MoViBio Research Group Server Infrastructure

P. Abellán J. Campos

Abstract

The Information System in a research group is a key component to suport the main tasks. This document contains the technical experience of our own research group Modeling and Visualization of Biomedical data (MoViBio) using a centralized server computer to provide some of the Information System services.

CONTENTS

1 INTRODUCTION

MoViBio [17] is a research group working on *Modeling and Visualization of Biomedical data* at Universitat Politcnica de Catalunya (UPC). Currently it is integrated in the Engineer Graphics Research Group (GIE) [13], Nanobioengineering Research Laboratory of Catalunya (CREBEC) [8], Biomedical Engineering Research Centre at UPC (CREB) [7], in the cooperative network Imagen Mdica Molecular y Multimodalidad of Spain (IM3) [14], and cooperating with the Volume Visualization and Artificial Intelligence research group (WAI) [31] at University of Barcelona (UB).

The main research lines are: multimodal data, time-varying data, nano-scale data and medical applications. The group is producing articles on all these subjects and also implementing related visualization software since 1993.

Recently the server infrastructure was revised and according to the requirements a new design was suggested. This report exposes the design, installation and configuration process of the new server infrastructure with two main *objectives*. The first objective is to document internally the whole process to reflect the design decisions for further revisions. The second objective is to provide some indications and suggestions useful to other research groups, specially to the national research network [14] to which MoViBio is attached.

The current organization of MoViBio research group involves people from different universities UPC, Universitat de Barcelona (UB) and Universitat Rovira i Virgili (URV). Thus, members are physically located in far places, and they need to share data, software, computation resources and general information to work together. This functions were initially covered sharing accounts in a main server (a Solaris operating system equipment). But, the current situation based on higher interaction requires a new server computer with some new services.

This report exposes the original services request, and the design decisions up to the current infrastructure to achieve those new requirements.

2 SERVER

Different requirements have arisen from the research group daily experience, which had to to be covered by the new server infrastructure.

There is need for communication among members of our group, with other research groups of the research network, and with any person interested in our work. Nowadays nobody denies that one of best channels of communication that exists is Internet, and therefore a Web server is the natural election for this requirement.

The different members of the group collaborate in software development and in documents elaboration. In order to automate this work, a repository that offers the possibility of storing and sharing working files, and facilitates the interaction between the different members of the group is necessary.

Also there is a need to share data files. Given to the nature of these files, which normally have read-only access and have a big size, they require a file server that allows access without the need to copy them locally.

In order to minimize the damage produced by hardware failures, or a human error, there is a need to make periodically backups of the server system, the stored data in the server, and the stored data in the user personal computers.

Thus, minimum services that the server must implement are:

- Web Server to store the research group public information
- Repositories to share software code and documents
- File server to share data files
- Backup system to warrant data persistency

2.1 Hardware

The software requirements force the hardware specifications.

The file server and the Web server require of fast access of reading from hard drive. It is also required that the server has a considerable capacity to store the data files.

In addition, a system to make backups will be necessary, so two levels have to be taken into account. The first level is related to a RAID [22] system with concurrence, in concrete a RAID 5 system, that allows to recover the system quickly if a hardware failure in a hard drive takes place. The second level will allow to recover the system from a massive failure in several discs or from any human error. It consists of making backups in removable storage devices, in our case DVDs and data tapes.

Although it is not required at this moment, we may execute intensive calculus applications in the server. Therefore we would need several fast CPUs, to be able to execute the applications and the services.

Because power cuts are common in the building where the server will be, an uninterruptible power supply unit [29] is required that provides a stabilized line.

Finally, this is the list of hardware requirements:

- Fast hard drive: Four SCSI [25] hard drive of 73 GB. That gives a 219 GB of capacity in a RAID 5 system.

- Fast CPU: Dual Processor (Intelx86 family)
- Great capacity of main memory for the applications and services: 2 GB
- A stabilized and Uninterruptible Power Supply: UPS
- A backup system
  - A RAID 5 system
  - Tape storage system
  - DVD writing unit

2.2 Operating System

Besides the hardware component, the other basic server component is the Operating System (OS). The initial options were among a UNIX family OS or a Microsoft Windows OS (Ms-Windows). All the services required could be provided by both families of OS. However our own experience showed us that we had longer uptimes on UNIX machines than in Ms-Windows ones. Basically it is thanks to the fact that the UNIX systems are prepared to have software updates without interrupting the service, while the Ms-Windows machines often need to be restarted.

On the other hand, we have identified all the software we need to offer the required services as free software [12]. This allows us to have a better control and customization of the software we run because we can access to the source code. In addition, the maintenance costs are lower for us, specially because the purchase and formation costs are significantly lower than the ones of propietary software. The purchase price mainly includes the distribution tasks, because the software itself is freely available. The formation costs are lower because our system administrators have the skills to learn by accessing the source code and the internet forum communities.

We could run all this free software collection on both OS, which is another degree of freedom of this kind of software. Inside the UNIX family OS are the GNU/Linux OS systems, which are entirely based on free software. This way, not only the applications, but the OS itself could have the advantatges of free software. Our experience has showed us that we could access to more secure updates on free software

than in propietary one, specially at OS level, where there is a large number of servers runing free software, so the comunity is larger and the capacity of finding and fixing bugs too.

