AN INTRODUCTION TO THE GLOBUS TOOLKIT

Giovanni Aloisio
University of Lecce, Lecce, Italy

Massimo Cafaro
University of Lecce, Lecce, Italy

Abstract
The Globus toolkit is a grid middleware being developed at the Institute of Sciences Information of the University of Southern California and at Argonne National Laboratories. In this paper we briefly introduce the core services and the functionalities provided; in particular we address the Globus Security Infrastructure, the resource management and the Grid Information Service.

1. INTRODUCTION

In the past, high performance applications ran on a single supercomputer; advances in both networking technologies and metacomputing environments are going to change this, enabling new classes of applications that can benefit from metacomputing.

The physical infrastructure is now widely available in the United States and is becoming widespread in Europe, although some efforts were undergone to improve the networks available. Using such high speed networks it is feasible to link together supercomputers, data archives, advanced visualization devices such as a Cave, and scientific instruments. But to really exploit these new interesting possibilities the heterogeneous and dynamic nature of the metacomputing environment must be taken into account.

Globus is a hierarchical, layered set of high level services built using a low level toolkit. Its purpose is to provide the middleware needed for grid and metacomputing applications, providing the foundations for building computational grids [1] linking together thousands of resources geographically distributed.

2. THE GLOBUS TOOLKIT

The term metacomputing [2] refers to computation on a virtual supercomputer assembled connecting together different resources like parallel supercomputers, data archives, storage systems, advanced visualization devices and scientific instruments using high speed networks that link together these geographically distributed resources.

We may want to do so because it enables new classes of applications [3][4] previously impossible and because it is a cost effective approach to high performance computing; we can classify the new applications as follows:

- desktop supercomputing;
- smart instruments;
- collaborative environments;
- distributed supercomputing.

Desktop supercomputing includes applications that couple high end graphics capabilities with remote supercomputers and/or databases; smart instruments are scientific instruments like microscopes, telescopes and satellites that require supercomputing power to process the data produced in near real time.

In the class of collaborative environments we find applications in which users at different locations can interact together working on a supercomputer simulation; distributed supercomputing finally is the class of application that require multiple supercomputers to solve problems otherwise too large or whose execution is divided on different components that can benefit from execution on different architectures.

The challenges to be faced before grid computing and metacomputing can be really exploited however include the following issues:

- scaling and selection;
- heterogeneity;
- unpredictable structure;
- dynamic behavior;
- multiple administrative domains.

Scaling is a concern, because we expect that metacomputing environments in the future will become larger, and resources will be selected and acquired on the basis of criteria like connectivity, cost, security and reliability.

Such resources will show different levels of heterogeneity, ranging from physical devices to system software and schedulers policies; moreover traditional high performance applications are developed for a single supercomputer whose features are known a priori, e.g. the latency; in contrast metacomputing applications will run in a wide range of environments thus making impossible to predict the structure of the computation.

Another concern is related to the dynamic behavior of the computation [5], since we cannot be assured that all of the system characteristics stay the same during the course of computation, e.g. the network bandwidth and latency can widely changes, and there is the possibility of both network and resources failure.

Finally, since the computation will usually span resources at multiple administrative domains, there is not a single authority in charge of the system, so that different scheduling policies and authorization mechanisms must be taken into account.

Globus is designed to tackle this issues, and its usefulness in the context of metacomputing was demonstrated in the I-WAY experiment [6][7] and in the Gusto testbed; among the distributed simulations that have been run using Globus there is SF-Express [8], the largest computer simulation of a military battle involving more that 100,000 entities.

3. GLOBUS SERVICES

Globus provides different services in order to enable the metacomputing environment:

- resource management;

- communication;
- information;
- security;
- health and status;
- remote data access;
- executables management.

Resource management is the aim of the GRAM (Globus Resource Allocation Manager) module [9][10]; it takes care of resource location and allocation; moreover it performs process management. The user can specify its application requirements using RSL, the resource specification language.

Resource location is needed, since in a dynamic environment applications cannot know in advance the exact location of the required resources. A broker is in charge of locating and acquiring (allocation) the resources by scheduling them and performing the required initialization.

The NEXUS library of communications [11][12][13] provide unicast and multicast communication services; moreover, different quality of service parameters [14] are taken into account, e.g. jitter, latency, bandwidth. The library can be exploited for the implementation of message passing [15], rpc, distributed shared memory, etc; it uses mechanism for negotiating the best communication method among parties and provide asynchronous remote service requests (RSR).

