Impact of request dispatching granularity in geographically distributed Web systems

Mauro Andreolini, Claudia Canali, Riccardo Lancellotti

University of Modena and Reggio Emilia
Department of Computer Engineering
{mauro.andreolini, claudia.canali, riccardo.lancellotti}@unimo.it

Abstract

The advent of the mobile Web and the increasing demand for personalized contents arise the need for computationally expensive services, such as dynamic generation and on-the-fly adaptation of contents. Providing these services exacerbates the performance issues that have to be addressed by the underlying Web architecture. When performance issues are addressed through geographically distributed Web systems with a large number of nodes located on the network edge, the dispatching mechanism that distributes requests among the system nodes becomes a critical element.

In this paper, we investigate how the granularity of request dispatching may affect the performance of a distributed Web system for personalized contents. Through a real prototype, we compare dispatching mechanisms operating at various levels of granularity for different workload and network scenarios. We demonstrate that the choice of the best granularity for request dispatching strongly depends on the characteristics of the workload in terms of heterogeneity and computational requirements. A coarse-grain dispatching is preferable only when the requests have similar computational requirements. In all other instances of skewed workloads, that we can consider more realistic, a fine-grain dispatching augments the control on the node load and allows the system to achieve better performance.

1 Introduction

A significant trend in the mobile Web is the growing amount of personalized contents required by users, that increasingly access the Web through heterogeneous and mobile devices. The underlying Web system needs to handle increasing percentages of dynamic generation and adaptation services to deliver personalized contents that match user preferences and device capabilities. A typical example is related to Web sites providing location-based or tourism services, that are commonly accessed by mobile users through devices with limited capabilities. These Web sites aim to offer personalized contents to provide the users with information tailored to their interests or current location. These contents are dynamically generated on the basis of user-related information and delivered, possibly after the adaptation of part of the contents (e.g., images) to match the requirements of the user device.

The delivery of personalized contents exacerbates the performance issues of the underlying architecture for a twofold reason. First, contents have to be generated and adapted at the moment of the request, since a pre-generation of formats for any combination of user preferences and device capabilities is simply unfeasible. Second, generation and adaptation of contents typically have much higher computational requirements with respect to the request/reply service of traditional static resources [2]. For these reasons, much interest of the research community has been oriented to achieve efficient content generation and delivery through geographically distributed systems. Replication guarantees the possibility of sharing the load of computationally expensive tasks, while location of nodes close to the network edge allows the reduction of network-related delays. Throughout this paper, we consider an architecture which is commonly used in distributed Web systems [10, 23], where a *core* component, typically consisting of a cluster of servers, is integrated with distributed *edge* nodes that are close to the clients (see, for example Figure 1).

In similar distributed Web systems the request dispatching mechanism and algorithms play a fundamental role to efficiently exploit the system resources and to provide load sharing among the nodes [19, 25]. We should consider that a user click for a Web resource originates multiple client requests for the resource template and the *components*, where a component may range from a fragment to a multimedia resource. Moreover, each component may require a different

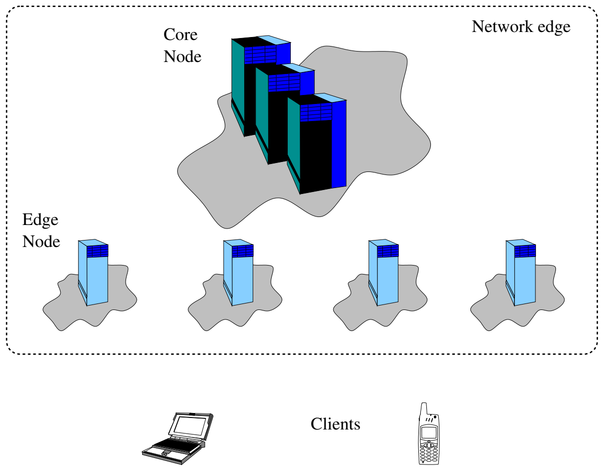


Figure 1. Distributed system for personalized Web contents.

generation or adaptation service. In this scenario, an important choice in the design of efficient dispatching mechanisms concerns the level of request granularity to operate on. We can ideally go from *coarse-grain* dispatching that assigns all requests for a single Web resources to the same node, to *fine-grain* dispatching that may assign each component of the same Web resource to a different node. Since the granularity of request dispatching has significant effects on the distribution of the computational load on the system nodes, it is important to understand which is the most convenient granularity to be used in a geographically distributed system supporting personalized Web contents.

