Appendix: Specific package issues and solutions

The following sections cover two Grid software packages. Each package has:

- an overview of its purpose and the different roles of the systems involved in it
- references to detailed protocol specifications where these are available, followed by a discussion of the issues involved in deploying the software on a network
- a final section examining other issues and good practices that may be relevant to the particular package.

Many Grid software packages have configuration options that make them easier to deploy on a network, though these are seldom enabled by default. For example, protocols that use ephemeral ports can often be configured to select from a particular range of port numbers to reduce the impact of allowing them through firewalls. In combination with an appropriate choice of logical or physical network topology, these can make deployment significantly easier. Discussions between network and Grid system managers should begin as early as possible in the process of designing a Grid deployment.

The packages described in this edition of the document are those for which information and experience were easily available at the time of writing. As new packages and versions are developed and experience gained, it is hoped to add sections for them to future editions.

Globus Toolkit™

Purpose

The Globus Toolkit™, developed by the Globus Alliance [Globus], provides a set of tools for building computational Grids. It supports a variety of different applications and architectures, so this section will describe individual tools and typical uses rather than a complete system. Each Globus deployment may use different combinations of tools and features so each will require its security measures to be tuned appropriately to suit the particular arrangement of components.

Globus provides three distinct groups of functions:

- resource discovery
- job submission and management
- file transfer.

A common set of authentication and encryption tools for all of these is provided by the GSI (Grid Security Infrastructure). At present (early 2005) there are three different versions of the Globus Toolkit™ either deployed or in development. This section describes versions 2 and 3 (GT2 and GT3), which have similar requirements of the network though there are some

differences in detail and port numbers, which are described in ‘Configuring Networks for Globus; below. Version 3 also includes some web services components which, along with the web services based Version 4 of the Toolkit (GT4 – released in April 2005), will be described in a future edition of this document.

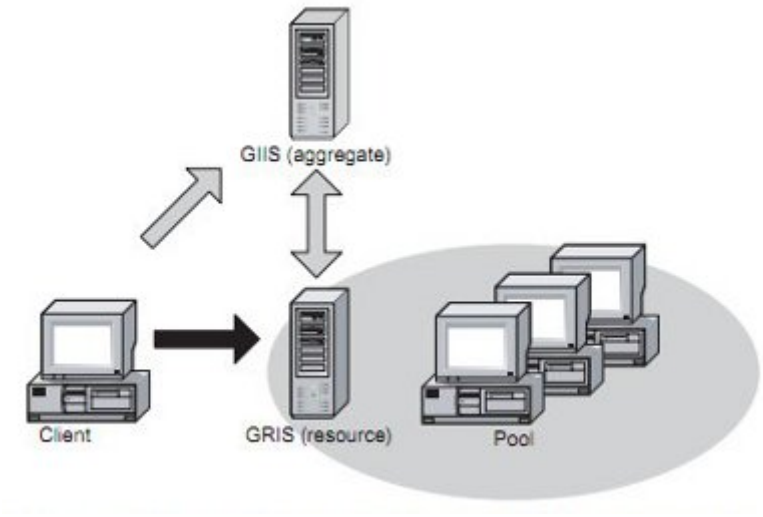
6.1.2 Systems Involved

The Globus Toolkit™ permits computers to play two different roles: as a client, submitting requests, or as a resource, receiving and processing requests. In the simplest model the roles are performed by different computers: the client is the workstation at which the user sits and the resource is the computer on which the request is processed. In other cases, for example when transferring data files or managing parallel processing jobs, the same computer may act as both client and resource. A resource may also be a CPU or storage pool running on a number of computers managed by some other software – Condor® (see section 6.2 Condor®), PBS (Portable Batch System) [OpenPBS] and GridEngine [GridEngine] are often used. In this case a controlling computer accepts requests through Globus and communicates them around the pool using the appropriate local software and network protocols. Scientific instruments, databases and network links may also be treated as resources. Requests may also be delegated using Globus, in which case the resource that initially accepted the request will act as a client to pass on the handling of all or part of the request to one or more other resources.

A person wishing to use Globus will normally do so in two separate stages: first investigating what resources are available, then submitting jobs to appropriate resources. These two steps involve different network flows, so they are described and illustrated separately.

