RC22482 (W0206-034) June 10, 2002
Computer Science

IBM Research Report

Global Caching Mechanisms in Clusters of Web servers

Marco Emilio Poleggi
Dipartimento di Informatica, Sistemi e Produzione
Universita di Roma “Tor Vergata”
Via de1 Politecnico 1, 00133 Roma - Italy

Bruno Ciciani
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Michele Colajanni’
Dipartimento di Ingegneria dell’Informazione
Universita. di Modena
Via Vignolese 905,411OO Modena - Italy

Research Division
Almaden - Austin - Beijing - Delhi - Haifa - India - T. J. Watson - Tokyo - Zurich

Global Caching Mechanisms in Clusters of Web Servers

Marco Emilio Poleggi
Dipartimento di Informatica, Sistemi e Produzione
Università di Roma "Tor Vergata"
Via del Politecnico 1, 00133 Roma - Italy
Tel: +39 06 72597336, Fax: +39 06 72597460
E-mail: poleggi@ing.uniroma2.it

Bruno Ciciani
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598, U.S.A.
E-mail: ciciani@us.ibm.com

Michele Colajanni*
Dipartimento di Ingegneria dell'Informazione
Università di Modena
Via Vignolese 905, 41100 Modena - Italy
Tel: +39 059 2056137, Fax: +39 059 2056129
E-mail: colajanni@unimo.it

Abstract

We investigate how to improve Web cluster performance through cooperation protocols for a shared use of the main memories of the server nodes (*global caching*). Many popular schemes to locate requested documents are based upon locality information sharing (namely, *Informed* cooperation); we also propose a novel approach for document lookup through a query broadcasting mechanism (namely, *On-demand* cooperation). Moreover, we consider the *Migration* and the *Handoff* mechanisms to retrieve successfully located documents. The four cooperation schemes have been evaluated through synthetic workload driven simulations. The experimental results show that, in a static Web scenario, the On-demand with Handoff scheme performs similarly to the Informed with Handoff scheme, while the On-demand with Migration scheme outperforms the Informed with Migration scheme. On the other hand, in a scenario with some dynamic requests, the On-demand cooperation experiences a significant performance degradation with respect to the Informed cooperation.

Keywords: Cluster-based Web systems, Resource management, Performance evaluation, Caching, Cooperation algorithms.

* **Corresponding Author**

1 Introduction

Locally distributed HTTP servers, often called *Web clusters*, are the main solution to face the ever increasing demand of Web-based services at low costs. Nevertheless, the aggregated power of many single machines, without any global resource management, often is not sufficient to stand peak loads of thousands of requests per second because the potentials of the cluster resources are not exploited at their best. As a consequence, under particularly heavy loads, the saturation of some servers can cause the Web cluster capacity to be far from the expected aggregate capabilities of the system. Nowadays, a single disk can hold a medium sized Web site, and the aggregated RAM of the server nodes might hold the entire static content of the Web site. Moreover, a commercial LAN, such as Fast-Ethernet or Gigabit-Ethernet, can convey much more traffic than that received by the Web cluster from the Internet. Different solutions are feasible for the cooperation among the server nodes that aim at better exploiting the aggregated RAM and network bandwidth of the Web cluster. To the best of our knowledge, commercial Web clusters do not use cooperative caching policies, whereas some examples exist in the research community [7, 10, 11, 8, 14].

The key idea of *global caching* is to rely on a distributed RAM storage for the Web documents, to be accessed through a cooperation protocol designed to avoid, as far as possible, accessing the local disk. Placing inside the server application a policy to make all the server nodes in a cluster cooperate to realize a kind of *global cache*, is a good way to improve the single server performance, and, consequently, the overall Web cluster performance too. This approach should be well suited to static Web scenarios, where most requests to the Web site are issued to read-only documents, typically of a Web hosting/publishing service site.

Cooperation mechanisms for global caching are mainly based upon a two-step interaction: firstly, the requested document must be located as it could reside in any of the cluster's caches (*global lookup*), then, once the file has been successfully located, it must be retrieved from some remote node (*document retrieval*). The global lookup phase is typically carried out through locality information sharing policies, which we refer to as *Informed*. We also propose a different global lookup policy based upon broadcasting of queries: we refer to this approach as *On-demand*. Broadcast-based cooperation solutions are mainly

motivated by the high bandwidth availability of current LANs.

The document retrieval phase is commonly accomplished either by fetching the file from a remote node (Migration) or by moving the TCP connection to a remote node (Handoff). This paper studies the two lookup alternatives, both of them combined with a Migration or Handoff mechanism for document retrieval, in different workload scenarios. It compares, through synthetic workload driven simulations, the performance of a Web cluster working without cooperation, and with the four cooperation schemes. The experimental results show that, when the requests are all directed to cacheable Web objects, the performance of the On-demand with Handoff schemes equals that of the Informed with Handoff policy, without overloading the system. The Informed with Migration scheme is outperformed by the On-demand with Migration scheme, but this latter can stress heavily the internal network. On the whole, the Handoff technique is preferable to the Migration technique, as this latter consumes high network bandwidth due to file transfers and update traffic. Another series of tests is dedicated to the estimate of the impact of uncacheable dynamic requests over the cooperation protocol. The results show that the On-demand cooperation loses effectiveness even at low percentages of dynamic requests, whereas the Informed cooperation exhibits a better scalability. On the whole, only the Informed with Handoff scheme can sustain a highly dynamic workload without losing completely the benefits due to cooperation.

The rest of the paper is organized as follows. In Section 2 we compare state-of-the-art architectures endowed with global caching capabilities. Section 3 gives a detailed discussion of the cooperation alternatives addressed in this paper. Section 4 provides the simulation and workload models we used for our experiments. In Section 5 we discuss the results of the performance evaluation. We outline conclusions and future work in Section 6.