Using a RSR, a handler on a remote machine can be invoked supplying a typed buffer of data in input. The advantage over traditional rpc mechanism is that the remote applications can be multithreaded thus avoiding blocking.

The Metacomputing Directory Service (MDS) [16] provides a uniform resource information service that allow distributed access to structure and data information about the metasystem status. The module allow both posting and receiving information, using the LDAP protocol (Lightweight Directory Access Protocol).

The Globus Security Infrastructure (GSI) module [17][18][19] is responsible for handling the authentication mechanism used to validate the identity of the users and resources; it makes use of both the Generic Security Service api (GSS) and of SSL (Secure Socket layer). Certificates are used as the principal authorization mechanism.

Fault detection and monitoring of health and status of the metasystem components is possible by using the tools provided by the Heart beat Monitor (HBM) module [20]. Using the HBM a process can be monitored and periodic heartbeats can be sent to one or more monitors.

Remote access to data via sequential and parallel interfaces can be obtained exploiting the Global Access to Secondary Storage (GASS) module [21]; supported operations include remote read, write and append. The Remote I/O (RIO) library [22] implements the MPI I/O interface.

The Globus Executable Management (GEM) is responsible for performing the necessary initialization and setup for the same executable at different sites; it is used to migrate the executables and eventually to compile the source code on the target machine if this step is needed.

4. THE GLOBUS SECURITY INFRASTRUCTURE

Grid aware applications must be designed bearing in mind since the beginning one of the most important issues, security. Therefore the Globus team developed an authentication system known as gssapi, the Generic Security Service api using a public domain implementation of SSL. This system

utilizes the RSA algorithm, thus employing both public and private keys. Authentication relies on X.509v3 certificates provided by each user in their directories, that identify them to grid resources.

5. SECURE SYSTEMS

Secure systems make it hard for people to do things they are not supposed to do; they are designed so that the cost of breaking any component of the system outweighs the rewards. The cost of breaking systems is measured in money, time and risk, both legal and personal. The benefits of breaking systems are control, money or information that can be sold for money. Therefore, the security of a system should be proportional to the resources it protects.

The term secure systems can be misleading; as a matter of fact it does not implies that systems are either secure or insecure, but it is used to denote systems that were designed with security requirements. It is worth noting here that every system can be broken, given enough time and money.

A Computational Grid consists of a set of valuable resources like parallel supercomputers, workstations, databases and smart instruments, so in order to prevent unauthorized accesses to the Grid we need a strong authentication mechanism. To show how Globus handle authentication, we begin reviewing some of the notions belonging to the theory of Cryptography.

6. CRYPTOGRAPHY

Cryptography [23][24] is the science of *secret writing*; it is a branch of mathematics called *cryptology*. Strictly related to cryptography is *cryptanalysis*, the science of breaking (analyzing) cryptography. Cryptography is the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication, and data origin authentication. Cryptography is not the only means of providing information security, but rather one set of techniques.

Confidentiality is a service used to keep the content of information from all but those authorized to have it. *Secrecy* is a term synonymous with confidentiality and privacy. There are numerous approaches to providing confidentiality, ranging from physical protection to mathematical algorithms which render data unintelligible.

Data integrity is a service which addresses the unauthorized alteration of data. To assure data integrity, one must have the ability to detect data manipulation by unauthorized parties. Data manipulation includes such things as insertion, deletion, and substitution.

Authentication is a service related to identification. This function applies to both entities and information itself. Two parties entering into a communication should identify each other. Information delivered over a channel should be authenticated as to origin, date of origin, data content, time sent, etc. For these reasons this aspect of cryptography is usually subdivided into two major classes: *entity authentication* and *data origin authentication*. Data origin authentication implicitly provides data integrity (if a message is modified, the source has changed).

Non-repudiation is a service which prevents an entity from denying previous commitments or actions. When disputes arise due to an entity denying that certain actions were taken, a means to resolve the situation is necessary. For example, one entity may authorize the purchase of property by another entity and later deny such authorization was granted. A procedure involving a trusted third party is needed to resolve the dispute.

A common way to protect information is for the sender to *encrypt*, or scramble, information before sending it. The receiver *decrypts*, or unscrambles, the information after receiving it. While in transit, the encrypted information is unintelligible to an intruder. *Encryption* applies a mathematical

function to transform the information, called *plaintext*, in order to make it unintelligible to anyone but the intended recipient. *Decryption* is the opposite process of transforming encrypted information, called *ciphertext*, to recover the original information.

The mathematical function used to encrypt or decrypt information is called a *cryptographic algorithm* or *cipher***.** It is worth noting that, since ciphers algorithms are widely known, confidentiality is based on a number called a *key* to be used with the algorithm to encrypt or decrypt previously encrypted information.