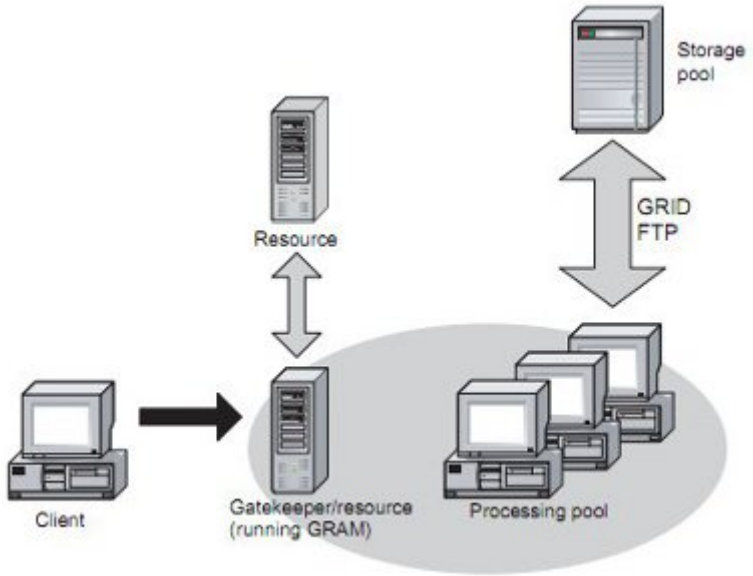
Resource discovery and monitoring services are provided by the MDS (Monitoring and Discovery System). This allows a client to find out what resources are accessible using Globus and, if the resources advertise this information, what CPU and data resources they provide. It also allows resources to provide real time updates and clients to query resources to determine characteristics that may be useful in deciding where and when to run a job. These may include the current status of a resource, the current CPU load, memory and disk space available. The most widely used version of MDS is MDS2, though this is now deprecated for security and support reasons, and alternatives provided in GT3 and GT4 should be used instead. In MDS2, each resource runs the GRIS (Grid Resource Information Service) software, which clients connect to in order to request information. Intermediary systems, known as MDS aggregate servers and running the GIIS (Grid Information Index Server), may also be used to collect information from a number of resources and provide some management of the load of requests made to the individual computers running GRIS. If aggregate servers are used then each resource must be configured to register itself with the aggregate server when it starts. Clients can, in principle, send requests either to the aggregate server or direct to the GRIS software running on the resource.

Figure 4: Globus: Using GRIS and GIIS to discover and monitor resources



[1]

Job submission and management are provided by the GRAM (originally Globus Resource Allocation Manager, but more recently defined as Grid Resource Allocation and Management). This essentially allows a client to request the remote execution of one or more commands by a command-line environment or web service, and to monitor the subsequent progress of the job. Commands may either be executed directly on the resource machine, or submitted to an existing scheduler running on the resource. In the original (non web services) versions of Globus, a resource providing a GRAM service is known as a gatekeeper. To submit a job, a client contacts the gatekeeper for the appropriate resource. The gatekeeper then handles all interactions with the client, as well as with any processing or storage pools that are required to execute the job. A resource may, in turn, act as a Globus client if part or all of the job is to be run on another Globus resource.



l management

[2]

In some cases, it may be necessary to transfer files between computers involved in a Globus job: for example, a job may require files to be fetched from one Globus resource for processing on another. File transfer between Globus resources is commonly performed using GridFTP, an incompatible variant of normal FTP that uses multiple data channels and TCP

tuning to obtain the best transfer rate. GridFTP also provides authentication and, optionally, encryption. GridFTP transfers may take place between any pair of Globus resources (provided at least one of them is running a GridFTP server) and may be controlled either by a process running on one of the resources involved, or by a third resource that is managing the job but not executing this part of it. In the latter case, both resources between which the file is transferred must be running a GridFTP server.

Overview of Network Flows

As described above, the different subsystems of Globus involve different network flows. For clarity these are described separately in this section, but it is likely that in any particular Globus deployment most, if not all, of the computers will be running more than one subsystem and so will participate in more than one type of flow.

Network connections between Globus systems make use of the GSI (Grid Security Infrastructure). The use of GSI does not alter the flows or ports used, but does mean that network traffic is authenticated and may be encrypted at the transport layer.

Resource Discovery and Monitoring

Versions 2 and 3 of the Globus MDS use different terminology and well known port numbers, but otherwise the same network flows are involved.

Each resource runs an information service, known in MDS2 as GRIS, which listens on a well known port for queries from clients. Clients then connect to the information service ports on resources they wish to query.

A resource may also be configured to register itself with one or more aggregate servers. In MDS, these run the GIIS. If this is configured, the resource will make a connection when it starts up to the well known port on the aggregate server. Regular updates about the status of the resource will be sent across this connection. The aggregate server may also establish its own connection to the information service on the resource if it needs to make additional queries. Clients connect to the same well known port on the aggregate server if they wish to make aggregate queries, or may still make direct queries to the information service on the individual resource.

Note that resources may also act as clients and send queries to information services on other resources.

Job Submission and Management (GRAM)

Job submission in Globus involves direct connections between a client and the resource to which the job is first submitted. During the course of execution a job may require a number of different connections between these two systems. There are some differences of detail between version 2 (GT2) and 3 (GT3) of the software.

When a client wishes to submit a job to a Globus resource it first connects to the gatekeeper process (under GT2), or the GRAM service (under GT3), on that resource, running at a well known port. In GT2 a Job Manager process is created for each new request, the client is informed of the ephemeral port on which its Job Manager is listening and makes a further

connection to that port to submit and manage the job. In GT3 job submission and management is done through the initial connection to the well known port. In some cases, files needed for the job will be on the client machine. These may be data files, executables, or batch files containing a sequence of commands to be executed. To transfer the files between the client and the resource, a process called staging is used. The client starts its own fileserver program, GASS (Global Access to Secondary Storage), and informs the server which ephemeral port it is listening on. The resource then makes a connection to that port to transfer the files. Staging may also be used to return output files to the client or to transfer the results of some management commands.

Clients may also request that they be notified of various events during the progress of their job. If notifications are required then the client will listen for them on an ephemeral port and the resource will make connections to this port when required to deliver a notification.