2 Related work

This section gives an overview of other works related to cooperative caching in a locally distributed Web system. The comparison is carried out mainly considering the dynamics of the cooperation, disregarding other minor issues. All the architectures are based upon Informed global lookup, and most of them adopt a Migration technique to retrieve Web documents. Many solutions represent a direct integration of functional component into an enhanced Web cluster, such as the *Web-server Accelerator* [14], the

DC-Apache [10] and Swala [8].

The Web-server Accelerator is based on a two-tier distributed cache hierarchy: the front-end tier is a specialized cache array and the back-end tier is an array of conventional Web-servers. The global lookup approach is Informed-based, whereas the document retrieval can work with both Migration and Handoff techniques. All the Web documents are statically partitioned among the nodes, thus, to locate a cached file is sufficient a hashing technique.

The DC-Apache is a distributed cooperative Web cluster built upon the popular Apache software [1]. The DC-Apache adopts an Informed-based approach with a Migration technique to share disk content among the cluster nodes, without managing main memory caches. The cooperation is assisted by meta-data in the form of a document graph, built at system start-up and updated dynamically upon changes in the location of the Web files. The requested files are located through the hyperlink matching given by the document graph, and, if necessary, fetched from a sibling server; the embedded hyperlinks are rewritten on-the-fly so that a subsequent request reaches the least loaded sibling. The finite cache issue is addressed through global awareness of the available space at a node, whereas we rely only upon local information.

Unlike the previous two architectures, which considers only static files, Swala is a clustered Web server specifically designed to cooperatively cache CGI results, namely dynamically generated objects. The paper is mainly focused on cache consistency issues, and the proposed caching engine relies upon operating system caching rather than managing directly a main memory cache. Global lookup is Informed-based as it uses a meta-data broadcasting, whereas is not clear what kind of document retrieval technique Swala uses.

Other solutions are related to distributed file systems, such as the TH-CluFS by Liu et al. [11] and the CCM by Cuenca-Acuna and Nguyen [7], both adopting a block, rather than file, Migration mechanism. TH-CluFS is an I/O balancing file system dedicated to Web clusters, based upon a global integration of RAM and disks into a unique cache. To reduce the migration overhead, documents can migrate periodically only within dynamically formed groups; periodical regrouping is used to balance the load. The Cooperative Caching Middleware (CCM) layer adopts an Informed cooperation approach wherein meta-information updates are "piggy-backed" asynchronously, i.e. when are requested blocks

of a file whose location has changed. CCM uses an approximate global LRU replacement policy, while we consider schemes relying on autonomous local-cache eviction. The experimental results substantially confirm the poor effectiveness of migration-based schemes.

3 Global caching

A *cache* is a temporary data storage placed in a strategic location with the goal of retrieving its content more quickly than from the original storage resource. A Web cluster has a lot of RAM distributed among the server nodes that can be used as a virtually unique cache to create a so called *global caching* architecture. In principle, all types of Web objects can be cached, however the dynamically generated documents, such as the *CGI* results, pose many consistency problems which are beyond the scope of this paper. Indeed, we want to focus on cooperation issues rather than on dynamic request management, hence, most considerations will refer to static Web objects. This assumption is not too strong for current Web sites, because most Web content is still of static kind [3] or slowly variable over time [6]. However, we also analyze how dynamic requests affect the cooperation performance in a series of dedicated experiments.

A basic example of Web cluster equipped with a global caching engine is illustrated in Figure 1. In this *global caching abstraction*, each server node has a local cache which is directly accessible for local lookups, while the other remote caches can be indirectly accessed through a cooperation protocol. Distributed file systems are often used to share files within a Web cluster, because of their transparent interface toward the user level applications. But this feature and other ones, such as the capability to work in a read/write environment and the need to solve cache consistency problems, require additional computation which increases the service latency. Therefore, we conceive a cooperation protocol driven at the user-process level, with limited operating system intervention, which also exploits a full replication of Web contents on the node disks. Our cooperation schemes are not formulated as a resource allocation problem, thus we rely on local cache replacement policy, rather than on some globally approximated algorithm, even though it could carry out a non-optimal exploitation of the global memory. The goal of the cooperation protocol is to improve the service performance by exploiting RAM-to-RAM transfers through the internal LAN, which can be even two orders of magnitude faster than a local disk-to-RAM

transfer.

The whole cluster operates as a *one-way* architecture that is, only the HTTP requests from Web clients come through the Web switch, whereas the replies from the Web servers return directly to the clients through a different Internet connection. Unlike other proposals, we assume that the Web switch in front of the cluster is not involved in the cache cooperation as it dispatches any incoming request at the level 4 of the ISO/OSI network protocol stack. A complete overview of main dispatching issues in a Web cluster is given in [5].

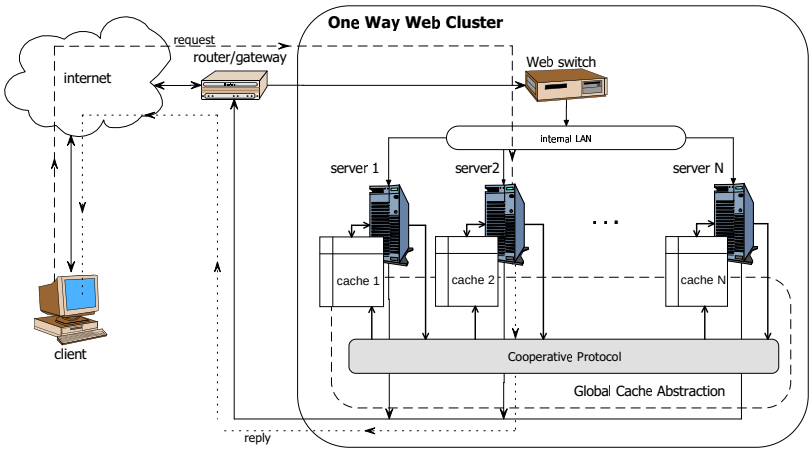


Figure 1. A basic global caching architecture.

