UK OGSA Evaluation Project[1]

Report 1.0

Evaluation of Globus Toolkit 3.2 (GT3.2) Installation

Created 16 July 2004, final version 24 September 2004

Document URL: http://sse.cs.ucl.ac.uk/UK-OGSA/Report1.doc

Report Editor: Paul Brebner, UCL[2]

Project Members:

Paul Brebner[3], Wolfgang Emmerich[4], (with assistance from Tom Jones[5]), Jake Wu[6], Savas Parastatidis[7], Mark Hewitt[8], Oliver Malham[9], Dave Berry[10], David McBride[11], Steven Newhouse[12].

Abstract

The initial goal of the UK OGSA Evaluation Project was to establish a Globus Toolkit Version 3.2 (GT3.2) test-bed across four organisations. This report details our experiences to date installing, securing and testing GT3.2. We discuss the project context and Globus Toolkit characteristics impacting the project, particularly the demanding nature of cross-organisational deployment, and the research nature of the middleware. The evaluation methodology is then driven by scenarios for Core Installation, "All Services" Installation, and Security. For each, we list the steps taken, and explore some of the issues encountered. In conclusion, we identify the need for improvements in the quality of grid middleware (in terms of ease of installation, documentation, and support); better support for remote tasks (including installation, deployment, testing, management, monitoring and debugging); and more scalable processes and tool support for security.

---

[1] "Establishment of an Experimental UK OGSA Grid", an EPSRC Funded Project.

[2] Email: P.Brebner@cs.ucl.ac.uk
[3] Computer Science Department, University College London
[4] Computer Science Department, University College London
[5] IT Support, Computer Science Department, University College London
[6] School of Computing Science, University of Newcastle upon Tyne
[7] School of Computing Science, University of Newcastle upon Tyne
[8] School of Computing Science, University of Newcastle upon Tyne
[9] National e-Science Centre, University of Edinburgh
[10] National e-Science Centre, University of Edinburgh
[11] London e-Science Centre, Imperial College London
[12] OMII, University of Southampton

Contents

1 Introduction

The initial goal of the UK OGSA Evaluation Project [1] was to establish a Globus Toolkit Version 3.2 (GT3.2) test-bed across the four organisations involved in the project (UCL, Imperial, Newcastle, Edinburgh).

What is OGSA and why did we choose to evaluate it by using GT3.2? At one level, OGSA is just a Service Oriented Architecture (SOA) specifically intended to support Grid applications. OGSA [25] specifies higher-level services that are motivated by, and designed to satisfy, a set of Grid functional requirements (Use Cases [24]) and also Grid non-functional requirements. The categories of services covered are:

- Infrastructure services
- Execution Management services
- Data Services
- Resource Management Services
- Security Services
- Self-Management Services
- Information Services

A useful OGSA specific glossary of Grid related terms is also available [26].

It is significant to note that OGSA is *not* a layered architecture. There is no clear boundary between layers, or precise location of services in tiers; services in higher *logical* tiers are not dependent on services in lower *logical* tiers. Rather, services are loosely coupled peers that can be combined in different ways to realise particular capabilities. These characteristics contribute to the perceived power and flexibility of SOAs, but have the potential to make the understanding, evaluation, and use of OGSA more complicated than traditional n-tier architectures and technologies.

OGSA Services will be built on core infrastructures (not necessarily one). At the time this project started, the chosen infrastructure was OGSI - the Open Grid Services Infrastructure. We decided to evaluate one implementation of OGSI, GT3.2, as an exemplar, as this was the only real candidate available at the time. Nevertheless, the evaluation and conclusions were designed to inform the development of future approaches to OGSA Grids, either built on GT, or using alternative technologies and products.

The scope of this report (1.0) is limited to the installation, configuration and testing of the Globus infrastructure itself. An evaluation of service deployment, performance, scalability and reliability, etc, will be covered in Report 2.0 to be produced at the completion of the project.

The logical phases of the evaluation (although they were not necessarily enacted in precisely this order) were as follows:

- Decide what version of GT3.2 and supporting software was required
- Acquire GT3.2 and supporting software
- Install supporting software on all sites
- Install GT3.2 core on all sites
- Configure GT3.2 core on all sites
- Test installation and configuration to ensure interoperability between all the sites
- Obtain certificates, configure and test security on all sites
- Install and test GT3.2 "All Services" on all sites

It would be possible to produce a blow-by-blow account of our experiences including: steps taken (logical and actual); issues; misunderstandings; problems with documentation; possible bugs encountered and workarounds; correct steps to take; etc. However, many GT3 installation "How To" guides have been produced, which is revealingly symptomatic of the real difficulties faced getting GT3 to work [2-12]. There are also many papers of a more general nature on the problems and benefits of GT3, evaluations, and experiences with deploying applications to GT3 [13-21, 31].

Regarding the installation of GT3, a few indicative comments from these papers are:

- "If the stars are in alignment, this might just work"
- "Installing Globus is a Nightmare"
- "… its installation is still painful to users …"
- "… installation of the whole product is far too hard"
- "… painful and difficult to install and maintain"
- "… unnecessarily difficult to install (with respect to the facilities offered)."

Subjectively, our general experiences were not substantially dissimilar. However, the aim of this report is not to cover the same ground again, but to explore some of the more generic and conceptual issues pertaining to the installation and debugging of grid middleware infrastructure across multiple organisations, using GT3 as an example.

The plan of the report is as follows. Section 2 explains the characteristics of the project and the GT3 software - thereby pre-empting some of the likely issues that will be encountered. Section 3 describes the evaluation scenarios, steps required, and problems discovered. Finally, Section 4 summarises the results and lists some recommendations. There are also two Appendices. Appendix 1 analyses the scalability of security administration, and Appendix 2 lists supplementary tools.

2.0 Project Context and GT3 Characteristics

2.1 Project context

In order to understand the reasons for, and the significance of, some of the problems encountered, some background project information is useful.

The platforms used in the test-bed included Linux on Intel machines at three sites, Solaris on Sun hardware at one site, and a Windows client machine (at the same site). All the test-bed sites were universities with their own departmental and university wide system administration, usage/access, and security/firewall policies

The experience of the project members at each site with Globus at the start of the project ranged from no prior experience, 3-4 months GT3.0 experience, to three years experience with GT1.X and GT2.X. Some sites had previous experience running inter-organisational grids based on GT2.

Because of these differences in platforms and past Globus experience, the issues encountered were not uniform across the test-bed sites. Consequently, a significant amount of project effort was spent trying to determine why something worked easily at some sites but not others.

2.2 GT3 Software Characteristics

The nature of the software (both the class and specifics) under investigation inevitably suggests problems that may be encountered during an evaluation of this sort. Following are some a-priori observations about the nature of GT3 and the type of issues that were likely to be found in the installation phase of the project (sometimes informed by the actual results).

