A Highly Available Cluster of Web Servers with Increased Storage Capacity

José Daniel García, Jesús Carretero, Félix García, David E. Singh and Javier Fernández

Resumen— **Web servers scalability has been traditionally solved by improving software elements or increasing hardware resources of the server machine. Another approach has been the usage of distributed architectures. In such architectures, usually, file allocation strategy has been either full replication or full distribution. In previous works we have showed that partial replication offers a good balance between storage capacity and reliability. It offers much higher storage capacity while reliability may be kept at an equivalent level of that from fully replicated solutions. In this paper we present the architectural details of Web cluster solutions adapted to partial replication. We also show that partial replication does not imply a penalty in performance over classical fully replicated architectures. For evaluation purposes we have used a simulation model under the OMNeT++ framework and we use mean service time as a performance comparison metric.**

Palabras clave— **Web Cluster, Web Switch, Content replication, Partial replication.**

I. Introduction

DUE to increasing development of Web based applications a huge interest has arised in building hardware and software for this kind of systems in such a way that solutions are scalable both in performance and storage capacity. Traditional solutions to this problem include software and hardware scale up [1], [2], [3]. A recent architectural alternative is the distributed Web server [4] where a set of server nodes are used to host a Web site. In those systems, performance scalability is achieved by adding new server nodes. In a distributed Web server, requests must be distributed among the server nodes.

Distributed Web server architectures may be classified in three families.

- *Cluster based Web systems* Cluster nodes own IP addresses are not visible from the clients. Instead, clients use an virtual IP address which corresponds to that of some request distribution device (Web switch). The Web switch [5] receives requests from clients and sends each request to some server node.
- *Virtual Web clusters* All the server nodes share a single common public IP address [6]. Each node receives all messages in a way that every request is discarded by all nodes except one.
- *Distributed Web systems* Each node has its own different publicly visible IP address [7], [8]. Request distribution is made by a combination of dynamic DNS [9] and request redirection.

Usually, solutions are based either on full content replication (every file is replicated in every server node) or full distribution (every file is stored in one and only one server node).

With full replication [10] the system is highly reliable and request distribution is easy to implement, as each request may be served by any server node. On the other hand, storage scalability is minimal, as the full storage capacity is limited by the server node with the lowest capacity. Furthermore, adding new server nodes to the system does not increase storage capacity.

With full distribution the system gives a lower reliability (a node failure makes some content unavailable) and request distribution needs to use some sort of directory service [11] to determine which node is storing the requested file. On the other hand, storage scalability is maximum, as adding a new node means increasing the total amount of available storage.

A third alternative is partial replication [12], [13], [14], where each file is replicated in a possibly different subset of the server nodes set. With partial replication reliability is higher than in the case of full distribution and lower than in the case of full replication. In terms of storage capacity, partially replicated solutions provide less capacity than fully distributed solutions but more than fully replicated solutions. However, an important difference with full replication is that partial replication provides storage capacity scalability (i.e. adding new server nodes increases the system storage capacity) while full replication does not.

This paper is organized as follows: section 2 discusses architectural alternatives for a distributed Web server under the restriction of partial replication of contents; section 3 presents our architecture for a single switched Web cluster; section 4 presents our architecture for a multiple switched Web cluster; section 5 shows details about our simulation model and the obtained results; section 6 summarizes conclusions; section 7 enumerates lines for future work.

II. Architectural Alternatives

Partial replication restricts the architectural alternatives for a distributed Web server. When a Web request arrives into the system, that request cannot be sent to any node in the system, as not every node stores every content. Besides every file is not stored in a single node, but in a set of server nodes. Thus, a mechanism to determine that set of server nodes is needed.

Virtual Web Clusters cannot be easily integrated with the idea of partial replication [15]. In that case, every request reaches to every node of the cluster.

Grupo de Arquitectura de Computadores, Comunicaciones y Sistemas. Univerisdad Carlos III de Madrid. e-mail: josedaniel.garcia@uc3m.es

Nodes not storing the requested content may ignore the request. But if two or more nodes store the requested content it is not simple to determine which node should take the responsibility of answering the request. Furthermore, in most cases communication among cluster nodes is not feasible, and this fact excludes the possibility that a node may notify others when it takes the responsibility of taking in charge of a request and its response.