A *symmetric cipher* (Fig. 1) uses the same key at the sending and receiving end; these ciphers are also called *private key* or *secret key* ciphers.

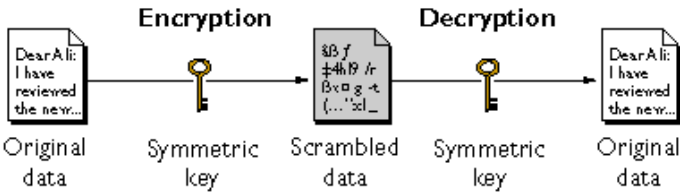

Fig. 1 Symmetric cipher

Symmetric algorithms can be divided into *stream ciphers* and *block ciphers*. Stream ciphers can encrypt a single bit of plaintext at a time, whereas block ciphers take a number of bits (typically 64 bits in modern ciphers), and encrypt them as a single unit. Implementations of symmetric-key encryption can be made highly efficient: the time delay to encrypt or decrypt information is not significant. Moreover, the use of symmetric-key encryption provides a degree of authentication, since information encrypted with one symmetric key cannot be decrypted with any other symmetric key.

Nevertheless, symmetric-key encryption can be effective only if the symmetric key is kept secret by the parties involved. Supposing that anyone else discovers the key, this in turn affects not only confidentiality but also authentication, since people with an unauthorized symmetric key is able to decrypt messages sent with that key and encrypt new messages and send them. This messages appear as if they originate from one of the parties who were originally using the key.

Asymmetric ciphers (Fig. 2), also called *public key* ciphers, make use of a pair of keys - a *public key* and a *private key* - associated with each entity that needs to authenticate itself or to sign or encrypt data. Each public key can be safely published, while the corresponding private key must be kept secret. The mathematical functions associated to public key ciphers are such that data encrypted with a public key can be decrypted only with the corresponding private key.

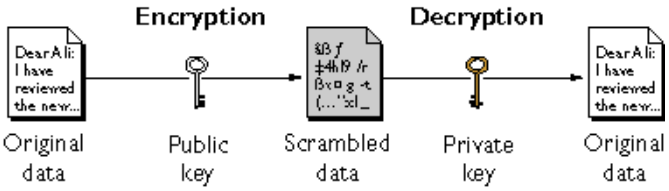

Fig. 2 Asymmetric cipher

To send encrypted data to someone, the information is encrypted using that person's public key. The person who receives the encrypted data decrypts it using the corresponding private key. With asymmetric ciphers there is no need to worry about other people keeping a secret, as in case of symmetric ciphers; asymmetric ciphers can also be used to generate a *digital signature:* data encrypted with your private key can be decrypted only with your public key. Anyway, asymmetric ciphers are computationally expensive.

Before showing the RSA algorithm, we need the following

DEFINITION For $n > 1$, let $\phi(n)$ denote the number of integers in the interval $[1,n]$ which are relatively prime to n. The function ϕ is called the Euler phi function.

FACT (properties of Euler phi function)

(1) If p is a prime, $\phi(p) = p - 1$

(2) If $gcd(m,n) = 1$, $\phi(mn) = \phi(m)\,\phi(n)$

ALGORITHM: Key generation for RSA public key encryption

Each entity creates an RSA public key and a corresponding private key as follows:

1) Generate two large random and distinct primes p and q, each roughly the same size.
2) Compute $n = pq$ and $\phi(n) = (p - 1)(q - 1)$
3) Select a random integer e, $1 < e < \phi$, such that $gcd(e, \phi) = 1$
4) Use the extended Euclidean algorithm to compute the unique integer d, $1 < d < \phi$, such that $ed = 1 \pmod{\phi}$
5) The public key is (n,e), the private key is d.

DEFINITION The integers e and d in the RSA key generation are called the *encryption exponent* and the *decryption exponent* respectively, while n is called the *modulus.*

ALGORITHM: RSA public key encryption

Bob encrypts a message m for Alice, which Alice decrypts.

1) Bob obtains Alice's authentic public key (n,e).
2) Represent the message m as an integer in the interval $[0, n - 1]$
3) Compute $c = m^e \bmod n$

4) Send the ciphertext *c* to Alice

Alice recover plaintext *m* from *c*

1) Use the private key *d* to recover $m = c^d \bmod n$

The strength of encryption is closely related to the difficulty of discovering the key; this in turn depends on both the cipher used and the length of the key. We can think of encryption strength in terms of the size of the keys: in general, longer keys allow stronger encryption of data. Key length is measured in number of bits.

