XtremWeb-HEP Version 7.6.0

Introduction

Version 1.2.0

Table of Contents

Table of Figures

1 Introduction

This document is part of the XtremWeb-HEP middleware documentation which contains different documents: , “XtremWeb-HEP Administrator Guide”, “XtremWeb-HEP Resource Provider Guide”, “XtremWeb-HEP Programmer Guide”, “XtremWeb-HEP User Guide”, and this one: “XtremWeb-HEP Introduction”.

The name “XtremWeb-HEP” is shortened as “XWHEP”.

Please note that the documentation versioning is different from the middleware one.

On October 2011 :

- “XtremWeb-HEP Introduction 1.2.0” has been released;
- “XtremWeb-HEP Programmer Guide 1.1.0” has been released;
- “XtremWeb-HEP Administrator Guide 1.1.0” has been released;
- “XtremWeb-HEP User Guide 1.3.0” has been released;
- “XtremWeb-HEP Resource Provider Guide 1.0.0” has been released.

1.1 Audience

Anyone who would like to contribute, use, install or administrate the XtremWeb-HEP middleware should read this document.

This document specifically introduces the XtremWeb-HEP middleware, its services and protocols.

1.2 Software license agreement

XtremWeb-HEP (XWHEP) is an open source software provided under the GPL2 license (http://www.gnu.org/licenses/gpl-2.0.html).

Because the program is licensed free of charge, there is no warranty for the program, to the extent permitted by applicable law. Except when otherwise stated in writing the copyright holders and/or other parties provide the program "as is" without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality and performance of the program is with you. Should the program prove defective, you assume the cost of all necessary servicing, repair or correction.

1.3 Reading conventions

Code examples appear in grey boxes

```
$> ./proxy-init.sh desktopgrid.vo.edges-grid.eu
Enter your private key passphrase:
Created VOMS proxy: O=GRID-FR,C=FR,O=CNRS,OU=LAL,CN=Oleg Lodygensky,CN=proxy
Proxy is valid until: Fri Nov 19 21:59:37 2010 CET
Proxy location: /var/folders/0R/0RQqVNNAEpqqeXUidaIL0++++TM/-Tmp-/x509up_u_mboleg
```

Variables appear in `courier font`, within '`${`' and '`}`', just as “`${VARIABLE}`”. They should be replaced by their values.

Some variables used in this document:

`${XWHEP_VERSION}`	is the current version of our middleware
`${XWHEP_HOME}`	is the directory where the middleware is installed
`${HOME}`	is the user home directory

Sometimes there are some inputs the reader must provide. These appear in `courier font`, within '<' and '>', just as “`<your input>`”.

1.4 Middleware Versioning

Middleware versioning is as follows:
- X is the major part of the version
- Y is the minor part of the version
- Z is the micro part of the version

(*) X reports a very important change : for example the communication protocol changes or a major new feature is modified/introduced/removed
When X changes, backward compatibility is not ensured

(*) Y reports a important but not critical change : backward compatibility is ensured

(*) Z reports a minor change : bug correction, documentation changes etc.

In September 2011, the last version of XtremWeb-HEP is 7.6.0.
In this document we refer that version as the current version and refer it as `${XWHEP_VERSION}`.

1.5 Documentation Versioning

Documentation versioning is as follow
- X is the major part of the version
- Y is the minor part of the version
- Z is the micro part of the version

1.5.1 Version 1.0.0

DATE : MAY 5TH , 2011
AUTHORS : O. LODYGENSKY
REVIEWERS :

Initial document.

1.5.2 Version 1.1.0

DATE : MAY 25TH, 2011
AUTHORS : O. LODYGENSKY
REVIEWERS :

- XWHEP 7.5.0 introduces full certificate usage

1.5.3 Version 1.2.0

DATE : SEPT, 2011
AUTHORS : O. LODYGENSKY
REVIEWERS : S. DADOUN, E. URBAH

- XWHEP 7.6.0 introduces URI pass through.

2 Introduction to XtremWeb-HEP (XWHEP) middleware

2.1 XWHEP features

XtremWeb-HEP (XWHEP) is based on XtremWeb[1] by INRIA. It is a middleware permitting to deploy a distributed data processing infrastructure (computing grid). XWHEP belongs to the so called “Cycle Stealing” family that uses idle resources. Like some other grid middleware stacks, XWHEP uses remote resources (PCs, workstations, PDA, servers) connected to Internet, or a pool of resources inside a LAN. Participants of an XWHEP infrastructure cooperate by providing their computing resources, such as processor, memory and/or disk space.

Figure 1 permits to compare the major grid middleware stacks using cycle stealing.

	Condor	**Boinc**	***XtremWeb-HEP (XWHEP)***	**XtremWeb by INRIA**	**OurGrid**
Volunteer resources	✓	✓			
Multi OSes	✓	✓			
Type	**cluster**	**desktop grid**			
Deployment	**per domain**	**global**			
Firewall bypassing	✗	✓			
Shared FS	✓	✗			
Authentication	**delegated**	**included**			
Authorization	**delegated**	✗	✓	✗	✓
X509	✗	✗	✓	✗	✓
Sandbox	?	**?**	✓	✓	✓ ✓
Data	✓	✓	✓ ✓	✗	✗
Multi users	✓	✗	✓	✓	✓
Multi applications	✓	✗	✓	✓	✓
Volunteer experience	✓	✓	✗	✗	✗
Large deployment	✓	✓	✗	✗	✗

Figure 1: Comparison of Major Grid Middleware Stacks using Cycle Stealing

XWHEP is a lightweight grid middleware. Its a Free Software (GPL), Open Source and non-profit software to explore scientific issues and applications of Global Computing and Peer to Peer distributed systems.
The XWHEP middleware allows to setup and run Distributed Systems. Such project infrastructure

1 http://www.xtremweb.net

may be based on a community of participants. For example, XWHEP allows a High School, a University or a Company to setup and run a Global Computing or Peer to Peer distributed system for either a specific application or a range of applications.

The middleware is written in Java 1.6 computing language.

2.2 XWHEP architecture

The XWHEP architecture contains three parts, as shown in figure 2:

- one or a group of several Servers is installed and maintained by system administrators to host centralized XWHEP Services such as the 'Scheduler' and the 'Result Collector';
- distributed parts deployed over the internet:
  - XWHEP Workers are installed by citizens on their PCs to propose their computing resources to be aggregated within an XWHEP infrastructure;
  - XWHEP Clients are installed by XWHEP users (scientists, for example) on their PCs to interact with the infrastructure. The XWHEP client permits users to manage the Servers and use the distributed resources (submit jobs, retrieve results...).