In a *Distributed Web System* any request may reach any node, but it is guaranteed that one request reaches to one and only one node. However, there is no guarantee that the node receiving the request contains a replica of the requested node. In that case the only solution is that the node receiving the request performs a redirection to another node which effectively contains the requested resource. This fact introduces additional complexities in the load balancing mechanisms.

Although it is not impossible to integrate the partial replication strategy into a *Virtual Web Cluster* or into a *Distributed Web System* we consider that such strategy can be more easily integrated into a *Cluster based Web System* (or *Web Cluster*). The main difference of a *Web Cluster* with other architectures is that it has a single point where every request arrives, namely the *Web Swtich*. In case of partial replication the Web Switch must be a content aware one [4]. That is, the switch must operate at level 7 of the protocol stack and it must parse each request before deciding to which server node that request will be routed. In a previous work [16], we showed that a Web Cluster with partial replication using a small amount of replicas per file offers a reliability equivalent to that of a fully replicated system and, at the same time, it offers a much higher storage capacity. However, reliability is negatively affected by the fact of using a single *Web Swtich* and thus having a single point of failure. This flaw is mitigated by using more than one Web Switch (2 or three are enough).

III. Single Switched Web Cluster Architecture

In a single switched Web Cluster, every request arrives into the Web Switch. The Web Switch is responsible for selection of the node which must serve the request (through a dispatching algorithm). The Web Switch is also responsible for sending the request to the selected node (through a request routing algorithm). Existing solutions do not apply because they usually assume either that every file is replicated in every node (full replication of contents) or that every file is stored in one and only node (full distribution of contents). However, general architecture is very similar to that of a standard Web Cluster (see fig. 1) where every server node has two network interfaces: one connected to the Web Switch network, and another one connected to high bandwidth output link.

In the case of partial replication the dispatching algorithm is dependent of the exact set of nodes effectively containing the requested file. Thus, the *Web Switch* must have the knowledge of the allocation of each file to nodes (the set of nodes where it resides). It is for that reason that content blind routing is not applicable. As the dispatching algorithm must use the allocation information, some sort of *directory service* is needed to keep such information. Although a generic directory service could lead to excessive delays, it is possible to build low demanding resources data structures with lower delays [11]. As an example, Luo et al. [17] implemented a data structure using URL formalization and multilevel hash tables. There solution stored the information related to 76000 files in 540 KB and the time needed for a query was about 1.12 microseconds.

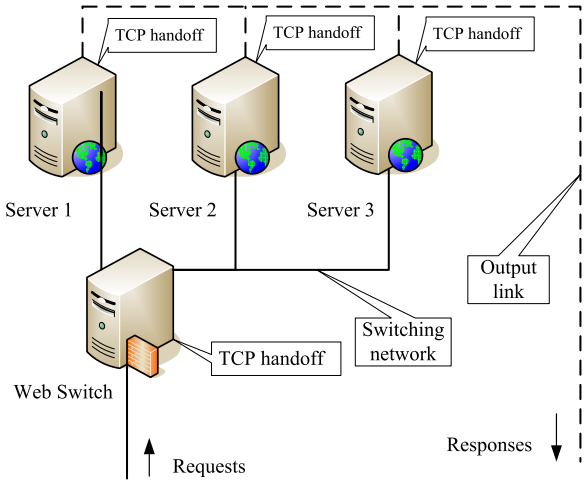

Fig. 1. Architecture of a single switched Web Cluster

Once the dispatching algorithm has selected a server node for a request, it is necessary to use some request routing mechanism to effectively send the request to the selected node. As we have already stated, request routing needs to be content aware. The routing mechanism may be implemented at the application level (e.g. TCP gateway [18]) or at the operating system kernel level (e.g. TCP splicing [19] or TCP handoff [20]).

