# A Quantitative Justification to Dynamic Partial Replication of Web Contents through an Agent Architecture

Enrique Torres[1], José Daniel García[2], Oscar Sanjuan[3], Luis Joyanes[4] and Rubén González Crespo[3]

[1]*Madrid Council IT Deparment, Madrid, Spain*
[2]*Universidad Carlos III de Madrid, Leganés, Madrid, Spain,*
[3]*Universidad Internacional de La Rioja (UNIR), La Rioja, Spain,*
[4]*Universidad Pontificia de Salamanca, Madrid, Spain*

***Abstract* — The most usual solution to improve the performance of a Web server is based on building a distributed architecture, where the Web server is offered from a set of nodes. The most widely distributed architecture is based on Web clusters including a Web switch. The Web switch is responsible for deciding which site's node must attend which request. When deciding where elements are stored the classical solution was to fully replicate all contents in every server node. However, partial replication may require a fraction of storage while offering the same level of reliability. In this paper we report a solution based on dynamic partial replication where the number of replicas for each file and its management is handled by an agent architecture. We compare our solution with full replication and with static partial replication both in terms of storage capacity consumption and service time. Our results show that our proposed solution provides equivalent performance with a better use of disk storage capacity.**

***Keywords* — Distributed computing, Computer network performance, Network servers, Cooperative systems.**

## I. Introduction

A Web site typically consists of a set of elements or resources, where each of them can be of a certain type (HTML page, image, video, file download, music, etc.). A page consists of a primary element that refers to a series of secondary elements (included in the page).

The Web site receives requests from clients, where each request for a web page typically consists of multiple requests to the server, one for each object included in that page. The client establishes a connection to the server for each of these requests, and receives the response for this connection [1], although some optimizations are possible to reduce the number of connections.

The most usual solution to improve the performance of a Web server is based on building a distributed architecture, where the Web service is offered from a set of nodes [2], acting as a logical single server and consequently giving a single server image.

There are a variety of solutions allowing the construction of distributed Web servers [3]. Those solutions are based either on total replication of contents (all contents are replicated in every server nodes) or on the distribution of contents (contents are distributed so that each element is in a single server node).

Between these two alternatives (total replication and total distribution), a third alternative can be found, namely partial replication, where a certain number of copies is performed for each element. This third alternative [4] [5] needs to determine a priori the number of copies required, which does not always respond to the real needs of the system or to the needs evolution over time.

An evolution of this third alternative dynamically adapts the number of stored replicas according to actual needs and modifies this number when change is needed [6].

This paper presents a quantitative evaluation performed on a prototype that follows this model in which the access time and required storage space is quantified.

The rest of this paper is structured as follows: Section II describes the architecture used in the distributed web server, Section III explains algorithms developed to allow dynamic replication, a prototype is presented in Section IV, results from evaluation are shown in Section V, finally conclusions are drawn in Section VI.

## II. A Dynamic Partial Replication Architecture Implementation by Agents

The most widely distributed Web system architecture is based on clusters [1]. In this architecture (Fig. 1) a distribution node between clients and servers is used: the Web switch. The Web switch is receiving all requests to the visible IP address through a request distribution algorithm, and decides which server node should process which request [3]

The use of an additional service network (Fig. 2) is a modification from the base architecture of a Web-based system cluster that can improve performance of adaptive allocation of contents, for the redistribution of those contents without affecting the main network of the cluster [7].