Once we decided to install a GNU/Linux OS, we had to choose among all the available distributions. There are many families of GNU/Linux distributions, but the most known are the Debian-based distributions and the RedHat-based ones. RedHat-based distributions are generally prefered by companies that need certification programs and guarantees among service companies, while Debian-based distributions are more used in environments where all these certifications are not mandatory. Due to our research group university nature, we're not tied to all these certification issues, so we could take more advantages from the most spread community-based distribution: Debian itself.

The strong point of Debian-based distributions is the desktop user target, but as our system is a server machine we prefered to use the core Debian distribution. This distribution is the largest in package number -software available- and it is being the most stable last years. Due to this stability feature, we choose the Debian stable branch instead of the testing or unstable branch. In fact, we had to start by the testing branch which was going to be stable in short. Nowadays the server is running in the new stable version. You can see all the installation details in Annex A.

3 SERVICES

After the description of the server machine in the previous section, this one is centered on the services that the server must provide to the research group. The following subsections expose the requirements of every service and the most important points of their implementation. The details about the procedure to implement them are collected in the Annexes section.

The main services required are a web-server to manage the group dynamic website, a repository to manage the source code of software projects and documents, a file server as a data file storage system and a backup policy to protect the research group data.

3.1 Web Server

The main new requirement was to provide a web site to expose the research group activities to the public in general. The site should be useful also as an intranet, to manage internal research group information.

3.1.1 Requirements

The website should be distributed edited by each group member, without requiring any webmaster intervention. Nowadays, these features are usually covered with a *Content Management System* (CMS) [5]. These systems allow plain users to edit the content following a controlled document work-flow. The site administrators can modify the work-flow policies to control the content, but the content itself is produced in a distributed way by each site editor.

There exists a great amount of different CMS, using different programming languages and targeted to different editor members. The easiest site management software from the point of view of a normal content editor user are: the wikis and the portal CMS. Both of them allow editing the content from a normal web-browser, without any special application.

A *wiki* is a web site based on a meaning network created by the terms that it contains. The most known applications of this technique is the Wikipedia [36], a complete encyclopedia edited by thousands of volunteers. The content generators don't need special web design knowledge, because the pages descriptions are based on plain text including a few format tags. The main tag is the one used to create a new concept. Once a new concept has been created inside a page, any other editor user can create a new page for it. This way the pages grow following a web topology.

On the other hand, there are *portals CMS* governed by conventional web pages definition language, the Hyper-Text Mark Up Language (HTML). Those portals provide also mechanisms to edit a web page through a web-browser, but the way those documents are linked follow the hyper-text definitions. These mechanisms are based on the link between files in a folder-tree structure.

3.1.2 Software

In a first phase, the wiki concept was the one choosed because it was easier to introduce to the research group members. These members became used to edit web pages following the simple wiki syntax, and get control of their own contents very soon. The choosed wiki was the *ZWiki* [38], an implementation running on the *Zope Application Server* [37]. Among all wiki implementations [35] [34], the ZWiki has the common basic syntax and runs on an application server instead of being an standing alone application like WikiMedia [16], TWiki [28], PHPWiki [19] or UseModWiki [30]. This feature was interesting for the research group because the services related to the web server were expected to grow up. So an applications server [1], able to run on the same context more than one web application was more suitable.

Zope [37] is an applications server coded in python programming language very extensible [21]. There are other applications servers more powerful like Jakarta Tomcat [27], WebSphere [33] or JOnAs [15]. But Zope was choosen because it has a basic user orientation, so the manager interfaces are easier to use by those research group members in charge of system administration issues. Zope has its own web server and database manager -an object oriented database- by default, so it does not require any other software to run the hosted applications. In addition Zope is the hosting application server of the CMS choosen.

In a second phase, as the users increase their web page creation level with the wiki edition system, the portal CMS based on HTML was introduced. Zope is one of the Content Management Framework (CMF) [4] that provides the interface to program a CMS. This way, the transition between the original Wiki and the new portal CMS was more comfortable to final users, but also to the system administrators. However, the main reason to choose this CMF is that it hosts the *Plone* [20] CMS. This CMS was choosen among others [6] because it implements a vision of the CMS based on work-flows of documents in combination with their hierarchical organization inside a virtual file-system. The information unit of Plone are the objects, mainly those representing documents, stored in the Zope object-oriented database. These document objects have associated a work-flow which determines their possible status and properties. They are indexed by their identifier key, so they can be located following the wiki style (meaning network). But also organized inside a container hierarchy that provides all the conventional file-systems properties -like permissions, folders and soft links- that can be accessed using a web interface or integrated to other file-

systems using web-DAV [32]. In addition, there are a lot of already implemented components following this architecture that provide common web portal features like: calendars, news panels and forums. This way, Plone can be seen at the same time like a wiki and a document work-flow CMS, that it lets editors control the contents from the upper conceptual level up to the bottom implementation level as if they were working with a conventional web server based on the file-system.

3.1.3 Installation and configuration

The installation and configuration details of the Plone CMS can be found in Annex B. The main process follows the following steps:

1. *Package installation*: mainly Zope and Plone.

2. *Basic configuration*: switch to port 80, create the administrator user, create a Plone instance inside Zope and redirect the entry page to the new Plone instance.

3. *Member management*: create the users and groups, assign them permissions and avoid visitors to join the site.

4. *Layout configuration*: customize layout, logo, charset and language.