Note that resources may also act as clients when themselves submitting jobs or requesting file transfers. File Transfer (GridFTP)

When transferring large files between Globus resources it is common to use the GridFTP protocol. File transfers may occur between any pair of resources and may be requested by a job running on either resource, or by a job running elsewhere. Like traditional FTP, GridFTP commands are sent over a control connection established by the computer requesting the transfer to a well known port on the computer at the other end of the transfer. The data contained in the files is transferred over one or more data connections between ephemeral ports. If only one data connection is used then it will be established from the requesting computer, but if multiple data connections are used then these will be opened whichever direction the files are flowing. ‘PUT’ and ‘GET’ commands will therefore result in connections being created in opposite directions.

Within Resource Pools

Where a Globus resource consists of more than one computer, communications within the resource may use any appropriate software and network protocols. The configuration of networks and flows within a pool cannot therefore be described in general. For Condor® pools, more information is contained in a later section of this document; information about the PBS can be found at [OpenPBS] and Grid Engine at [GridEngine].

Protocol Specifications and Configuration

The excellent document ‘Globus Toolkit™ Firewall Requirements’, by Von Welch [Welch], describes the various protocols used by Globus and their options for working with firewalls. This document also contains a useful table of port numbers for GT2 and GT3. Welch and Mulmo have also written an overview of the issues: ‘Using the Globus Toolkit™ with Firewalls’ [Mulmo & Welch]. 6.1.5 Configuring Networks for Globus

Globus uses a client-server model and TCP connections, but the complexity of some of the protocols can cause problems on networks where ephemeral ports or inbound connections to clients are restricted to protect them from attack across the network. Clients and servers will often be at different locations, connected by a WAN, so this type of restriction may well be encountered. Globus can be configured, as will be discussed, to significantly reduce the

exposure to external threats. It is also possible to carry the Globus protocols in secure pointto-point tunnels. Even if clients are restricted to outbound TCP connections only, it is still possible to use Globus to discover resources and submit jobs to them, though only a reduced set of functions will be available.

Globus Resources (Servers)

The networking requirements of a Globus resource are of a similar order of complexity to a traditional FTP server, though Globus aware firewalls are not yet available as off-the-shelf products. The MDS information services are straightforward as they use a single well known TCP port (2135 for GT2, 8080 for GT3) to contact the GRIS service or Index service on each resource. If aggregate servers are used then resources also need to be able to make outbound TCP connections to the same well known port on the aggregate servers they are configured to register with.

The GRAM job submission service in GT3 uses the same well known TCP port (8080) for inbound connections. In GT2, GRAM has its own well known TCP port (2119) and the Job Manager needs inbound connections to a range of ephemeral TCP ports, though the port range can, and should, be restricted by configuring the resource as described later. In both versions of the software, the resource may need to make outbound connections to ephemeral TCP ports on the client.

Most resources will also need to support GridFTP, which requires the well-known TCP control port (2811) plus a restricted range of ephemeral TCP ports. The same connections may also be made outbound from the server. For dedicated Grids it may therefore be possible to identify a relatively small number of external IP addresses from which such connections will be required and prevent connections from elsewhere, however, in other cases this will be impractical. Those resources that delegate part or all of their jobs also need to be able to act as clients to other resources for which they will need to make and accept connections as described in the next section.

The security of Globus resources should be managed in the same way as any other computer that provides services to the Internet. No unnecessary services or software should be installed on them and perimeter routers should be configured to block connections to ports other than those required to provide the intended services. The software and operating system should be kept up to date, especially with security patches. In particular, any Grids still using versions of MDS2 distributed by the Globus Alliance should be planning to replace this deprecated software. Network and server logging should be configured and checked for any signs of misuse.

Globus Clients

Running the Globus client software may alter the network requirements of a workstation because of the requirement to allow inbound connections from resources to clients. Some networks will have chosen to protect their workstations by a security model that only allows outbound connections from these computers. Such networks are likely to have problems deploying the standard Globus Toolkits™ and are likely to need modified versions of Globus and/or their existing security precautions. A number of technical solutions for this problem are outlined in the following sections, including limiting the Globus port range (recommended in

any case), restricting the Globus resources used, tunnelling and using a subset of Globus. The best solution for any given situation will depend on factors such as the intended use of the network and Grid systems, the security management available for individual systems, and a local assessment of risk.

Providing the MDS information services should not be a problem for network configuration as this requires only an outbound TCP connection from the client to a well-known port on the resource or aggregate server (2135 for GT2, 8080 for GT3).

The initial submission of a job to the GRAM service on a resource is also done by an outbound TCP connection to a well known port (2119 for GT2, 8080 for GT3). Under GT2, an additional outbound TCP connection is required to the Job Manager. The exact port number for this will vary, but will be in a limited range if the resource has been configured to restrict its port usage. However, if the client wishes to use staging of files, or to be notified of events in the processing of the job, this requires TCP connections to be made from the resource back to the client. The range of ports used by clients can, and should, be limited by configuring the client as described later, and may be as small as ten ports per simultaneous user [Mulmo & Welch]. If this is not sufficient to satisfy local policy then a different technical approach will be needed, as described in the following sections.

If inbound connections to clients are allowed then these will expose the clients to threats from the Internet. The security of the client computers must be actively managed through initial configuration and subsequent patching of the operating system and other software.

