UPGRADE is the European Journal for the Informatics Professional, published bimonthly at <http://www.upgrade-cepis.org/>

Publisher
UPGRADE is published on behalf of CEPIS (Council of European Professional Informatics Societies, <http://www.cepis.org/>) by Novática <http://www.ati.es/novatica/>, journal of the Spanish CEPIS society ATI (*Asociación de Técnicos de Informática,* <http://www.ati.es/>)

UPGRADE monographs are also published in Spanish (full version printed; summary, abstracts and some articles online) by Novática

UPGRADE was created in October 2000 by CEPIS and was first published by Novática and INFORMATIK/INFORMATIQUE, bimonthly journal of SVI/FSI (Swiss Federation of Professional Informatics Societies, <http://www.svifsi.ch/>)

UPGRADE is the anchor point for UPENET (UPGRADE European NETwork), the network of CEPIS member societies' publications, that currently includes the following ones:
- Informatica, journal from the Slovenian CEPIS society SDI
- Informatik-Spektrum, journal published by Springer Verlag on behalf of the CEPIS societies GI, Germany, and SI, Switzerland
- ITNOW, magazine published by Oxford University Press on behalf of the British CEPIS society BCS
- Mondo Digitale, digital journal from the Italian CEPIS society AICA
- Novática, journal from the Spanish CEPIS society ATI
- OCG Journal, journal from the Austrian CEPIS society OCG
- Pliroforiki, journal from the Cyprus CEPIS society CCS
- Pro Dialog, journal from the Polish CEPIS society PTI-PIPS
- Tölvumál, journal from the Icelandic CEPIS society ISIP

Cover page designed by Concha Arias-Pérez and Diego Blasco-Vázquez
"Abacus" / © CEPIS 2008
Layout Design: François Louis Nicolet
Composition: Jorge Llácer-Gil de Ramales

Editorial correspondence: Llorenç Pagés-Casas <pages@ati.es>
Advertising correspondence: <novatica@ati.es>

UPGRADE Newslist available at
<http://www.upgrade-cepis.org/pages/editinfo.html#newslist>

ISSN 1684-5285

Monograph of next issue (August 2008)

"EUCIP: A Model for Definition and Measurement of ICT Skills"

(The full schedule of **UP**GRADE is available at our website)

CEPIS UPGRADE
The European Journal for the Informatics Professional
http://www.upgrade-cepis.org

Vol. IX, issue No. 3, June 2008

Monograph: Technology-Enhanced Learning
(published jointly with Novática*)
Guest Editors: *Carlos Delgado-Kloos and Fridolin Wild*

UPENET (UPGRADE European NETwork)

CEPIS NEWS

* This monograph will be also published in Spanish (full version printed; summary, abstracts, and some articles online) by **Novática**, journal of the Spanish CEPIS society ATI (*Asociación de Técnicos de Informática*) at <http://www.ati.es/novatica/>.

UPC's Moodle Platform: A Study on Architecture and Performance

Marcos Montero-Torres

This article describes a design and implementation project for a Moodle architecture capable of providing service to a community of 30,000 users under criteria of scalability, performance, and high availability. In addition to the design of the architecture, we look at the design and development of a series of performance tests which allow us to enhance the efficiency of the system and reliably establish the validity of the platform in terms of environment dimension sufficiently in advance of its actual implementation. The aim is also to be able to repeat this type of performance analysis on a regular basis ahead of future modifications of the Moodle platform.

Keywords: Atenea, Digital Campus, JMeter, Moodle, Performance, UPC, UPCnet.

Author

Marcos Montero-Torres has a degree in Computer Science Engineering from the *Universitat Politècnica de Catalunya* (UPC). He has worked at UPCnet as project manager, mainly specializing in large e-mail and Web system installations and high availability solutions. He has been fully involved in the design and deployment of Atenea, the e-Learning platform at the UPC. He currently works as IT Product Manager at UPCnet. <marcos.montero@upcnet.es>.

1 Introduction

During the academic year 2005/2006, UPC carried out a pilot implementation project on a new Moodle-based e-Learning platform. This pilot project had a limited scope within the university, providing a small number of subjects to a community made up of 4,700 learners and 700 teachers.