5. *Main structure creation*: create the main page and the top level folders to classify the content.

6. *Members contribution*: instruct members to introduce and manage the content, and let them populate the site.

In addition to these main steps, a few extra configurations were needed to provide the services required:

- *Wiki data migration*: import the original wiki data.

- *Member list*: create a direct link to the member list, and show the full name in it.

3.1.4 Contents

The structure given to the web site to accomplish the main objectives of diffusion and working documents repository is:

- *Main page*: research group description, web site content description and links to integrated research networks.
- *Research Lines*: every research line has its own folder.
  - *Description page*: entry page exposing the *Aim*, *Topics*, *Funding*, *Results* and the *Contact person* for that research line.
  - *Bibliography*: list of reviewed bibliography and summary cards about each entry.
- *Publications*: folder containing all research group publications.
  - *Chronological list*: all publications ordered by publication date, linked to their summary card.
  - *Summary cards*: publications summary cards including the *Title*, *Authors*, *Publication details*, a *Reference image*, and a *Bibtex description* an the *PDF file* to download.
- *Projects*:
  - *Description page*: entry page exposing the *Reference data* (title, main project, coordinators), a brief *Description*, *Participants*, *Objectives* and *Results*.
  - *Other data*: extensive documentation and files related are stored in the same folder.
- *Courses*: teaching activities of the research group.
- *Member list*: list of all the members showing their picture and full name. The entries link to each member personal page.
- *Links*: external links to interesting sites related to the research group activities.
- *Search engine*: advanced search engine to locate the information inside the site, based on contents, author, date of publication and topics.

3.2 Repositories

Usually, in a research group different members work on the same software project, articles, or data. They really work on the same group of files, even on the same file at the same time.

These changes made separately must be unified. Normally to make this task manually is not trivial, specially when working on software projects. Even in cases that are trivial it can be necessary several hours to complete the task. A repository system make this work automatic or semiautomatic, increasing the collaboration of different members in a project.

3.2.1 Requirements

The repositories system must facilitate the collaborative work, allowing manipulation, edition and update of files by different users concurrently. It must make automatically the unification of the these modifications with the minimum user intervention. It would be also worth if it could control the different versions of each file, so changes could be reverted or to return to a concrete version of a file.

The repositories will store files of software code and documents. This allows the collaboration of different users in software development and the document composition (papers, reports, ...).

Different research groups exist, each one working in different projects. Thus, each group must have a different repository with restricted access to its members. In any case, anonymous read access will be allowed.

3.2.2 Software

There are different systems that achieve the requirements, and all of them offer additional features, like the possibility of having more than one branch in the same project.

The Concurent Versions System (CVS) [10] is the control version system more used at the present. Its development began in 1986 and therefore it is a very mature and stable system. It is a centralized system that can be used locally and remotely by a client/server architecture. The connection with the server can be made with different protocols including ssh, rsh, and pserver, its own protocol that does not need any other additional service.

Subversion [26] arises to cover certain necessities that CVS does not satisfy. It has most of CVS characteristics and a very similar interface. In addition, it includes characteristics like version control of directories, copies, movements, as well as file metadata. The connection can be made by a HTTP-based network through WebDAV/DeltaV protocol, or by its own protocol. The migration of a project stored in CVS repository to Subversion repository is as easy as possible, because Subversion developers have the intention to replace CVS.

Arch [2], unlike the other solutions, is a distributed control version system. It is the youngest system of all and also the most complex to use.

We have chosen CVS between all the possibilities because it fulfills all the requirements and it is well known and used by most of the research group members. In addition, given the facility of migration to Subversion, it would be always easy to migrate from CVS to it.

3.2.3 Server Installation and configuration

The CVS installation and configuration have the following steps:

1. Installation of the necessary packages: cvs
2. Creation of a group in the system for each research group.
3. Creation of a repository for each research group.
4. Permission modifications such that only the suitable group can access to the repository.

The connection with the server can be made by the protocols *pserver* and *ssh*. For security reasons the second option has been chosen, since the *pserver* protocol keeps and transmits passwords with a trivial codification easy to break.

In section C there is a detailed description of the CVS installation and configuration.

3.2.4 Client management

The installation in the client system has the following steps:

- Installation of the necessary packages:

```
$ apt-get install cvs
```

- Creation of a project work copy :

```
$ cvs -d :ext:user@server:/repository checkout project-name
```

Once the local copy has been created we could work normally. When we want to update the repository version with the changes we made in our private copy, we will simply have to execute the following command in the main directory of the project:

```
$ cvs commit
```

If we want to update our local copy with the changes that other members have made to the repository version, we need to execute the following command in the main directory of the project:

```
$ cvs update
```

To add files, once created in the work copy, we have to notify to the CVS server that they are added to the repository with following command:

```
$ cvs add filename
```

If we want to delete a file, first we must remove it from the work copy and we need to notify the CVS server with this command:

```
$ cvs rm filename
```

3.2.5 Organization

Each research group has its own repository to store files of the software projects and documents.

Each software project will be consider as a project/module of the repository. The organization of the files in the project will be a decision of the members that work in that project.

The documents will be considered as an additional project/module of the repository. Within this module they will be organized in directories according to the document type (paper, report, poster, ...). All the own files of a document will be in a directory with a descriptive name, that will include the document year of publication.