2.2.1 Research Software Toolkit vs. Production Quality Product

GT3 isn't intended to be production quality software. It's an open source research toolkit, and therefore lacks extensive high-quality support and documentation. In fact, in some ways OGSA isn't really intended to be a product at all; rather, it's a set of services that are designed to be built on to solve higher level grid problems. There is no particular reason for all of the OGSA services to be supplied by one provider or be implemented in the same underlying technology. However, as the representative candidate for an OGSA implementation infrastructure, GT3 could optimistically be expected to provide a production-quality user-friendly integrated grid middleware solution. However, given the research nature of GT3, this is an unrealistic expectation, and in practice there is only basic in-built tool support for many aspects of the installation, configuration, and management process (as documented later). It's also very command-line, Linux, and script oriented.

2.2.2 Web Services Standards

GT3 is based on non-standard Web services technology. It allows applications to be exposed as Grid services, and wraps other legacy grid functionality as Grid services. This makes it confusing to learn – there is a substantial learning curve to this sort of hybrid technology. It is difficult to know in advance what standard Web Service technologies/practices/tools are supported and will work smoothly with GT3 (e.g. is

there support for SOAP attachments?[13]). Some standard SOAP tools (E.g. JMeter, used for load/functional testing) don't automatically work with GWSDL (The Globus specific augmented version of WSDL). The notion of stateful service instances causes problem for tracing/debugging (e.g. standard proxy interception debugging approaches such as TCPMon can't be used). The choice of development environments that directly support GT3 is also likely to be small due to this divergence from commercial web services standards.

2.2.3 Platform and other Dependencies

GT3 is not 100% portable across platforms. Part of it is written in Java, and therefore in theory can be easily installed and run on any platform. However, binaries are not always available for all versions, requiring some compilation, even of Java code. The bulk of the legacy functionality only runs on UNIX platforms, and it is primarily designed for Linux. Support, testing, and binaries for other UNIXes (e.g. Solaris) may not be as good. Using the right version of supporting software (e.g. Ant, JUnit, compilers, Javac, etc) is also expected to be critical.

2.2.4 Version Churn/Moving Target

Because of reported backwards incompatibility issues with earlier versions of GT3.X we decided to use the latest version publicly available during the project. This necessitated constant upgrades and rework over the first few months of the project as GT3.2alpha, GT3.2beta and GT3.2final released in quick succession. Typical of pre-release software, some problems went away and new ones appeared between versions. Also, because we were not dealing with "full releases" the documentation was often out of synchronisation, inconsistent, or just wrong or incomplete, some knowledge of previous versions was assumed (which because the different project sites had varying prior experience with Globus wasn't always the case), and some missing components (e.g. tools, executables, scripts) had to be obtained from previous versions.

2.2.5 Systems Software and Site Specific Systems Administration Policies

Because GT3 is *perceived* as critical systems infrastructure (i.e. comparable with other production quality systems software), with security requirements, who is responsible for it, and the way in which it is installed and managed can be important. Some sites decided to have it installed and managed by the systems administrators, others had dedicated Globus administrators for this role (who then had to interact with systems administrators for specific tasks), while other sites combined both Globus and systems administrator in the one role.

These choices impacted time to install, security, ability to test/debug, and installation and management approaches.

Aspects of site-specific systems administration policies which interacted with GT3 installation are as follows.

[13] The answer for this is "yes" and "no" – it depends on a number of factors.

- The procedures for providing remote users with Globus accounts, including requesting accounts, account creation, and notification to users, and testing and support.
- The provision of remote access for testing. Alternatives included remote login with password, and remote login with SSH.
- Understanding and requesting host access for users. Access restriction policies included: No restrictions, access from specified client machines only, access to specified ports only. Keeping track of all this information to conduct interoperability testing across all the sites was complex, as (for example), no two sites had the same port number for Globus, or exactly the same access policy.
- The site where the Systems Administrators installed GT3 required extra work to support grid administrator/developer roles. For example, extra effort was needed to allow other users to deploy and undeploy services, change configuration information, and start/stop the Globus container. Some of these were achieved with the aid of setuid programs, and changes to the default Globus configuration files and users environments.

2.2.6 "Legacy" vs Web Service architectures

There was initially some conceptual confusion about the necessary actual order (based on assumed dependencies) of installing and testing the GT3 components. One approach based on the use of "legacy" Grid APIs (e.g. The Master Managed Job Factory Service – MMJFS - new to GT3.2, and designed to expose *legacy* code as a generic grid service, but not as a "first-order service" – i.e. a unique service with it's own WSDL) assumed that the "All services" package, and full security, needed to be in place before any remote testing could begin. Another approach, motivated by a standard Web Services architecture, assumed that only the "core container", with some test services deployed, was necessary as a first step, and that security wasn't needed until later. Following the first approach initially meant delays due to trying to solve installation, security, and access/interoperability problems simultaneously.

2.2.7 Testing and debugging across organisations and firewalls

Not surprisingly (given the nature of the problem) this was not as straightforward as we would have liked. Traditional tools such as "ping" and tcp tracing/routing programs aren't guaranteed to work (in fact, they're just about guaranteed not to) as not all the required ports/protocols are supported end-to-end across multiple organisations and firewalls. There are multiple places where something can, and probably will, go wrong, and little chance of easily finding out what and where. Also, proxy based tracing of SOAP messages (e.g. using TCPMON) doesn't work in conjunction with GT3 stateful service instances. In the commercial world there exists relatively sophisticated tool support (E.g. [38]).

2.2.8 Remote Management and Monitoring

Mature distributed middleware typically provides extensive integrated GUI tool support for remote management and monitoring. This includes management of infrastructure, and deployed applications. It allows remote discovery of nodes, and static/dynamic information about the infrastructure, services, and deployed application, including version information, which services/application are deployed and running, lifecycle information, and resources used, including history and current state, and logging information (including exceptions) [34, 35, 36, 38].

Some products provide the ability to remotely include and manage nodes for clusters, and automatic single point deployment of server and client side applications (e.g. an application deployed on one node becomes available on all the nodes in the cluster, or client-side code is automatically updated on client machines) [37].

This information and capabilities are critical for a variety of tasks including: management of installations, services and applications; performance analysis/engineering and capacity planning; deployment/re-deployment and un-deployment of applications; configuring applications correctly; composition of new applications; detection/diagnosis and rectification of exceptions, etc.

Now, it is possible that GT3 supports all of these features - in theory – however, the end-user would probably need to write all the tools themselves, or modify/integrate/combine existing 3rd party tools to provide the required end-to-end lifecycle management. Because of the effort required to obtain and incorporate extra tools and products, and because the evaluation was limited in scope to OGSA/GT3, and was of limited duration, we could not justify supplementing the GT3 functionality in this way.

2.2.9 "Legacy" Software

Shortly after starting this project using GT3.2 it became apparent that we were essentially dealing with *legacy* software, that was not going to be developed or supported further *in it's current form* by the Globus team[14]. This was presumerably because they had decided that, due to the lack of any take up commercially or by standards bodies of the Open Grid Services Infrastructure (OGSI - which is the way GT3 implements OGSA), they would concentrate on gaining wider acceptance for Web Service standards that support the Grid, and only produce (reference) implementations of these standards [39]. Thus, GT4, which will support Web Services Resource Framework (WS-RF) rather than the now defunct OGSI, was announced in February 2004. This was to be available August 2004, but has since been pushed back to January 2005.