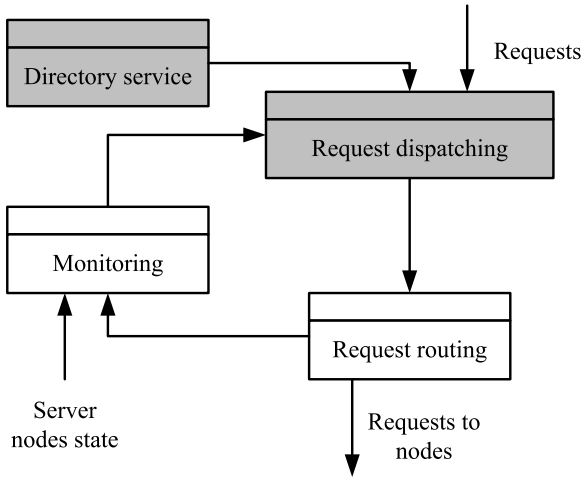

Fig. 2. Internal architecture of Web Switch software for a single switched Web Cluster

Several modules are needed in the Web Switch software, as depicted in fig. 2:

Request dispatching Receives requests and selects the node which must serve each request. The request dispatching algorithm uses replica allocation information (given by *directory service*) and node state information (given by *monitoring* module).

Request Routing Receives dispatched requests and performs request routing to selected server node. We propose the usage of TCP handoff for efficient request routing.

Directory service Efficiently keeps replica allocation to node information.

Monitoring Keeps node state information (provided periodically by each node) which is used by the *request dispatching* module. It also keeps trace of routed requests information (provided by the *request routing* module).

IV. Multiple Switched Web Cluster Architecture

Single switched Web clusters present a reliability drawback. As they have a single switch, that element becomes in a single point of failure. That is, if there is a failure in the Web switch, the full system fails. To avoid this flaw, we propose a new architecture: the multiple switched Web Cluster based on the concept of *distributed Web switch*.

The main idea is the usage of a set of Web switches as front end of the Web cluster. This allows a high increase of the global reliability [16]. To distribute request arrival among the Web switches we use the combination of two mechanisms dynamic DNS and request redirection among switches.

Dynamic DNS [21] has been widely used in conjunction with *Distributed Web Systems* to share requests from clients among a set of publicly available nodes. However this technique is of limited usefulness due to DNS caching as several experiments [22] have showed.

To complement dynamic DNS each Web switch may redirect some requests to another Web switch of the front end. In our system, when a Web switch is highly loaded, it redirects incoming requests to another Web switch. That redirection may be performed by means of standard HTTP redirection.

Fig. 3 shows the flow of a request in multiple switched Cluster. When a client starts a request it first issues a DNS resolution request (1) which is answered by the DNS server (2) with the IP address of a Web switch in the front end of the cluster. The client, then sends a HTTP request to the selected Web switch (3), which may answer with a redirection to another Web switch (4). In that case, the client resends the HTTP request to the newly selected Web switch (5). The Web switch selects a server node through its dispatching algorithm and resends the HTTP request to it (6). At last, the selected node answers the client (7).

For the solution to work properly, each Web switch must monitor its own state and periodically interchange that information with the rest of Web switches. This allows that each switch has enough information to decide when it is able to transfer some requests to another switch to achieve some degree of load balancing at the front end level. It is important to remark that transferring a request to another switch has a lower cost than keeping it in the original switch. That is because transfer operation may be performed in a content blind manner at the TCP level of the protocol stack and may be easily integrated in the operating system kernel. When a switch takes the responsibility of managing a request it selects a server node and it routes the request to that server node.