This architecture corresponds to most of well known Global Computing Projects:

- The XWHEP central services permit the XWHEP administrator to manage registered applications.
- An XWHEP Client prepares data that are needed to successfully compute jobs. These data may be stored in the XWHEP infrastructure or in any repository as soon as the data can be described by an URI and are accessible through the network. There is no limit on data size, but data exceeding a few hundred mega-bytes may take very long to download or upload.
- An authorized XWHEP Client registers the application on the XWHEP infrastructure.
- A Client prepares jobs (units of work) containing the reference of a registered application, optional parameters, optional references to additional files.
- Finally, the Client submits the prepared jobs to the XWHEP Scheduler.
- Independently, XWHEP Workers contact the XWHEP Scheduler to get jobs suitable for their architecture.
- In response, the Scheduler sends a suitable job description to a Worker.
- For each file referenced by the job which is not present in the local cache of the Worker yet[2], the Worker fetches the file from the XWHEP Data Repository or from an External Data Server.
- As soon as a job has finished on Worker side, the Worker contacts the XWHEP Result Collector to send the results.

2 To reduce bandwidth usage, the worker manages a local cache in *least recently used (LRU)* mode. Each time the computer is restarted, the worker automatically cleans this local cache. This ensures that the worker does not use to much local disk space, but the worker may then sometimes need to download the same file from run to run.

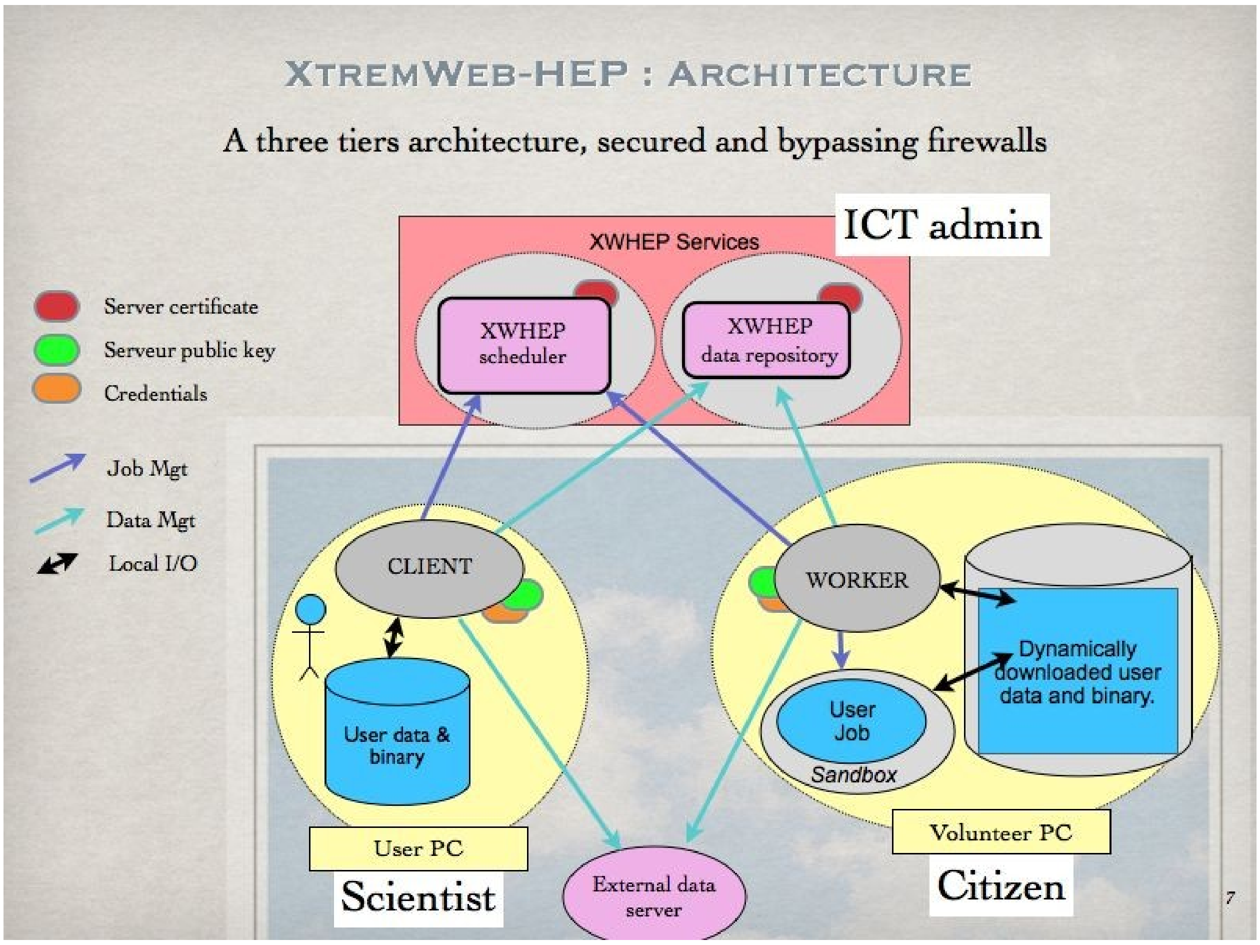


Figure 2: XWHEP Architecture

3 XWHEP hardware and software requirements

The XWHEP middleware, written in Java 1.6 computing language, is highly portable.

To run any component of the middleware, you need:

- a machine running Windows, or Linux or Mac OS X
- java runtime environment 1.6 or above

4 XWHEP software stack

The middleware is composed of different parts: the centralized server; the distributed client, worker, monitor and bridges. These parts can be layered in four slices as shown in figure 3:

1. the "OS layer", at the bottom;
2. the "service layer" on top of the OS layer;

3. the "middleware layer" containing distributed parts as the client, the worker and the monitoring tools;
4. finally the "application layer" containing the user application submitted as jobs and running on top the the worker.