The comparison between different levels of dispatching granularity has been widely investigated for the delivery of static contents, for example in cluster-based Web systems [5, 4], where fine granularity is preferred for load sharing purpose, and in CDNs [19], where Kangasharju *et al.* [18] demonstrates the superiority of coarse-grain dispatching. Even many recent proposals of systems for dynamic contents [12, 24] rely on DNS-based approaches, that operate a coarse-grain dispatching, or on application-layer redirection that statically assigns requests on the basis of the current application placement over the system nodes. However, to the best of our knowledge, this is the first paper that investigates the impact of different dispatching granularity on the performance of geographically distributed systems providing personalized contents that are dynamically generated. Understanding the impact of different dispatching granularities is particularly important in a scenario where generation and adaptation services place a significant load over the system and make less effective common techniques that are traditionally used in content delivery, such as caching or prefetching [6, 23, 11].

This paper evaluates how different granularity choices in request dispatching affect the user-perceived performance of a distributed Web system composed by a core node and geographically distributed edge nodes. We consider three dispatching mechanisms, namely *coarse-grain*, *medium-grain* and *fine-grain*, that are described in Section 3. Our analysis shows that the choice of the best granularity for request dispatching strongly depends on the characteristics of the workload in terms of heterogeneity and computational requirements. Experimental results demonstrate that, in a scenario where all the provided services have similar computational requirements, dispatching mechanisms with different granularity lead to similar performance, but a coarse-grain dispatching is preferable since it causes less overhead on the system. On the other hand, for highly heterogeneous workloads, a fine control of the request dispatching is of fundamental importance to achieve good performance, since it provides the flexibility necessary to efficiently share the load among the nodes of the distributed Web system.

The rest of this paper is structured as follows. Section 2 outlines the geographically distributed Web system for personalized contents considered in this paper. Section 3 presents the dispatching mechanisms operating at different levels of granularity. Section 4 describes the prototype and the experimental setup. Section 5 compares the performance achieved by the considered dispatching mechanisms. Section 6 discusses some related work. Section 7 summarizes our main results and conclusions.

2 System overview and operations

In this section we describe the distributed Web system that represents the reference scenario for this paper. As shown in the model of Figure 1, we consider a distributed architecture [10, 23] consisting of a centralized core node integrated with geographically replicated edge nodes. The use of replicated edge nodes represents a common solution for a system providing personalized Web contents for performance reasons and also because most mobile devices access the Web through some intermediary node. The considered architecture is an evolution of the systems where the geographically distributed nodes are reverse proxies [26] that cache and serve static content. Our approach follows the recent trend of modern systems that exploit servers on the network edge to replicate services (possibly including generation and adaptation), as can be observed in recent literature [24, 19, 17] and in CDN solutions [27, 12]. It is worth to note that we focus on a scenario where the content provider manages the entire architecture and controls all services and information hosted on the core and the edge nodes. However, the results of the paper can be extended to the case where a CDN houses or hosts the edge components of the distributed architecture.