Nevertheless, it was decided to proceed with the evaluation using GT3.2 as there was no other obvious candidates for an OGSA infrastructure, and we expected to discover technology independent issues and solutions that could be applied to future versions of GTX (or other OGSA technologies).

---

[14] Strictly speaking, only the OGSI parts of GT3 are legacy. These are being re-factored into a set of "WS" standards, which will be supported in GT4. Also, pre-WS software in GT3 will still be supported and in the future will be made compatible with the new "WS" standards.

3 Evaluation Scenarios and Results

We now look in more detail at the GT3.2 installation related scenarios, and the issues discovered related to each. The list of scenarios is as follows:

3.1 Scenario Install GT3.2 core
3.2 Scenario Install GT3.2 "All Services"
3.3 Scenario Security

If we had followed the idealised logical order mentioned above, we would have performed the scenarios in the order 3.1, 3.3, 3.2. Instead, we initially tried 3.2, 3.3.

The format of the scenarios (with a few variations) follows the pattern: Name, Goal, Pre-conditions, Post-conditions, Steps, Evaluation. Also note that deployment wasn't an explicit scenario in this round of the evaluation, as it was assumed that only the sample services that come with GT3 would be used, and are automatically deployed to the container upon start-up.

3.1 Scenario Installation of GT3.2 core package (test container)

Goal

Install GT3.2 core package with test container

Preconditions

Know what port GT3.2 will use, and ensure firewall and access requirements are met.

Postconditions

Services deployed in container can be invoked from outside firewall on the port specified, by permitted users from allowed machines.

Steps

- Download
  - And install JVM and Ant if not already available on machine
  - The binary version of the core package
- Install
  - Unzip
  - Generate launcher scripts
- Configure
  - Set GLOBUS_LOCATION
  - Set other environment variables
  - There are other container specific settings, but the defaults should do initially. In a production environment tuning of container settings (e.g. threads), and the use of the "–server" JVM engine is needed.
- Start container
  - Choose container port
  - Change to bin directory

- Run globus-start-container script with required port

Evaluation

Installing GT3.2 core only (if the binary is available, on a single machine, using the supplied "test container", and compared with other scenarios) is relatively straightforward in theory.

However, because our initial approach attempted to get the "All Services" package working (with Security enabled, and, at one site, with Tomcat instead of the test container), the theory and the practice didn't converge, and it proved to be a significant learning curve for non-Linux sites, or those that had limited Globus experience.

Nevertheless, even focussing just on the basic scenario, there are still a number of things that can cause problems with something this simple.

1 Port number conflicts

First, the default port used by GT3.2 container may be in use by other software, requiring a different port number to be chosen. This port number must be communicated with the intended users of the node (or better still, a central registry containing resource information). However, even once another port is chosen, this does not preclude future contention for this port, as it may be used by another service not currently running. It is also possible that between shutting the container down (e.g. for maintenance, or for deploying/un-deploying new services) and trying to restart it again on the same port another service may have been started on that port, thereby preventing GT3.2 from using it. There doesn't seem to be a generic solution for this problem of dynamic port management, including reservation and history. The tools available on UNIX can only report the ports that are currently in use, not which ports have been used. It is also not possible to reserve a port for the use of particular service. The only solution is "good management" of the server, whereby all software using ports is centrally installed and managed to ensure coordination. Of course, this problem isn't specific to GT3.2 alone.

2 Remote Host Access

The second problem can be remote access. Depending on site policies, access to some machines is only permitted from specified external machines, or to/from specified ports. For things to work correctly the client and server machines must allow access on the required ports, and all client machines must be permitted to access the server. Moreover, there may be multiple firewalls and security policies in place between the client and server machines making it non-trivial to determine what the current policies are, who is responsible for each, to request changes to allow access, and test. Consequently it may also be impossible to ensure that the same port can be used for all of the Globus containers in a given Grid Virtual Organisation, due to site specific port access policies (and also due to port conflicts).

3 Remote visibility of installation information

The ability for remote users to determine information about middleware installed on nodes is important to aid resource selection and use, and installation testing and debugging.

Ideally, therefore, a remote user could determine if GT3.2 is installed on a machine, including the version, container type, what parts of GT3.2 are installed, and if the container is currently running. However, this capability is limited in the current version. There is a ping script (globus-ping-container), but this gives only basic information. Using the test container, there is no information available for a standard browser to find (c.f. Tomcat, which provides some management information, lists deployed services, and optionally allows remote management); and telnet only indicates that *something* is listening on a host:port. The Globus service browser can be used, but it gives limited information about the installation and state, and isn't much help for debugging. It would be possible to write and deploy a new service to provide detailed management and configuration information.

4 Division of administration roles

The installation and management of Globus by Systems Administrators adds another level of complexity. One site decided to follow this tactic:

- Because of the nature of GT3 as systems software.
- Because of IT management policies - Globus was to be installed on a shared departmental, rather than project owned, Solaris machine.
- As a way of breaking the project work up based on roles – the role of installer/administrator seemed well defined.
- Because a real production grid is likely to be managed in part by different people, and parts of an organisation, requiring more specialised roles, and co-ordination between them.

However, there were a number of problems with this division or roles. Systems administrators typically had limited prior experience of Globus and GT3, and they were not directly part of the research project (e.g. they were not on project mailing list, so did not have access to exhaustive up-to-date configuration information, and project context). Extra "set-uid" scripts had to be written to provide access to non-administrator users, to scripts in "GLOBUS_LOCATION/bin" directory (e.g. deployment, logging, start/stop container commands, etc), and changes of configuration files and environment to give them write permissions for selected installation and deployment related subdirectories. This added an extra layer of complexity and indirectness/opaqueness.

The Systems Administrators were also used to dealing with production quality systems software, which is easy to install and administer. However, GT3.2 did not always conform to these expectations.

5 Tomcat as the Globus container

One site assumed that Tomcat was the default container for testing, so installed this initially. However, the role division problems (above) made this slightly more

complicated, particularly for deployment, which required changing the default deployment directory location and providing access to different deployment scripts.

However, there may be major benefits of using Tomcat as the GT3 container. The other UK OGSA evaluation project [22] reported that it is possible to manage Tomcat installations centrally, but give users independent virtual environments[15]:

> "Tomcat supports multiple Web application contexts." [22]

However, this may have a deleterious impact on security and deployment:

> "A potential drawback of multiple application contexts is that files located in various Web application contexts need to be accessible to Tomcat, thus enabling developers to access someone else's files." [22]

Also, in their project, deployment is done manually (presumably because the ant deployment files would need to be modified for Tomcat/contexts).

In a production environment it would be worth investigating better ways of isolating the deployment, execution and management of users/services from each other, while ensuring the users have control over these activities.

3.2 Scenario Install "All Services" version of GT3.2

Goal

Install GT3.2 "All Services" package, to enable use of MMJFS [27] to run jobs.

Preconditions

Security must be installed and enabled. See scenario 3.3. Alternatively, security can be performed as part of this scenario (and in fact, may have to be).