After a client request has been accepted, the Web switch dispatches it to an HTTP server. If the requested document is not found in the local cache of the first contacted server, a cooperation protocol is activated. This scheme is based upon two phases, each of them can be implemented in different ways.

Global lookup. To find a copy of the requested document in the caches of other server nodes requires some *cooperation policy* which aims to provide the nodes with a global view of the system caches. Existing architectures are all based upon mechanisms for meta-information sharing which locate directly the requested documents: this approach can be defined as **Informed**. In a Web cluster endowed with an Informed cooperation policy each node is completely aware of the content of all the other caches, therefore, when the requested document is not found in the local cache (*local cache miss*), it can contact directly the node which is presumed to have a copy of the file in its cache. The mechanisms currently adopted to update the global state of the system are the asynchronous broadcast [8] and the asynchronous

piggy-backing [10]: updates are sent immediately upon changes in each local cache. Our cooperation schemes adopt a simpler solution based on periodical (synchronous) broadcast: exchanged information includes the log of all the changes occurred in the local cache from the last updating instant.

We propose a new approach to global lookup based upon a total unawareness of the contents of the cache nodes. In the case of a local cache miss, the node which receives the client request has to ask some other nodes whether they have or not a valid copy of the document in their cache: because of this kind of interaction we call this approach **On-demand**. The main motivation for this new scheme is the high bandwidth availability of the network interconnecting the cluster's nodes. We rely on a fully distributed policy, wherein queries are sent through a broadcast communication mechanism. This avoids the bottlenecks of centralized solutions, yet it can generate a significant traffic overhead, as each local miss on a node triggers a broadcast followed by a possible reply from all the other nodes. As we will see in Section 5, the network bandwidth consumption may become a crucial issue for the On-demand approach.

Remote document retrieval. Once found the requested document on a remote server node, the file must be returned to the client. The operations needed to this purpose involve only the first contacted node (A) and the node (B) that has been selected in the global lookup phase. A classical solution coming from distributed systems is to make the file migrate from the remote node (*Migration* technique), whereas a solution peculiar to a Web cluster consists in delegating the remote node to serve the request (*Handoff* technique).

The **Migration** mechanism is the most widely adopted, and it can be block-based [7, 11] or file-based [10, 14]. Our proposal is based upon file migration because we feel this solution more suitable to global caching, as a Web interaction always entails file transfers. The entire process works in the following way. The node A gets from the node B the requested cached document/block and sends it to the client. Each transaction keeps unchanged the number of document/block replicas, as the requested document/block migrates from one node to another with no local copy. The goal is to avoid the competition for global cache space at the expense of high network overheads.

The **Handoff** mechanism was originally conceived for request dispatching purposes, and can be

directly adopted in a global caching cluster. It works as it follows. The node A delegates the node B to complete the Web transaction: the connection is transparently "handoffed", that is, the corresponding TCP flow is rerouted from the node A to the node B. The latter node will reply directly to the client without any other intervention by the node A. This mechanism moves the load of the HTTP connection onto the node B, but it does not transfer any document. A feature that makes the handoff a general purpose technique [12, 14], which should work fine whatever global lookup policy is adopted.

4 Simulation model

4.1 System model

The Web cluster model consists of a set of N Web server nodes and a front-end Web switch. Each node has a disk, a CPU, a certain amount of RAM which is used for caching purpose, and two network links: the former for the internal network through which it receives HTTP requests dispatched by the Web switch and exchanges GCP data with the other nodes; the latter for the external network, through which the server node sends HTTP replies directly to the client (one-way interaction). Internal network links and hard-disks are modeled as FCFS queues characterized by a service time proportional to the amount of data to be transferred. Each transfer from one node to another engages two links, with no switching delay. The maximum theoretical bandwidth of such network model is $(N+1)/2$ times greater than the bandwidth value of the single link. Each CPU is a round-robin server with a fixed time-slice and a programmable service time, whereby each operation requested specifies its duration.

The model processes and their main interaction paths are illustrated in Figure 2. Each node has the same frame consisting of a fully replicated set of Web files on its disk, and of a server application, which has the following components.

RAM Cache: this is the local main memory storage of Web documents.

HTTP Daemon: this thread is responsible of accepting the HTTP requests coming from the HTTP clients. It can access directly the local RAM Cache, whereas the other remote RAM Caches can be looked up via the **GCP Client** routine: this latter plays the client role of the Global Caching Protocol, by sending *service requests* to a remote "Cache Manager" and waiting for replies from it.