We can now describe the authentication mechanism exploited in the Globus Toolkit, i.e., digital certificates.

7. DIGITAL CERTIFICATES

A *certificate* is an electronic document whose aim is to identify an entity and to associate that identity with a public key. Certificates are similar to a driver's license or a passport, common IDs that provide a generally recognized proof of a person's identity. People who want to get a driver's license apply to a government agency, in this case the Department of Motor Vehicles, which in turn verifies their identity, ability to drive, address, and other information before issuing the license.

People get a certificate in much the same way, applying to a *Certificate authority* (*CA*), an entity that validate identities and issue certificates. CAs can be independent third parties or organizations running their own certificate-issuing server software.

When a CA issues a certificate, a particular public key is *binded* to the name of the entity the certificate identifies. We can say that, for all practical purposes, a certificate is just a public key signed by the issuing CA. This means that the public key certified will work with the related private key owned by the entity identified by the certificate. Since we trust the CA, we trust the identity certified by the CA.

A certificate *always* includes other information like the name of the entity it identifies, the date in which the certificate is issued, an expiration date, the name of the issuer of the certificate, a serial number, etc. Moreover, a certificate must always include the digital signature of the issuing CA.

Certificates are used both for *client authentication*, i.e. the confident identification of a client by a server, and for *server authentication*, i.e. the confident identification of a server by a client. The process of authenticating a user to a server works as follows. The client digitally signs a randomly generated piece of data using the user's private key and sends both the certificate and the signed data to the server. The server authenticates the user's identity using the user's public key found in the certificate.

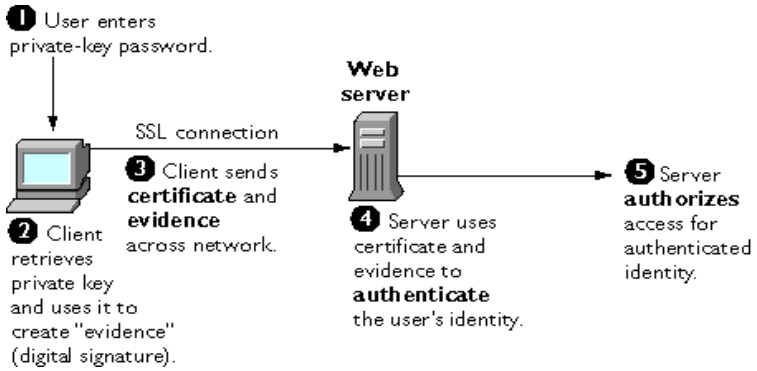

Fig. 3 Certificate based client authentication

Certificate-based authentication is to be preferred to password-based authentication because it is based on what the user has (the private key) as well as what the user knows (the password that protects the private key). The client asks for the password that protects the user's private key the first time the client needs to access it during a given session. After entering this password once, the user doesn't need to enter it again for the rest of the session, even when accessing other SSL-enabled servers.

In the SSL protocol the client uses the private key to sign some data that has been randomly generated on the basis of input from both the client and the server. This data and the digital signature can be used to prove the private key's validity. As a matter of fact, the digital signature can be created only with that private key and can be validated using the corresponding public key against the signed data, which is unique to the SSL session.

7.1 Globus certificates

There exist a number of different certificates to identify several entities. CA certificates are used to identify Cas; both client and server software exploit CA certificates to determine what other certificates can be trusted. Client SSL certificates are used to identify clients to servers using the SSL protocol and server SSL certificates are used to identify servers to clients. Server authentication in SSL may be used with or without client authentication, but server authentication is a requirement for an encrypted SSL session. Client certificates in Globus are compliant to the X.509 v3 certificate specification; an X.509 v3 certificate binds a *distinguished name* (DN) to a public key. A DN is composed of name-value pairs, such as uid = doe, that uniquely identify the certificate *subject*.

The following is an example of DN:

Subject: C=US, O=Globus, O=University of Lecce, OU=HPC Lab, CN=Massimo Cafaro

uid=cafaro,e=massimo.cafaro@unile.it,

cn=Massimo Cafaro,o=University of Lecce, c=IT

The abbreviations shown have these meanings:

- uid: user ID
- e: email address
- cn: the user's common name
- o: organization
- ou: organization unit
- c: country

DNs may also include other name-value pairs.