DOI: 10.9781/ijimai.2015.3311

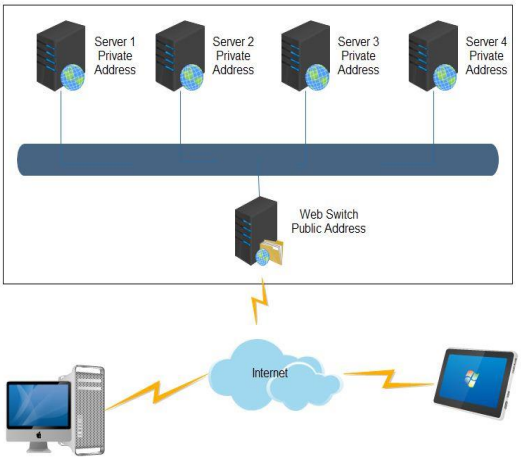



Fig. 1. Cluster Web based General Architecture

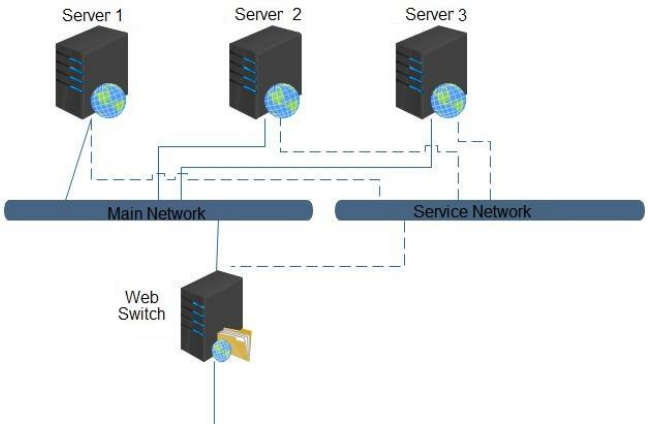



Fig. 2. Cluster Web based General Architecture with an Additional Service Network

Development through intelligent agents simplifies decision making for each component, as each agent behaves in an autonomous way and interacts with the rest of agents [8].

The needed implementation to develop a Web site with partial replication needs to perform activities in the Web switch as well as in server nodes.

- The Web switch must contain all the necessary logic to allocate requests to servers (Request Distributor Agent), and to account them in order to modify the popularity and to determine whether the popularity change implies a change in replication (Replica Control Agent). This machine must also contain the module allowing the administrator to add or remove items (External Control Agent). The Web switch scheme is presented in Fig. 3.
- The server node must include the required logic to obtain an item when the Web switch decides that there should be an additional replica on that node making a request to other server node (Content Control Agent), and the component performing requests resolution by accessing disk (Disk Access Agent). The server node scheme is presented in Fig. 4.

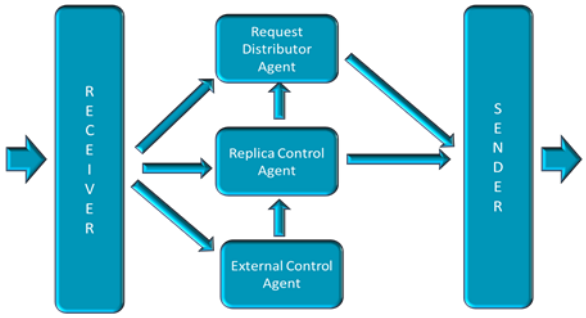



Fig. 3. Agents in a Web switch

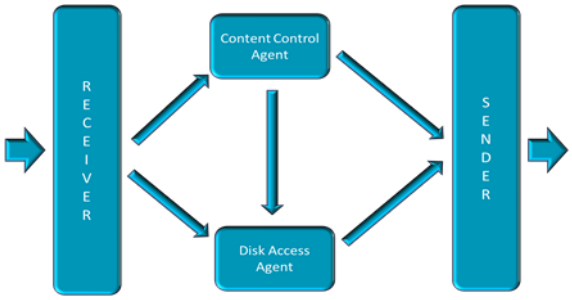



Fig. 4. Agents in a Server Node

To build the previous model, it is necessary to specify the algorithm used by each of the components, on the understanding that the receiver and transmitter modules only transfer requests reaching each system

## III. Dynamic Replication Algorithms

Having established the architecture to be used, it is also necessary to explain the proposed dynamic replication algorithm, which will be placed in the above mentioned replica module. This algorithm tries to dynamically optimize the number of copies depending on the number of requests it receives to each file. Consequently, it will increase the number of replicas of those files which are in high demand to meet actual requests while it will reduce the number of copies of those files which are in low demand in order to free up storage space in nodes [9].

The dynamic replication faces three issues: number of copies for each file, choosing the nodes where the copies are stored, and when the algorithm is executed.

- **Number of copies**: The algorithm assigns all files once, causing the free storage of nodes to decrease by a certain percentage. This percentage decreases the probability of files to be requested and, while this value is greater than 0, the system includes a copy of that file. This will give the total number of copies of file, so we compare this number with the number of copies already in the system and add or remove copies depending on whether the new number is higher or lower [10]. The algorithm for obtaining the number of copies is shown in Algorithm 1.

```
assign TRUE to insert
while insert is equal to TRUE
    assign FALSE to insert
    for all i = 0 to FileNumber
        if FileProbability i > 0
            decrease size i of FreeSpace
            increase CopyNumber
            assign TRUE to insert
    for all i = 0 to FileNumber
        decrease (1-FreeSpace)/TotalSpace
            of FileProbability i
```

Algorithm 1: Obtaining the number of copies.

