than running the job locally, uses the same `condor_submit` command introduced above to re-submit the job to the Grid testbed's Condor cluster. Then, whichever machine is available to run the job does so, and returns its output.

Typically, user interaction with the Grid testbed will use the above commands. Condor-G is also useful, but is not explained here for the sake of brevity and because the difference between Condor and Condor-G seems to be a source of confusion.

Though the testbed works, and seems to work well, there are still a few improvements that could be made. These changes would require a significant amount of time and re-working, and some of them are inappropriate until the technologies involved stabilize, but they are still worth consideration.

5 Further areas of work

Should the Grid testbed prove its worth, there are some avenues that we should explore before expanding its size. These possibilities are listed in Table 3.

Table 3: Further areas of work

- Web-based front-ends are desirable for some Grid applications. Services such as webmail would make user interaction easier, and would dovetail with the next-generation OGSA (Open Grid Services Architecture) that the Globus team is currently investigating.
- A high-performance network architecture, using gigabit ethernet, would greatly increase performance without moving beyond consumer hardware prices.
- A centralized, very large file server would bring storage availability and performance far beyond what is available in the Testbed now.
- Per-node Grid metadata could be integrated into the Globus information services infrastructure. Currently, only the console node provides information: the number of cluster nodes is available, but not any details about their hardaware.
- A good, on-line information repository for users (including bug tracking and FAQ) would be invaluable both for users, and to ease the administrative workload.
- Technologies such as AFS or Intermezzo could be included in the network once they stabilize and their interactions with Globus become better understood.

6 Conclusion

This report has explained the circumstances and details surrounding and involving the UVIC Grid testbed. As a result of my work, UVIC has a contained, safe environment in which they may evaluate the Grid and some of the technologies that will be used in its development.

While the merits of the Grid have yet to prove themselves, and the testbed has yet to acquire a large user base, the Grid testbed is ready for use as soon as it is needed.

Glossary

References

[Bac02] Charles Bacon. Globus 2.0 admin guide. `http://www.globus.org/admin/guide.html`. Viewed Apr. 26, 2002., Pub. 2002.

[Bry88] Bill Bryant. Designing an authentication system: a dialogue in four scenes. `http://web.mit.edu/kerberos/www/dialogue.html`. Viewed Apr. 25, 2002., Pub. 1988.

[Don00] Michael Donnelly. An introduction to ldap. `http://ldapman.org/articles/intro%5Fto%5Fldap.html`. Viewed Apr. 26, 2002., Pub. 2000.

[Fre02a] Turbo Fredriksson. Implementing ldapv3. `http://www.bayour.com/Implementing%5FLDAPv3/Implementing%5FLDAPv3.html`. Viewed Apr. 26, 2002., Pub. 2002.

[Fre02b] Turbo Fredriksson. Openldap, openssl, sasl and kerberosv howto. `http://www.bayour.com/LDAPv3-HOWTO.html`. Viewed Apr. 26, 2002., Pub. 2002.

[Var00a] Various. Kerberos: The network authentication protocol. `http://web.mit.edu/kerberos/www`. Viewed Apr. 26, 2002., Pub. 2000.

[Var00b] Various. Openldap 2.0 administrator's guide. `http://www.openldap.org/doc/admin`. Viewed Apr. 26, 2002., Pub. 2000.

[Var02a] Various. The babar homepage. `http://www.slac.stanford.edu/BFROOT/www`. Viewed Apr. 26, 2002., Pub. 2002.

[Var02b] Various. The globus project. http://www.globus.org. Viewed Apr. 26, 2002., Pub. 2002.

Appendices

An awful lot of technical documentation was also written for this co-op semester. For the sake of BC's rainforests, it has not been printed along with this document and is available by request, or by viewing http://grid.phys.uvic.ca/~gsmecher.