The files that are used in different documents (bibliographies, styles, ...) will keep in a subdirectory of the main directory of the document module called *util*. These files will be referenced relatively by the documents that use them, this way facilitates the creation of a local copy.

3.3 File server

In addition to the repositories destined to development files -program codes or documents- a place to store files that, by their nature, are almost not modified, is required. These are normally data files -for example graphical databases or medical images- that can reach a large size.

These files usually are used by a great number of members of a work group. Duplicating them in the user machines adds an unnecessary cost of space and could cause a mess of different versions of the same file.

A repository that controls the different versions does not seem suitable for this type of files, since these systems are oriented mainly to small size text files. The data files used usually are binary and big size, hundreds of Megabytes and even several Gigabytes.

A file server would allow users to use those data files with no need to store them in their computer.

3.3.1 Requirements

Although GNU/Linux users predominate in the research group, there are also Microsoft Windows users. So the file server must be accessible from both systems.

Each work group will have an area where they will be able to store their files, and any member of the group will be able to make all type of modifications on the files and directories structure. Each group will be the one in charge to define the behavior policy of their members and the organization of their files.

The system must allow to work directly on the file server, that is to say, the users must not have the necessity to copy the files to their local hard drive to be able to work with them.

It is also necessary to provide an alternative mechanism to access the data from machines outside the local network, this way the members of the group that are in other locations could copy the data files and could work with them in a more efficient way.

3.3.2 Software

Between the different existing possibilities, we mainly have considered to install a Samba [24] and/or NFS [18] server. They are less complex than other solutions.

The requirement to give access to Microsoft Windows and GNU/Linux users pointed to use an hybrid solution. This way, Microsoft Windows users could connect by SMB protocol, used by files and printers sharing of Microsoft Windows, and GNU/Linux users could access by mounting the remote system of files in a local directory by the NFS protocol.

This solution has the disadvantage that the users and permissions control in NFS are local. That forces to have control on the computers clients and to have a centralized authentication system. Since in the work environment each user has total control on his/her computer -even some users have their own laptop- we have discarded to share the file system by NFS. Besides, the Samba server has the possibility to store the users and passwords, which facilitates the administration of the server. What is more, the GNU/Linux Samba client allows the user to mount a shared resource in a local directory, being left a totally transparent access for the user.

For security reasons the access to Samba server outside the local network will not be allowed. The users who are outside the local network will have to copy the files in their machine by means of safe protocol SSH, with the program scp for example.

3.3.3 Server installation and configuration

The installation and configuration of a Samba server are very simple. In summary, they consist of the following the steps:

- Installation of the necessary packages.

  ```
  $ apt-get install samba libpam-smbpass
  ```

- Configuration of the shared resources.

  - Remove the default shared resources (homes, . . .).
  - Share the wished resources.

In the Annex D we can see the detailed installation and configuration of a Samba server.

3.3.4 Client installation and configuration

As Samba reproduces the native Microsoft Windows resource sharing system, these kind of client platforms need no extra installation. Only use the navigation tool and/or map a unit to the shared resource.

In GNU/Linux we will need to install a SMB client and an application to be able to mount the shared resources:

```
$ apt-get install smbclient smbfs
```

At this moment we can access by means of the program smbclient or the tools of navigation that provide some desktops like KDE or Gnome.

If we want to mount the resource in a local directory we will have to modify the `/etc/fstab` file and add an entry similar to the following one:

```
//<server>/<resource> /<mount_directory> smbfs
    noauto,user,rw,uid=<user>,gid=users,username=<user2>  0 0
```

We must replace these entries by their corresponding value:

- `<server>`: Server name
- `<resource>`: Shared resource name
- `<mount_directory>`: Path to the mount directory
- `<user>`: Users name in the local machine
- `<user2>`: Users name in the remote machine (Samba server)

This way the resource is not mounted automatically in the system boot, and the user must mount it before accessing. To be able to automate it, we must put the password in a plain text file with the consequent security risk.

3.4 Backups

The last specified service was the backup of all the research group data. The following section explains how is this service provided taking into account that it requires special hardware components.

3.4.1 Requirements

The backup requirements specified cover two main areas: *user local data protection* and *group centralized data protection*. The first area can be seen like one more service provided by the server, while the second one is related to the server maintenance. Thus, a mechanism is provided to transfer user local data to the server, and later on a backup policy is implemented in the server to guarantee the persistence of all the stored data.

The service offered to the research group members to backup their local data should be fully automated to avoid that them have to perform any kind of system administration task. On the other hand, the main server backup policy can use human intervention because there is a server system administrator. We only need to keep history changes in the server data backup, thus the user uploaded data will have also history snapshots. Both of them, need to be periodic to ensure backup data is always up to date.

Finally, another vision of backup has been implemented: the *persistence of services*. This means that not only the server data persistence must be guaranteed, but the services themselves should also be guaranteed at a certain level. In the case of the MoViBio research group, at least the web site should be alive even if there are problems related to the web server. This way, people in general can always access to research group public information, and group members could always access to the published working documents.

3.4.2 Hardware

The backup system includes certain hardware component to achieve the specified requirements. All the server data -which includes the user desktops backups- reside in the server storage system. Thus, it is necessary to ensure data persistence of the server hard disk.

The low level chosen strategy consists of the use of hardware disk mirroring to prevent disk failures.