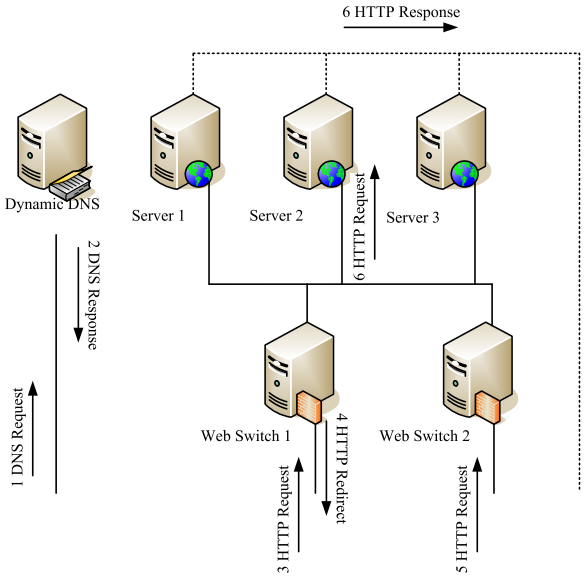

Fig. 3. Request routing in a multiple switched Web cluster

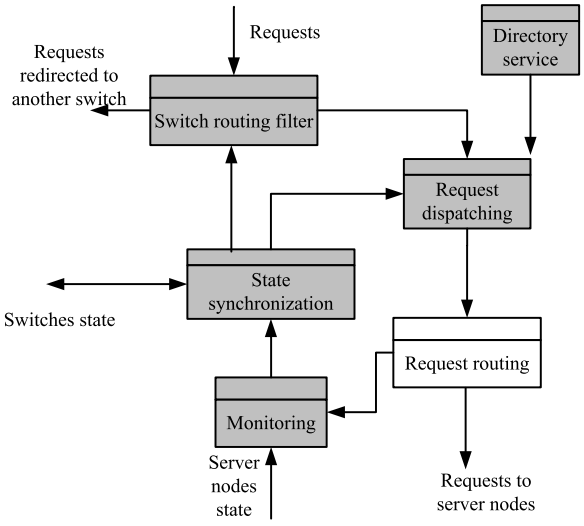

Fig. 4. Internal architecture of Web Switch software for a multiple switched Web Cluster

The structure of the software running in each server node (see fig. 4) is more complex than the one used for a single switched web cluster. Main modules are listed below:

Switch routing filter Decides if an incoming re-

quest may be processed by the current switch or if there is a better switch to serve the request. The decision is based in the load state of every switch. That information is provided by the state synchronization module.

Request dispatching Receives requests not discarded by the switch routing filter and selects the node which must serve each request. The request dispatching algorithm uses replica allocation information (given by directory service) and system state information (given by state synchronization).

Request routing Receives dispatched requests and performs request routing to selected server node. We propose the usage of TCP handoff for efficient request routing.

Directory service Efficiently keeps replica allocation to node information. To reduce communication overheads we place a directory service replicated in every Web Switch of the system.

Monitoring Keeps node state information (provided periodically by each node). It also keeps trace of routed requests information (provided by the request routing module). The information it gathers is provided to the state synchronization module to build an integrated system state view.

State synchronization Periodically notifies its own statte to the rest of Web switches. This allows that every switch has knowledge about the state of other switches. That information is integrated with information about server nodes provided by the monitoring module to help switch routing filter and request dispatching modules to make their own decissions.

V. Evalution

Even if reliability and storage capacity are key issues for a Web server, these goals must be achieved without loss of performance or with a minimum loss. To evaluate performance of our solutions we have built a simulation model using the OMNeT++ 3.0 framework (`http://www.omnetpp.org`).

For our simulations we have used a Web cluster with 16 server nodes. We have set up 800 client machines which are continuously performing Web requests to the cluster. Requests are routed using a one-way strategy (responses from server nodes use a different connection so they do not return through the Web switch). The request dispatching algorithm has been fixed to round robin over the set of server nodes storing each file. Table I shows the main simulation parameters used in the evaluations.

We have performed our evaluations for different replication strategies:

FREP Full replication of contents. Every file is replicated in every node.

R2 Every file is replicated in two nodes.

R4 Every file is replicated in four nodes.

RV Every file is replicated in a variable number of nodes. The number of replicas for a file is selected using popularity of the file and size as critera.

TABLA I
Simulation parameters used in the evaluation.

Parameter	Distribution
Number of included files	Pareto ($\alpha = 2.43, k = 1$)
Main file size (body)	Lognormal ($\mu = 7.63, \sigma = 1.001$)
Main file size (tail)	Pareto ($\alpha = 1, k = 10240$)
Embedded files size	Lognormal ($\mu = 8.215, \sigma = 1.46$)
Inter session time	Pareto ($\alpha = 1.4, k = 20$)
Requests per session	Inverse gaussian ($\mu = 2.86, \lambda = 9.46$)
Inactivity time	Pareto ($\alpha = 1.4, k = 1$)
Parsing time	Weibull ($\alpha = 1.46, \beta = 0.382$)

We have used two test scenarios: a single switched Web cluster (SSWC) and a multiple switched web cluster (MSWC) with three Web switches. For each scenario we performed 35 simulation realizations with different random seeds. Table II shows mean service time and variance for SSWC and Table III shows mean service time and variance for MSWC.

Replication	**Mean Time**	**Variance**
FREP	5.422234027	0.001745808
R2	5.422842417	0.001739406
R4	5.422525610	0.001755285
RV	5.422836699	0.001744325

TABLA II
Mean service time and variance for request processing in a Web cluster for different replication strategies.