In order to compile "All Services", the correct versions, and in some cases, "brands", of Java, Ant, JUnit, C compiler, YACC/Bison, GNU Tar, etc are needed.

Postconditions

An authorised client on each node can run remote applications/jobs on all the other nodes using MMJFS. Note that this is a very basic goal, but nevertheless depends upon the installation, configuration, and correct operation of the core container, the MMJFS component of "All Services", access and security between users and nodes, and the deployment of an application/job on each node.

We make no attempt to test *all* the services available in "All Services" package in this version of the evaluation and report. Instead, we focus on MMJFS and Security.

---

[15] However, we have not determined if "independent virtual environments" are separate containers/JVMs, which would be necessary to prevent conflict between users services. For more information on application contexts see Tomcat docs [23].

Some other services will be evaluated in the context of different architectural and design choices in the next round of the evaluation and report.

Steps

1 Ensure/obtain/install correct versions of supporting software.
2 Install All Services: The exact steps depend on the packages/components required. Everything other than core and GARS requires compilation. See [40].
3 Configure All Services: Again, the exact steps depend on the packages/components required. See [41].

Evaluation

1 What was needed, and in what order

The execution and evaluation of this scenario was somewhat problematic, as exactly which parts of "All Services" were required (and why), and how they were to be configured, wasn't obvious initially. Part of the problem was due to security. Not everything required to set up security was available in the core GT3.2 package. Also, some projects had no prior experience with the "legacy" Globus components, and there was some confusion about what could be done with the core package, and how much of "All Services" was initially required do enable even basic functionality.

The confusion arose in part because this scenario was attempted before it was strictly necessary. In hindsight, it is preferable to get the basic/test container working and tested, and then progress to Security and All Services. In fact, for 3.2final the order we followed was as follows: GT3.2 test container; security; Tomcat container; and then All Services (except that some of All Services had be installed to enable security, so this wasn't a clean division).

2 Testing All Services

In order to debug/test All Services (the MMJFS goal), we had to request and obtain host and client certificates from a Certification Authority (CA). We used the UK e-Science CA [28]. Accounts had to be requested and created for GT3 security on all nodes, for all users, and client certificate subjects and account mappings configured in gridmap files on all nodes for all users. Testing accounts, or SSH login, for users also had to be provided as there was no other easy way to allow users to check logs, and server side exceptions, etc, remotely.

A laborious incremental n-squared process of testing was proposed and partially followed in an attempt to ensure confidence and increasing levels of access-ability and interoperability between the four nodes of the test-bed for each step. In a real production environment however this approach would be unworkable due to difficulties meeting security and account requirements, and limits to scalability.

3 Multiple versions/containers

We encountered concerns with supporting multiple versions or copies/containers of Globus on the same machines. This was due to conflicting port numbers, caution about whether GT2 and GT3 would run safely together, different installations sharing certificates and security configuration (E.g. security unexpectedly working or failing due to use of /tmp by default for certificates), and uncertainty about running multiple versions of All Services (using MMJFS).

4 Installation state

There was also no easy way of checking what parts of Globus had been installed, if they were configured and working correctly, inspecting the settings for each container/installation (without having to manually inspect XML files), or checking security configurations. Furthermore, there was no way of doing any of these remotely.

5 Platform and component/version dependencies

There are five packages to GT3, and the relationships and dependencies between them are confusing. Two are pure Java, one is "everything" (All Services), and the rest (subsets of "All Services") are legacy services in written in 'C'. Linux is the platform of choice for Globus for development and testing, leaving other platforms less well supported. We had problems with compilation of All Services on Linux (related to installation of previous versions), and Solaris.

There were also problems with the re-use of existing versions or configurations of required supporting tools. The easiest solution was typically to start from scratch each time and install GT3 specific versions of all the required supporting software.

3.3 Scenario Security (Core, or All Services)

Note that there is some overlap between this scenario and the last, given the inter-dependency of "All Services" and Security.

Goal

A If performed after "Install Core", and prior to "All Services" Scenarios, then: Secure container, and allow access to authorized users, with specified security protocols, to specific services.

B If performed after, or as part of, "All Services" Scenario, then: Allow the remote execution of jobs via MMJFS for authorized users.

Preconditions

Core package and container installed, and running, and non-secure service invocation tested.

Postconditions

A Secure services can't be called without required security (either due to server configuration or client protocol/identity), and non-secure services can still be called without security, or:

B Jobs can be executed remotely by authorized users.

Steps

- Host certificate
  - Each host requests and obtains certificate from CA
  - Generate Globus format certificate
  - Install certificate
  - Install CA certificate
  - Configure
  - Generate proxy
  - Test/verify certificates and proxies
- Client certificates (Split between client and host)
  - Each client requests and obtains certificate (client) from CA
  - Request/create Globus user account for each host (client/host)
  - Generate Globus format certificate (client)
  - Send certificate subject to hosts (client)
  - Add client subject/Globus account to gridmap file (hosts)
  - Setup security on client side (client)
    - This includes installing CA certificate
  - Generate proxies (client)
  - Test/verify certificates and proxies (client/hosts)
- Services
  - Create deployment security descriptors, send to hosts (developer)
  - Deploy service (hosts)
  - Modify client code to call services with required security protocols (developer)
  - Test/verify service level security (client)

Evaluation

1 Packaging of security infrastructure

Some sites reported having to install the All Services package in order to get security working, due to the lack of some of the security infrastructure in the core package. However, this introduced a problem as the client security protocol had to be changed to continue to work with the All Services version of the security (related to GRIM credentials).

2 Multiple containers and security interactions

Multiple containers running on the same machine by default share the same host credentials. Consequently, it is harder to isolate security configurations across containers (for example, to run one container to run with no security, and another with

security enabled, for testing purposes). Also, changes in security settings for one container can accidentally prevent others from working correctly. In theory this can be addressed by configuring each container to use different proxy certificate.

3 Client side security (on Windows)

At the time of writing the report, we have not got this working correctly, but suspect a problem with the CA certificate on the client machine. Determining the minimum client side installation and configuration to enable security is not obvious. This could be addressed by making an explicit "client" (with and without security) package available.

4 Testing, viewing, and verifying security

There is no obvious or transparent way of determining and checking the security configuration of containers and services, apart from examining the XML deployment files, or invoking the services and examining exceptions[16]. Invocation of the services is the only way of doing this remotely. There is also some confusion in the documentation about the correct way of setting up security infrastructure and specifying security requirements for different *levels* of the system including Virtual Organisations, clusters, servers, containers, factories, services, and instances.

5 Debugging secure calls

We discovered that a proxy interceptor approach (e.g. TCPMON) could not be used to aid GT3 service call debugging. This is inevitable given that stateless instance model of GT3. Any sort of proxy service intervention will not work (without modification[17]) as the address returned by factory services to clients will always have the address:port hash of the server, not the proxy, thereby calls to it will bypass the proxy thereafter.