- **Storage node**: When assigning copies of files to different nodes, it is important to note that nodes in the proposed system have a finite storage space and this aspect must be taken into account [11]. In addition, cloned nodes should be avoided as much as possible, and an appropriate algorithm should be used. A widely used solution addressing the same problem [7] is the backpack greedy algorithm with prior sorting of replicas by size.
- **Activation time**: The system should consider to increase the number of replicas of a file when quality of service is compromised [6]. That is, when the response time of a file exceeds a preset time, the situation is called timing failure. The maximum response time for each file and the probability that the response time is met will be established by the administrator. The latter value is considered because it is admissible that a small percentage of timing failures does not compromise the quality of service and therefore while failures are below the threshold the number of copies will not be increased. When the request ratio that meets the established response time is below the established probability, the algorithm is activated to optimize the number of copies of the files. Moreover, every time a new file is added, the number of copies must be recomputed for all the files, using the probability given by the Pareto distribution for the new file.

## IV. Model Built

To quickly create a model to perform the evaluation, we decided to build a simulation of the real situation.

To build model we used the OMNeT ++ with the INET framework. This tool has already been used to create a large number of projects, such as developing a full suite of TCP/IP at the Karlsruhe University [12], a framework for computer architectures simulation [13], a file storage model for distributed systems [14], a simulation model for IEEE 802.15.4 [15], or make a performance analysis of a handover level 2 in IPv6 mobile networks [16] among many others.

Three VLANs are required for this model, the first routes the client requests to the Web Switch, the second links the Web Switch with the server nodes, and the third one is used as internal service network. A simplified diagram is shown in Fig. 5.

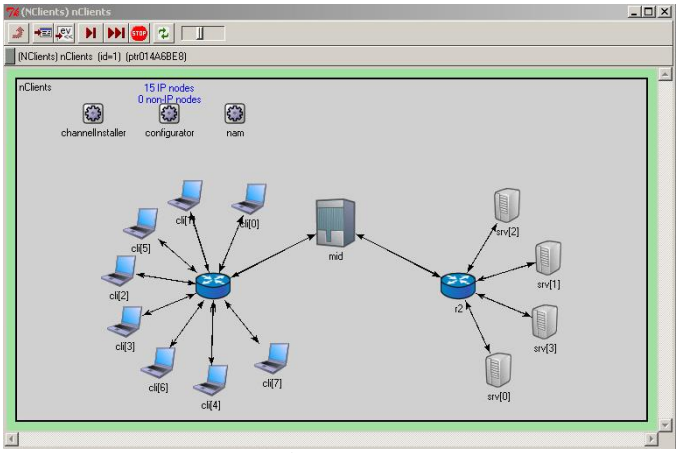



Fig. 5. Model Architecture simplified to 8 clients

Behavioral patterns of a web server based on experimental data were presented in several papers [17] [18]. In this paper, an adaptation of this model has been made to the case of a Web cluster.

A Web site consists of a set of elements or resources, where each may be of a certain type (HTML page, image, video, file download, music, etc.). Set E (see Equation 1) can be defined as the set of all elements that make up the Web site.

$$E = \{e_1, e_2, \ldots, e_N\} \quad (1)$$

Each element of set E can be a primary element or a secondary element. Thus, the E set can be expressed as the union of set $E_p$, primary elements, and set $E_s$, secondary elements (see Equation 2).

$$\begin{aligned} E &= E^p \cup E^s \\ E^p &= \{e_1^p, e_2^p, \ldots, e_n^p\} \\ E^s &= \{e_1^s, e_2^s, \ldots, e_m^s\} \end{aligned} \quad (2)$$

The number of secondary elements included per each primary element can be modeled by a Pareto distribution [19] (see Equation 3).

$$f(x) = \frac{\alpha k^\alpha}{x^{\alpha+1}} x \geq k \quad (3)$$

A web page can be defined as a pair consisting of a primary element and a set of secondary elements, and this allows define the W set as all the site's web pages (see Equation 4).

$$\begin{aligned} w_i &= \{e_i^p, e_{i1}^s, \ldots, e_{in}^s\} \\ W &= \{w_1, w_2, \ldots, W_m\} \end{aligned} \quad (4)$$

The size of the elements is modeled [19] by a lognormal distribution (see Equation 5).

$$f(x) = \frac{1}{x\sqrt{2\pi\sigma^2}} e^{-\frac{(\ln x - \mu)^2}{2\sigma^2}} \quad (5)$$

Experimental studies have established values of α = 2.43 and k = 1 to the number of secondary elements [19], as well as values μ = 9.537 and σ = 1.318 for size of primary elements and μ = 8.215 and σ = 1, 46 for the size of secondary elements [20] [21].