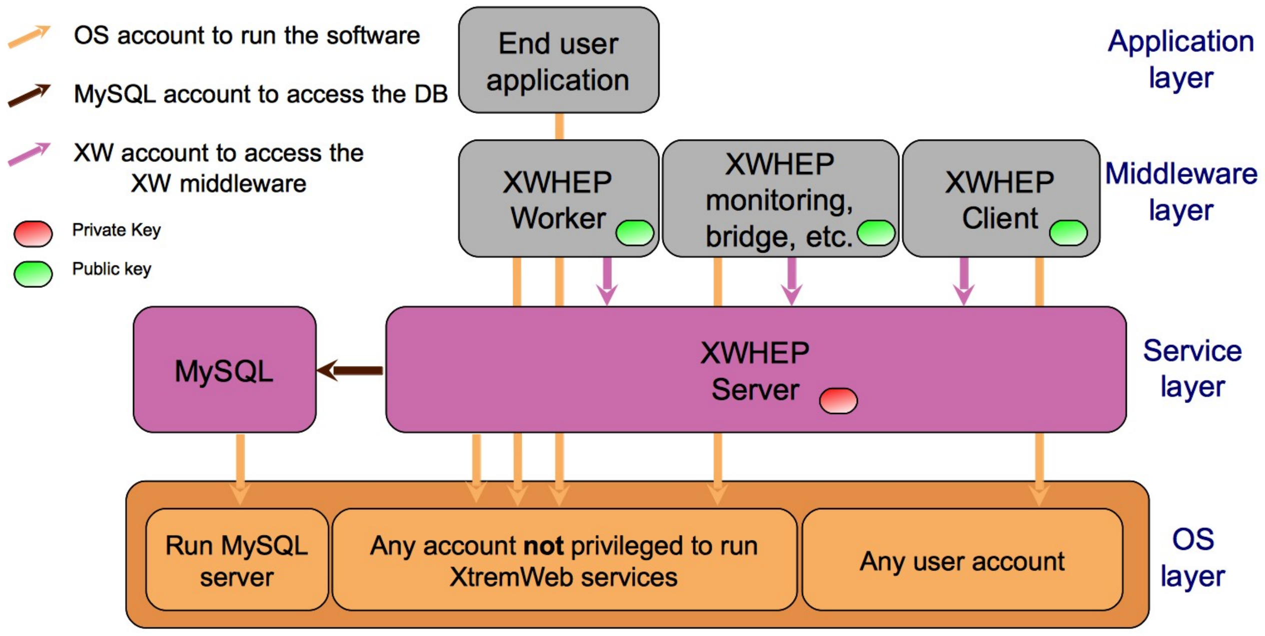

OS account to run the software
MySQL account to access the DB
XW account to access the XW middleware
Private Key
Public key
End user application
Application layer
XWHEP Worker
XWHEP monitoring, bridge, etc.
XWHEP Client
Middleware layer
MySQL
XWHEP Server
Service layer
Run MySQL server
Any account **not** privileged to run XtremWeb services
Any user account
OS layer

Figure 3: XWHEP Software Stack

5 XWHEP communication protocols

Since the platform is distributed, communications are a key point of the XWHEP middleware. The XWHEP middleware tries to be communication protocol agnostic by allowing different low level protocol as well as several high level ones. Developers may implement their own protocols by extending *CommServer* and *CommClient* abstract classes. (Please refer to the "XtremWeb Programmer Guide"). The middleware designs low level protocols as *"communication layer"* and high level ones as *"communication handlers"*.

The XWHEP middleware provides two communication layers : *UDP* and *TCP*, as well as three communication handlers described by their *URI scheme*: *HTTP*, *ATTIC*[3] and the XWHEP middleware own one: *XW*.

That last has been introduced to follow the desktop grid requirements and implements a high level of security (see chapter 7).

3 http://www.atticfs.org/

6 Description of the objects managed by XWHEP

For the description of the objects managed by XWHEP, please refer to the “XtremWeb-HEP Programmer Guide”.

7 Security considerations for XWHEP

The middleware ensures security at different levels. In this chapter we detail security in the client-server interaction. Security services and protocols in this context can be introduced as follows :

7.1 Secure resources of XWHEP

Each part of the middleware must access to some different components. It must be clearly understood that these accesses need to pass different security barriers. The different accounts need to present “credentials” to pass these security barriers.

This section details credential considerations that the middleware installation packages automatically manage at installation time.

This may sound obvious, but each part of the middleware needs to run on some computing resource. To do so, an account is used on that computing resource. Since the server and the worker parts run as “daemon” (a program that runs indefinitely in the background, and not under the direct control of a user), the middleware does not install these parts under privileged account for security reasons, but needs a non privileged user account. The client part is under the responsibility of the user who uses it, and runs under the user account on the user machine. This access type is shown as orange arrows in figure 3, labelled “OS account to run the software”.

The middleware needs a database (*DB*) where it stores all objects. A DB account is then necessary to connect to the DB server. One may note here that the XWHEP server comes with an embedded DB engine; even with that embedded engine, an account is still necessary. This access type is shown as a brown arrow in figure 3, labelled “MySQL account to access the DB”.

The middleware itself introduces a new account type needed to access the platform: register applications and data, submit jobs etc. When connecting to a XWHEP service, distributed parts must present valid credentials (usually an X509 certificate). These credentials permit to determine if that connection is authorized and what actions are permitted. This access type is shown as purple arrows in figure 3, labelled “XW account to access the XW middleware”.

Finally, the volunteer resources that run the end user jobs may also be secured by the usage of “*sandboxing”* technology. This is described in the “XtremWeb-HEP Resource Provider Guide”.

7.2 Secured communications of XWHEP

As sensitive data may be transferred over the network, the XWHEP middleware systematically ensures communications privacy by using TLS encryption. This ensures security by using encryption and by identifying communicating entities.

7.3 Authentication of XWHEP Server, Client and Worker

Server authentication is implemented by the usage of an electronic key pair: the server private key (shown in red in figure 2) is securely stored on server side, and the server public key (shown in green in figure 2) is enclosed inside the installation packages of the client and worker.

This permits to certify the server identity: distributed XWHEP components establish a session with a Server only if this server proves that it owns the private key corresponding to the public key.

On the other side, the server doesn't accept connection without receiving its own public key.

Being connected to the server is not sufficient to execute a command on server side. Each Client must present valid credentials so that the server can securely identify who is connecting.

In XWHEP, two credential types are accepted by the server :

- login/password
- X509 certificate

Using login/password

Login/password is a key pair that uniquely defines an identity. Such identities must be inserted by the XWHEP administrator.

It is the administrator responsibility to securely send login/password pair to whom it may concern.

Using X509 certificate

The XWHEP middleware is able to use X509 certificates. This is an IETF standard ensuring certificate owner identity. It is out of scope to detail certificate in this document.

The middleware authenticates the communication initiator presenting an X509 certificate by verifying the certificate authority (CA) that signed the certificate.

It is the XWHEP administrator responsibility to manage its list of trusted CA certificates.

The authentication process, if successful, defines an identity. This identity is used on server side to execute commands sent by the distributed component (client, worker). An identity has several attributes: first name, last name, email etc.

An identity has also some critical attributes involved in security: credentials (login and password, or X509 certificate), user group membership and authorization.

The XWHEP middleware manages user groups. Any user may belong to at most one user group. User groups aim to confine accesses so that members of the group are allowed to access objects defined within the group. These objects confined in a group are not accessible by users who don't belong to the user group.

7.4 XWHEP authorizations

Being authenticated by the server is not sufficient to execute a command on server side. Each authenticated entity has a usage level called "*USER RIGHTS*", that defines authorization.

Authorization allows or denies execution of requested action.

Authorizations are defined as enumerations from least to most privileged. Please refer to the "XtremWeb-HEP Programmer's Guide".

The SUPER_USER authorization allows all access rights on all XWHEP objects.. In the XWHEP middleware, we call "*administrator*" an identity associated to a SUPER_USER authorization.

The STANDARD_USER authorization allows to do actions such as INSERT_JOB, INSERT_DATA etc.

The general case is as follow: user rights *R* allow to perform an action *A* if and only if :

```
0 < A <= R
```

The only one specific case is: WORKER_USER. Presenting credentials associated with this user rights, each XWHEP worker can :

- never list anything (LIST_APP, LIST_DATA etc.)