Every X.509 certificate consists of two sections:

- The data section
- The signature section

7.1.1 The data section

The data section must contains:

- A number that references the version of the X.509 standard supported by the certificate
- The certificate's serial number. Every certificate issued by a CA is required to have a serial number that must be unique among the certificates issued by that CA
- Information concerning the user's public key, including the cipher and a representation of the key itself
- The DN of the CA that issued the certificate
- A starting date and an expiration date that determine the time frame during which the certificate is considered valid
- The DN of the certificate subject, also called the *subject name*
- Optional *certificate extensions*

7.1.2 The signature section

The signature section must contains:

- The cipher used by the CA to digitally sign the certificate
- The CA's digital signature

Here it is an example Globus certificate:

issuer :/C=US/O=Globus/CN=Globus Certification Authority

subject:/C=US/O=Globus/O=University of Lecce/OU=HPC Lab/CN=Massimo Cafaro

serial :052B

Certificate:

Data:

Version: 3 (0x2)

Serial Number: 1323 (0x52b)

Signature Algorithm: md5WithRSAEncryption

Issuer: C=US, O=Globus, CN=Globus Certification Authority

Validity

Not Before: Sep 27 13:59:59 1999 GMT

Not After : Sep 26 13:59:59 2000 GMT

Subject: C=US, O=Globus, O=University of Lecce, OU=HPC Lab, CN=Massimo Cafaro

Subject Public Key Info:

Public Key Algorithm: rsaEncryption

RSA Public Key: (1024 bit)

Modulus (1024 bit):

00:c0:e3:40:15:f2:2e:e7:43:9e:62:e2:5d:aa:ce:
3d:52:9c:dd:7d:6b:d7:59:c0:fe:14:26:3b:83:51:
19:d7:1a:ed:50:ed:e0:9c:52:dd:bb:4b:88:c0:91:
2a:4e:b2:21:e4:be:a3:35:69:59:84:73:c9:8c:cc:
f4:8b:45:4c:79:ef:b5:93:01:3d:ba:15:8d:af:3b:
84:62:56:b8:4f:dd:b9:42:10:93:85:bd:01:e0:85:
d0:09:9a:fb:0c:eb:a0:fe:0d:6f:3e:5f:42:9d:3d:
80:52:18:af:a0:64:2d:b1:76:a4:e9:97:ea:64:ef:
fc:5e:3e:d1:18:11:7d:3e:91

Exponent: 65537 (0x10001)

X509v3 extensions:

Netscape Cert Type:

0x40

Signature Algorithm: md5WithRSAEncryption

4b:22:4e:23:87:cc:30:0b:cf:78:c6:62:99:34:e0:f8:19:7f:
2a:a0:2e:3d:86:82:60:8c:6e:b1:e7:7d:8c:1c:b0:d2:9a:69:
41:27:2b:69:95:9b:7f:07:d9:06:b2:7b:1d:bd:3e:f7:24:35:
c8:60:4b:40:34:03:e5:97:34:0e:5c:82:73:23:f6:ab:43:c2:
34:cd:95:14:99:02:e8:73:63:e2:0f:3a:a1:77:51:f2:de:74:
9b:ff:1d:84:68:46:23:94:a3:c4:ac:c4:a6:7b:ac:bb:13:23:
09:0e:a7:01:b0:17:8b:d1:01:10:fb:6a:2f:32:da:8a:e4:fe:
da:65

-----BEGIN CERTIFICATE-----

MIICOjCCAaOgAwIBAgICBSswDQYJKoZIhvcNAQEEBQAwRzELMAkGA1UEBhMCVV
MxDzANBgNVBAoTBkdsb2J1czEnMCUGA1UEAxMeR2xvYnVzIENlcnRpZmljYXRpb24gQXV0
aG9yaXR5MB4XDTk5MDkyNzEzNTk1OVoXDTAwMDkyNjEzNTk1OVowZzELMAkGA1UEBh
MCVVMxDzANBgNVBAoTBkdsb2J1czEcMBoGA1UEChMTVW5pdmVyc2l0eSBvZiBMZWNjZ
TEQMA4GA1UECxMHSFBDIExhYjEXMBUGA1UEAxMOTWFzc2ltbyBDYWZhcm8wgZ8wDQ
YJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMDjQBXyLudDnmLiXarOPVKc3X1r11nA/hQmO4
NRGdca7VDt4JxS3btLiMCRKk6yIeS+ozVpWYRzyYzM9ItFTHnvtZMBPboVja87hGJWuE/duUIQ
k4W9AeCF0Ama+wzroP4Nbz5fQp09gFIYr6BkLbF2pOmX6mTv/F4+0RgRfT6RAgMBAAGjFTAT
MBEGCWCGSAGG+EIBAQQEAwIAQDANBgkqhkiG9w0BAQQFAAOBgQBLIk4jh8wwC894xm
KZNOD4GX8qoC49hoJgjG6x532MHLDSmmlBJytplZt/B9kGsnsdvT73JDXIYEtANAPllzQOXIJzI/
arQ8I0zZUUmQLoc2PiDzqhd1Hy3nSb/x2EaEYjlKPErMSme6y7EyMJDqcBsBeL0QEQ+2ovMtqK5
P7aZQ==