Compared to a pure Service Oriented Architecture, which supports the rich use of a variety of proxy services, this is a weakness in OGSI/GT3. It is possible that the move to OGSA/GT4/WS-RF will facilitate the use of proxies through the provision of "handle Resolvers" (from WS-Addressing and WS-RenewableReferences, which will work differently to the comparable OGSI mechanism [42]).

SOAPMonitor (part of Axis) would work as a substitute, as it is not a proxy approach, (but rather uses a handler), although it requires installation on the host/container. Appendix 2 contains a list of suggested supporting tools.

6 Client-side security configuration

Ideally security could be set up with no code modification on either server or client sides. However, some code modification is required on the client side in order to call services with specified security protocols. Admittedly the changes required are only a few lines of code [33]. However, it should be possible to do this declaratively, rather

---

[16] However, there is a 3rd party script which provides some information about a Globus configuration: http://gtr.globus.org/article.php?story=20040727151719575

[17] It is conceivable that proxies could be modified to overwrite this information with their own addresses.

than programmatically, with client-side deployment descriptors (E.g. J2EE client side security can be specified this way at deployment time).

7 Security Installation and Management Scalability

To set up authentication and authorisation, each user must have a client certificate, an account on each node in the grid, and an entry in the container gridmap file on each node (which is a mapping between their certificate subject and their local account). However, because GT3 allows finer grain (per service and method level) security, multiple gridmap files may be required per node. If the current security configuration and management process is blindly followed, without better tool support, it is unlikely that it will be scalable for an increasing number of sites and users. See Appendix 1 for an analysis of the scalability of the current security mechanisms.

Ideally there will be improved, and/or automated, processes, to centralise (or federate), automate, replicate (to ensure adequate availability), and synchronise security for VOs and grids. Indeed, security management is a recognised problem in the grid and enterprise computing areas. An increasing number of projects are working on solutions to these issues, and better tools and procedures will hopefully become available [29, 30, 45, 46, 47]. However, it will be important to ensure that these work seamlessly with services. For example, we believe that CAS [47] does not work with grid services yet. Other more detailed evaluations of Globus security have also been undertaken [31, 32].

It is desirable for the Grid community to fully understand the scalability and usability requirements for the security of large-scale grids, the limitations of the current technologies, and to develop and standardise appropriate solutions and procedures, with the goals of ensuring interoperability at the infrastructure (not just the application) level, and seamless integration with OGSA/Globus.

4 Recommendations

We conclude with some recommendations for procedures and technology, and finally with a summary of our findings.

4.1 Procedures

Problems due to the technology, or lack of experience with it, can be mitigated (at least to some extent) by improved procedures. Here are three suggestions from our experiences.

Install and test incrementally. An incremental installation and testing approach starting with the core package only, and ensuring interoperability before moving onto All Services or Security, gives the best chance of success. However, it is still not certain that All Services and Security can be treated independently, as All Services depends on Security functionality, and some sites reported that they needed to install (parts of) All Services to enable even basic container level security. It would be useful to provide distinct packages for core plus container level security, and client side security only.

Understand grid and site requirements. It is essential to understand, document, and communicate Globus and site-specific port and client-machine access requirements, restrictions, and procedures to everyone involved (users, local and organisational systems administrators), and conduct adequate systematic testing. A number of times we found that errors had been introduced by manual processes, but were not detected early enough.

Repository of critical information. Keeping a central repository of critical information can help to avoid redundancy and inconsistency, and can be constantly checked and corrected by everyone (users/Globus/host administrators). This can usefully include information about: client machine addresses; host machine addresses and Globus ports; information on what is installed on each node, including packages, supporting software, and versions; and user names, affiliations and certificate subjects.

4.2 Technology: What could be better about GT3.2?

Installation. A more fully automated install procedure is desirable. For example, some J2EE application servers take only 5 minutes to install from scratch on a fresh machine. This could be achieved in a number of ways, including making the installation process more robust and platform independent, providing more package options for specific uses/target platforms (e.g. client side only package), and more pre-tested binary package options. The necessity of compiling some packages substantially increases the effort required to install GT3.2 on some platforms.

Deployment Tools. XML configuration and deployment files are a "good thing", as long as humans don't have to create, edit, or read them. It is desirable to have tool support for producing deployment descriptors, verifying them before/during deployment (so that errors due to syntax errors or incorrect deployment semantics are found prior to run-time), and viewing/changing settings post-deployment (E.g. as part of a management console).

Management console. In order to more easily manage large-scale grids, it is desirable to provide a management console with information about, and control over the following:

- Nodes which are part of a given grid, or VO, and
  - Ability to include and remove nodes from a grid/VO.
  - Information such as OS, size/type of machine, and what state they are in – up or down, etc.
- Machine and container resources (capacity, history and current usage)
- Support software installed, including the container (version, brand, etc)
- Globus software installed (services, packages, version, configuration, and status, etc)
- Security, deployment and resource usage information for the container (e.g. what users are authorised for container/services, instance cycling rates, JVM performance statistics, etc)
- Services deployed, deployment information, state, number in use, and resource usage and history
- Access to logs, including informational, event, and error logs

- Deploy/undeploy/redeploy services (ones that are already installed on the machine, or better still, ones that aren't yet installed on the machine – i.e. allow local or remote deployment).
- Start/stop/restart container(s)

Because the roles involved in using and managing a grid infrastructure will frequently cross software and site boundaries, the software infrastructure must be designed to support location independent multiple role scenarios, including remote access to management functions.

Application Server or Grid Services. These types of capabilities are present on some Application Servers (although Open Source Application Servers typically rely on combinations of 3rd party products to achieve similar functionality).

There are two critical issues in building generic grid infrastructures: Interoperability and Implementability. Interoperability is achieved by conformance to protocol and interface standards. This was partially the goal of GT3, by implementing OGSA using OGSI (although the maturity of OGSA was insufficient to provide interoperability at that level, and instead relied more on OGSI to do so). It also tackled the problem of ease of development of grid services, by providing a container and component model (based explicitly on J2EE) to manage service lifecycles [53].

One of the tensions in evaluating Globus is that, on one hand, it is a J2EE-like clone technology, and can therefore be evaluated as an Application Server; on the other hand, it provides a set of high-level grid services, and can be viewed as a service provider, largely independent of the underlying development model and hosting environment. I.e. Is it middleware infrastructure, or a set of services?

If Globus continues to take the middleware path, then it needs to be realised as production quality middleware, at least comparable to enterprise products, and given the anticipated goals of grid computing, probably of even higher quality, usability, manageability and scalability. However, these are non-trivial qualities, leading to some alternative approaches.

One of the problems with OGSI was lack of uptake by alternative vendors. The grid community anticipates that the strategy of obtaining wider acceptance for Grid related Web Services standards, such as WS-RF, will result in a convergence of Grid and Web standards, enabling common infrastructure to be used for both [40]. OGSA will therefore be more easily implementable using commercial off-the-shelf middleware that supports these common standards. However, it is not obvious that an *identical* infrastructure will necessarily result. For example, [54] points out that WS-RF and WS-Notification have an implied complex programming model. It is unlikely that all vendors will agree with the philosophy of the model, or choose to implement the standards in a uniform way. An alternative view maintains that Grid services can already, and probably therefore should, be built on existing (or at least, emerging) standards (Web services) and infrastructure (component technologies and middleware) [55].