The UPC already had its own e-Learning platform, developed entirely within the university itself and which enjoyed a significant degree of implementation and usage. After evaluating a number of options and verifying the viability of using Moodle from an educational viewpoint and its appropriateness to the needs of UPC's teaching staff, it was decided to use it as an e-Learning tool throughout the entire university. To this end a project was designed to extend the new digital campus (to be known as Atenea) to the entire UPC community.

In this new phase the digital campus was to provide service to a community of 30,000 students and 3,000 teachers, offering around 4,000 Moodle courses. A volume such as this meant that it would not only be a key tool within the university's learning environment, but it would also be one of the largest Moodle installations in any Spanish-speaking university. And as such, its implementation had to be nothing short of a complete success. One of the key success factors (though not the only one) was that the platform providing this service should meet a number of requirements in terms of performance, scalability, and availability which would ensure that Moodle would function correctly under extreme load conditions and would be able to respond easily to future extensions of the service, whether in terms of number of users, number of academic subjects, or load capacity.

After an architecture design project and a performance study to ensure its correct dimensioning (as described later in this article), the system finally started up in September, 2006.

2 Architecture Hardware

2.1 Architecture During the Pilot Project

In the first version of the pilot project, the Moodle architecture comprised 2 separate layers (Figure 1):

- **Front-end layer**: made up of 3 Debian + Apache servers. There is a generic Domain Name System (DNS) name for the campus containing the Internet Protocol (IP) addresses of the 3 servers. Front-end load distribution is performed transparently using RoundRobin which provides the DNS resolution mechanism.
- **Back-end layer**: made up of a single database server with a PostgreSQL database management system (DBMS) plus an Network File System (NFS) exported disk space to which the 3 front-ends access. The back-end server stores data in local partitions using RAID[1] 5.

2.2 Present Architecture

The basic concepts of the pilot project are maintained in the new architecture: a single instance of Moodle to serve the entire UPC community and physical separation of the Web servers and database servers. Special emphasis is also given to improving possibilities of scalability, load distribution and high availability.

The definitive architecture comprises 4 separate layers (Figure 2):

Load balancing:

A cluster comprising 2 servers with Linux Virtual Server (LVS) and High Availability (HA) is responsible for balancing the traffic between the various HTTP front-ends. They enable the load to be distributed "intelligently", while improving availability by dynamically eliminating non-operational front-ends. They also allow the system to grow or shrink transparently (for maintenance purposes, say) by adding nodes to or eliminating nodes from the front-end,

[1] Redundant Array of Inexpensive Disks

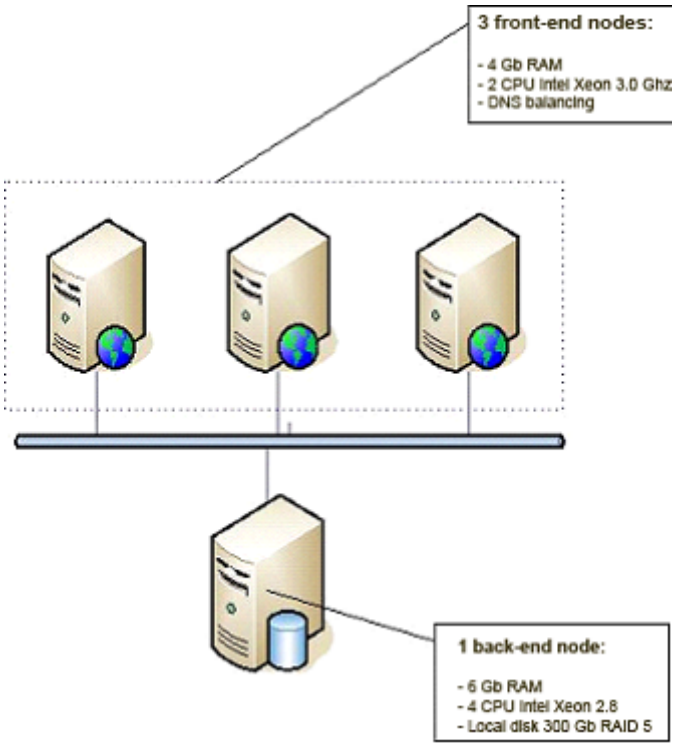

Figure 1: Architecture of the UPC Moodle Platform During the Pilot Project.

without impacting on the service.