- never insert anything (INSERT_APP, INSERT_DATA etc.)
- never delete anything (DELETE_APP, DELETE_DATA etc.)
- ask for a pending job
- send a heartbeat signal
- retrieve an application by providing the application unique identifier (UID)
- download data
- upload result for the job it has computed (this is checked by the server)

7.5 XWHEP ownership

In XWHEP, each object has an owner. By default, the owner has full access to his/her objects.

Any user may have access to an object he/she does not own, if this object permits it.

For example, any user belonging to a user group may have access to an object owned by a member of the same user group, if this object permits group access.

Accesses are defined by object access rights presented in next chapter.

7.6 XWHEP access rights

Being allowed to execute a command on server is not sufficient to execute this command on an object managed by the server.

All objects (applications, data, jobs etc.) have associated access rights. XWHEP implements this "*a la*" linux file system access rights. Access rights allow or deny access to the object it is associated with.

XWHEP assigns Read, Write [4] and Execute access rights separately for:

- the owner of the object;
- members of the group the owner belongs to (if any);
- others.

Access rights of an object can by modified by its owner or by an XWHEP administrator.

But it is never possible to increase access rights, except for administrator.

4 *"write" includes "modify" and "delete".*

There is one exception where access rights are fully bypassed so that access is always allowed: an XWHEP administrator is always allowed to access (read, write, execute) any object. This exception permits to modify access rights in any manner (even increasing).

Access rights are defined in figure 4:

U_R U_W U_E G_R G_W G_E O_R O_W O_E

Where

U_R = User (owner) read access rights

U_W = User (owner) write access rights

U_E = User (owner) execute access rights, if applicable

G_R = Group read access rights

G_W = Group write access rights

G_E = Group execute access rights, if applicable

O_R = Other read access rightss

O_W = Other write access rights

O_E = Other execute access rights, if applicable

Examples in octal notation:

```
755 defines
 - full accesses for the owner;
 - read and execute accesses to the owner group, if any;
 - read and execute accesses to others.

750 defines
 - full accesses for the owner;
 - read and execute accesses to the owner group, if any;
 - no access to others.

700 defines
 - full accesses for the owner;
 - no access to the owner group, if any;
 - no access to others.
```

Figure 4: XWHEP Access Rights

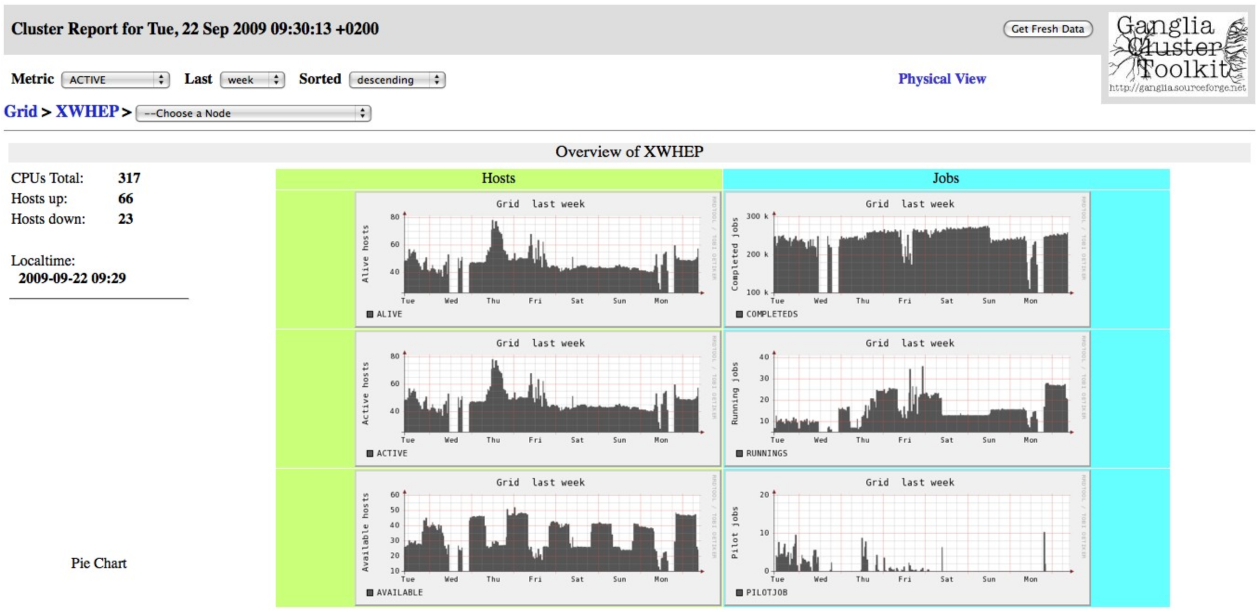
7.7 XWHEP logging and bookkeeping

The middleware proposes protocols, services and tools to post analyse events. This may especially be useful in case of errors or unconsidered utilization of the middleware.

As detailed in previous sections, users have associated credential defining their identity and authorisations. All actions executed through the middleware are logged with informations such as the date, the user credential and the action itself. The middleware comes with necessary tools to post analyse actions occurred in a deployment.

The middleware also proposes to use Ganglia[5] as the realtime monitoring system. Ganglia is a cluster and grid monitoring system; it comes with a web interface to easily display realtime monitoring.

Figure 5 shows some ganglia statistics generated by the middleware Ganglia plugin.


Cluster Report for Tue, 22 Sep 2009 09:30:13 +0200

Metric ACTIVE Last week Sorted descending

Grid > XWHEP > --Choose a Node

Overview of XWHEP

CPUs Total: 317
Hosts up: 66
Hosts down: 23

Localtime:
2009-09-22 09:29

Figure 5: XWHEP Ganglia Monitoring Page

7.8 XWHEP confinements

Since XWHEP assigns an authorisation to each identity and access rights to each object, XWHEP effectively confines all objects it manages. Inside XWHEP, a combined setup of authorisations and access rights is call a confinement.