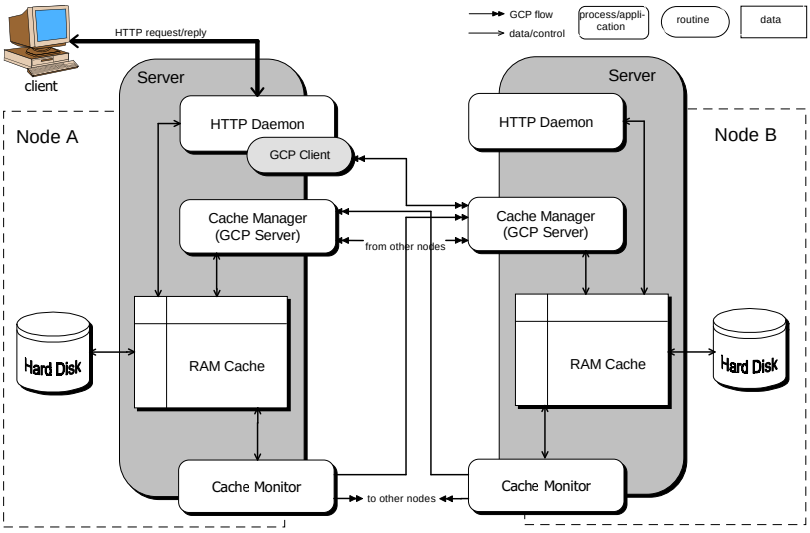


Figure 2. Main components of the system model.

Cache Manager: this thread plays the server role in the GCP, by accepting two kinds of requests: service requests from remote GCP clients, to which a reply must follow, and *update issues* from remote "Cache Monitor"s, which trigger modifications on the local meta-information.

Cache Monitor: this thread is responsible for checking periodically the local meta-information to update the state of the global cache. Any change is broadcasted (update issue) to the other nodes; these messages do not require replies, and are sent only under the Informed policies.

The assumptions underlying our simulation model are conceived to isolate the performance of the GCP. We assume that the cluster's ingress and egress points are not a performance bottleneck, which means that, with reference to Figure 1:

- the router/gateway can sustain both the ingoing and the outgoing TCP/IP traffic with no overhead;
- the Web switch can sustain the ingoing requests traffic at the kernel level with no overhead;
- all external network links have infinite capacity (ideal network);
- the kernel level computation on each node has negligible overhead in comparison with the application level computation.

The first assumption is straightforward, as current network devices can easily sustain traffic rates far over the megabit per second. The second assumption guarantees that all the inbound HTTP traffic can

reach the servers, subjecting them to the highest load pressure as it is generated by the client processes. We make the third assumption because we are not interested in network latency measurements on a geographical scale. The last one is due to the fact that the handoff mechanism requires additional kernel computation to re-map a TCP connection onto a node different from the one which established it. However, the connection cut-off mechanism prevents each node from serving, on average, no more than 450 requests per second: this means that each node has at least $2.2ms$ to process a request, which is an order of magnitude greater than a typical TCP handoff latency, as the measure of $194\mu s$ reported in [12].

The main simulation parameters are shown in Table 1. Note that the LAN bandwidth value is that of a real cost-effective network hardware. The cache size has been chosen to allow a certain degree of redundancy for the most popular Web documents: the global cache, whose size is equal to $N*cache\ size$, can ideally hold as much content as the entire working-set's core plus a further percentage. The access to the cache is barred to the files whose size is greater than the "cacheability size threshold". We choose the LFU*-Aging (Least Frequently Used with bounded reference counters and aging mechanism) algorithm as the cache replacement policy, because it solves the problem of the "one-timer" documents and reacts dynamically to the variations of document popularity [2]; A_{max} is the maximum allowed age, beyond which the reference counters are halved. Each message exchanged during a GCP transaction has a fixed size (200 bytes) header and, possibly, a payload large as much as the file conveyed. The "request timeout" specifies how long a node will wait a reply to its GCP service request from a remote Cache-Manager, before servicing locally the Web request. Locality information are updated with a period given by the "meta-data update interval". The system load metric is the number of active connections at each node: the two "cut-off threshold" parameters define the maximum sustainable load for HTTP Daemons and Cache Managers, beyond which any further connection request is refused. The percentage of requests for uncacheable objects is the driver parameter of the second test series.

4.2 Workload model

The system is subject to a synthetic workload generated according to the main guidelines found in the literature [3, 4]. The data set of the workload model is outlined in Table 2, where, for each component,

COMPONENT	PARAMETERS	VALUE
General	Number of nodes	8
	Dispatching policy	round-robin
Hardware	LAN bandwidth	100 Mbps
File system	Number of Web files	5000
Local Cache	Cache size	5 MB per node
	Cacheability size threshold	250 KB
	Replacement policy	LFU*-Aging (A_{max} = 600 sec)
GCP	Message size	$\geq$ 200 B
	Request timeout	2 msec
	Meta-data update interval	10 sec
Load Management	HTTP Daemon cut-off threshold	100 active connections
	Cache Manager cut-off threshold	10 active connections
Uncacheable Objects	Request share	0$\div$50% (variable)
	Processing throughput	1 KB/sec

Table 1. Simulation parameters.

is showed the probability density function with its parameters. The Zipf's function for the popularity of Web documents is normalized through the constant C. The file size probability function is hybrid: it is composed of a Lognormal body and of a Pareto tail bounded to a maximum value s_{max}. The resulting file system has a mean file size of $7.5KB$, whereas the working-set of the synthetic HTTP workload exhibits a mean request size of $15.8KB$. The working-set's core amounts to approximately $37MB$. The *mean inter-arrival time* (μ) for client processes is variable as it is used to drive the first experiment series. The HTTP workload is generated through a session-based mechanism, but neither pipelining nor keep-alive mechanisms are used, that is equivalent to an HTTP/1.0 stream. The simulator creates a client process, which goes through a certain number of session, each corresponding to an HTML page request, which in its turn triggers a certain number of requests for the embedded objects. Client processes are generated with an exponentially distributed inter-arrival time; the number of sessions issued by client processes is exponentially distributed, whereas the number of requests composing each session is distributed according to a Pareto function. Our analysis has a granularity at the level of a single hit request: this means that, from the Web-server's point of view, each request is related to a file transfer, being it an HTML page or an embedded object.