-----END CERTIFICATE-----

8. RESOURCE MANAGEMENT

Globus provides a flexible Resource Specification Language (RSL) that allows users to express their application's constraints. The resource management architecture (Fig. 4) allow easy allocation and co-allocation of computational resources. The Globus Resource Allocation Manager (GRAM) (Fig. 5) provides such services despite the heterogeneity of the resources managed.

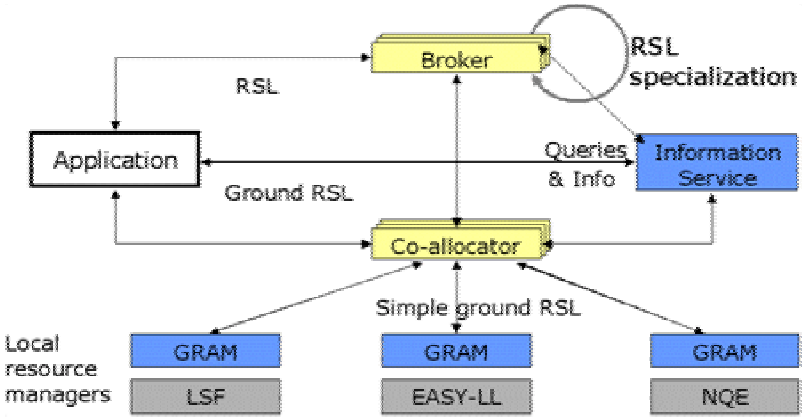

Fig. 4 Resource Management Architecture

RSL syntax closely resembles LDAP filters syntax; using RSL users can easily inquire about resource features like machine type, memory, cpu, clock speed and number of processors available in case of parallel machines. RSL also allow to specify for a job the directory to be used, where the executable can be found, the command line arguments and environment variables that may be needed.

The GRAM allows to run jobs remotely, and provides an api for submitting, monitoring, and terminating jobs. When the user submits a job, a request is generated and forwarded to a daemon called the gatekeeper running on the remote computer. The gatekeeper manages the request creating a job manager responsible for the job. The job manager is in charge of starting and monitoring the remote application, it also communicates job state changes to the user and terminates when its job has terminated execution either normally or by failing.

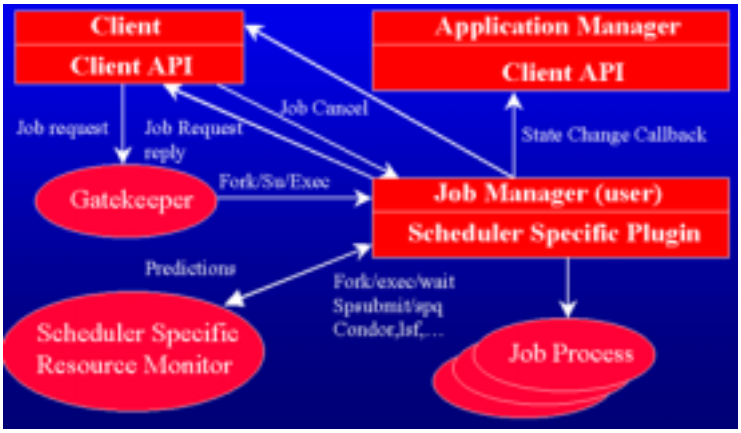

Fig. 5 GRAM

The GRAM supports a multiple queues scheduling model, in which users or resource brokers submit job requests that are registered as pending jobs. A job during its lifetime undergoes state changes according to the state diagram in Fig. 6.