Activity of a client is determined by a sequence of sessions and downtime between sessions (*Downtime*) as show in Fig. 6.

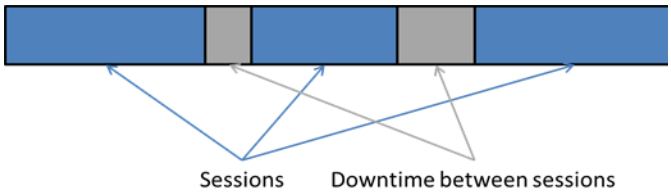



Fig. 6. Customer's activity along the time

*Downtime* can be modeled by a Pareto distribution [19]. Experimental studies [22] use values α = 1.4 and k = 20.

During a session, a client visits a set of Web pages (*Requests per session*), starting with the entry page to the website. Before moving to the next page, it evaluates the contents of the current page for a given time (*Inactivity time*) as shown in Fig. 7.

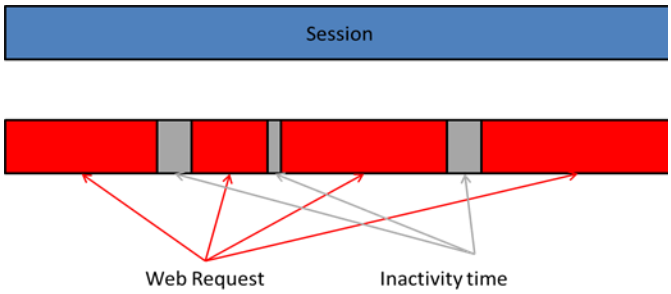



Fig. 7. Session activity

The *Requests per session* can be modeled quite accurately by an inverse Gaussian distribution [23] (see Equation 6) with experimental values of μ = 3.86 and λ = 9.46 [20] [21].

$$f(x) = \sqrt{\frac{\lambda}{2\pi x^3}} e^{-\frac{\lambda(x-\mu)^2}{2\mu^2 x}} \quad (6)$$

Several studies have modeled the downtime by a Pareto distribution with experimental values k = 1 and α = 1.4 [20] [21].

After receiving the response to the primary element, the client scans the content (*Scan time*) before generating multiple request and multiple connections, up to the maximum indicated by the *Degree of concurrency*, one for each secondary file.

The *Scan time* is modeled by a lognormal distribution with experimental results of μ = 360.4 and σ = 106.5 [18].

The *Degree of concurrency* has been modeled as a constant which sets the number of connections.

In the server the different elements involved in an file access [23] are included to compute the access time to each of the stored files. The file requests arrive to the File Manager that passes it to the I/O Manager.

The File Manager receives file requests to be read/write and queries the File Table. In this article was used a UNIX-like file system structure [24], with some simplifications that do not affect the access time. First the Index Node Table is looked up to obtain the disk blocks that must be accessed and if the file uses indexing blocks, Fig. 8, the system keeps the access order to know the addresses of blocks.

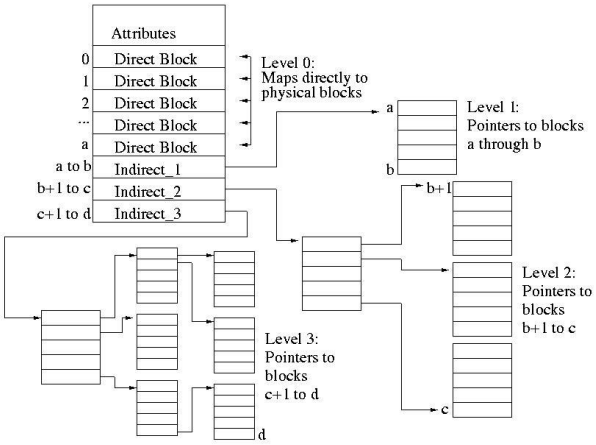



Fig. 8. Index Node Table in UNIX

A call to the I/O Manager is made for each block, which computes the physical address, scheduling disk accesses according to these addresses and computes access time based on the current position of the head and the physical parameters of each of the disks [24]. It is possible to set both the rotational speed, the speed of movement of the heads and the transfer rate.