COMPONENT	MODEL	PROB. DENSITY	PARAMETERS
Document Popularity	Zipf	$Cr^{-\beta},\ r = 1, ..., 5000$	$\beta = 1$
File Size Distribution (body)	Lognormal	$\frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\ln x - \mu)^2}{2\sigma^2}},\ x > 0$	$\mu = 7.796,\ \sigma = 1.624$
File Size Distribution (tail)	Bounded Pareto	$\alpha k^{\alpha} x^{-\alpha-1},\ x \geq k$	$s_{max} = 1MB,\ k = 75,$ $\alpha = 1.47$
Client Inter-arrival Time	Exponential	$\frac{1}{\mu} e^{-x/\mu},\ x > 0$	$1/\mu = 15 \div 70$ (variable)
Sessions per Client	Exponential	$\frac{1}{\mu} e^{-x/\mu},\ x > 0$	$\mu = 10$
Requests per Session	Pareto	$\alpha k^{\alpha} x^{-\alpha-1},\ x \geq k$	$\alpha = 1.33,\ k = 2$

Table 2. Workload model.

5 Experimental results

We evaluate the performance of four global cooperation schemes: Informed with Migration (Info-M), Informed with Handoff (Info-H), On-demand with Handoff (Onde-H) and On-demand with Migration (Onde-M). These schemes are also compared to a Web cluster where the server nodes do not use any cooperation mechanism (No-Coop). All the reported values are the aggregate results of multiple experiments referring to the whole cluster. The main performance metrics of interest for this paper are:

Document Hit Ratio (DHR): this is the ratio of the number of documents found in the global cache to the total number of requested documents.

Request Response Time: it is measured at the client side as the time elapsed between the generation of a request from the client and the arrival of the last packet in response. Delays in the external network are not included.

Connection Refusal Rate: this is the number of refused request connections per second. It is useful to estimate the performance loss when the system saturates.

Protocol Bandwidth: it estimates the costs of global cooperation, in terms of network overhead. It is measured as the number of megabits per second transmitted over the internal LAN by the cooperation protocol.

In the following sections we present the results of tests for two scenarios: all the requested objects are static and cacheable, some requests are dynamically generated and not cacheable.

5.1 Static workload scenario

This scenario is characterized by requests to static Web objects which can be always cached. Two system configurations have been considered: a finite size cache cluster ($5MB$ per node) to evaluate the peak performance of the cooperation policies, and an infinite size cache cluster to evaluate the cooperation overhead.

The experiments are driven by a mean number of client process arrivals per second ranging from 15 to 70. The corresponding arrival rates of requests range from 700 to 3300 requests per second on average: such load conditions drive the cluster toward a saturation state, whereby no more than 8% of the incoming requests are refused by the cluster in the worst situation.

COOPERATION SCHEME	DHR%	
	worst [var]	best [var]
Non-cooperative (no-coop)	61.6	61.6
On-demand with Handoff (Onde-H)	73.6 [+12.0]	81.9 [+20.3]
On-demand with Migration (Onde-H)	67.9 [+6.3]	73.6 [+12.0]
Informed with Handoff (Info-H)	79.2 [+18.2]	85.1 [+23.5]
Informed with Migration (Info-M)	64.2 [+2.6]	65.3 [+3.7]

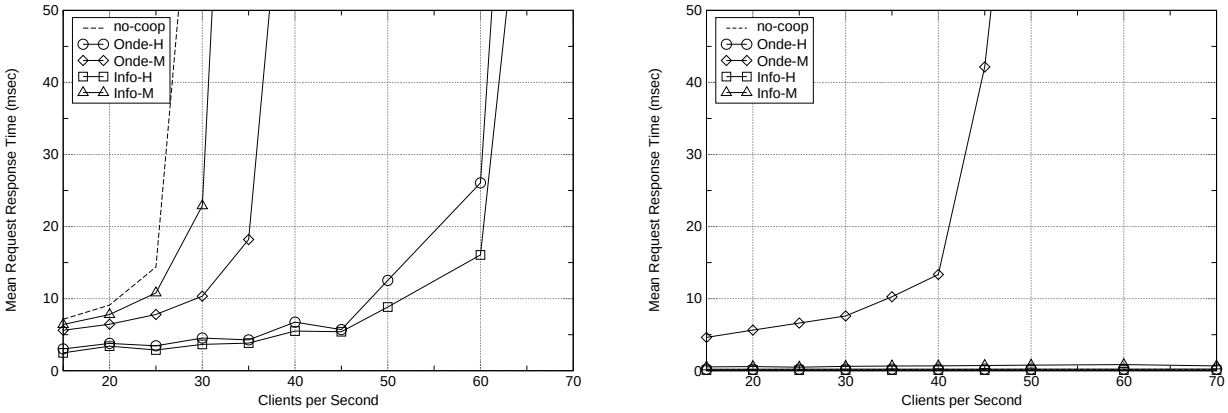
Table 3. Static workload scenario: mean DHR comparison for the finite cache cluster.

We first show in Table 3 the worst and best mean DHR values recorded during the whole test series of the finite cache cluster. We observe that our system and workload models do not tend to penalize the basic Web cluster (No-Coop). Indeed, even with No-Cooperation, the DHR values are always higher than 60%. The results in this table show that only the Handoff-based schemes are truly effective, whereas the Migration-based ones exhibit minor improvements. The Onde-H scheme performs a little worse than the Info-H scheme, presumably due to the query broadcasting which generates a significant network overhead. The poor effectiveness of the migration-based schemes could be a consequence of the caching dynamics which change the location of files more frequently under the Migration technique rather than with Handoff, wherein cache content changes only as a consequence of local replacements.