5 http://ganglia.sourceforge.net/

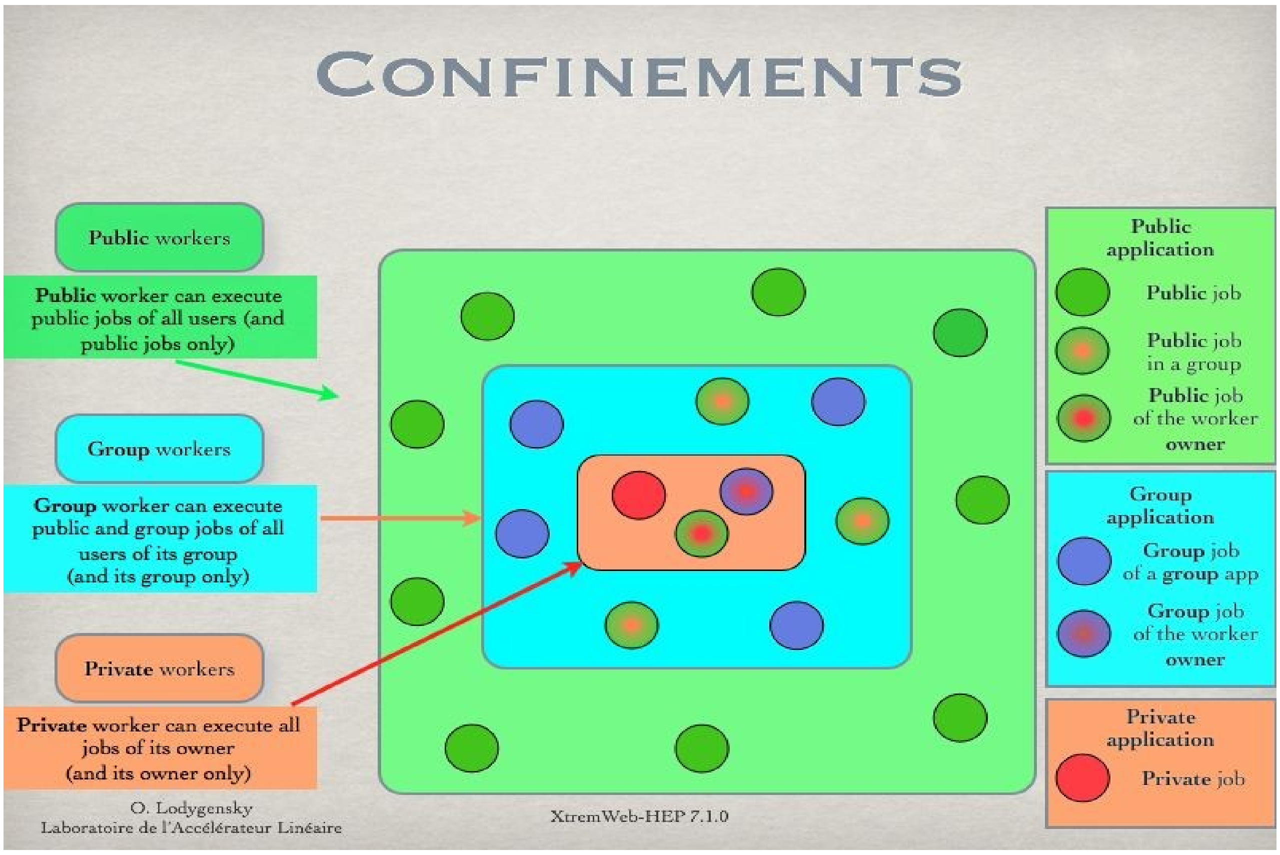
All applications, all data, all jobs are confined. But also all users, all clients and all workers.

Confinements aim to restrict accesses so that distributed entities (clients and workers) are allowed or denied to access (*read, write, execute*) confined objects.

The XWHEP middleware helps to define as many confinements as possible combinations of authorizations and access rights.

In this chapter, we present the three major confinements:

1. public
2. group
3. private


Confinements

Public worker can execute public jobs of all users (and public jobs only)

Group worker can execute public and group jobs of all users of its group (and its group only)

Private worker can execute all jobs of its owner (and its owner only)

Figure 6: XWHEP Confinements

Confinement paradigms are shown in figure 6 representing the three majors confinement levels : *public*, *group* and *private*.

Following sub chapters detail these confinements.

7.8.1 XWHEP public confinement

This is the default confinement. It is shown in green in figure 6. In a public confinement, everything is public, all distributed entities (workers, clients) have read and execute accesses to public objects. The write access, if set, is reserved to the owner only.

A public confinement is made of public objects:

- public applications;
- public jobs;
- public data;
- public workers;
- public clients;
- public users.

7.8.1.1 Access rights for public confinement

Public access rights are defined to allow owner full accesses, group read and execute accesses as well as read and execute accesses to others.

The octal value of the public access rights is : **755**.

7.8.1.2 Management of a public confinement

The XWHEP administrator is the only user allowed to manage a public confinement. Administrator authorization is the only one that permits to insert, delete and modify public users, public workers and public applications.

7.8.1.3 Resources aggregation within a Public confinement

The XWHEP administrator is the only user allowed to perform resource aggregation in a public confinement. To do so, the administrator must create an identity associated with WORKER_USER authorization.

This identity must not belong to any user group.

It is then the administrator responsibility to prepare worker installation packages that will install worker with this identity. All deployed workers will then connect to the server and present this identity.

Such workers will be automatically public.

There is no way to modify this.

Public workers execute public jobs only.

7.8.1.4 Usage of a public confinement

Any user can read and execute public objects.

Users who don't have administrator authorization can not modify public objects they don't own.

Users can send public data.

Users can submit jobs referring public applications.

Public applications must refer public data only.

Jobs referring public application are automatically public.

There is no way to modify this.

Public jobs must also refer public data only.

7.8.1.5 Behaviour of workers within a public confinement

All workers are allowed to access all public objects.

Worker confinement depends on the identity a worker presents when it connects to the XWHEP server. A worker is *public* if this identity is associated with WORKER_USER authorization and does not belong to any user group.

Public workers are not allowed to access non public objects (group and private objects).

This is why jobs referring public application are automatically public and must refer public data only.

7.8.2 XWHEP group confinement

This confinement confines accesses within a group of users only. By default all entities defined within a user group have read and execute accesses to group objects.

A group confinement is made of group objects:

- group applications;
- group jobs;
- group data;
- group workers;
- group clients;
- group users.

7.8.2.1 Access rights for group confinement

Group access rights are defined to allow owner full accesses, group read and execute accesses and to deny accesses to others.

The octal value of the group access rights is : **750**.

7.8.2.2 Management of group confinement

The XWHEP administrator is the only user allowed to create a group confinement. The administrator should delegate group management to a group administrator. A group administrator is an identity defined in a user group and associated to ADVANCED_USER authorization.

A group administrator is allowed to insert, delete and modify group objects (users, workers , applications....).

7.8.2.3 Resources aggregation within a group confinement

The group administrator is allowed to manage resources aggregation in a group confinement. To do so the group administrator must create an identity within its group and associated with WORKER_USER authorization.

It is then the group administrator responsibility to prepare worker installation packages that will install a group worker with this identity. All deployed workers will then connect to the server and present this identity.

Such workers will be automatically group workers.

There is no way to modify this.

Group workers can execute any public jobs.

Group workers may also be restricted to execute public jobs owned by members of its group.

Group workers can execute group jobs owned by members of its group.

(this includes group <u>and</u> public jobs)

7.8.2.4 Usage of a group confinement

Any member of the group can read and execute group objects. Others can't.

Users who are not administrator of the group can not modify group objects they don't own.

Users can send group data.

Users can submit jobs referring group applications.

Group applications must refer public data or data of its group only.

Jobs referring group application are automatically group jobs.

There is no way to modify this.

Group jobs may refer any public data and data of the group.

7.8.2.5 Behaviour of worker within a group confinement

Public workers are **not** allowed to access group objects.

Worker confinement depends on the identity a worker presents when it connects to the XWHEP server. A worker is a *group worker* if this identity is associated with WORKER_USER authorization and belongs to a user group.