4.3 Summary

We have identified the need for better quality "grid" middleware and infrastructure (easier to install and use, high quality documentation, and responsive support). Whether the best route to this is by using production quality Grid specific middleware, or "standard" Commercial or Open Source products, augmented with "Grid" service standards, and with OGSA services deployed on them, is an open question. However, it is obvious that e-Scientists and their support staff are unlikely to have the interest, skills, or time to expend on overly demanding middleware.

Second, there is a need to support remote distributed tasks for the increasing variety of roles in grid computing cutting across organisations, technologies and tools, and traditional role boundaries, including installation, deployment, testing, management, monitoring, and debugging.

Finally, the scalability/n^2 problem needs addressing. More scalable processes for installation, security, testing, and deployment, are required, with appropriate support from tools.

Appendix 1 Scalability of security configuration and administration

Because of the complexity of the security configuration of GT3.2, it is useful to re-examine the approach taken and evaluate the implications for a larger number of nodes and users.

A.1.1 Client related security

The steps followed in the evaluation to set-up security (authentication and authorisation) for every client on every node, are briefly as follows.

The administrator on every node is responsible for:

- Obtaining and entering client machine addresses to local access list (if required).
- Creating local user accounts for each client.
- Obtaining and entering client certificate subjects in the GT3 container gridmap file.
- Removing client certificates when users are no longer allowed access to the facilities (Users may still have a valid certificate, but no longer belong to the project/organisation which has access rights).

Each client must:

- Request and obtain a certificate from the CA, and manage their own certificate (including passwords).
- Request an account from each node, supplying all the required information (possibly including client machine addresses).
- Convert their certificate to the Globus format.
- Send their certificate subject to each node.
- Install their client certificate and the CA certificate on each machine used by them as a client machine.
- Generate a Globus proxy certificate for/on their client machine(s).

Globus allows coarse grain security to be specified in the container gridmap file. The security settings in this file apply to all services deployed in the container. However, finer grain (per service, and per method) security settings may be optionally specified in service specific security configuration and gridmap files [43, 44]. The more specific security configurations take precedence.

Due to the possibly large number of service specific security configuration files, how they are constructed, deployed, managed and maintained is important. Service specific security files would typically be produced as part of the deployment process, and ideally, authorisation would be *role based* (I.e. All users from some virtual organisation or group would be authorised to use a service, or set of services). However, there is no role based security in GT3 at present, and because user accounts are specific to each node, a different service specific gridmap file may be required for each node. User certificates may also have to be added to, and removed from, the gridmap files over the lifetime of the service. How this process will be managed and

synchronised across multiple nodes for large numbers of services and users is critical. The issue of local user accounts is examined further in A.1.3.

A.1.2 Scalability example

A simple thought experiment shows the scale of the problem, even just for managing container level security.

Imagine a linearly (in terms of nodes and users) increasing grid. Assume 1 new node is added every week, and that every node comes with 10 new users (a node is more likely to correspond to a group in an organisation, rather than an individual). Thus, each site must create 10 new local user accounts, and add 10 new users to the gridmap file, each week. However, each new node must add all the existing users as well. There will obviously need to be an adequate mechanism for obtaining information about the existing users.

The problem is largely one of repetition and synchronisation across all sites. By the time there are 100 nodes, there are 1000 users, and in the last week a total of 1990 "add user" actions across all the sites (i.e. 10 users times 100 nodes, plus 990 users for the last node to catch up on), a cumulative total (over 100 weeks) of 100,000 "add user" actions – 1 million minutes of node administrators time (assuming 10 minutes per action[18]).

From a client point of view, a timely and bounded/predictable response time to their request to "join the grid" (to be allowed to use all the resources available) will be expected. If there is a large disparity in the time it takes to gain access to some nodes, effort will be wasted in finding out where the process is up to, and why. There will also need to be a mechanism to discover the machines each user has been granted permission to use so far[19].

Note that from the perspective of node administrators, there does not appear to be any direct benefit for each node in this process. All that is happening is that more users are getting access to their resources, although this is obviously reciprocal, as their users are also gaining access to an increasing resource base. In terms of total cost, the site administration overhead of this security model is something like £300,000 for 100 nodes and 1000 users. This is assuming a cost of £18/hour, and a nominal 10 minutes per user to obtain a user's certificate/subject, obtain a fax form with name, address, and agreement to local conditions of use, set up account, add users machine to list of allowed machines for access, add user to gridmap file, stop Globus container, and restart container. However, the total cost of adding users as the number of nodes increases grows. So, for 10 nodes, the cost per new user is £57, for 100 nodes the cost per new user is £597, and for 1000 nodes the cost per new user is £5,970.

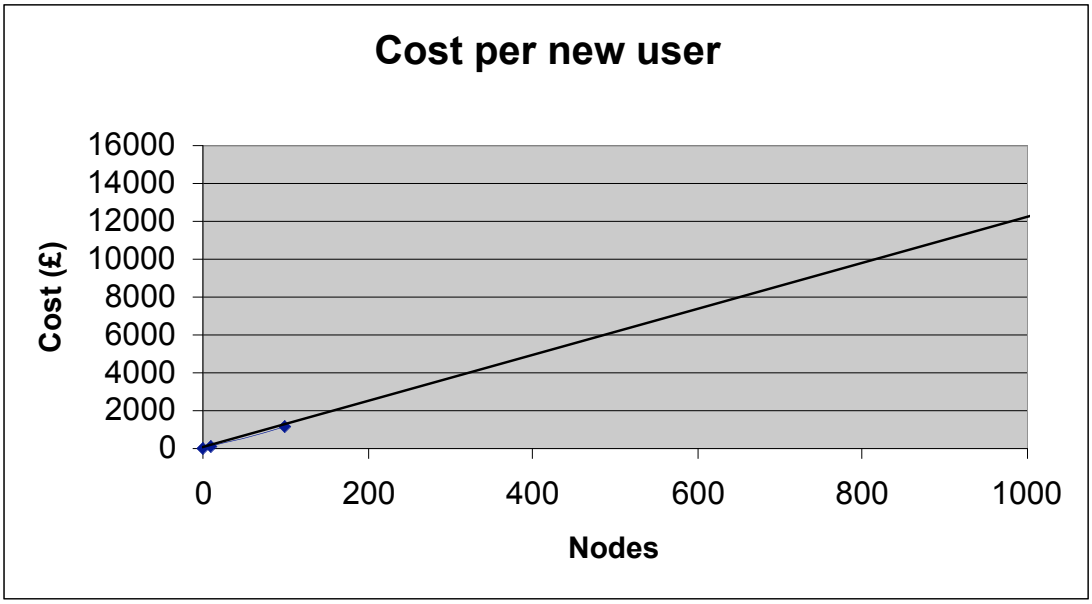
From the client perspective this process is also likely to become increasingly un-scalable in the existing form. Each client has to request and obtain a certificate from the CA, and then request an account on each grid node. This typically involves filling

[18] And also assuming that each new node is starting from scratch and must set up their own users – it doesn't make a large difference to the calculations either way.