The purchase order included instructions to get four hard-disks of the same size. This way it has been possible to create a Redundant Array of Independent Disks [22] of level 5. This level provides data striping at the byte level and also stripe error correction information. Once the RAID virtual disk was created using the Basic Input/Output System [3] setup tool, the virtual capacity was the same than the three of the four hard-disks. The remaining capacity is used to store the redundancy information that allows one disk failure. So, currently the data persistence can be guaranteed even if one of the hard-disk fails. In such case, the purchased hardware allows to change the damaged disk without stopping the server.

The high level strategy is based on external storage systems to prevent possible damages to the server room, like fires, floods or abrupt movements. These systems allow to store the storage media in a different place than the original data. The chosen solutions have been DVD and tape storage systems.

The server includes a tape unit with a cartridge that is changed every working day by the administrator. An automatic process copy all the server's sensitive information every night -of working days- to the current tape. This information is basically that which couldn't be recovered by standard software: includes users/services data, configuration files and patched software, and excludes operating system and original software.

There are two backup tape sets, with five tape each one. This way, there is a different tape for every working day during two weeks. The backup scheme performs a full backup every Monday and incremental backups the rest of working days. This strategy provides at least a one week backup window. The administrator has to pay attention to the holidays in case he/she is not going to be able to change the tape to prevent erasing previous data. The most interesting would be to have a tape robot to automatically change the tapes, but it was out of our budget.

A part from the mirror system -which provides data persistence continuosly- and the tape system -that provides data persistence daily- we have the DVD system to provide data snapshots once a month. The DVD system is operated also by the administrator which perform full backups on this media at the end of every month. The storage of the DVDs is performed in other university buildings, while the tape sets are stored in the same building but not in the server's room.

Due to the last point of the backup requirements about the web server persistence, one other hardware component is needed. The backup system includes another computer, in our case a workstation, prepared

to serve web pages. The web pages are a static copy of the server's web site collected automatically every night. In case there would be a major problem involving the server, the Domain Name System [11] entries will be changed, so the web browser would point to this backup computer to obtain the web pages. When implementing this kind of strategy, it is important to prepare the automatic process not to overwrite the previous static copy in case the original server is not working. Otherwise, the previous valid static copy could be lost (see Annex E.2 for further details).

3.4.3 Software

The software component of the backup system include user and server applications. The user data present at the users personal computers is automatically copied to the server each day, and the server data persistence is guaranteed with different systems.

The transfer of data from the personal computers to the servers is done thanks to the file server service (see 3.3). Once the clients have attached the direct path to the server hard-disk, everything they store in that path is transfered to the server. This process could be automated using different backup tools, but the one chosen has been rsync [23]. Rsync is a simple tool to keep synchronized a mirror of directory tree in another directory. Despite its simplicity to use in a command line script, it includes a lot of different options to specify the details of what need to be kept up to date. As all the clients are GNU/Linux based personal computers, the backup script has been added to the crontab [9]. Thus, user data is mirrored every night to the server.

On the other side, the server data has different levels of backup as explained in the previous section. Besides from the software pieces needed to control the RAID, tape unit and DVD burning, there are also scripts to deal with the dynamic web data. The web-server used is the Zope Application Server (see 3.1 section), which has integrated its own object oriented database. The Zope database contains all the data necessary to restore the dynamic web sites in any other Zope installation. Thus, copying the unique database file is enough to make a daily backup of the web-server content. In addition, a static copy of the website is downloaded to the secondary web-server to make the information available in case of main server failure (see Annex E.2 for further details).

4 RESULTS AND CONCLUSIONS

The experience of providing the research group with a server machine has arisen different issues. Some of these issues have been technical decisions as the ones explained in the sections of this document, while some other issues have been more related to organization topics involving human interaction. All of them are part of what currently is our group's information system.

The system has been working for half a year and the results are quite satisfactory to all the members. The web server is used continuously by people that check information about our research group but also by group members to keep the internal information. However, the most of this internal information is stored in the server repositories, which are the most used service. On the other hand, the file server service is the less used because it is only collecting the information which is not placed in the previous two points. Finally, the backup service is been acting automatically each night and up to the moment it is not been necessary to recover any information.

After all this initial period, we are studying the possibility of providing a Central Process Unit (CPU) service to allow members to use the server's CPU to perform their jobs.

REFERENCES

[1] Application Server. http://en.wikipedia.org/wiki/Application_server.

[2] Arch Revision control system. http://www.gnu.org/software/gnu-arch.

[3] Basic Input/Ouput System. http://en.wikipedia.org/wiki/BIOS.

[4] Content Management Framework. http://en.wikipedia.org/wiki/Content_Management_Framework.

[5] Content Management System. http://en.wikipedia.org/wiki/Content_management_system.

[6] Content Management System Matrix. http://cmsmatrix.org.

[7] CREB Biomedical Engineering Research Centre. http://www.creb.upc.es.

[8] CREBEC Nanobioengineering Research Laboratory. http://www.nanobiolab.pcb.ub.es.

[9] crontab. http://en.wikipedia.org/wiki/Crontab.

[10] CVS Concurrent Versions System. http://www.cvshome.org.

[11] Domain Name System. http://en.wikipedia.org/wiki/DNS.

[12] Free Software Fundation. http://www.fsf.org/.