Most of the time used in the resolution of an HTTP request corresponds to the operation of disk access for the file associated with the request, which depends largely on the size of the file.

## V. Evaluation Results

We compared different options for content distribution, in order to evaluate whether replica allocation policy affects performance.

In all alternatives it has been simulated the behavior of 200 clients making requests on a web cluster system for 7 days of simulated time.

The alternatives evaluated were:

| | |
|---|---|
| TREP | Total Replication |
| SPREP | Static Partial Replication |
| DPREP | Dynamic Partial Replication |

Iteration over the servers for each of the files is used as requests allocation policy. First server that contains the file requested is selected both **SPREP** and **DPREP**.

The first performance metric used is the *HTTP Request Service Time*, which corresponds to the time between the time

when the client sends a request for a file and the time when it receives the response. Table 1 shows the average results of the three alternatives when the number of servers is increased.

TABLE I
SIMULATION RESULTS FOR THE AVERAGE SERVICE TIME OF A WEB SITE WHEN THE NUMBER OF SERVERS IS INCREASED

| Number of servers | TREP | SPREP | DPREP |
|---|---|---|---|
| 4 servers | 1,82601405 | 1,33961388 | 1,34021504 |
| 5 servers | 1,57350203 | 1,59601244 | 1,58789328 |
| 6 servers | 1,62635074 | 1,76683025 | 1,75924017 |
| 7 servers | 1,45933568 | 1,53470846 | 1,50389218 |
| 8 servers | 2,01486739 | 1,99620641 | 1,98940237 |
| 9 servers | 1,28018552 | 1,24920451 | 1,31952067 |
| 10 servers | 1,4591151 | 1,4672851 | 1,4619481 |

To determine whether the difference in Average Service Time of a Web Site is significant, an analysis of variance test has been performed, with the results shown in Table 2. The test was performed for a value of α = 0, 05

TABLE II
VARIANCE TEST RESULTS FOR THE AVERAGE SERVICE TIME OF A WEB SITE WITH A=0,05

| | |
|---|---|
| F | 0,0110429 |
| F Critical Value | 3,55455714 |
| Probability | 0,98902453 |

Fig. 9 shows graphically the above average values. It can be easily seen that the Average Service Times are similar.


Fig. 9. Average Service Time of a Web Site

Another metric to evaluate the evolution of the storage space is the number of files stored in each server. The average results were shown in Table 3.

TABLE III
FILES' COPIES STORED IN EACH SERVER

| Number of servers | TREP | SPREP | DPREP |
|---|---|---|---|
| 4 servers | 10400 | 8361,00 | 8157,00 |
| 5 servers | 10400 | 7953,20 | 7684,60 |
| 6 servers | 10400 | 7681,33 | 6427,00 |
| 7 servers | 10400 | 7683,43 | 5818,90 |
| 8 servers | 10400 | 7683,52 | 5853,50 |
| 9 servers | 10400 | 7683,52 | 5859,85 |
| 10 servers | 10400 | 7614,88 | 5876,13 |

If we consider the distribution of files is the same for primary and secondary files, and according to several studies [20] [21] it is possible to calculate the number of secondary files associated to a primary as it follows a Pareto distribution with α = 2, 43 k = 1, (see Equation 7).

$$E(X) = \frac{\alpha\, k}{\alpha - 1} = 1{,}6993 \tag{7}$$

We can deduce that the function that determines the number of primary and secondary files based on the total number of files stored on a server is given by the expressions in Equation 8.

$$primaries\ (n) = \frac{n}{2{,}69} \qquad secondaries(n) = \frac{1{,}69\, n}{2{,}69} \tag{8}$$

And since the distribution following the primary and secondary files [17] is known [18], we can compute the average of these values (see Equation 9).

$$\begin{aligned} E_{prim}(X) &= e^{7{,}63 + \frac{1{,}001^2}{2}} \\ &= 3398{,}1977 \\ E_{secon}(X) &= e^{8{,}215 + \frac{1{,}46^2}{2}} \\ &= 10730{,}0125 \end{aligned} \tag{9}$$

Using the number of files expressed in Table 3 we can compute the average storage required in each server, as show in Table 4.