Group workers are allowed to access public objects and objects defined in the group.

Group workers are not allowed to access private objects or objects defined in other groups.

This is why jobs referring group application are automatically group jobs and must refer data accessible within the group (public data and data defined in the group).

7.8.3 XWHEP private confinement

This confinement confines accesses to a single user only. All entities defined as private deny accesses to all users but the owner.

A private confinement is made of private objects:

- private applications;
- private jobs;
- private data;
- private workers.

7.8.3.1 Access rights for private confinement

Private access rights are defined to allow owner full accesses and to deny accesses to all others.

The octal value of the private access rights is: **700**.

7.8.3.2 Management of a private confinement

Any user can manage its own private confinement.

7.8.3.3 Resources aggregation within a private confinement

Any user is allowed to manage its own private resource aggregation.

A resource is private when its presents owner credentials **not** associated to WORKER_USER authorization (belonging or not belonging to any user group).

It is then the user responsibility to prepare worker installation packages that will install a private worker with its own identity. All deployed workers will then connect to the server and present this identity.

Such workers will be automatically private workers.

There is no way to modify this.

Private workers execute jobs owned by its owner only.

(this includes private <u>and</u> group <u>and</u> public jobs)

7.8.3.4 Usage of a private confinement

The owner is the only user allowed to access his/her private objects.

Users can create private data.

Users can submit jobs referring private applications.

Private applications must refer data accessible by the application owner.

Jobs referring private application are automatically private jobs.

There is no way to modify this.

Private jobs may refer any data accessible by its owner.

7.9 Data confidentiality

Data may circulate over the network, from client to server, from server to workers, and back. Communication are secured as described in chapter 7.2 .

The middleware enforces data confidentiality according to the authorization and access rights mechanisms described above. The confinement ensures that :

- private objects are sent to private clients and workers only;
- group objects are sent to group clients and workers only;
- public objects are freely accessible.

XWHEP provides absolutely <u>no</u> guarantee as soon as an object is sent to a distributed entity.

It is the user responsibility to properly use confinement mechanisms to ensure confidentiality and restricted access to sensitive data.

X509 certificates are automatically private data. This ensures that this very sensitive data will never be downloaded by any one but the owner. The middleware is written in such a way that workers can never access certificates.

7.10 Data integrity

The middleware ensures data integrity by checking data size and MD5 checksum on each access. It is the responsibility of the XWHEP administrator to ensure integrity of data stored on server side.

7.11 Data persistence

The XWHEP middleware does not propose anything on this field.

It is the end user responsibility to ensure persistence of his/her data. For example by using BitDiew[6] P2P storage or EGI Storage Elements.

The administrator may remove any data from the server without any formal notification to the data owner.

It is the administrator responsibility to avoid removing any data that may lead to some problems regarding registered applications and submitted jobs.

7.12 Result certification

The XWHEP middleware does not propose anything on this field.

It is the end user responsibility to verify the results of his/her jobs.

6 http://www.bitdew.net/

8 Bridging XWHEP to other grids

There are various computing grids, also known as "*Distributed Computing Infrastructure*" (*DCI*). Currently, these various grids are powered by different incompatible grid middleware stacks, and users of one grid have extreme difficulty to get access to resources managed by other grids.

An important grid type is the "*Service Grid*" (*SG*) aggregating geographically distributed cluster-like resources as a coordinated federation of independently managed computing sites. Inside Service Grids, the job servers do *not* wait that computing resources *pull* waiting jobs, but the job servers *broker* incoming jobs and *push* them to computing resources assessed as adequate. The most notable SG middleware stacks are Globus, gLite, ARC, Unicore, Naregi and Genesis.

8.1 Xtremweb-HEP plugin of the 3G Bridge

In order to bridge these various grids powered by different middleware stacks, the EDGeS and EDGI projects funded by the European Commission have designed, implemented and put in production a "*Generic Grid to Grid Bridge*", shortened as "*3G Bridge*" and shown in Figure 7.

The XWHEP team has contributed with the "Xtremweb-HEP plugin of the 3G Bridge". An infrastructure using this plugin currently permits users of gLite, ARC and Unicore to transparently take advantage of resources managed by desktop grids powered by XWHEP.

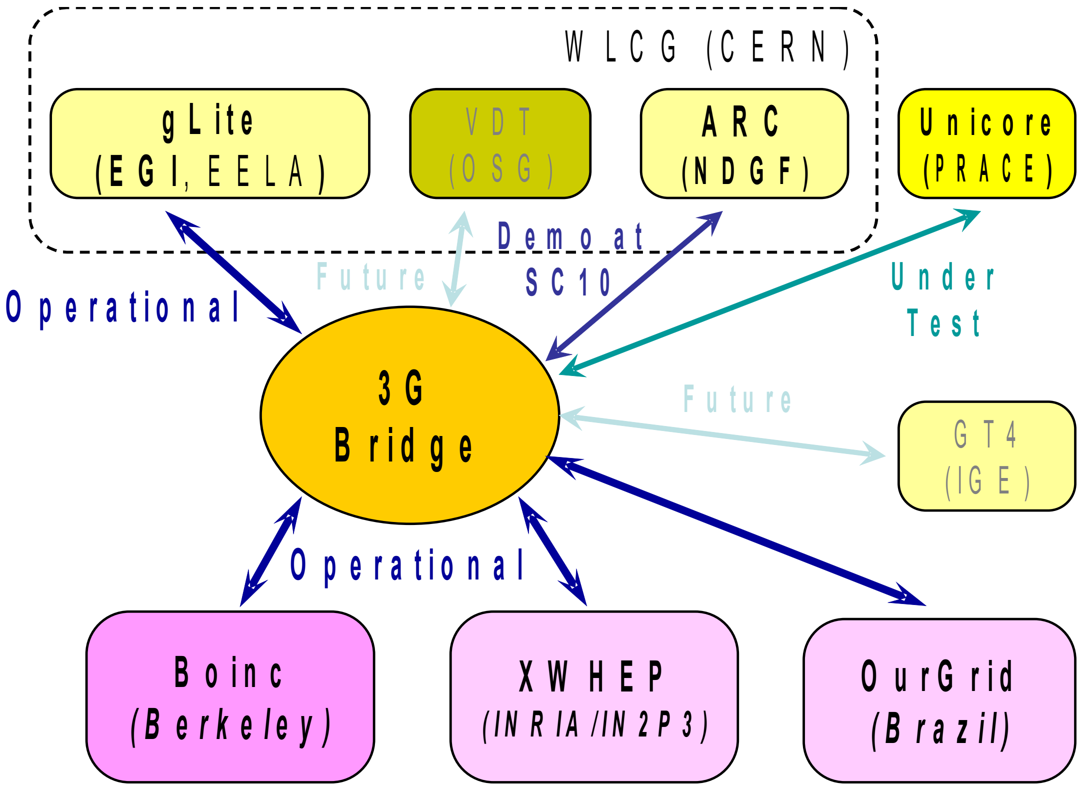


Figure 7: Grid middleware stacks bridged by the 3G Bridge

8.2 XWHEP bridge to gLite

XWHEP allows users to use not only volunteer, but also gLite "*Service Grid*" (*SG*) resources aggregating and using cluster-like resources, by proposing an innovative resource sharing technology: The XWHEP *bridge* aggregates SG resources in a Desktop Grid (DG) deployment by submitting to the SG a *pilot job (PJ) containing an XWHEP worker*. XWHEP implements the DG to gLite bridge so that, under certain circumstances, jobs submitted to an XWHEP scheduler can be run by both types of resource.