In Figures 3(a) and 3(b) we show the mean response time for a Web cluster with finite and infinite cache, respectively. Focusing on the knee of the diverse curves in Figure 3(a) (finite cache), we see that Onde-H and Info-H move the cluster saturation point to 60 *clients/sec*, which is twice better than the No-Coop performance (25 *clients/sec*). On the other hand, Onde-M achieves minor improvements,

though it outperforms Info-M. Surprisingly, the Onde-H scheme performs the same as the Info-H one in a wide load range, despite of an inferior cache hit rate (DHR): this is due to the higher probability of locating the requested document, which is typical of the On-demand approach. We also measured the system throughput, whose values (not shown) confirm the above remarks.

In the infinite cache scenario, local cache replacements never occur. All the Web files are prefetched in the caches at start-up time in two ways: in the No-Coop cluster each node caches all the files, while in the cooperation-based clusters all the files are circularly allocated on different nodes, in order to serve via global cooperation all the local cache misses. The mean response time in the infinite cache configuration (Figure 3(b)) demonstrates that the global cooperation is subject to heavy overhead only when the Onde-M scheme is used.



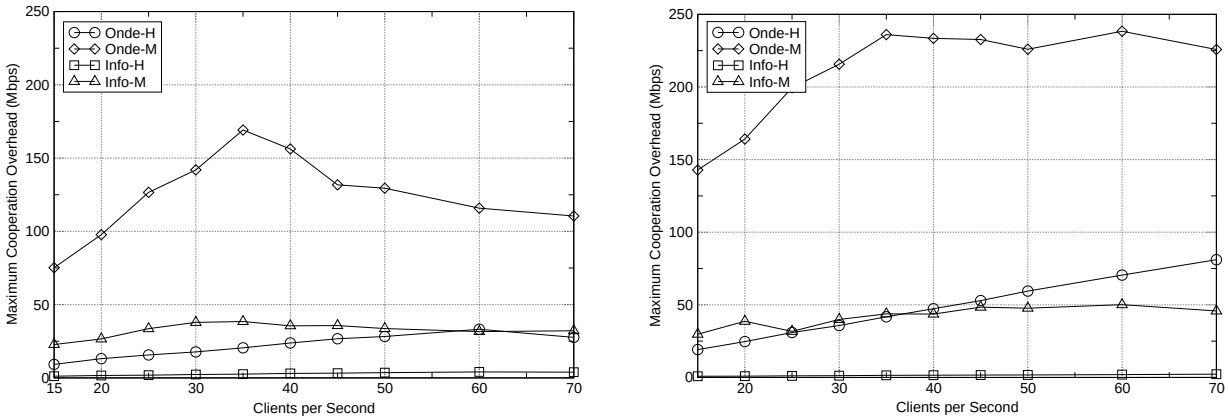
(a) finite cache

(b) infinite cache

Figure 3. Static workload scenario: mean request response time.

We evaluate also the maximum network bandwidth consumed by the GCP, with finite (Figure 4(a)) and infinite cache (Figure 4(b)). The observed values are obtained by measuring the number of bytes coming out from each node. Depending on the global caching scheme, the infinite cache configuration allows us to evaluate the upper or the lower bound of bandwidth consumption. As local cache replacements never occur, when the Handoff mechanism is used, the file location never changes. On the other hand, the Migration technique causes location changes at every local cache miss. Thus, if compared with the query traffic of the On-demand approach, the periodic information exchange traffic of the Informed

approach is negligible with the Handoff mechanism. On the contrary, this traffic represents a large percentage of the total network overhead when the Migration mechanism is used because this latter requires a lot of update messages. Therefore, the curves related to maximum bandwidth consumption in the infinite cache scenario denote a lower bound for the Info-H scheme, because it has negligible update and no query traffic. For the other schemes, the reported results denote upper bounds, because of high query broadcast traffic (Onde-H), high query broadcast and migration traffic (Onde-M), high update broadcast and migration traffic (Info-M).



(a) finite cache

(b) infinite cache

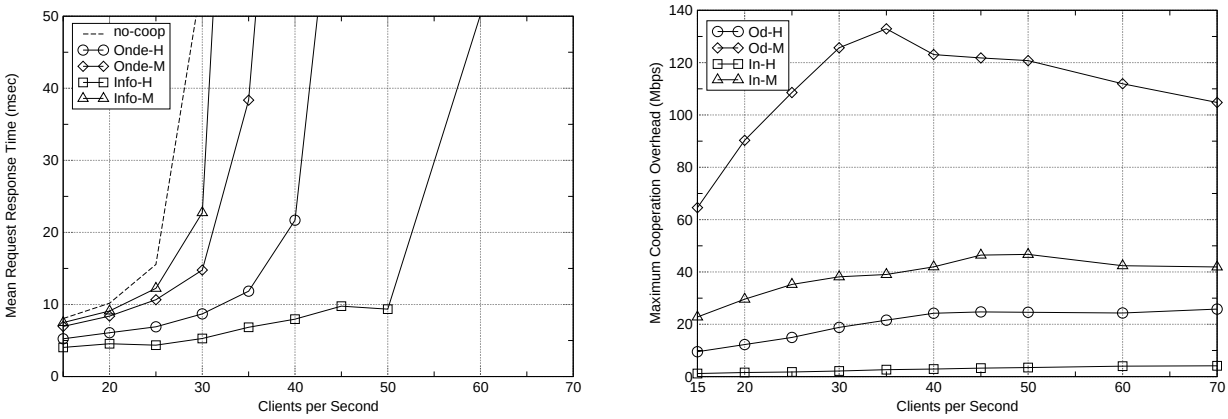
Figure 4. Static workload scenario: maximum internal B/W consumption due to global cooperation.