Configuring the Ephemeral Port Range

The exposure to threat of computers running Globus can be significantly reduced by limiting the range of ephemeral port numbers that will be used for inbound connections and configuring routers to block attempted connections to ports outside this range. This is of benefit to both resources and clients. The port range used by each system is set using the environment variable GLOBUS_TCP_PORT_RANGE: a number of ways to do this are described in [Welch]. Setting this range on clients or servers means that the local router can be configured to allow inbound connections only within this range (plus the static well known ports associated with any services that are being provided). If a client site restricts outbound TCP connections then its router and firewall will also need to be configured with the port range of any resources that are being accessed by clients.

Restricting Globus Resources

A further possibility for reducing exposure to threats in some Globus deployments is to list the resources that a particular site's clients or resources will need to communicate with. If this is a relatively small and static list then a firewall or router can be configured to only allow Globus protocols (and callbacks in particular) to and from the IP addresses on the list. This is the 'clique grid' described in [Hillier], in the main section of the document. These configurations can be made easier by careful allocation of IP addresses, for example ensuring that Globus clients or resources are grouped in IP address ranges that can be addressed in CIDR notation. If systems with the same function are grouped together, router access control lists can use the CIDR notation (which may well be implemented in hardware and so be interpreted at line speed) rather than listing individual addresses.

Tunnelling

Since the Globus Toolkit™ only uses TCP connections, it is possible to set up encrypted tunnels to carry these connections. Such tunnels use a single well known port (for example 22 for SSH) to carry a number of separate connections between two endpoints. This requires a much smaller opening in a firewall or router configuration, and the connection can be established permanently in the preferred direction.

The use of ephemeral ports makes the configuration of tunnels quite complicated. It is likely that all the possible ephemeral ports will need to be set up in advance, so the port range should be reduced as much as possible as described above. Placing all the connections in a single tunnel may also remove the performance benefits of GridFTP over conventional FTP, so tunnels are most likely to be suitable for lower speed client to resource communications. Finally tunnels also need to be set up between individual IP addresses, so sites with a large number of clients or resources are likely to find other methods more suitable.

A paper by Graupner and Reimann [Graupner & Reimann] explains how to set up SSH tunnels to carry GT2 protocols, and the same principles could also be used for GT3. It is also likely that other point-to-point VPN systems could be used in the same way.

Globus without Callbacks

On some networks, the requirement to make connections from resources to user clients will not be acceptable. In these circumstances it is still possible for clients to discover Globus resources (MDS does not require callbacks) and submit jobs to them but the functions available will be limited. In particular it will not be possible to use staging of input or output files to transfer them between the client and the resource and it will not be possible to notify the client of events in the course of job execution. Clients will therefore need to use other methods (for example conventional passive-mode FTP) to transfer and access files, and will need to check the status of their jobs manually.

Other Issues

Network Address Translation and Dynamic Address Allocation

The use of callbacks by Globus needs special treatment if client machines have IP addresses that are either not public or are not static. The former case is common when private IP

addresses (as defined in RFC1918) are used within a site with external traffic passing through a device that performs NAT (Network Address Translation) at the perimeter of the site. Provided the client machine has an address that is allocated permanently to it, a combination of Globus environment variables and settings on the NAT device can allow the protocols to work as described in [Mulmo & Welch]. The client machine must first be assigned its own Globus port range and told to report its Globus hostname as the externally visible name of the NAT device. The NAT device must then be configured to forward all inbound connections to that port range to the private IP address of the client. If more than one machine on the internal network may act as a Globus client they must all have different Globus port ranges assigned.

If, however, the IP address of a client changes during the lifetime of a job then callbacks will not work. This may occur for both private and public IP addresses if, for example, the client obtains its IP address using DHCP (Dynamic Host Configuration Protocol) and the lease is shorter than the duration of the job. If this situation is unavoidable, then only the facilities described in the previous section ‘Globus without Callbacks’ will be available.

Certificates

Globus relies on digital certificates to prove that users are authorised to access systems and resources. Identity certificates are used to prove a user’s identity. Short lived proxy certificates may then be generated from the identity certificate to enable a Globus resource to act on behalf of the user. Ensuring that these certificates remain private and cannot be used by intruders to masquerade as legitimate users is therefore critical to the security of a Globus system. The greatest technical weakness of this model is that certificates must, at various

times, be stored in files on computer disks. Personal identity certificates are often stored on the user’s workstation. A passphrase is generally used to encrypt the certificate file before it is written to disk, though it is hard to enforce the use of passphrases, or to ensure that those chosen cannot easily be guessed. Proxy certificates are generally stored unencrypted on disk on Globus resources. The privacy of disk storage is therefore a significant factor in keeping a Globus-based grid environment secure. Unfortunately, there have been a number of incidents where this protection has been undermined. Permissions on the certificate files have been left too open, poor passphrases have been used or the disk partition may have been part of a network accessible filesystem that could be read from anywhere on the Internet. Firewalls can protect against at least the last of these mistakes by blocking the protocols (for example, AFS (Andrew File System) or NFS (Network File System)) used to access networked filesystems, thus at least containing the scope of accidental exporting of certificate files. An alternative approach is taken by the MyProxy system, which keeps the personal identity certificates on a central certificate store and requires users to authenticate themselves using a strong password, Kerberos or one time password system across an encrypted link to gain access to their identity certificate. This allows enforcement of the quality of the credentials used to protect the identity certificate and may also allow more pro-active management of the certificate store computer than is possible for an end-user workstation. Since MyProxy allows proxy certificates to be generated within the certificate store, the identity certificate never needs to leave that computer. Proxy certificates obtained from a MyProxy server will be stored unencrypted on filesystems during the processing of a job, so there is still a need to ensure the privacy of some filesystems. MyProxy can also allow users to access their Globus credentials from anywhere on the Internet, rather than being tied to the particular workstation that holds their identity certificate. If MyProxy is used, both client and server machines need to

be able to make TCP connections to port 7512 on the MyProxy server. The original MyProxy concept is described in [Novotny] and the current release of the software is at [MyProxy].