8.2.1 Bridging security for XWHEP job submission to gLite

The main concern of the XWHEP bridge is security. The XWHEP bridge must conform to SG security requirements, as well as logging and bookkeeping. To achieve these goals, XWHEP introduces unique DG features: authorization and access rights. These must be understood as those well known in clustering environment. Any user willing to use the platform must have valid credentials which define and ensure its identity as well as its authorizations. Those last allow (or deny) possible actions. For example, an user may be allowed to submit jobs, but not permitted to insert applications etc. If the possibility of an action relies on user authorizations, the success of an action depends on object access rights. All these are described in chapter 7 .

The security in SG is ensured by X509 certificates which must be signed by a known certificate authority (CA). These certificates are very sensitive data which must be securely kept by owner. They are designed with a long term life time (generally twelve months) and must definitely never be duplicated. They are secured by a password the owner must be the only one to know. As grids delegate credentials to services, grid users can generate a short life time X509 certificate (generally twelve hours) known as *X509 proxy* (or simply *proxy*). User delegation to grid services may be performed by migrating proxy; this is considered as an acceptable compromise since communications are always encrypted ensuring proxy integrity and point to point authentication. On another hand, if a proxy was to be maliciously used, its short life time would then limit the misusage.

Following these concepts, XWHEP can use X509 proxy, as any SG service since communications are encrypted and services authentication ensured by the usage of keys pair. An DG user can connect using such a proxy. The proxy is then safely transferred and copied in XWHEP data repository which ensures its integrity. The usage of this proxy being confined to the owner only thanks to XWHEP AR as explained above: in XWHEP, proxy is a private data, defined with a private access. This especially ensures that the proxy will never be sent to any volunteer computing resource. Finally logging and bookkeeping are ensured on both grids (DG and SG) since the same X509 proxy is used.

An user submitting a job to XWHEP using a proxy gains a potential access to an SG resource. The resource that will finally run the job is still unpredictable. It depends of resources usage. The job will be sent to the first available one, which may be a DG one, as well as an SG one. To ensure its security and integrity, X509 proxy will never be downloaded by any XWHEP resource, neither a volunteer one, nor a pilot job one.

8.2.2 Architecture of the XWHEP bridge to gLite

The XWHEP bridge is the only distributed part of XWHEP that uses the X509 proxy. It is a *daemon* (a program that loops for ever) that must be run on a machine where both XWHEP and SG client middleware are installed. Grid administrators must ensure the security and the integrity of the XWHEP bridge, since it performs some critical actions.
The XWHEP bridge periodically looks for XWHEP jobs submitted with a valid X509 proxy. For each job, the XWHEP bridge downloads the proxy and uses it to perform SG actions necessary to start a pilot job.
If the proxy is not valid, SG actions fail. The XWHEP bridge does not consider this as an error since the job has then no chance to be computed on any SG resources, but only on DG ones (and the XWHEP bridge does not have to report this as an error since it does not manage DG itself).
If the proxy is valid, the XWHEP bridge will monitor the pilot job and report status until completion or error. To ensure the pilot job will compute proxy owner job, the XWHEP bridge configures it as a private resource : an XWHEP resource that can only compute job for a given user.

Figure 8 shows the overall infrastructure: XWHEP and SG and the XWHEP bridge.