TABLE IV
AVERAGE STORAGE NEEDED IN EACH SERVER

| Number of servers | TREP | SPREP | DPREP |
|---|---|---|---|
| 4 servers | 79,39 GB | 63,82 GB | 62,27 GB |
| 5 servers | 79,39 GB | 60,71 GB | 58,66 GB |
| 6 servers | 79,39 GB | 58,63 GB | 49,06 GB |
| 7 servers | 79,39 GB | 58,65 GB | 44,42 GB |
| 8 servers | 79,39 GB | 58,65 GB | 44,68 GB |
| 9 servers | 79,39 GB | 58,65 GB | 44,73 GB |
| 10 servers | 79,39 GB | 58,13 GB | 44,85 GB |

Fig. 10 shows graphically the average storage needed in each server in the different evaluations performed.

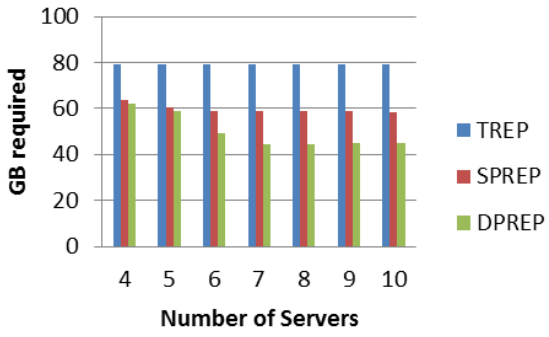



Fig. 10. Average Storage needed in each server

If we also consider the number of servers in each of the simulations, it is possible to compute the average total number of files stored in the Web site, as shown in Table 5.

TABLE V
AVERAGE TOTAL NUMBER OF FILES IN THE WEB SITE

| Number of servers | TREP | SPREP | DPREP |
|---|---|---|---|
| 4 servers | 41600 | 33444 | 32628 |
| 5 servers | 52000 | 39766 | 38423 |
| 6 servers | 62400 | 46088 | 38562 |
| 7 servers | 72800 | 53784 | 40732 |
| 8 servers | 83200 | 61468 | 46828 |
| 9 servers | 93600 | 69152 | 52739 |
| 10 servers | 104000 | 76149 | 58761 |

Allowing us to establish the percentage of space needed in the different solutions versus TREP, as shown in Table 6.

TABLE VI
USAGE RATE VERSUS TREP

| Number of servers | TREP | SPREP | DPREP |
|---|---|---|---|
| 4 servers | 100% | 80,39 % | 78,43 % |
| 5 servers | 100% | 76,47 % | 73,89 % |
| 6 servers | 100% | 73,86 % | 61,80 % |
| 7 servers | 100% | 73,88 % | 55,95 % |
| 8 servers | 100% | 73,88 % | 56,28 % |
| 9 servers | 100% | 73,88 % | 56,34 % |
| 10 servers | 100% | 73,22 % | 56,50 % |

These rates are graphically shown in Figure 11.

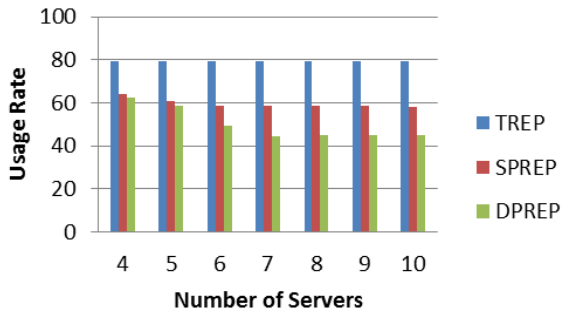



Fig. 11. Usage Rate versus TREP

## VI. CONCLUSIONS

This paper presented a web cluster architecture allowing partial replication of website contents with dynamic adaptation of the number of replicas.

A prototype was developed to test the architecture. The proposed structure is intended to provide fault tolerance, simplicity and distribution taking advantage of the use of an agent architecture.

The results of this prototype are presented in this article, evaluating both performance and storage capacity.

We have seen no evidence to reject the hypothesis that the Request Service Time does not depend on content allocation policy. Even if we admit that there are differences, they never exceed 0.03%.