Globus References

[Globus] The Globus Alliance: http://www.globus.org/ [3] [visited 22/03/2005]

[Graupner & Reimann] Graupner, S & Reimann, C, Globus Grid and Firewalls: Issues and Solutions in a Utility Data Center Environment: http://www.hpl.hp.com/techreports/2002/HPL-2002-278.html [4] [visited 22/03/2005]

[GridEngine] Grid Engine: http://wiki.gridengine.info/wiki/index.php/Main_Page [5] [visited 22/03/2005]

[Mulmo & Welch] Mulmo, O & Welch, V, Using the Globus Toolkit™ with Firewalls: http://www.grids-center.org/news/clusterworld/0304Grid2.pdf [6] [visited 22/03/2005]

[MyProxy] MyProxy Online Credential Repository: http://grid.ncsa.uiuc.edu/myproxy/ [7] [visited 22/03/2005]

[Novotny et al] Novotny, J, Tuecke, S & Welch, V, An Online Credential Repository for the Grid: MyProxy. Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10), IEEE Press, August 2001: http://www.globus.org/research/papers/myproxy.pdf [8] [visited 22/03/2005]

[OpenPBS] Portable Batch System: http://www.openpbs.org/ [9] [visited 22/03/2005]

[Welch] Welch, V, Globus Toolkit™ Firewall Requirements, version 5: http://www.globus.org/toolkit/security/firewalls [10] [visited 10/05/2005]

Condor®

Purpose

Condor® [Condor] is a workload management system, developed and used over the past 15 years by the University of Wisconsin-Madison, designed to handle CPU intensive sequential or parallel jobs. It provides familiar batch job facilities:

- users submit jobs to a queue
- these jobs are scheduled taking account of local and sitewide policies and priorities
- jobs, and the resources they run on, are monitored and managed
- reports on job progress and completion can be provided to the user.

Condor® can be used to manage dedicated clusters of CPUs (e.g. Beowulf systems), but can also make use of spare CPU cycles on computers that are used intermittently, such as desktop PC workstations. Under Condor®, both jobs and computer resources can express requirements and preferences, giving the job submitter some control over the risks he runs and the resource provider some control over the tasks he accepts, though both are reliant on the statements of requirement being truthful.

Although Condor® pre-dates the idea of Grid computing, it is often used to manage CPU resources within Grid projects. The Condor® software has also been developed to fit smoothly into a Grid environment. The Condor-G package allows Condor® to be used to manage resources controlled by Globus, while an interface is available for Globus to provide authentication, authorisation and job submission to Condor® resource pools. ‘Flocking’ and the configuration of individual Condor® pools may allow individual Condor® installations to be joined into resources crossing multiple organisations.

Systems Involved

The Condor® software contains three main functional components: job management and resource management and pool management. For a computer to be able to process jobs under Condor® it must be running the resource management component. Such computers are referred to in the Condor® documentation as execute machines. For a computer to be able to submit and manage jobs to a Condor® pool it must run the job management component: such computers are referred to as submit machines. Each pool also requires one computer to act as central manager, collecting information about the computers in the pool and allocating jobs from submit machines to execute machines. Additional computers may be configured to take over if the central manager fails. If required, more than one component may be run on the same computer. Condor® pools can therefore have different structures:

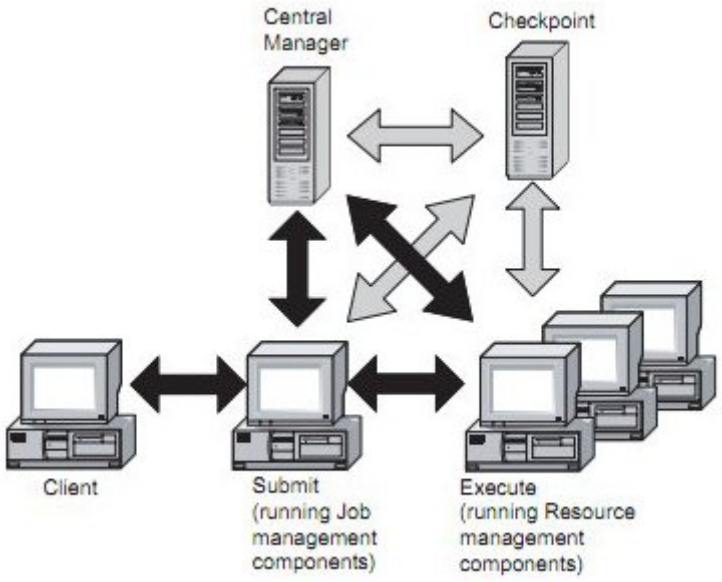
- a hierarchical pool might consist of one submit machine to receive, monitor and manage jobs and a number of execute machines to process them
- or all the computers in the pool may run both submit and execute components, allowing jobs to be submitted and run from, and on, any of the computers in the pool.