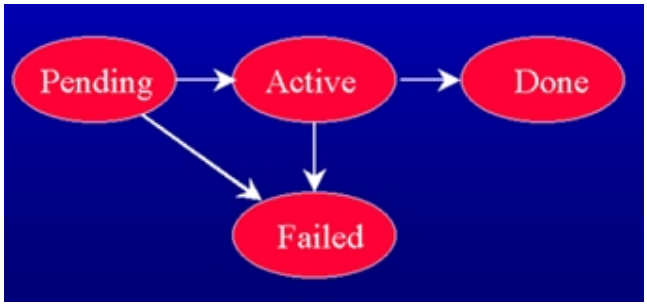

Fig. 6 State transitions of a Globus Job

Globus provides the user with the possibility of submitting a multirequest, i.e., a specification of a job whose execution requires the simultaneous allocation of multiple resources. Co-allocation of resources is managed by the DUROC (Dynamically Updated Request Online Co-allocator) as shown in Fig. 7 and is an atomic operation with a simple semantics. Either all of the requested resources are available and Globus co-allocate them, or none of them get scheduled. Pending request can be edited dynamically by adding new nodes or removing failed ones, and are delayed to last possible minute, using a barrier synchronization mechanism.

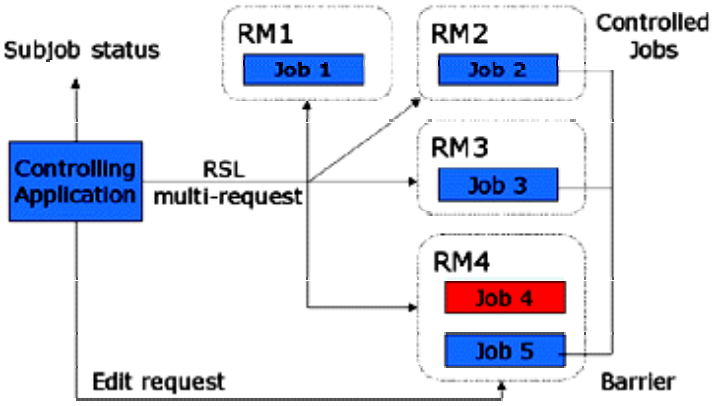

Fig. 7 DUROC Architecture

9. THE GRID INFORMATION SERVICE

The Globus toolkit provides the users with a directory service called Grid Information Service (formerly known as Metacomputing Directory Service or MDS). Aim of this service is to allow grid applications to easily locate and determine features of resources. The GIS can be queried to find where resources with a required architecture, installed software, available capacity, network bandwidth etc. are, or given a specific resource, to find its distinguishing features.

Basically, the users can query a centralized server (this is true for Globus v1.1.2 as of this writing, but the upcoming Globus release v1.1.3 is going to change this, implementing the GIS as a true distributed service, much like the DNS) using the LDAP (Lightweight Directory Access Protocol) api or the globus command *grid-info-search*.

The GIS can be populated with a variety of useful information. This information is published both as a *white pages directory*, in which information is related to a particular *distinguished name*, and as a *yellow pages directory* in which information is categorized using a number of *object classes* defined by the Globus team. These classes are currently outdated, and a new set of schemas is being developed jointly with the US Grid Forum.

The GIS provides a flexible information infrastructure critical to grid applications; such applications should harness the information published in the GIS to determine the available resources and the state of the computational grid. The information could also be exploited to optimize the applications, by tuning them at runtime if a reconfiguration procedure is needed on the basis of both static and dynamic contents.

Information accessible via the GIS include, but is not limited to:

- computing resources features like IP address, installed software, system administrator, networks connected to, OS version, current load;
- network features like bandwidth and latency, protocols and logical topology;
- Globus infrastructure features like hosts and resource managers.

The uniform information infrastructure provided by the GIS is scalable and permits efficient access to dynamic data, moreover it is based on a standard (LDAP).

10. CONCLUSIONS

We have described the core services composing the Globus toolkit, which is becoming the de facto middleware standard to build grid enabled applications. In particular, we addressed the Globus Security Infrastructure explaining related concepts along the way, the resource management architecture and the Grid Information Service.

A complete description of the Globus toolkit is beyond the scope of this paper, nevertheless interested readers will find more information at http://www.globus.org.

REFERENCES

[1] Ian Foster, Carl Kesselman, “The Grid. Blueprint for a new computing infrastructure”, Morgan Kaufmann, San Francisco 1999

[2] C. Catlett, L. Smarr, “Metacomputing”, Communications of the ACM, 35 (1992), pp. 44-52

[3] W. Benger, I. Foster, J. Novotny, E. Seidel, J. Shalf, W. Smith, P. Walker, “Numerical Relativity in a Distributed Environment”, Ninth SIAM Conference on Parallel Processing for Scientific Computing, Apr. 1999

[4] Gregor von Laszewski, Mei-Hui Su, Joseph A. Insley, Ian Foster, John Bresnahan, Carl Kesselman, Marcus Thiebaux, Mark L. Rivers, Steve Wang, Brian Tieman, Ian McNulty, “Real-Time Analysis, Visualization, and Steering of Microtomography Experiments at Photon Sources”, Ninth SIAM Conference on Parallel Processing for Scientific Computing, Apr. 1999

[5] W. Smith, I. Foster, V. Taylor, “Predicting Application Run Times Using Historical Information”, Proc. IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing, 1998.