HTTP front-ends:

Hardware structure remains the same as before, comprising 3 Linux servers Linux with Debian + Apache. The servers have two 2.8 GHz Xeon CPUs and 4GB of RAM. A number of enhancements have been made to the software which we will comment on later in this article.

Back-end:

Comprising two servers, of which only one provides service to the campus:

■ 1 primary back-end server. 6GB of RAM and 4 CPUs. Contains an DBMS PostgreSQL v8.1 and a 2Gb Host Bus Adapter (HBA) Emulex LPE9802 board for connection to a Storage Area Network (SAN).

■1 back-end backup server. This is a machine very similar in terms of hardware to the above mentioned server and with an identical configuration.

The decision to use this device was based on the fact that, at the time of implementing Moodle at UPC, the only available candidates (due to constraints imposed by Moodle) for use as a DBMS were MySQL and PostgreSQL. Neither of these two platforms were yet able to provide clustering support that was sufficiently mature (or of an acceptable complexity). For this reason we decided to use a system without automated high availability. A single server is responsible for managing access to data, but the platform has been equipped with a backup node which, in combination with SAN storage, ensures rapid service recovery in the event of a hardware failure.

SAN storage:

The data is located in specific volumes in a SAN comprising a Bull FDA 2400 disk array and 2 fibre-channel Brocade Silkworm 3850 switches. The data volumes are composed of RAID 6 partitions (providing 2 redundancy disks) of 120GB for the database and 200GB for the "Moodledata" file system.

3 Simulation Tool

3.1 Selection Criteria

The choice of the simulation tool was determined by its capacity to perform realistic tests that would allow us to check the Moodle's performance throughout the UPC. The essential factors that have been taken into consideration are:

■ Possibility of establishing and checking *usage quality* criteria: for example, the generation of statistics based on the response time for each HTTP request or the number/percentage of bad requests.

■ *User concurrency* management: the tool must be able to simulate simultaneous accesses to the system by different users. It must be able to manage their authentication and each user's particular characteristics.

■ *Scalability*: it is essential for the tool to be able to generate simulations under very high loads in order to meet the ever growing needs of the digital campus environment.

■ Possibility of performing *realistic tests*: not any simulation will do. It is not enough to send HTTP requests to the system at random. To ensure that the test is reliable, it is necessary to use browsing patterns similar to those of the

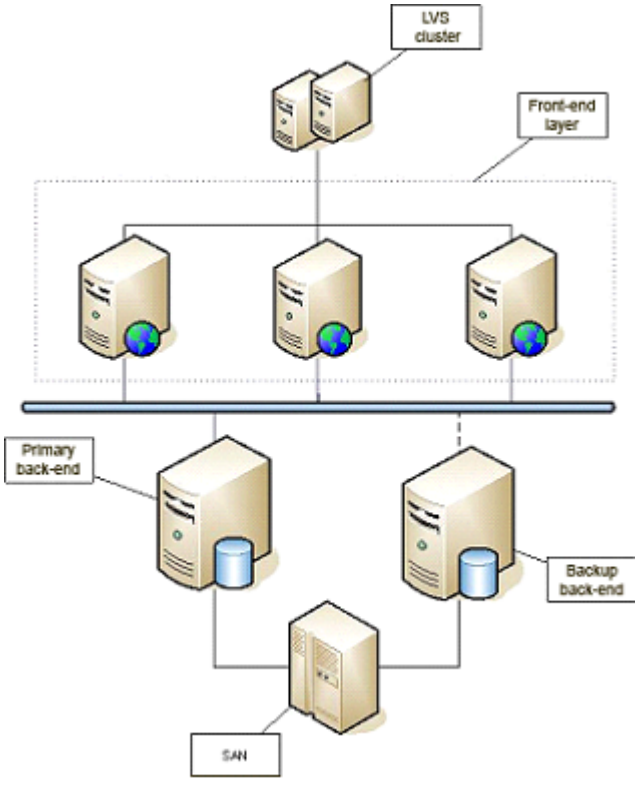

Figure 2: Present Architecture of the UPC Moodle Platform.

system's users, to consider session times similar to theirs, and even to use time intervals between different requests that are as realistic as possible. In short, it is important to be able to design usage profiles that are tailored to the operating environment, and for the tool to be able to reproduce such profiles on a large scale.