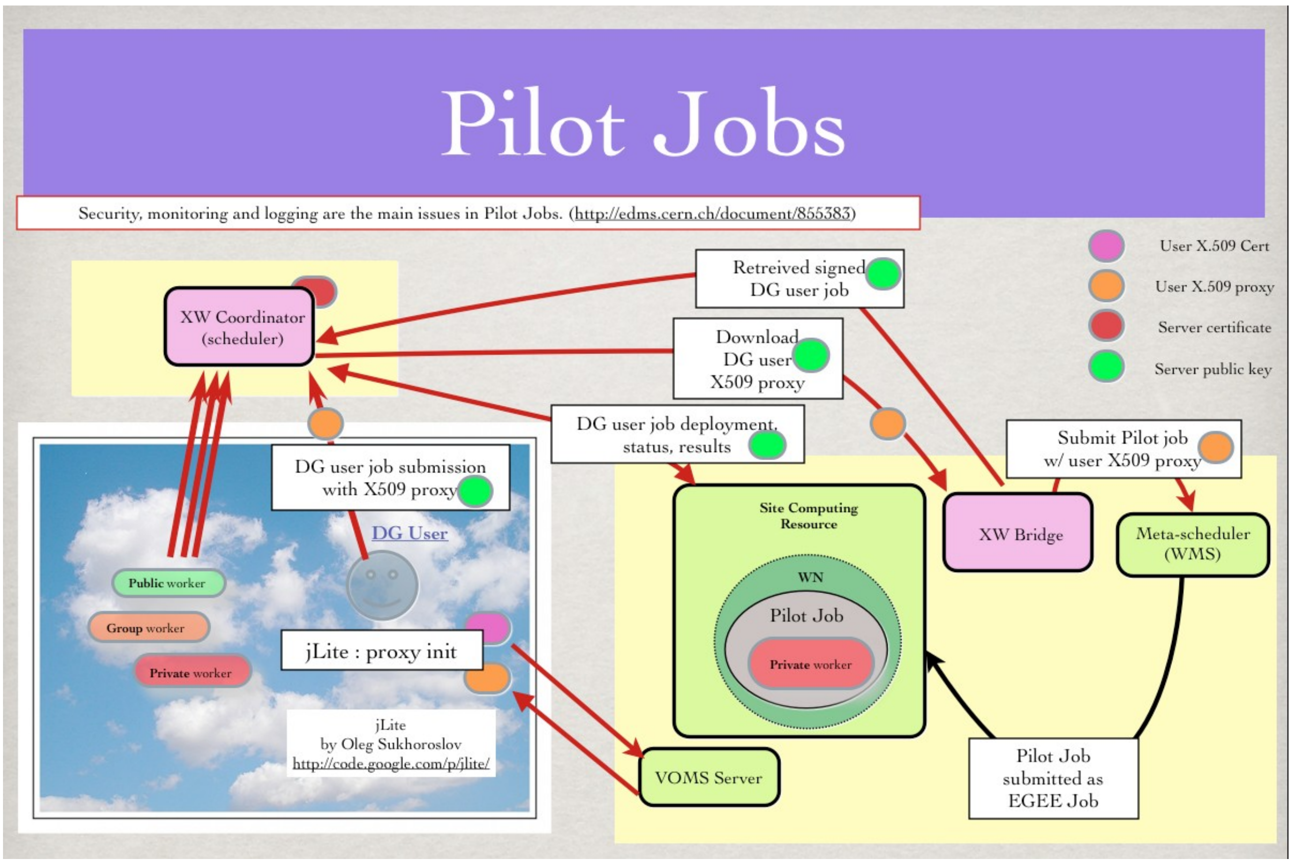

Pilot Jobs
Security, monitoring and logging are the main issues in Pilot Jobs. (http://edms.cern.ch/document/855383)
User X.509 Cert
User X.509 proxy
Server certificate
Server public key
XW Coordinator (scheduler)
Retreived signed DG user job
Download DG user X509 proxy
DG user job deployment, status, results
DG user job submission with X509 proxy
DG User
Public worker
Group worker
Private worker
jLite : proxy init
jLite by Oleg Sukhoroslov http://code.google.com/p/jlite/
VOMS Server
Site Computing Resource
WN
Pilot Job
Private worker
XW Bridge
Submit Pilot job w/ user X509 proxy
Meta-scheduler (WMS)
Pilot Job submitted as EGEE Job

Figure 8: XWHEP Pilot Jobs

8.2.3 Usage of SG resources by the XWHEP bridge

One can easily understand that there may be a resource overbooking. Since there are delays in the submission path, it may happen that the XWHEP bridge submits a pilot job that will start up after all jobs have been computed by other resources. Our experience shows that gLite resources are effectively used at an average rate of 75% by our XWHEP bridge. Figure 9 shows resources usage:

- The blue line shows the amount of jobs submitted with an X509 proxy, hence able to be computed by any gLite resources.
- The green line shows the total amount of completed jobs, independently of their submission type (with or without X509 proxy), hence computed by XWHEP and gLite resources.
- The red line shows job failures (data unavailable etc).

This figure also shows two different bar types : a black and a blue one.

- The black bar shows the amount of connected gLite resources. These resources are allocated when the XWHEP-->gLite bridge has found XWHEP jobs submitted with X509 proxy. When such a job is found, the XWHEP bridge downloads the end user proxy and uses it to submit an XWHEP pilot job to gLite (a pilot job being an XWHEP worker that will connect to the XWHEP server and request an XWHEP job to compute).
- The blue bar shows the amount of job effectively computed by gLite resources.

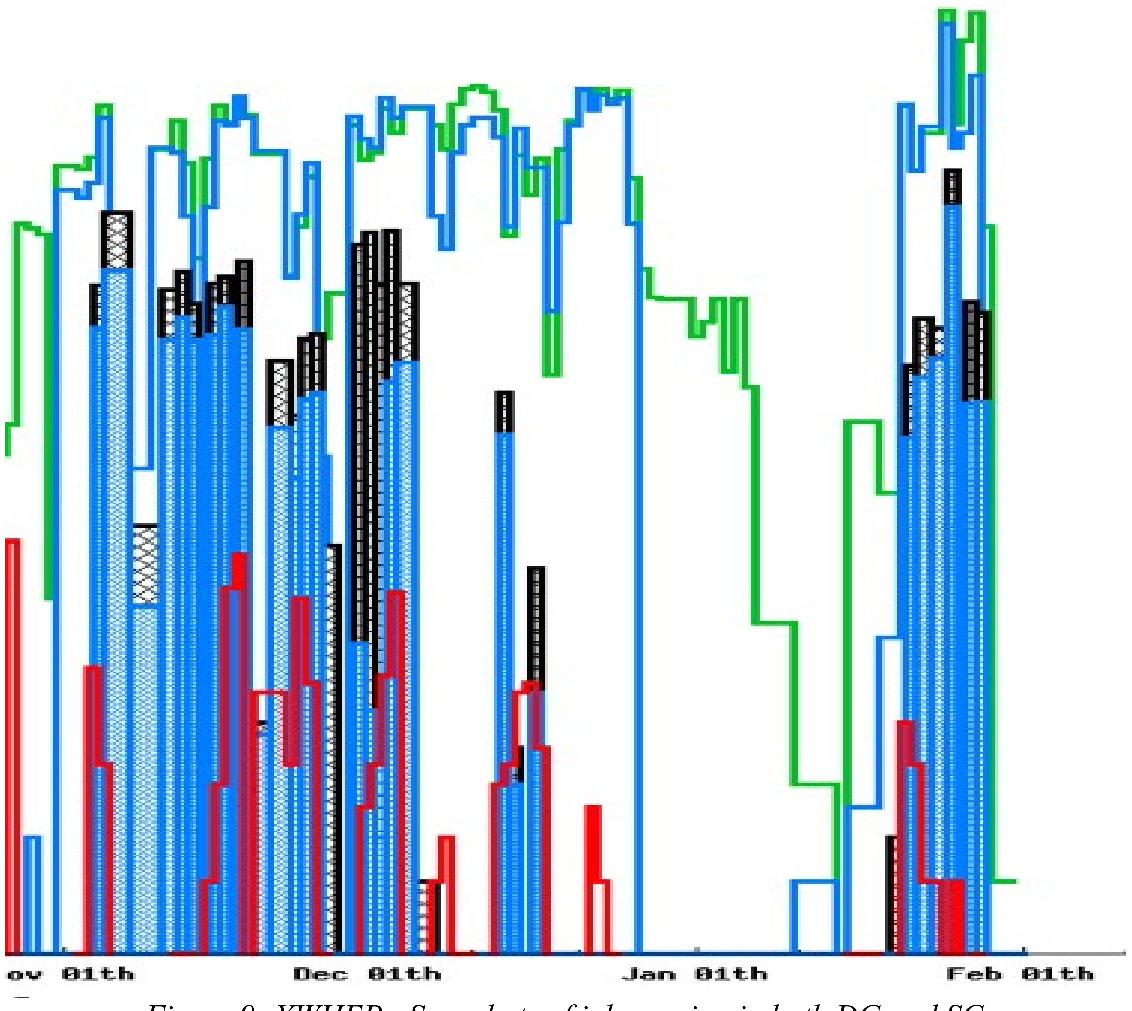


Figure 9: XWHEP : Snapshots of job running in both DG and SG

9 Contacts for XWHEP

If you need help, you can contact the XWHEP team : xtremweb (at) lal.in2p3.fr

XWHEP website : http://www.xtremweb-hep.org

You can find RPM and Mac OS X packages at this address :
`http://www.xtremweb-hep.org/spip.php?article52`

Search page for Virtual Organizations managing SG members for access to SG resources :
http://operations-portal.egi.eu/vo

Example of server permitting to request VO membership :
https://grid12.lal.in2p3.fr:8443/voms/vo.lal.in2p3.fr/Siblings.do

10 Third party libraries for XWHEP

jLite library :
`http://code.google.com/p/jlite/`

Bouncy Castle library :
`http://www.bouncycastle.org/`

Jetty library :
`http://jetty.codehaus.org/jetty/`

11 Index