Finally, each pool may optionally have one or more checkpoint servers to provide disk space for storing the intermediate results of jobs.

To create and initiate a Condor® job, a user may sit at a submit machine, or may log on to a central submit machine across the network from a computer that is not running any part of the Condor® software. For the purpose of the following discussion such computers that do not run Condor® are referred to as clients.

Figure 6 (overleaf) shows how these components form a simple Condor® pool with a single submit host. The network flows that occur within the pool are discussed in more detail in the following section (the optional checkpoint server and its flows are shown in lighter grey shading in the figure).

Figure 6: Condor®: Job and resource management



[11]

Overview of Network Flows

Flows Within the Condor® Pool

The activity within the Condor® pool is controlled by the central manager, which runs two daemons: the condor_collector, which runs on a well-known port number, and the condor_negotiator, whose port number can be configured. To join the pool, each computer starts the appropriate daemons for its role(s) as submit, execute or checkpoint host and then registers itself with the central manager, informing it what services are offered and what ports should be used to contact them. After registering, each host sends regular status updates to the central manager, by default every five minutes. In a quiescent pool, therefore, network traffic consists of a series of messages from each host in the pool to the central manager.

When a submit host has a job to run, it informs the central manager of this. The central manager selects a suitable execute host, or hosts, and informs the submit and execute hosts of their temporary relationship. The execute and submit hosts then communicate directly to transfer the executable code and data, to run the job and return the results. Subsequent traffic patterns depend on the Condor® 'universe' that has been chosen for the pool. In the 'Standard Universe', all file handling is done by the submit host so there is likely to be a continuous flow of requests and data between the two machines throughout the course of a job's execution. In the 'Vanilla Universe', file handling is done locally by the execute hosts so there will be bursts of traffic between the submit and execute hosts at the start and end of the job. If a checkpoint host is available in the pool, the execute host may communicate with it to send periodic dumps of the status of the job, otherwise any such dumps will be sent to the submit host. There will also be periodic status updates between the submit and execute hosts and between these hosts and the central manager. A Condor® pool that is processing jobs will therefore generate direct network traffic flows between submit, execute and checkpoint hosts. Potentially, any pair of hosts of different types may need to communicate depending on how jobs are allocated around the pool.

Flows into and out of the Condor® Pool

File Servers

Many Condor® pools will operate as described above, with all file access being performed either locally or using the Condor® protocols for remote file access. However, it is also possible to set up a pool based on a shared networked file system, using the NFS or AFS protocols. If this type of pool is used, each host in the pool will need to be able to send and receive network traffic to the computer(s) that hold the shared file systems.

Authentication

Condor® systems can use authentication at two levels. Users should have to prove their identity to demonstrate that they are authorised to submit jobs to the pool. This authentication step can also be used to gain access to the program and data files needed for the job, and to allocate quotas and other limits on the use of resources to ensure that the pool is used fairly. User authentication may be performed by a central submit host when the user logs in to it, or by Condor when a job is accepted by the execute host, so these hosts will need some way to verify that the user is who they claim to be. In many cases this will require access to a networked authentication server, so at least the submit hosts will need to send and receive network traffic to that server. Within the pool, execute hosts may run jobs either as the logged in user, under a dedicated Condor® user account or as user ‘nobody’, depending on how they are configured. Execute hosts may therefore also need to be able to communicate with a networked authentication server. Various options for user authentication are supported, including Globus GSI, Kerberos and Windows®, but not all of these are yet supported on all platforms. In particular, there is no common authentication method supported that covers both Windows and UNIX®/Linux® platforms.

Within the pool, it is also recommended that submit, execute and central manager computers authenticate each other to prevent rogue computers attempting to join the pool. At the simplest level this can be done using IP addresses. Condor® can, for example, be configured to prevent hosts with IP addresses from another site from joining a pool other than through the ‘flocking’ mechanism (see below), or to permit only nominated remote hosts to query the status of jobs. However, it is also possible to configure hosts to require authentication for particular types of request. For example, it might be acceptable to query the status of the pool without user authentication, but to submit a job to a particular execute host might require proof of the submitting user’s identity. Depending on the mechanism chosen, the hosts in the pool may need to contact a networked authentication server to verify the identity of their peers, or users, and will therefore need to be able to send and receive network traffic to that server.

Job Submission from Outside the Pool

Where jobs are initiated from outside the Condor® pool, the user's client computer needs to be able to communicate with the Condor® submit host. Condor® jobs are submitted from a command line, so simple terminal access to the submit host is sufficient for this. Since the user will normally need to provide login credentials to the submit host this command line access should be provided across an encrypted link: in most cases the SSH protocol is used for this.

The user also needs to be able to put data and program files on the submit host or the shared filesystem, and to create and edit submit description files for their jobs on that host. While it would be possible for program and data files to be transferred manually to the submit host, and for job submission files to be edited there using the terminal interface, users are likely to prefer more user friendly options using either file transfer or networked file system protocols. Either of these methods is likely to require additional network protocols to be available to the submit host.