Replicacin	**Media**	**Variance**
FREP	5.436501603	0.003001719
R2	5.436536060	0.002974351
R4	5.436385993	0.003007949
RV	5.436293051	0.003026502

TABLA III
Mean service time and variance for request processing in a Web cluster with three Web switches for different replication strategies.

The obtained result show slight overheads in case of partial replication over the case of full replication. To estimate if there is difference in obtained results we have performed an analysis of variance (ANOVA) with $\alpha = 0.05$ over our simulation results (see Table IV.

Thus, there is no evidence that the usage of a partial replication strategy affects negatively performance in our simulation experiments. This is true for both our experiment of a single switched Web cluster and for our experiment using a front end of three Web switches. It is remarkable to note that even if we

Value	SSWC	MSWC
F	0.00169626	0.000144015
F critical value	2.67117796	2.67117796
Probability	0.999903073	0.999997599

TABLA IV
Analysis of variance for simulation results

admitted that performance is not equal differences in performance are below 0.5% which makes our solution very useful as it offers a very good tradeoff between reliability and storage capacity.

VI. Conclusions

In this paper we have presented the advantages of partial replication over full replication of Web contents: higher storage capacity, reliability and minimal or null performance penalty. We have explored the architectural alternatives and the advantages of Web clusters over other architectural families. Starting with the Web cluster architecture, we have defined the modifications needed over a standard Web cluster to be able to cope with the restrictions derived from partial replication of contents.

Our architecture for single switched Web clusters is based on an efficiently implemented directory service capable of determining the server nodes set where the requested file is effectively stored. Request dispatching algorithms need to use that information to route every request to a node which contains the requested file.

In case of multiple switched Web clusters we use several switches as a front end for the system. Request arrival is distributed over the set of switches by a combination of dynamic DNS and HTTP redirection. The software running each Web switch uses a replicated directory service as primary information source for the dispatching algorithm. Besides, switches periodically interchange information to keep their state information synchronized.

Performance evaluations have been conducted using a simulation model and leading to the result that the performance penalty is, if existing, minimal. Thus additional complexities of the solution are justified by the obtained advantages, namely higher storage capacity and a reliability equivalent to that of a fully replicated solution.

VII. Future Work

A key issue in partial replication is the determination of the number of replicas needed for each element and the allocation of replicas to server nodes. We are working to find algorithms which help to determine the replication number for each element so that reliability is not compromised. We are specially interested in the case where storage capacities and performance characteristics of server nodes are heterogeneous, as we see that case of special interest for cluster upgrading.

The number of replicas for each file and its allocation into server nodes should change over time, as load conditions of nodes and preferences of users evolve. Our current solution is of static nature. We plan to define a dynamic replication strategy to evolve as the system situation changes.

VIII. Acknowledgements

This work has been partially supported by the Spanish Ministry of Science and Education under the **TIN2004-02156** contract.

Referencias

[1] Vivek S. Pai, Peter Druschel, and Willy Zwaenepoel, "Flash: An efficient and portable Web server," in *Proceedings of the USENIX 1999 Annual Technical Conference*, Monterey, CA, June 1999, USENIX Association, pp. 199–212.

[2] Vivek S. Pai, Peter Druschel, and Willy Zwaenepoel, "IO-Lite: A unified I/O buffering and caching system," *ACM Transactions on Computer Systems*, vol. 18, no. 1, pp. 37–66, Feb. 2000.

[3] Gaurav Banga, Peter Druschel, and Jeffrey C. Mogul, "Resource containers: A new facility for resource management in server systems," *Operating Systems Review*, vol. Special Issue, Winter 1998, pp. 45–58, 1999.

[4] Valeria Cardellini, Emiliano Casalicchio, Michele Colajanni, and Philip S. Yu, "The state of the art in locally distributed Web-server systems," *ACM Computing Surveys*, vol. 34, no. 2, pp. 263–311, June 2002.

[5] Trevor Schroeder, Steve Goddard, and Byrav Ramamurthy, "Scalable web server clustering technologies," *IEEE Network*, vol. 14, no. 3, pp. 38–45, May 2000.