3.2 JMeter

The tool chosen to perform the performance tests was Apache JMeter. Developed by the Apache Software Foundation, it is a 100% Java application which is commonly used to perform functional tests and measure performance. It was originally designed to test Web applications, but since then it has evolved and it now features increased functionality and can be used to perform a wider range of tests.

It can be used to simulate very high loads on servers, networks, or specific applications in order to verify their capacity or analyse general performance under different load conditions.

Its main features include:

- Load modelling. Based on real logs from Apache or by generating model users specific to our application.
- Authentication and cookie management for each user.
- Provides for the verification of quality criteria.
- Possibility of cluster installation for simulations requiring very high loads.

4 Performance Tests

The performance tests conducted on the system had two complementary purposes: firstly to certify the validity of the chosen technological architecture and its ability to handle the kind of load volumes expected from UPC's community of students and teachers (30,000 students, 3000 teachers). And, secondly, to identify improvable aspects of the platform in order to provide it with sufficient response capacity and, once that has been achieved, to provide us with the best possible idea as to which elements might be improved in the event of future extensions.

4.1 Preparation of the Test Scenario

Prior to the performance of the load tests, a usage profile was produced for the digital campus based on the system's real usage data during the pilot project. The intention was to obtain the most reliable information possible about the browsing style of users on campus: session times, most accessed sections, time intervals between requests etc.

In order to obtain this data it was necessary to select the busiest periods during the pilot project. The logs were extracted from the web servers and all logins corresponding to non-teaching periods were eliminated.

With the aid of a Web statistics program (AWStats) the periods with the highest concurrency levels were selected and, on the basis of that data, the following parameters were obtained:

- Average session time: 7.25 mins.
- 50.26 hits/user.
- 90% logins belonging to learners, 10% to teachers.
- List of most accessed URLs.

Once the most relevant aspects defining user behaviour had been determined, two different browsing profiles were constructed; one for teachers and one for learners. Access to Moodle by other user behaviour profiles is so marginal compared to these two as to be not worth including in the load tests.

The profiles were constructed by combining the results obtained from studying the specific browsing history of users experienced in the use of Moodle. By doing this we aimed to ensure a minimum level of access to the most important areas of the campus in each session.

The next step was to set up a small test environment (made up of just 1 front-end server and 1 back-end server), and copy the operating database to it. On top of the original content of the database, we added a total of 3000 test users (300 teachers and the rest students), all enrolled in various subjects created expressly for the test. The aim was to have a database that, rather than being made up of recently created tables and containing minimal data, would have the size and "possible degradation" typical of a database in regular use, with the addition of new users and subjects.

4.2 Running Tests: Bottlenecks and Their Solutions

Once the test scenario had been set up, we embarked on a 2 month testing programme, first using a test environment and then running two final tests in the operating environment with its definitive configuration.

The complexity of the tests was gradually increased. We started with very simple concurrency levels (a single user to test the validity of the profiles, 10-20 simultaneous users, 50-100 users, 200 users...), and finished with the two large scale tests under the real operating environment.

We chose this gradual approach because it allowed us to discover, step by step, where the system's bottlenecks were and to apply the most appropriate solutions before moving on to the next test.

Thus, the performance of the system gradually improved and, by the time we ran the last two tests under the operating environment, we were already looking at thoroughly tuned system.

For those last two definitive tests under the operating environment we used a JMeter cluster to be able to generate the required load. The aim was to check whether the system was capable of handling 1,000 Moodle users working simultaneously. To this end the following tests were performed as shown in Table 1.

It should be noted that simultaneity does not refer merely to HTTP requests, but rather to the number of users in the system at the same time running part of a session, as defined previously in our user profiles. In other words, we are looking at 1,200 or 1,500 seven minute sessions involving different users and overlapping in time.

Below we present a summary of the most serious bottlenecks detected in the platform as the tests progressed, and the solutions that we found for them. They are grouped together according to the layer of the platform involved:

Load balancers:

Once the balancers were installed, the next step was to optimally tune the values of the various time-outs of *ldirector* and define an appropriate test to verify the HTTP service (in our case, the query of a *.gif* file).