[6] I. Foster, J. Geisler, W. Nickless, W. Smith, S. Tuecke, “Software Infrastructure for the I-WAY High Performance Distributed Computing Experiment”, Proc. 5th IEEE Symposium on High Performance Distributed Computing. pg. 562-571, 1997

[7] T. DeFanti, I. Foster, M. Papka, R. Stevens, T.Kuhfuss, “Overview of the I-WAY: Wide Area Visual Supercomputing”, International Journal of Supercomputer Applications, 10(2), pp.123-130, 1996

[8] S. Brunett, D. Davis, T. Gottshalk, P. Messina, C. Kesselman, “Implementing Distributed Synthetic Forces Simulations in Metacomputing Environments”, Proceedings of the Heterogeneous Computing Workshop, Mar. 1998

[9] I. Foster, C. Kesselman, C. Lee, R. Lindell, K. Nahrstedt, A. Roy, “A Distributed Resource Management Architecture that Supports Advance Reservations and Co-Allocation”, Intl Workshop on Quality of Service, 1999

[10] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, N. Karonis, S. Martin, W. Smith, S. Tuecke, “A Resource Management Architecture for Metacomputing Systems”, Proc. IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing,1998

[11] I. Foster, J. Geisler, C. Kesselman, S. Tuecke, “Managing Multiple Communication Methods in High-Performance Networked Computing Systems”, Journal of Parallel and Distributed Computing, 40, pp.35-48, 1997

[12] I. Foster, C. Kesselman, S. Tuecke, “The Nexus Approach to Integrating Multithreading and Communication”, Journal of Parallel and Distributed Computing, 37, pp.70-82, 1996

[13] I.Foster, C. Kesselman, S. Tuecke, “The Nexus Task-Parallel Runtime System”, Proc. 1st Int'l Workshop on Parallel Processing, pp. 457-462, 1994

[14] T. C. Lee, C. Kesselman, J. Stepanek, R. Lindell, S. Hwang, B. Scott Michel, J. Bannister, I. Foster, A. Roy, “The Quality of Service Component for the Globus Metacomputing System” , Proc. IWQoS '98, pp. 140-142, 1998.

[15] I. Foster, N. Karonis, “A Grid-Enabled MPI: Message Passing in Heterogeneous Distributed Computing Systems”, Proc. SuperComputing 98

[16] S. Fitzgerald, I. Foster, C. Kesselman, G. von Laszewski, W. Smith, S. Tuecke, “A Directory Service for Configuring High-Performance Distributed Computations”, Proc. 6th IEEE Symp. on High-Performance Distributed Computing, pp. 365-375, 1997

[17] I. Foster, C. Kesselman, G. Tsudik, S. Tuecke, “A Security Architecture for Computational Grids”, Proc. 5th ACM Conference on Computer and Communications Security Conference, pp. 83-92, 1998.

[18] I. Foster, N. T. Karonis, C. Kesselman, S. Tuecke, “Managing Security in High-Performance Distributed Computing”, Cluster Computing 1(1) pp. 95-107, 1998.

[19] I. Foster, N. Karonis, C. Kesselman, G. Koenig, S. Tuecke, “A Secure Communications Infrastructure for High-Performance Distributed Computing”, 6th IEEE Symp. on High-Performance Distributed Computing, pp. 125-136, 1997

[20] P. Stelling, I. Foster, C. Kesselman, C.Lee, G. von Laszewski, “A Fault Detection Service for Wide Area Distributed Computations”, Proc. 7th IEEE Symp. on High Performance Distributed Computing, to appear, 1998.

[21] J. Bester, I. Foster, C. Kesselman, J. Tedesco, S. Tuecke, “GASS: A Data Movement and Access Service for Wide Area Computing Systems”, Sixth Workshop on I/ O in Parallel and Distributed Systems, May 5, 1999

[22] I. Foster, D. Kohr, R. Krishnaiyer, J. Mogill, “Remote I/O: Fast Access to Distant Storage”, Proc. Workshop on I/O in Parallel and Distributed Systems (IOPADS), pp. 14-25, 1997

[23] Bruce Schneier, Applied Cryptography, 2nd edition. John Wiley & Sons, 1995.

[24] Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone, Handbook of Applied Cryptography**,** CRC Press, 1996