[6] Sujit Vaidya and Kenneth J. Christensen, "A single system image server cluster using duplicated MAC and IP addresses," in *Proceedings of the 26th Annual IEEE Conference on Local Computer Networks (LCN 2001)*, Tampa, FL, USA, Nov. 2001, IEEE, pp. 206–214.

[7] Luis Aversa and Azer Bestavros, "Load balancing a cluster of web servers: using distributed packet rewriting," in *Conference Proceedings of the 2000 IEEE International Performance, Computing, and Communications Conference (IPCCC 2000)*, Phoenix, AZ, USA, Feb. 2000, IEEE, pp. 24–29.

[8] Valeria Cardellini, "Request redirection algorithms for distributed web systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 14, no. 4, pp. 355–368, Apr. 2003.

[9] T. Brisco, *DNS Support for Load Balancing. RFC 1794*, Internet Engineering Task Force, Apr. 1995.

[10] Bill Devlin, Jim Gray, Bill Laing, and George Spix, "Scalability terminology: Farms, clones, partitions, and packs: Racs and raps," Technical Report MS-TR-99-85, Microsoft Research. Advanced Technology Division, Dec. 1999.

[11] G. Apostolopoulos, D. Aubespin, V. Peris, P. Pradham, and D. Saha, "Design, implementation and performance of a content-based switch," in *Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2000)*, Tel Aviv, Israel, Apr. 2000, vol. 3, pp. 1117–1126, IEEE.

[12] Jos D. Garca, Jess Carretero, Jos M. Prez, Flix Garca, and Javier Fernndez, "A distributed web switch for partially replicated contents," in *Proceedings of the 7th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2003)*, Orlando, FL, USA, July 2003, vol. VIII, pp. 1–6.

[13] Ling Zhuo, Cho-Li Wang, and Francis C. M. Lau, "Document replication and distribution in extensible geographically distributed web servers," *Journal of Parallel and Distributed Computing*, vol. 63, no. 10, pp. 927–944, Oct. 2003.

[14] S. S. H. Tse, "Approximate algorithms for document placement in distributed web servers," *Transactions on Parallel and Distributed Systems*, vol. 16, no. 6, pp. 489–496, June 2005.

[15] Jose Daniel Garcia, Jesus Carretero, Felix Garcia, Javier Fernandez, Alejandro Calderon, and David E. Singh., "A quantitative justification to partial replication of web

contents," in *International Conference on Computational Science and its Applications*. May 2006, vol. 3983 of *Lecture Notes in Computer Science*, pp. 1136–1145, Springer Verlag.

[16] Jose Daniel Garcia, Jesus Carretero, Felix Garcia, Alejandro Calderon, Javier Fernandez, and David E. Singh, "On the reliability ofweb clusters with partial replication of contents," in *First International Conference on Availability, Reliability and Security, 2006. ARES 2006.* IEEE, April 2006, pp. 617– 624.

[17] Mon-Yen Luo, Chun-Wei Tseng, and Chu-Sing Yang, "URL formalization: An efficient technique to speedup content-aware switching," *IEEE Communication Letters*, vol. 6, no. 12, pp. 553–555, Dec. 2002.

[18] Emiliano Casalicchio and Michele Colajanni, "A client-aware dispatching algorithm for web clusters providing multiple services," in *Proceedings of the tenth international conference on World Wide Web*, Hong Kong, May 2001, pp. 535–544, ACM Press.

[19] D. A. Maltz and P. Bhagwat, "TCP splice for application layer proxy performance," *Journal of High Speed Networks*, vol. 8, no. 3, pp. 225–240, June 1999.

[20] Vivek S. Pai, Mohit Aron, Gaurov Banga, Michael Svendsen, Peter Druscheland Willy Zwaenepoel, and Erich Nahum, "Locality-aware request distribution in cluster-based network servers," *ACM SIGPLAN Notices*, vol. 33, no. 11, pp. 205–216, Nov. 1998.

[21] Daniel Andresen, Tao Yang, and Oscar H. Ibarra, "Toward a scalable distributed www server on workstation clusters," *Journal on Parallel and Distributed Computing*, vol. 42, no. 1, pp. 91–100, Apr. 1997.

[22] Michele Colajanni, Philip S. Yu, and Daniel M. Dias, "Analysis of task assignment policies in scalable distributed web-server systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 9, no. 6, pp. 585–600, June 1998.