[13] GIE Engineer Graphics Research Group. http://www.lsi.upc.edu/∼lluis/.

[14] Imagen Mdica Molecular y Multimodalidad. http://im3.rediris.es.

[15] Java Open Application Server. http://jonas.objectweb.org/.

[16] GPL Multimedia Wiki. http://www.mediawiki.org.

[17] MoViBio Research Group. http://truja.lsi.upc.edu/movibio.

[18] Network File System. http://en.wikipedia.org/wiki/Nfs.

[19] Wiki implementation on PHP. http://phpwiki.sourceforge.net/.

[20] Open source CMS. http://plone.org.

[21] Python Programming Language. http://www.python.org.

[22] Redundant Array of Independent Disks. http://en.wikipedia.org/wiki/Redundant_array_of_independent_disks.

[23] rsync. http://rsync.samba.org/.

[24] Samba Project. http://www.samba.org/.

[25] Small Computer System Interface. http://en.wikipedia.org/wiki/Scsi.

[26] Subversion Version control system. http://www.subversion.org.

[27] Servlet container. http://jakarta.apache.org/tomcat/.

[28] Enterprise Collaboration Platform Wiki. http://twiki.org.

[29] Uninterruptible Power Supply. http://en.wikipedia.org/wiki/Uninterruptible_power_supply.

[30] UseMod Wiki implementation. http://www.usemod.com.

[31] WAI Volume Visualization and Artificial Intelligence Research Group. http://wai.maia.ub.es.

[32] Web-based Distributed Authoring and Versioning. http://www.webdav.org/.

[33] Websphere application and transaction infrastructure. http://www-306.ibm.com/software/websphere/.

[34] Comparision of wiki engines. http://www.onlamp.com/pub/a/onlamp/2004/11/04/which_wiki.html.

[35] List of wiki engines. http://c2.com/cgi/wiki?WikiEngines.

[36] Shared encyclopedia. http://en.wikipedia.org.

[37] Zope Application Server. http://www.zope.org.

[38] Zope Wiki. http://zwiki.org.

All the documentation in this section is a summary of the practical directions useful to install and configure the software mentioned in the previous sections.

A OPERATING SYSTEM

This section contains two main parts: a first one focused on the RAID configuration and a second one centered in the operating system installation once the storage-system is been defined.

A.1 Hardware RAID configuration

- First boot of the Dell Server machine
- Ctrl-M : Mangement Utility
- Management - Configure
  - Clear configuration

  - New configuration
  - Set all disk: Online (using enter key)
  - F10 Configure
  - array ”Span-1”
  - F10 Configure
  - Logical Drive configured : RAID 5
  - Accept & Save

- Management - Initialize
  - Select Logical Drive 0
  - F10 initialize
- Esc & Ctrl-Alt-Del
- REBOOT

A.2 Debian Net Install

- insert Debian Net Install (Sarge) CD
- boot options: expert26
- extra modules 1: evms-udeb, e1000, mptscsih, megaraid
- debian dist.: stable
- Partition disk 219Gb (all partitions in ext3)
  - /boot 256MB
  - swap 2 GB
  - / 10 GB

  - /home 5 GB
  - /tmp 10 GB
  - /usr 20 GB
  - /var 10 GB
  - /var/log 4 GB
  - /var/spool 4 GB
  - /var/lib/ldap 500MB
  - /home_1 144 GB
  - free 10GB
- kernel-image-2.6.8-1-386
- install the GRUB boot loader on a hard disk, to the master boot record
- Problem when rebooting: system halted after Real Time Clock
  - reboot with a Knoppix distribution
  - Mounting all partitions and changing root:

```
mount -t ext3 /dev/sda3 /mnt/sda3
mount -t proc none /mnt/sda3/proc
mount -t ext3 /dev/sda1 /mnt/sda3/boot/
(all the other partitions)
chroot /mnt/sda3 /bin/bash
```

  - Adding “testing” to apt: /etc/apt/source.list
  - install: kernel-image-2.6.10-1-686-smp
- finish Debian installation

B WEB SERVER

This section contains the details about the Zope Application Server installation and configuration. To short these notes, we use the acronym ZMI to express “Zope Manager Interface”, which can be accessed by default at http://localhost/manage.

B.1 Installation

1. Installing packages

   apt-get install zope zope-atcontenttypes zope-btreefolder2 zope-cmf1.4 zope-cmfactionicons zope-cmfcalendar1.4 zope-cmfcore1.4 zope-cmfdefault1.4 zope-cmfformcontroller zope-cmfplone zope-cmfquickinstallerto zope-cmfsin zope-cmftopic1.4 zope-dcworkflow zope-epoz zope-externaleditor zope-formulator zope-groupuserfolder zope-linguaplone zope-plonearticle zope-ploneerrorreporting zope-plonelanguagetool zope-plonetranslations zope-pts zope-speedpack zope-zaaplugins zope-zattachmentattribut zope2.7 zope2.7-archetypes zope2.7-generator zope2.7-mimetypesregistr zope2.7-portaltransforms zope2.7-validation zopectl plone zope-cmfplone zope-linguaplone zope-plonearticle zope-ploneerrorreporting zope-plonelanguagetool zope-plonetranslations

2. Switch zope to port 80 (initially is 9673)

   - edit /etc/zopectl/zopectlrc and /etc/zopectl/default.conf
     - HTTP-Port: 80
   - /etc/init.d/zope restart