HTTP front-ends:

	#PC cluster JMeter	#Hits/sec	#Hits/hour	Concurrent users
Test 1	12	600	2,100,000	1,200
Test 2	18	800	2,900,000	1,500

Table 1: User Concurrency Tests Using JMeter Performed on UPC's Moodle Platform.

■ Installation of a PHP accelerator (eAccelerator). As Moodle code is almost entirely based on PHP, this is the obvious way of tuning it. However, the improvement rates obtained when an accelerator is installed may vary according to a number of factors. In our tests we saw a significant, but not excessive, improvement (an increase in system capacity of around 15% compared to the tests without an accelerator).

■ Apache processes. This is the priority aim of this phase. The concurrent capacity of the Web server depends, to a large extent, on the memory available (the performance of the CPU is already improved by the PHP accelerator).

While the performance of the Apache Web server is magnificent, the use of PHP imposes a number of constraints: since PHP is not considered to be a thread-safe product, it is not possible to run Apache under the mpm-worker module, which is where it performs best. Moodle (PHP, actually) forces us to use the mpm-prefork module, running Apache in multi-process mode instead of multi-thread.

This means that, in order to handle each HTTP request, Apache spawns a child process which consumes additional memory. Thus, the amount of processes Apache can run in our system is physically limited by the available memory divided by the memory occupied by any one of these processes.

From our measurements we observed that the overhead of an Apache process ranged between 10 and 12MB (reaching as high as 15MB depending on system conditions). Assuming a system with 3GB of free memory, we could have no more than 300 processes running on each front-end.

To raise this limit without increasing the physical memory of our servers we took a close look at the HTTP traffic and concluded that over 50% of the requests handled by the Web server corresponded to static content, more specifically small images (.gif, .png, .jpg) which are served over and over again in headers, footers, icons, logos, etc.

To increase the capacity of the system, a second Apache server was installed in the front-end servers with an extremely simple configuration. It was deliberately compiled to exclude most of the modules that a standard server has; due to this simplicity only 2MB of RAM are occupied by Apache processes.

This server, which we call "Tiny-Apache", is the Web server listening at port 80. For each request that arrives, all it does is to determine whether the content is static or dynamic, serve the request if its static content, and otherwise send it to the standard Apache server (which now listens at port 8080).

In this way, most HTTP requests received by the system are handled by a server which only occupies 2MB of memory instead of the 10-12MB previously, thanks to which the number of requests the system is able to handle simultaneously is greatly increased.

Back-end:

The steps taken in the back-end were aimed mainly at arriving at the necessary configuration to ensure that the DBMS was able to support a very high number of simultaneous connections; in our case, 1,500 connections.

Apart from configuring the PostgreSQL server, we had to alter the values of certain memory management related parameters in the Linux kernel.

Experience has shown that the number of connections to the database tend to stay at quite a low level even when the system receives a very high number of simultaneous requests. If the server has sufficient memory and CPU, requests tend to be served very quickly.

However, external factors may mean that connections last longer and, therefore, the number of simultaneous requests may rise very rapidly under high load conditions. This is why it is important for the system to be able to absorb a great many requests simultaneously. In any event, the regular and sustained appearance of a very high number of connections to PostgreSQL is a sure sign that the system has a problem of some sort (whether due to hardware, slowness of the network, a missing index in a table, costly queries, malfunction of a Moodle module, etc.).

5 Conclusions

5.1 Validity of the Platform and Current Usage Figures

The most obvious conclusion to be drawn now that the improvement measures are in place and all the performance tests have been conducted is that now we can be sure that the new Moodle platform has sufficient capacity to handle the expected load once nearly all the students and subjects have been incorporated into the digital campus.

In fact, the figures obtained during the first semester that the system has been in use provide clear confirmation of this conclusion on a day-to-day basis. The Table 2 summarizes this data:

These figures represent average values recorded during 2007 and demonstrate the undeniable success of the platform. As we can see, a third of the total number of students at UPC use the digital campus platform every day, some of them more than once. And in the course of a month, practically all the students accessed the digital campus on at least one occasion.