It has been shown that the necessary storage volume is much greater in the case of full replication that in the other cases. Dynamic partial replication is placed at the other extreme, and storage capacity required is the least of all strategies studied.

## REFERENCES


[1] T. Berners-Lee, R. Fielding and H. Frystyk, "Hypertext Transfer Protocol - HTTP/1.0. RFC 1945", Internet Engineering Task Force, Mayo 1996.

[2] V. Cardellini, E. Casalicchio, M. Colajanni, and P.S. Yu, "The state of the art in locally distributed Web-server systems", *ACM Computing Surveys*, vol. 34(2):263–311, 2002.

[3] T. Schroeder, S. Goddard and B. Ramamurthy, "Scalable web server clustering technologies", *IEEE Network*, vol. 14(3), pp. 38–45, Mayo 2000.

[4] García, J.D., Carretero, J., García, F., Singh, D.E. y Fernández, J.: "A Highly Available Cluster of Web Servers with Increased Storage Capacity", *XVII Jornadas de Paralelismo*, pp. 109-114, Albacete, Septiembre 2006.

[5] Garcia, J.D., Carretero, J., Garcia, J., Sánchez, L.M. y Garcia, F.: "A Web Cluster Architectural Proposal for Balancing Storage Capacity and Reliability by using Partial Replication", *International Journal of Computer Systems Science and Engineering*, vol. 28(3):191-202, CRL Publishing Ltd, Reino Unido, Mayo 2013.

[6] Torres, E., Sanjuan, O., Joyanes, L., García, J.D. y González, R.: An Architecture For Management Of Distributed And Redundant Web Storage With Intelligent Agent Systems And Emerging Techniques. En IEEE Latin America Transactions, vol. 6(6):524-528, IEEE Xplore, Octubre 2008.

[7] J.D. Garcia, "Propuestas Arquitectónicas para servidores Web distribuidos con réplicas parciales", Ph.D. dissertation, Computer Science Dept., Universidad Carlos III de Madrid, Junio 2005.

[8] Gutierrez, C.: "An Analysis Architecture for Communications in Multi-agent Systems", *International Journal of Artificial Intelligence and Interactive Multimedia (IJIMAI)*, vol 2(1):65-72, 2013.

[9] Torres, E., Sanjuan, O., Joyanes, L., García, J.D. y Pelayo B.C.: "A Multi-Agent based Proposal for the Management of Distributed and Redundant Storage". *Proceedings of the 2008 International Conference on Artificial Intelligence (ICAI 2008)*, pp. 566-571, Las Vegas, Nevada, USA, CSREA Press, Julio 2008.

[10] Torres, E., Sanjuan, O., Joyanes, L., García, J.D., González, R. y Ríos, S.: "Management of Distributed and Redundant Storage in High Demand Web Servers for Heterogeneous Networks Access by Agents", *International Symposium on Distributed Computing and Artificial Intelligence (DCAI 2008)*, vol. 50, pp. 123-131, Advances in Soft Computing, Springer, Enero 2009.

[11] Torres, E., Sanjuan, O., Joyanes, L., García, J.D. y González, R.: "Arquitectura Para La Gestión De Almacenamiento Web Distribuido Y Redundante Mediante Sistemas De Agentes Inteligentes Y Técnicas Emergentes". *6th International workshop on practical applications on agents and multi-agent systems (IWPAAMS2007)*, pp. 319-328, Salamanca, Noviembre 2007.

[12] Kaage, U., Kahmann V., Jondral F: "An OMNeT++ TCP Model", *Proceedings of the 15th European Simulation Multiconference (ESM 2001)*, Praga, Junio 2001.

[13] Núñez, A., Fernández, J., Carretero, J., García, J.D. y Prada, L.: "SIMCAN: A SIMulator Framework for Computer Architectures and Networks!, *First International Workshop on OMNeT++*, pp. 8, Marsella, Francia, Marzo 2008.

[14] Núñez, A., Fernández, J., Carretero, J., García, J.D. y Prada, L.: "New Techniques for Modelling File Data Distribution on Storage Nodes", *41st Annual Simulation Symposium*, pp. 175–182, Ottawa, Canada, Abril 2008.

[15] Kirsche, M. y Schnurbusch, M.: "A new IEEE802.15.4 Simulation Model for OMNeT++/INET", *First OMNeT++ Community Summit*, Hamburg, September 2014.