3. Creating first user

   - http://localhost:80/manage = from now will be called ZMI
     - use installation password with user ”Admin”
     - acl_users - Add: new user with: Manager & Owner
     - Logout and login with the new user

4. Creating an instance

- ZMI and Via the drop-down list in the top-right corner: add a ”Plone Site”

5. Redirecting main page

   - ZMI-RootFolder ; edit index_html and leave just

     ```
     <dtml-call "RESPONSE.redirect(_['URL1']+'/instance_name/')">
     ```

B.2 Configuration

1. User and group creation

   - You better create users from inside Plone, but it seem at least the first one should be created from ZMI
     - ZMI-RootFolder-PloneInstance-acl_users-Groups: to create Administrators (set Roles-Manager/Member/Owner/Reviewer) or plain Users (set Roles-Member)

2. Avoid visitors join the site

   - Removing link : Plone-Setup-ZMIPlone ; portal_registration ; uncheck ”visible”
   - Removing capability : Plone-Setup-ZMIPlone ; Root-Security ; Uncheck Acquire ”Add portal member” and you should check ”Add portal member” for the ”Manager” role

3. Activating advanced editors

   - Plone-Configuration-Add/Del product: active Epoz ; now you can go to each user preference’s page to choose the advanced editor as default

4. Change the logo

   - Website logo: http://plone.org/documentation/howto/custom-logo
   - Favicon: ZMI/portal_skins/plone_images/favicon.ico ; click on customize, and upload your *.ico 16x16 pixels

5. Charset

   - ZMI ¿ portal_properties ¿ default_charset : set your charset (example: iso8859-15)

6. Accessing the Member List without performing a search

   - Create a ”Member List” folder, with a default document ”index_html” that contains:

     ```
     <script>
     document.location='http://<server>/<plone-site>/Members/member_search_results'
     </script>
     ```

   - Creating a “Member tab entry”:

     – ZMI/portal_actions: change the action of the entry ”Members” to this

       `string:$portal_url/Members/member_search_results`

   - Show Full Name in members search result

     – ZMI/portal_skins/plone_forms/member_search_results: click in customize and update this

       ```
       * <metal:block tal:repeat="result batch">
           <div class="card" tal:define="name result/getUserName;
             fullname python:result.getProperty('fullname','');
             home python:container.portal_membership.getHomeUrl(name,
         verifyPermission=1);
             portrait python: here.portal_membership.getPersonalPortrait(name);">
           <a href="#" tal:attributes="href home">
             <img src="defaultUser.gif"
              alt=""
              border="0"
              width="75"
              height="100"
              tal:attributes="src portrait/absolute_url" />
             <br />
             <span tal:content="fullname"> member id</span>
           </a>
           <br />
       ```

```
            </div>
        </metal:block>
```

C CVS INSTALLATION AND CONFIGURATION

To install an configure the CVS server we must execute this commands.

Install the necessaries packages:

```
$ sudo apt-get install cvs
```

Answer the questions about configuration:

```
Where are your repositories? Left it default o left it blanc.
Do you wish to fix invalid repositories? ignore (We create our repositories manually)
Do you want the history files in your repositories rotated weekly? Yes
Should the CVS pserver be enabled?  No
```

Make a directory for all de repositories:

```
$ sudo mkdir /home_1/cvs
```

Add a group for all the users of CVS:

```
$ sudo groupadd cvs
```

Change the directory's group:

```
$ sudo chown root:cvs /home\1/cvs
```

Remove permissions for the ”others” from the directory:

```
$ sudo chmod o-rx /home_1/cvs
```

Make a soft link for easy access:

```
$ sudo ln -s /home_1/cvs /cvs
```

Create a system group for every research group:

```
$ sudo groupadd cvs-movibio
```

Create a repository for every research group:

```
$ sudo cvs -d /cvs/movibio init
```

Change group and permission of the repositories to avoid access of non group's members:

```
$ sudo chown -R root:cvs-movibio /cvs/movibio
$ sudo chmod -R o-rwx /cvs/movibio
```

Set the sticky bit for the group in the repositories, so the files will belong to this group and all group members can modify these files:

```
$ sudo chmod g+s /cvs/movibio
```

For security reasons, remove write access to the group for the directory CVSROOT of each repository:

```
$ sudo chmod g-w /cvs/movibio/CVSROOT
```

Add users to the corresponding group:

```
$ adduser pabellan cvs-movibio
```

D SAMBA INSTALLATION AND CONFIGURATION

Install necessary packages:

```
$ sudo apt-get install samba libpam-smbpass
```

Answer the questions about configuration:

```
Workgroup/Domain Name? GIE
Use password encryption? Yes
Modify smb.conf to use WINS settings from DHCP? No
How do you want to run Samba?  daemons
Create samba password database, /var/lib/samba/passdb.tdb?  Yes
```

Edit `smb.conf` file:

```
$ sudoedit /etc/samba/smb.conf
```

- Change security level to user: `security level  = user`
- Comment homes section, we don't want to share user's homes.
- Comment printers section, we don't want to share any printer.
- Only allow access from local network: `hosts allow = 147.83.29.`

Create a directory to hold shares:

```
$ sudo mkdir /home_1/samba
$ sudo chown root:users /home_1/samba
$ sudo chmod 0755 /home_1/samba
```