In fact, on weekdays the number of Moodle sessions established is systematically close to 1,000 during practically the entire time period between 10h and 22h. At times of intensive use of the platform (at the end of semesters) there were sustained peaks in the system of over 2,000 simultaneous users that had hardly any impact on server load or speed of response.

	Different users	Number of logins	Activity log
Daily	12,000	32,000	320,000
Weekly	20,000	140,000	1,200,000
Monthly	25,000	400,000	4,500,000

Table 2: Average Current Usage of UPC's Moodle Platform, 2007.

The column "Activity log" refers to logins contained in Moodle's log table. In this table Moodle keeps a record activities performed by users of the system. This is much appreciated by the teachers as it allows them to know what material has been of most interest to their students, the most commonly performed exercises, which sections of a subject have been visited most, etc. The number of entries in this log table is an infallible indicator of users' level of activity regarding the tool.

If we look carefully at the table we can see that, given these daily access figures, the complexity and length of today's "real" sessions differ to a certain extent from (i.e. are lower than) those used in the model employed to test the system. It is to be expected that once the platform is extended to serve a much wider public and its use becomes more generalized, usage patterns will vary substantially. A greater frequency of access means sharing the workload between the various sessions and involves both a reduction in the complexity of the tasks performed in each session and a shorter duration of those sessions. In any event, these reductions are offset by the overall increase in the number of requests and, under these new conditions, the system should still ensure (as in fact is the case) sufficient capacity to handle the high demand for connections to the system.

5.2 Other Conclusions: Key Success Factors

In addition to conclusions regarding whether or not the chosen platform is valid, there a series of results (lessons rather) that may be drawn from this project which, while not so obvious as the above conclusions, may be of more or less importance to the table.

Usage profiles: a key success factor. If we wish to be sure of success, the most important factor when conducting performance tests is to arrive at usage profiles that are as close as possible to reality (or at least to the reality that we wish to reproduce). While we need to be aware that these are laboratory tests, it is important to reproduce the scenario in as realistic a manner as possible.

Not all Moodle modules have the same impact on the performance of the system and therefore it is important to give each component a similar weighting in the tests as they have in real daily use.

Changes to the platform or software require new tests. A performance test conducted on our platform today is only valid provided that there are no changes to the system's hardware or software. Adding processors or memory, or increasing the speed of the network will have an impact (mostly positive) on performance. But it is very difficult to quantify this impact a priori. Changes to software, whether due to using a different version of Moodle, using new modules, using a different version of the Web server, the database server, or any other basic software component, are delicate and their impact on the ultimate behaviour of the digital campus is, in many cases, an unknown quantity. For this reason any change in the system requires new tests so that we can be sure that the platform will continue to behave as we expect it to. However, it will not always be necessary to embark on a massive project to carry out these checks. Depending on the importance of the changes we may choose not to perform any checks, or we may decide to carry out some relatively simple tests merely to compare the performance of the versions we are familiar with to that of the new versions.

Need for a stable test environment. To ensure that this project enjoys a certain degree of continuity we need to have a stable test environment. Ideally this would involve having an exact replica of the operating environment, but that is not always so easy to achieve. However, such a replica is not in fact essential, although it is important that the two environments are easily comparable (e.g. do not set up a test environment based on a server with an integrated front-end and back-end if the two components are separate in the actual operating environment). This will allow us to extrapolate the results obtained in the tests and minimize the need to resort to real environment, the availability of which tends to be very limited for such purposes.

References

K. Douglas. "PostgreSQL" (2nd edition).
R. Bowen, K. Coar. "Apache Cookbook: Solutions and examples for Apache administrators".
B. Laurie, P. Laurie. "Apache, the definitive guide" (2nd edition).

Online resources

PostgreSQL: documentation. <http://www.postgresql. org/docs>.
The Linux Virtual Server Project: <http://www.linux.virtualserver.org>.
Wikipedia. <http://www.wikipedia.org>.
Apache JMeter. <http://jakarta.apache.org/jmeter/>.
Apache performance tuning. <http://httpd.apache.org/docs/2.0/misc/perf-tuning.html>.
Using Moodle: Hardware & performance. <http://moodle.org/mod/forum/view.php?id=596>.
Sakai project . <http://sakaiproject.org/>.