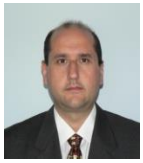
[16] Çetin, G., Çetin, A. y Özkaraca, O.: "The analisys of Layer-2 Handover performance for mobile IPV6 using OMNeT++ Simulation Tool", *Mugla Journal of Science and Technology*, vol. 1(1):34-38, 2015.

[17] Mah, B.A.: "An empirical model of HTTP network traffic", *Proceedings of the Conference on Computer Communications (INFOCOM'97)*, vol. 2, pp. 592-600, Kobe, Japón, 1997.

[18] Choi, H.K. y John O.L.: "A behavioral model of web traffic", *Proceedings Seventh International Conference on Network Protocols (ICNP'99)*, pp. 327-334, Toronto, Ontario, Canada, IEEE, Octubre 1999.

[19] Barford, P. y Crovella, M.: "Generating representative web workloads for network and server performance evaluation", *Performance Evaluation Review*, vol. 26(1):151-160, Junio 1998.

[20] V. Cardellini, M. Colajanni, and P.S. Yu, "Geographic load balancing for scalable distributed Web Systems", *Proceedings of the 8th International Symposium on Modeling, Analisys and Simulation of Computer and Telecommunication Systems (MASCOTS'00)*, pp. 20-27, IEEE, San Francisco, CA, USA, Agosto 2002.
[21] V. Cardellini, "Request redirection algorithms for distributed web systems", *IEEE Transactions on Parallel and Distributed Systems*, vol 14(4), pp. 355-368, Abril 2003.
[22] Wallerich, J.: "Design and implementation of a www workload generator for the ns-2 network simulator", *http://www.net.t-labs.tu-berlin.de/~joerg/nsweb/doku/*, Noviembre 2001. [consulta 1 de Abril de 2015].
[23] Stalings, W.: "Operating Systems: Internals and Design Principles (8th Ediдtion)", Ed. Prentice Hall, 2015.
[24] Carretero, J., Miguel, P., García, F. y Pérez, F.: Sistemas Operativos: una visión aplicada 2ª Edición, Ed. McGraw-Hill 2007.



**D. Enrique Torres** has a degree in Computer Science from the Polytechnic University of Madrid. He has been Assistant Professor at the Faculty of Computer Science from the Pontifical University of Salamanca. Ha has also been an Adjunct Professor at University Carlos III of Madrid. He is currently a systems engineer at the Madrid Council.

**Dr. José Daniel García** has a degree in Computer Science from the Polytechnic University of Madrid and a Computer Science PhD from Universidad Carlos III de Madrid.He is Associate Professor in Computer Science Department from University Carlos III of Madrid, where he previously served as Assistant Professor. Previously he was Lecturer in the Computer Science Faculty from Pontifical University of Salamanca. Dr. García has worked as consultant and software engineer in projects for several international companies like FCC, Siemens, DMR Consulting, Telefónica, or ING Bank. His main research interests including parallel and distributed systems, programming languages and programming models for applications improvement.

**Dr. Oscar Sanjuan** has a degree in Computer Science from the Pontifical University of Salamanca, where he also earned his PhD in Computer Science Engineering, and he is PhD in Computer Science from Oviedo University. He has been Area Director of Software Engineering at the Pontifical University of Salamanca, lecturer and researcher at the University of Oviedo and Assistant Professor at the University Carlos III of Madrid. Currently he is Engineering Director at ElasticBox Inc.

**Dr. Luis Joyanes Aguilar** has a degree in Physics Science from the Complutense University of Madrid and he has a degree in Military Higher Education from General Military Academy of Zaragoza. He earned his PhD in Computer Science from Oviedo University and his PhD in Sociology from Pontifical University of Salamanca. Also, he is Honorary Doctor from the Private University Antenor Orrego of Trujillo in Perú. He is Professor of Languages and Computer Systems from Pontifical University of Salamanca and he is member of Knowledge Management Committee of AEC (Spanish Quality Association) and AENOR (Spanish Agency for Standardization).

**Dr. Rubén González** has a degree in Computer Science from the Pontifical University of Salamanca, where he earned his PhD in Computer Science Engineering. He has been Area Director of Operating Systems at the Pontifical University of Salamanca and Graduate Director from Higher School of Engineering and Architecture from Pontifical University of Salamanca. Currently, he is Director of the Higher School of Engineering from International University of La Rioja and Director of the AENOR Chair in Certification and Quality and Technology Standards.