Create a directory to be shared:

```
$ sudo mkdir /home_1/samba/movibio
$ sudo addgroup movibio
$ sudo chown root:movibio /home_1/samba/movibio
$ sudo chmod 2770 /home_1/samba/movibio
```

Edit samba configuration file and add the new shared resource:

```
$ sudoedit /etc/samba/smb.conf
```

```
[movibio]
     path = /home_1/samba/movibio
     browseable = yes
     writable = yes
     create mask = 0000
     directory mask = 0000
     force create mode = 0660
     force directory mode = 0770
     public = no
     force group = movibio
```

Notify server that must read the configuration file:

```
$ sudo killall -HUP smbd
$ sudo killall -HUP nmbd
```

To synchronize system passwords and samba passwords edit `/etc/pam.d/passwd` file and add this line:

```
+ password required pam_smbpass.so nullok use_authtok try_first_pass
```

Add users in Samba database:

```
$ smbpasswd -a username
```

E BACKUPS

E.1 Clients

- rsync installation:

```
apt-get install rsync
```

- backup script content:

```
#!/bin/bash
ORIG=/home/<user>/

rsync -CRauvzogtp --delete --max-delete=100 --rsh=/usr/bin/ssh \
$ORIG/dir/to/copy1 \
$ORIG/dir/to/copyN \
<usuari>@<server>:~/backup/
```

- to avoid human intervention:
  - to skip password demand:

```
ssh-keygen -t rsa
scp ~/.ssh/id_rsa.pub <server>:~/tempo
ssh <server> "cat ~/tempo >> ~/.ssh/authorized_keys ; rm ~/tempo"
```

- daily execution:
  - crontab -e

```
0 <time> * * * /home/<user>/backup.sh 2>&1 1>/home/<user>/backup.sh.log
```

E.2 Webserver

- On server backup system
  - adduser bkpusr
  - create script: /home/bkpusr/scripts/bkp_zope.sh

```
#!/bin/bash

echo "ini: ‘date --iso-8601=seconds‘"

SRC_FILE=/var/lib/zope/instance/default/var/Data.fs
CUR_FILE=Zope-Data-$DATE.fs.tar.gz
TARGET_DIR=/home/bkpusr/data/
DATE=‘date --iso-8601=date‘

tar cvzf $TARGET_DIR/$CUR_FILE /var/lib/zope/instance/default/var/Data.fs

scp $TARGET_DIR/$CUR_FILE edanna.lsi.upc.edu:~/Truja/Zope-Data.fs.tar.gz

echo "end: ‘date --iso-8601=seconds‘"
```

  - allow execution: chmod u+x bkp_zope.sh
  - daily execution:
    - crontab -u bkpusr -e

```
1 1 * * * /home/bkpusr/scripts/bkp_zope.sh 1>>/home/bkpusr/logs/bkp_zope.log 2>&1
```

    * `ssh-keygen -t dsa`
    * paste the destine `/.ssh/id_dsa.pub` to the server `/.ssh/authorized_keys` and the opposite.

- Alternative static webserver

  – Install a minimal webpage server application in the alternative server machine: for example apache

    * apt-get install apache
    * vi /etc/apache/httpd.conf

```
[...]
DocumentRoot /var/www/bkpusr
AddDefaultCharset UTF-8
[...]
<IfModule mod_alias.c>
  #Alias /images/ /usr/share/images/
  # Comment this line, because Plone has a directory with the same name
</IfModule>
[...]
```

    * /etc/init.d/apache restart
    * http://localhost

  – Create static copy destine directory

```
mkdir /home/bkpusr/data/static
chown bkpusr.bkpusr /home/bkpusr/data/static/
chmod a+rx /home/bkpusr/data/static/
ln -s /home/bkpusr/data/static /var/www/bkpusr
```

  – Create static copy script:

    * `/home/bkpusr/scripts/static_www.sh`

```
#!/bin/bash
```

```
TARGET_HOST=<main_server>
STARTING_URL=http://<main_server>/
DESTINE_DIR=/home/bkpusr/data/static
DATE=‘date --iso-8601=date‘
RECURSIVE_LEVEL=inf

echo "ini: ‘date --iso-8601=seconds‘"

rm -rf "${DESTINE_DIR}_old"
mv $DESTINE_DIR "${DESTINE_DIR}_old"

mkdir $DESTINE_DIR
cd $DESTINE_DIR

wget --timestamping --no-host-directories \
     --quiet \
     --recursive --level=$RECURSIVE_LEVEL --convert-links --page-requisites \
     --exclude-directories="/user"\
,"/<site>/undo_form"\
     --reject="robots.txt"\
,"*\?*"\
,"document_view*"\
,"sendto_form"\
,"search*"\
,"search_form"\
,"login_form"\
,"join_form"\
,"mail_password_form"\
,"*month\:int*"\
     --header=’Accept-Charset: iso-8859-15’ \
     --domains=$TARGET_HOST --no-parent "$STARTING_URL"
```

```
scp -rp $DESTINE_DIR <alternative_server>:~/<destine_dir>


echo "end: `date --iso-8601=seconds`"
```

* Update permissions: `chmod u+x static.sh`
* Daily execution: `crontab -u bkpusr -e`

```
2 2 * * * /home/bkpusr/static.sh 1>>/home/bkpusr/logs/static.log 2>&1
```