Figure 4(b) shows that the migration-based schemes can stress heavily the system, as testified by the unevenness of their bandwidth consumption curves. In the realistic case of limited cache (Figure 4(a)) we observe that all the schemes, but Onde-M, generate a hardware-sustainable network traffic. The frequent file transfers and the subsequent local cache competition should be the main reason of the poor performance of the migration-based schemes. Indeed, Info-M seems to suffer mainly from low network transfer speed, whereas Onde-M clearly saturates the internal network: the flattening of their curves confirms that their saturation points are 30 and 35 *clients/sec*, respectively. The Info-H scheme has a negligible bandwidth consumption under all load conditions.

5.2 Dynamic workload scenario

This Web scenario contains some percentage of requests to dynamically generated objects, like CGI invocation results. As these objects are typically short-lived files, we choose not to cache them. Each request for an uncacheable objects loads the CPU of the target server for a time proportional to the object size.

Increasing load pressure tests. This performance stress test series has been executed with increasing load pressure from 15 to 70 *clients/sec*, and with a 2% of dynamic requests: this latter value is close to those reported in [6] about short-lived objects, and confirmed by statistics we extracted from two popular Web server traces (NASA and ClarkNet [15]).



(a) mean request response time

(b) maximum B/W consumption

Figure 5. 2% dynamic request share test.

As shown in Figure 5(a) the system response time exhibits evident changes with respect to the static workload scenario only for Onde-H, which is now outperformed by Info-H. The bandwidth consumption graph (Figure 5(b)) confirms that Onde-H saturates the system at a load pressure of 40 *clients/sec*, while Info-H can easily sustain 50 *clients/sec*. The other GCP schemes do not exhibits high variations in their performance, when the percentage of dynamic requests is low. The throughput measurements are not shown, as they do not change the above conclusions.

Increasing dynamic share tests. The goal of this series of experiments is to evaluate the robustness of the system in presence of dynamic uncacheable objects. The tests are driven by a percentage of dynamic requests ranging from 0% to 20%, while keeping the load arrivals to 50 *clients/sec*.

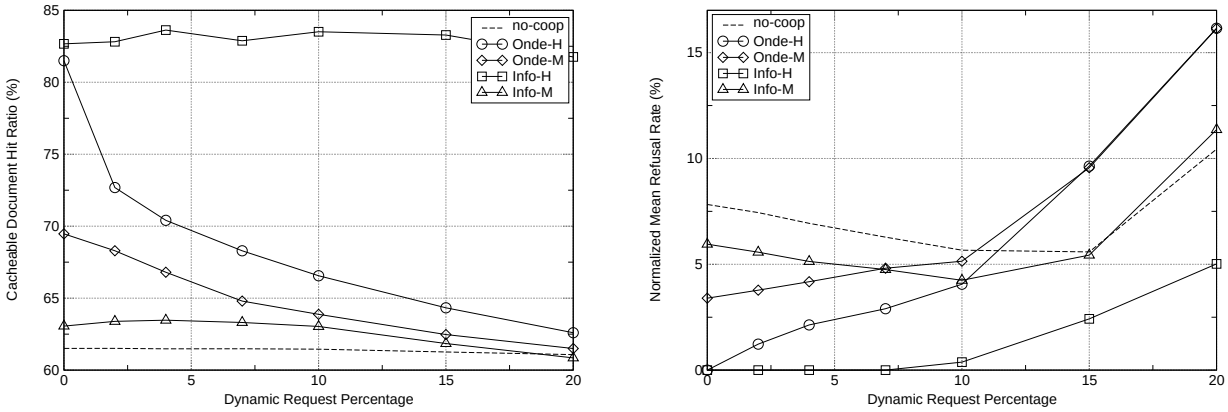
The dynamic processing increases the response time of each server, thus driving the cluster toward a deep saturation state. The connection refusal rate has been normalized to the No-coop cluster's values.

We observe from Figure 6(a) that just a 2% of dynamic requests can reduce markedly the DHR of the Onde-H scheme, whereas the others lose performance with a less steep trend. However, this DHR decrease does not affect in the same way the refusal rate of the various schemes, as Figure 6(b) shows: Onde-H and Onde-M exhibit an ever increasing rate of refused connections; Info-H and Info-M have a flat and a decreasing trend till about a 10% of dynamic requests, then their performance start worsening. As the size of the static working-set decreases when more dynamic requests are sent to the cluster, the subsequent greater cache availability antagonizes the higher service times: the trend inversion of Info-M means that this scheme is more sensitive to cache room competition than to CPU cycle consumption. On the other hand, the On-demand schemes require quick responses from remote Cache-Managers. Hence, the effects of the reduced CPU availability seems to prevail over the increasing of cache room insofar the cooperative system performs worse than the non-cooperative one. The Info-H configuration is the only scheme capable of bearing massive percentage of dynamic requests, thanks to the absence of cache competition and to the synchronous lookup interaction.

5.3 Summary of the performance study

The overall remark about the performance study is that the On-demand approach to global caching can be effective as much as the Informed approach, though it could require plenty of network bandwidth. In particular, when the system is subjected to an increasing load of static cacheable requests, we observe the following results.

- The Handoff-based schemes exhibit almost the same performance results, outperforming twofold a cluster with no cooperation mechanism. The internal bandwidth consumption of the Handoff mechanism is negligible under the Informed approach, and well sustainable under the On-demand approach.



(a) mean cacheable DHR

(b) mean connection refusal rate (normalized)

Figure 6. Dynamic workload tests: 8 servers, 50 clients per second.