[19] Although the more general problem is discovering the services and methods they are authorised to use.

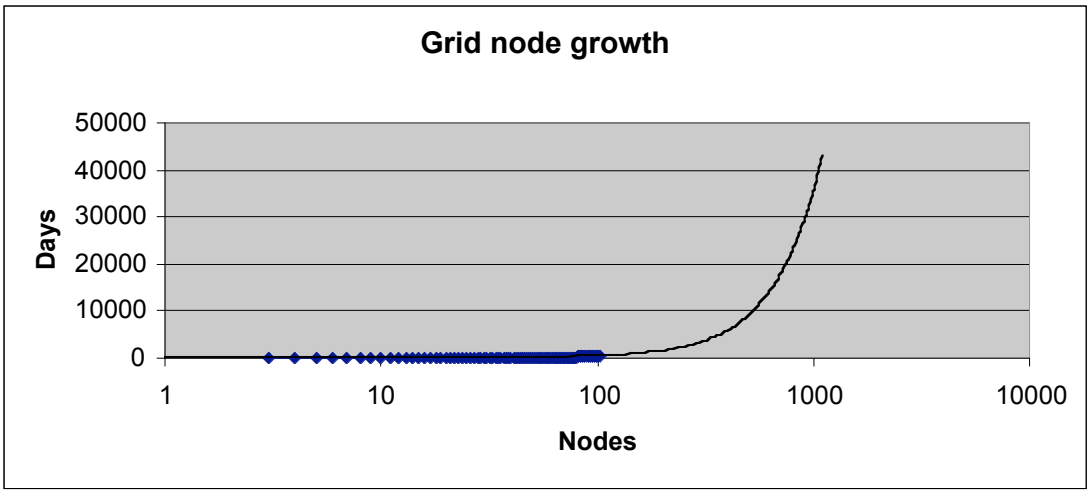
in and signing paper work (possibly also obtaining signatures of supervisors etc) and sending it to the system administrators of each node (by fax or letter). The clients also have to send their certificate subject information to the administrators of each site, and possibly the IP addresses of their machines. Thus, each new user must send this information to all of the current number of nodes, and then subsequently to each new node that joins the grid.

The above calculations do not take into account the cost for the *users*. Assuming it takes 10 minutes for a user to send certificates and supporting information to each node, the *total cost* (client and host nodes) is just double the above (host only) amounts. I.e. The total cost for 10 nodes, per new user, is £114; for 100 nodes the total cost per new user is £1194; and for 1000 nodes the total cost per new user is £11,940. The graph below shows the total cost, per new user, for increasing numbers of grid nodes.



There is also the added complexity that some nodes may be members of multiple Virtual Organisations, requiring users to be added/deleted based on their membership of these VOS. This means that gridmap files cannot simply be shared across nodes

Another question is the time these processes will take. The bottleneck will be the new nodes, as they have to add an increasing number of users in a fixed period of time. In fact, by the time the 20th node joins it takes a week (35 hours) to process all the existing users and add them to the new node. This implies a limit to the growth of the grid, at least with this simplistic model of security configuration and management. Assuming a 24x7 support environment the following graph shows the time required to grow the grid to a given number of nodes:



Of course, this is all entirely hypothetical, as realistically, this process will not be blindly followed for an increasing number of nodes/users. Also, some of the assumptions are bound to be wrong. For example, the number of new users per node may be lower in some cases, the number of users or nodes won't grow linearly, new nodes will be added concurrently (but the cost per new node is still high with increasing nodes), etc.

A.1.3 Local user accounts

One of the more problematic aspects is the requirement to map user certificates to unique local accounts in gridmap files. This is the default approach we took, but a number of alternatives are possible as follows.

Role based security is a typical approach to reducing the complexity of security for large scale enterprise systems. If users were mapped to roles in GT3.2, then the complexity of managing the gridmap files is reduced. However, authentication is still a problem, requiring user certificates, proxy certificates, and certificate subjects. One simplification is to allow anonymous users – people who share the same credentials - so that the nodes only have to know about the mapping between classes of anonymous users and roles. However, this allows the possibility for rogue users to misuse their roles without being able to be traced as individuals. Moreover, it is difficult to see how the access rights of individual users could be revoked.

Alternatively, discussions on the globus-discuss newsgroup suggested that a common account could be used in gridmap files (every user would still have a unique certificate). However, there would then be no privacy or isolation between users mapped to the same account, and the common account would function more as a "role. Nevertheless, this may be a reasonable compromise. The idea of "virtual accounts" is developed in [45].

Appendix 2 Supporting Tools

During the course of this project we became aware of some third-party tools that could supplement the basic GT3 toolkit, or provide inspiration for GT3 specific versions.

The IBM autonomic deployment framework is designed to automate deployment [48]. Manageability Services for Linux [49] is an example of using the low level OGSI services to build higher-level services, but is not a product yet. Java Certificate Services (JCS) is a tool that helps with managing grid certificates, and provides functionality for CSR (certificate request) creation, and X509 certificate creation (useful for GT3 user for host certificates) [50].

The Axis SOAPMonitor [51] has been mentioned previously. In theory it allows inspection of SOAP messages (but not protocol specific data without any additional development, special configurations or restarting of the application server. However, we experienced some difficulties installing and enabling this with GT3.2. Ideally it would be part of the default GT3.2 configuration.

The Tomcat servlet container [52] makes information available remotely (e.g. about the status of the server, and what services are deployed to it), can have user administered virtual servers, supports deployment (possibly even remote deployment) and can be administered remotely. There is anecdotal evidence that Tomcat provides excellent QoS in terms of reliability and performance/scalability. So far our test results have not been conclusive, so will be reported in the next report (2.0). However, based purely on the extra management functionality it provides, there is a good argument for making Tomcat the default Globus Toolkit container, and providing refined support and integration of it with Globus.

References

1. UK OGSA Evaluation Project web site, http://sse.cs.ucl.ac.uk/UK-OGSA/

2. From Zero to GT3, Ian Stokes-Rees, Oxford, http://www-pnp.physics.ox.ac.uk/~stokes/twiki/bin/view/DIRAC/GT3Express

3. GT3 Installation and Configuration Guide, Sheng JIANG, http://www.cs.ucl.ac.uk/staff/sjiang/webpage/GT3-install-guide.htm

4. GT3 Installation and Grid Service Development, http://testbed.gridcenter.or.kr/kor/technical_doc/KISTI-IBM-0522/ppt/day2-06-GT3%20installation&development.ppt