Flocking

Flocking allows jobs submitted in one Condor® pool to be run in another if there are insufficient resources available in the pool from which the job was submitted. If flocking is configured, then a submit host may ask a central manager host in a different pool if that pool can accept the job for execution. This requires the submit host to be able to communicate with Condor® daemons running on the central manager of another pool. If the job is accepted, the execute host(s) in the second pool need to be able to communicate directly back to the submit host using the Condor® ephemeral port range. Since the identity of these execute hosts may not be known to the administrator of the first pool, and may change as new hosts are added to or removed from the pool, it will be hard to configure any firewall or router for these communications without creating a considerable risk that unintended traffic would also be permitted. Flocking is therefore likely only to be practical either where the two pools are within a single unrestricted network or where the membership of each pool is well defined and relatively static. Flocking might also be used where the pool on which the jobs are executed is configured as a dedicated server farm with a contiguous range of published IP addresses and can be accessed from a limited number of submit hosts on each client network.

Protocol Specifications and Configuration

The Condor® manual [CondorMan] contains information about configuring the many options in the package. Some information about ports and firewall traversal is contained in sections 3.10.8 'Port Usage in Special Environments' and 3.7.6 'Condor w/ Firewalls, Private Networks and NATs'.

Configuring Networks for Condor®

Flows Within the Pool

As described above, a Condor® pool involves a mesh of bi-directional network flows between submit, execute and checkpoint hosts, as well as bi-directional flows between each individual host and the central manager. Condor® pools are therefore simplest to set up if there are no network level restrictions (firewalls or router Access Control Lists) between the hosts making

up the pool. However if such controls are unavoidable, it is possible to restrict Condor® to use a limited range of ports. The main central manager daemon, condor_collector, runs by default on the fixed port number 9618. Depending on the version of Condor®, the condor_negotiator daemon may be on the fixed port number 9614 or an ephemeral port. Both of these ports can be configured to any unused value in the range 1024 to 65535. The condor_ckpt_server daemon, which runs on checkpoint hosts, uses four fixed port numbers, 5651-5654, which cannot be changed. (Note that these ports are not registered with IANA and so may also be used by other applications.) Communication to these services normally use UDP for efficiency, for advertisements and updates, but within a single pool can be set to always use TCP where UDP is not sufficiently reliable.

Other Condor® daemons on the central manager, submit, execute and checkpoint hosts use ephemeral ports allocated by the operating system at the time the services start. By default, these may be anywhere in the range 1024 to 65535, but a smaller range can be specified using the HIGHPORT and LOWPORT configuration macros. The number of ports required depends on the components of the Condor® system running on the host, and a number of other factors, as discussed in section 3.10.8.2 of [CondorMan]. Submit hosts require five ports plus another five for each job they control; execute hosts require five plus another five for each concurrent job they execute; the central manager can be configured to require only five ports but, by default, requires twenty-one. A pool where each submit host controls one job at a time, and each execute host runs a single job, could therefore work with as few as ten ephemeral ports, but this would require manual changes to the standard configuration of the central manager.

Some operating systems impose a delay before an ephemeral port can be re-used, which means that either the range of ephemeral ports needs to be increased or the Condor® daemons will occasionally be unable to communicate with other hosts. If Condor® traffic is being passed through a firewall the port range should be set as small as possible, but the correct setting for each pool will depend on local circumstances. Published descriptions of Condor®

pools describe ranges of 11 [Thain] and 1001 [NotreDame] ports. Finding the correct range for any individual installation may require experimentation: if the port range is set too small, the error message ‘Failed to bind to any port wihtin xxx - yyy’ may appear in the log files. The port ranges are set by editing the configuration files. This can be done individually on each machine, or centrally, by making a single configuration file available to the whole pool on a shared filesystem from the central manager, as described in [CondorMan]. If port ranges are restricted in this way, routers or firewalls within the Condor® pool need only permit TCP and UDP traffic to these port numbers, plus the well known condor_collector port on the pool manager, and the checkpoint ports on any checkpoint servers.

These Condor® flows only need to pass between hosts in the pool, so routers or firewalls on or beyond the perimeter of the pool can, and perhaps should, restrict access to these port numbers. Doing so will help to protect the pool from misuse, but will require anyone checking the status of the pool, or a running job, to first log on to the pool by other means. Flocking will also not be possible. The use of the UDP protocol makes Condor® more vulnerable to datagrams with forged IP addresses, so routers around a Condor® pool should also be configured to do ingress filtering for misused local source addresses.

Another option for setting up Condor® pools on a network with internal controls is to use

private addresses from the ranges allocated by RFC1918. Each host in the pool is allocated an RFC1918 address in addition to its public address, and is configured to only use this private address for communication within the Condor pool. RFC1918 requires that traffic using such addresses must not pass the point where an organisation's network connects to the public network or that of any other organisation, but allows their free use within an organisation's own network. Internal firewalls and routers within the network can therefore allow traffic between private addresses to flow freely, while controlling traffic to public addresses. Routers and firewalls at the perimeter must block all traffic using the private addresses. Condor® traffic therefore sees an unrestricted network within the organisation and an impassable perimeter, exactly what the Condor® system favours, while traffic with public addresses can be controlled as the local security policy demands. A malicious host within the perimeter of the organisation can, however, gain access to the pool simply by assigning itself its own RFC1918 address.

Flows into and out of the Pool

As discussed above, a number of network flows need to pass between the Condor® pool and other systems. Details of these will depend on local configurations, but some of the options are summarised in the following sections.