- The migration-based schemes yield minor performance improvement with high communication overheads. The On-demand policy outperforms the Informed one, but it can saturate the internal network.

When adding a certain percentage of request for dynamically generated uncacheable objects we observe the following results.

- Only the Informed with Handoff scheme can sustain gracefully a highly dynamic workload, whereas the other schemes undergo a performance loss insofar they approximate the non-cooperative system. In particular, the On-demand with Handoff scheme is the most sensitive to the dynamic request processing.
- When the size of the static working-set decreases, there exists a threshold of dynamic request percentage before which the Informed GCP schemes tolerate the load and after which all GCP schemes start losing performance. In particular the handoff-based schemes end up outperformed by the non-cooperative cluster.

Table 4 summarizes this performance study by reporting the most significant results: the values are normalized to the corresponding ones of the non-cooperative cluster. The response times are those recorded at a load pressure of 25 *clients/sec*.

SCENARIO → / COOP. SCHEME ↓	STATIC		2% DYNAMIC		10% DYNAMIC	
	DHR	response time	DHR	refusal rate	DHR	refusal rate
On-demand with Handoff	+33%	-77%	+18%	-84%	+8%	-29%
On-demand with Migration	+19%	-47%	+11%	-50%	+4%	-10%
Informed with Handoff	+38%	-81%	+35%	-100%	+36%	-94%
Informed with Migration	+6%	-26%	+3%	-27%	+2%	-26%

Table 4. Normalized performance comparison.

6 Conclusions

We have analyzed and compared the main methodologies for cooperative caching in Web clusters. Current cooperative solutions rely on meta-information sharing to locate cached documents (Informed-based), but we show that is also possible to build a cooperation scheme upon a distributed query broadcasting mechanism (On-demand). We show that this latter approach is effective and can outperform the Informed-based one. We also considered the impact of the Handoff and Migration document retrieval mechanisms on cooperation. The experimental results show that, in a static workload scenario, when the Handoff mechanism is adopted the On-demand policy performs the same as the Informed policy at well sustainable bandwidth costs, and it outperforms the latter when using the Migration mechanism. In a scenario with some dynamically generated uncacheable requests, the On-demand approach works fine only when the percentage of dynamic requests is very low: after a certain threshold, both the On-demand schemes are outperformed by the Informed with Migration scheme. On the other hand, the Informed with Handoff scheme exhibits always a good robustness. The query broadcasting mechanism should make the On-demand cooperation well suited to highly volatile scenarios, wherein requests are cacheable but remain valid for a short time.

References

[1] The Apache Web server. http://www.apache.org.

[2] M.F. Arlitt, "A Performance Study of Internet Web Servers", Master Thesis, Department of Computer Science, University of Saskatchewan, Saskatoon, Saskatchewan, Canada, 1996.

[3] M.F. Arlitt and T. Jin, "Workload Characterization of the 1998 World Cup Web Site", *IEEE Network*, Vol. 14, No. 3, pp. 30-37, May/June 2000.

[4] P. Barford, M.E. Crovella, "Generating Representative Web Workloads for Network and Server Performance Evaluation", Proc. of *Performance '98/ACM SIGMETRICS '98*, Madison, WI, USA, June 1998.

[5] V. Cardellini, E. Casalicchio, M. Colajanni, P.S. Yu, "The state of the art in locally distributed Web-server systems", *ACM Computing Surveys*, Vol. 34, No. 2, pp. 1-49, June 2002.

[6] L. Cherkasova, M. Karlsson, "Dynamics and Evolution of Web Sites: Analysis, Metrics and Design Issues", Proc. of *IEEE Symp. on Computers and Communications*, Hammamet, Tunisia , July 2001.

[7] F.M. Cuenca-Acuna, T.D. Nguyen, "Cooperative Caching Middleware for Cluster-Based Servers", Proc. of *10th IEEE Int'l Symp. on High Performance Distributed Computing (HPDC-10)*, San Francisco, CA, USA, Aug. 2001.

[8] V. Holmedahl, B. Smith, and T. Yang, "Cooperative Caching of Dynamic Content on a Distributed Web Server", Proc. of *IEEE Symposium on High Performance Distributed Computing*, Chicago, Illinois, USA, July 1998.

[9] IRCache: the NLANR's Global Caching Hierarchy. Available at http://ircache.nlanr.net.

[10] Q. Li, B. Moon, "Distributed Cooperative Apache Web Server", Proc. of *10th Int'l World Wide Web Conference*, Hong Kong, May 2001.

[11] W. Liu, M. Wu, W. Zheng, M. Sheng, "Design of an I/O Balancing File System on Web Server Clusters", Proc. of *2000 Int'l Workshop on Scalable Web Services* (in conjunction with *ICPP'00*), Toronto, Ontario, Canada , August 2000.

[12] V.S. Pai, M. Aron, G. Banga, M. Svendsen, P. Druschel, W. Zwaenepoel, E. Nahum, "Locality-Aware Request Distribution in Cluster-based Network Servers", Proc. of *ACM 8th Int'l Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS-VIII)*, San Jose, CA, Oct. 1998.

[13] A. Santoro, B. Ciciani, M. Colajanni, F. Quaglia "Two-Tier Cooperation: A Scalable Protocol for Web Cache Sharing", Proc. of *IEEE Int'l Symp. on Network Computing and Applications (NCA'01)*, Cambridge, MA, Feb. 2002.

[14] J. Song, E. Levy-Abegnoli, A. Iyengar, D. Dias, "Design Alternatives for Scalable Web Server Accelerators", Proc. of *IEEE Int'l Symp. on Performance Analysis of Systems and Software*, Austin, TX, April 2000.

[15] The W3C Web Characterization Repository. http://repository.cs.vt.edu