5. GT3 alpha installation (for Redhat), http://www.cs.binghamton.edu/~xling/gt3-alpha-installation.html

http://dps.uibk.ac.at/index.pl/quick_start_for_gt3

6. Installation Instructions for GT3.2 beta,
   http://esc.dl.ac.uk/Testbed/gt3.2-installation.pdf

7. List of online GT3 installation instructions,
   http://www.casa-sotomayor.net/gt3-tutorial/multiplehtml/ch02.html

8. Globus toolkit 3.2 for HP systems,
   http://h30097.www3.hp.com/globus/gt3/

9. Gridlab, GT3.2 pre-webservices installation,
   http://www.gridlab.org/WorkPackages/wp-5/admin/gt3.html

10. Installation of Globus 3.2, Peter Troger,
    http://www.dcl.hpi.uni-potsdam.de/research/grid/testbed/

11. Globus 3.2 installation documentation,
    http://www-unix.globus.org/toolkit/docs/3.2/installation/install_support.html

12. GT3 Installation on Windows,
    http://www.bigdogsoftware.org/

13. Information Grid Toolkit: Infrastructure of Shanghai
    Information Grid, Xinhua Lin, Qianni Deng, and Xinda Lu,
    http://www.springerlink.com/media/FL9XA5LVLLCXV2JHPQ0X/Contributions/X/Y/3/4/XY34RRULXTL6618E.pdf

14. Problems with GT3 and some possible solutions, Guy Rixon,
    http://wiki.astrogrid.org/bin/view/Astrogrid/GlobusToolkit3Problems

15. Jonathan Chin and Peter V. Coveney, Towards tractable toolkits for the Grid:
    a plea for lightweight, usable middleware,
    http://www.realitygrid.org/lgpaper21.htm

16. Globus Installation and Maintenance Issues,
    http://edg-wp2.web.cern.ch/edg-wp2/docs/globusinstall-0.2.html

17. OGSA/GT3 evaluation at Cern:

    Report,
    http://lcg.web.cern.ch/LCG/PEB/GTA/GTA_OGSA/presentations/acat03-ogsa.ppt

    Project,
    http://lcg.web.cern.ch/LCG/PEB/GTA/GTA-GT3testbed/GTA-GT3testbed.htm

    Performance results,
    http://agenda.cern.ch/askArchive.php?base=agenda&categ=a035448&id=a035448/transparencies

18. Guy Rixon, Grid Experience in the Virtual Observatory, *http://www.ivdgl.org/documents/document_server/uploaded_documents/doc--979--ppnp-workshop-ggf10-rixon.ppt*

19. APAC Compute Grid – Infrastructure Project, http://www.apac.edu.au/project_proposals/GRID/Grid%20IP%20Computing%20Infrastructure%20Project%20v4.pdf

20. UK OGSA Evaluation Project links, http://sse.cs.ucl.ac.uk/UK-OGSA/resources.html

21. Implementing a Distributed Master/Slave Grid Service with GT3, Gregor Mair, http://dps.uibk.ac.at/~gregor/mandel.pdf

22. OGSA Report Version 2.3, http://www.cpc.wmin.ac.uk/ogsitestbed/GEMLCA/Draft/OGSA_report_version_2.3.doc

23. Tomcat documents: http://jakarta.apache.org/tomcat/tomcat-4.1-doc/html-manager-howto.html

24. OGSA Use Cases, http://www.ggf.org/documents/GWD-I-E/GFD-I.029.pdf

25. The Open Grid Service Architecture, Version 1.0, http://www.gridforum.org/Public_Comment_Docs/Documents/draft-ggf-ogsa-specv1.pdf

26. OGSA Glossary of Terms, http://www.gridforum.org/Public_Comment_Docs/Documents/draft-ggf-ogsa-glossary-v6.pdf

27. The Master Managed Job Factory Service (MMJFS), http://www-106.ibm.com/developerworks/grid/library/gr-factory/

28. UK e-Science Certification Authority, http://www.grid-support.ac.uk/ca/ca.htm

29. PERMIS, http://sec.isi.salford.ac.uk/permis/

30. Bruce Beckles, Removing digital certificates from the end-user's experience of grid environments, UK eScience All Hands Meeting 2004, http://www.allhands.org.uk/proceedings/papers/250.pdf

31. Grid Security for Dummies, http://www.ogsadai.org.uk/docs/OtherDocs/SECURITY-FOR-DUMMIES.pdf

32. Critical Evaluation of Current Approaches to Grid Security, http://myweb.lsbu.ac.uk/~haidaran/Grid%20Security-Ali-N-Haidar.pdf

33. A Secure Client, http://www.casa-sotomayor.net/gt3-tutorial/multiplehtml/ch13s03.html

34. Brebner, P. (Ed.), Evaluating J2EE Application Servers, July 2002, CSIRO Publishing and Cutter Consortium, http://www.cmis.csiro.au/paul.brebner/pubs/MTE%20App%20Server%20Comparison%20v2.1.pdf

35. Borland Enterprise Server, and Deployment Op-Center, http://www.borland.com/bes/, http://www.borland.com/opcenter/index.html

36. JOnAS management console, http://jonas.objectweb.org/current/doc/Admin.html

37. Novell extend (formerly SilverStream eXtend), http://extend.novell.com/Website/app/extend/ProductsLanding

38. AmberPoint, http://www.amberpoint.com/solutions/exception.shtml

39. The Future of Grid and Web Services, http://www.lesc.ic.ac.uk/events/stevetueke.html

40. GT3.2 Installation guide, http://www-unix.globus.org/toolkit/docs/3.2/installation/install_installing.html#buninstall

41. Configuring GT3.2, http://www-unix.globus.org/toolkit/docs/3.2/installation/install_config.html

42. From OGSI to WS-Resource Framework: Refactoring and Evolution, http://www.globus.org/wsrf/specs/ogsi_to_wsrf_1.0.pdf

43. Core Developer's Guide, http://www-unix.globus.org/toolkit/docs/3.2/core/developer/message_security.html#Service

44. GT3.2 Tutorial, Access Control with Gridmaps, http://www.casa-sotomayor.net/gt3-tutorial/multiplehtml/ch15s01.html

45. The Dynamic Sessions Project, http://www-unix.mcs.anl.gov/~keahey/DS/DynamicSessions.htm

46. Virtual Organisations Membership Service (VOMS), http://hep-project-grid-scg.web.cern.ch/hep-project-grid-scg/voms.html

47. Community Authorization Service (CAS), http://www.globus.org/security/CAS/GT3/

48. IBM autonomic deployment framework, http://www-106.ibm.com/developerworks/autonomic/library/ac-deploy/

49. Manageability Services for Linux, http://www-106.ibm.com/developerworks/library/gr-manageservices/

50. Mange X.509 certificates in your grid with Java Certificate Services, http://www-106.ibm.com/developerworks/grid/library/gr-jsc/?ca=dgr-lnxw06ManageX.509

51. Axis SOAPMonitor, http://ws.apache.org/axis/java/soapmonitor-user-guide.html

52. Tomcat, http://jakarta.apache.org/tomcat

53. An Interview with Argonne's Steve Tuecke, February 2003, http://www-106.ibm.com/developerworks/java/library/j-tuecke.html?dwzone=java

54. An early evaluation of WSRF and WS-Notification via WSRF.NET, http://www.cs.virginia.edu/~humphrey/papers/EarlyEvalWSRF.pdf

55. WS-GAF, http://www.neresc.ac.uk/ws-gaf/index.html