File Access

If a networked file system is used across the Condor® pool, all computers in the pool will need to be able to communicate with the fileserver(s). If NFS is used, the fileserver needs to be visible on UDP (and in some cases TCP) ports 111 (portmapper) and 2049 (NFS) as well as a number of ephemeral ports used by mountd and other associated RPC (Remote Procedure Call) services. For AFS, at least UDP ports 7000-7007 inclusive are required.

E-mail

The Condor® submit host sends e-mail messages to inform the user when certain events occur in the processing of their jobs. Depending on the parameters of the job, this may be when the job completes, suffers an error or creates a checkpoint. Condor® execute hosts use e-mail to inform the Condor® administrator of problems with their daemons while the central manager also sends e-mail to alert the administrator to problems with the pool. In each case e-mail is sent using a system command. Unless these e-mails are delivered within the computers that generate them, machines in the Condor® pool need to be able to send SMTP messages to a mailserver for delivery.

Authentication

Depending on their configuration, the submit and execute host(s) in a Condor® pool may need to authenticate the identity of users before accepting jobs from them. For a few authentication methods, this can take place within the Condor® pool without reference to external systems, but in most cases, some or all of the hosts in the Condor® pool will need to be able to access an external user authentication system. Condor® supports a wide range of authentication systems on different platforms, so the protocols required between the Condor® hosts and the authentication systems will vary. Whatever authentication systems are chosen,

the network controls around and within the pool need to be configured to allow them to work.

Client Access

To submit a job to a Condor® pool, the user must enter a command at the command line when logged in on a submit host. If submit hosts are workstations that users can access directly then this creates no additional network requirement, but in other pools the user will access the submit host remotely. This is normally achieved using SSH as an encrypted replacement for telnet. In these cases the network needs to allow SSH connections on TCP port 22 between the user's workstation and the submit host.

Flocking

As discussed above, if flocking is enabled then submit hosts need to be able to exchange the full range of Condor® protocols with the central manager and execute hosts in the pool to which jobs flock. This requires traffic to the configurable condor_collector and condor_negotiator ports on the pool's central manager. Connections from the execute hosts to the submit host will have destination ports within the ephemeral port range defined for the submit host's pool, whereas connections from the submit host to the execute host will have destination ports within the range defined for the execute hosts' pool.

Other Issues

System Security

Condor® clusters are often hosted on machines that do not otherwise provide networked services. For example, workstation clusters are often configured with no externally visible services and protected by firewall or router configurations that prevent all inbound connections. With these precautions, patching to correct remotely accessible vulnerabilities may have a lower priority. However, installing Condor® on such machines creates the possibility that they will be attacked across the network, so their security management regime may well need to be altered. A number of sites have addressed the issue of patching by including Condor® in a standard software image that can be maintained centrally and rolled out automatically to all workstations. This avoids the need for frequent manual software updates to individual workstations. Although network traffic within the cluster needs to be relatively unrestricted, it is desirable to establish a security perimeter around the cluster with incoming connections restricted to specified submit hosts, which can have more active security management regimes. By default the Condor® services need to be installed with super user privileges, which means that any security bug in the software is liable to give an attacker complete control of the attacked computer. Work is in progress to allow most of the Condor® system to run as an unprivileged user on Linux®/UNIX® systems, which will reduce this vulnerability when the amended version of the software is available.

Globus Extensions

If either of the Globus extensions to Condor® is used then the submit hosts concerned will also need to use the Globus protocols described in the previous section. A submit host running the Condor-G software acts as a Globus client, so will need to communicate with the

Globus Gatekeeper for resources accessible through it. A submit host running the Globus interface acts as a Globus Gatekeeper and will need to communicate with the Globus clients that submit jobs to the Condor® pool.

Condor® References

[Condor] Condor® Project Homepage: http://www.cs.wisc.edu/condor/ [12] [visited 23/03/2005]

[CondorMan] Condor® Team, Condor® Version 6.6.5 Manual: http://www.cs.wisc.edu/condor/manual/v6.6 [13] [visited 23/03/2005]

[NotreDame] About the CCL Condor® Pool: http://www.cse.nd.edu/~ccl/operations/condor/ [14] [visited 23/03/2005]

[Thain] Farming with Condor®: http://www.bo.infn.it/calcolo/condor/farming/siframes.html [15] [visited 23/03/2005]

Links
[1] http://community.ja.net/system/files/images/tg-deployinggrids-04.jpg
[2] http://community.ja.net/system/files/images/tg-deployinggrids-05.jpg
[3] http://www.globus.org/
[4] http://www.hpl.hp.com/techreports/2002/HPL-2002-278.html
[5] http://wiki.gridengine.info/wiki/index.php/Main_Page
[6] http://www.grids-center.org/news/clusterworld/0304Grid2.pdf
[7] http://grid.ncsa.uiuc.edu/myproxy/
[8] http://www.globus.org/alliance/publications/papers/myproxy.pdf
[9] http://www.openpbs.org/
[10] http://www.globus.org/toolkit/security/firewalls
[11] http://community.ja.net/system/files/images/tg-deployinggrids-06.jpg
[12] http://www.cs.wisc.edu/condor/
[13] http://www.cs.wisc.edu/condor/manual/v6.6
[14] http://www.cse.nd.edu/~ccl/operations/condor/
[15] http://www.bo.infn.it/calcolo/condor/farming/siframes.html