The reference architecture model, that is described in

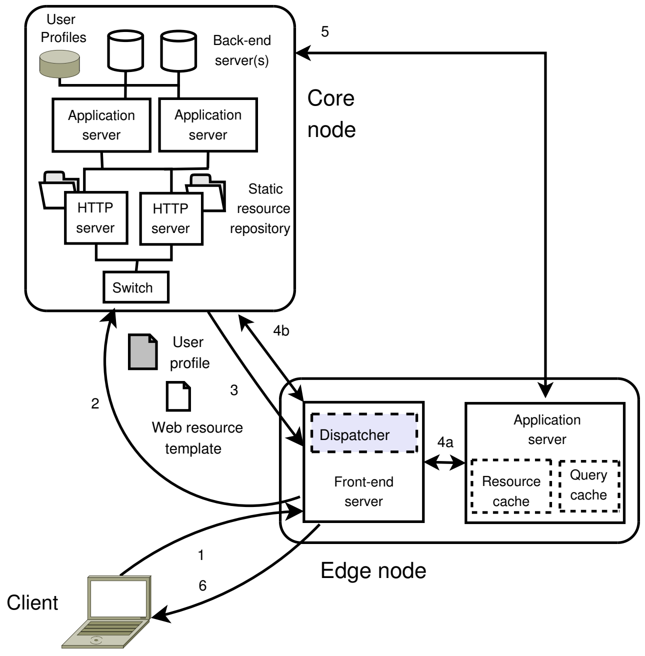


Figure 2. Detailed view of the distributed system supporting personalized Web contents.

Figure 1, is detailed in Figure 2. We can see that the core node is basically a three-tier Web system, where the first tier consists of the *HTTP servers*, the second tier hosts the *application servers*, and the third tier consists of *back-end servers* managing the data for the dynamic generation of contents. The core node maintains a *static resource repository* and a *user profile database*, that stores user information necessary to generate personalized contents. Each edge node consists of a two-tier Web system with a *front-end server* that receives client requests and an *application server* that carries out generation and adaptation functions. As shown in Figure 2, each edge node of the distributed Web system hosts a local *resource cache* that maintains a replica of the static resource repository on the core node. Every update to the static resource repository is propagated from the core node through push-based caching mechanisms to guarantee Web data consistency [14, 13, 22]. Each local resource cache also stores a copy of the user profiles that are necessary to allow edge nodes to generate and adapt Web contents. The user profiles are maintained on the edge nodes only for the period of time necessary to perform the assigned tasks. This choice allows the system to avoid any problem related to the consistency of replicated profiles, that may change frequently in modern Web systems (e.g., [15]). Finally, the edge nodes host a *query cache*, that stores the results of previous queries to the back-end database servers [13, 22].

The steps to serve a client request for a Web resource are detailed in Figure 2. We recall that a user click for a Web resource originates multiple requests for the resource template and the components, each of them may require a different generation or adaptation service. After receiving a client request (Step 1), the edge node determines if that request denotes a new user session. We define a user session as a sequence of Web resource requests issued from the same user to the same site during a limited time interval. In the case of a new session, the edge node contacts the core node (Step 2) to retrieve the user profile and a template that lists the components of the Web resource (Step 3). If the client request refers to an already established session, the edge node retrieves from the core node only the template of the Web resource, because it already owns the necessary profile information. The *dispatcher* running on the edge node executes one of the dispatching mechanisms described in Section 3, that decide which Web resource components will be processed by the local edge node (Step 4a) and which by the core node (Step 4b). If the processing of a resource component requires database information, the application server of the edge node may contact directly the back-end servers of the core node (Step 5) or interact with the local query cache. After the generation/adaptation of each component of the Web resource, this is sent to the client by the edge node (Step 6).

3 Request dispatching

In this section we describe multiple dispatching mechanisms using different levels of granularity in the distribution of requests for Web resource components among the edge and the core nodes. All the considered dispatching mechanisms aim to improve performance by moving as much computation as possible on the network edge to reduce network-related delays, as suggested by some of the best practices in efficient Web content delivery [24, 17, 12]. To avoid overloading the edge nodes, the dispatching mechanisms have to take into account information on the system status. As most generation and adaptation tasks are CPU-bound operations, we choose the CPU utilization of the nodes as the most representative index of the system status [1]. We follow a threshold-based approach, that is commonly used in literature [3], where the tasks originated by a client request are assigned to the edge node until this node reaches a utilization threshold U_T. When the edge node utilization U_E is higher than the threshold, some or all requests for other Web resource components are forwarded to the core node with different strategies depending on the granularity exploited by the dispatching mechanism.

We present three dispatching mechanisms, namely *fine-grain*, *medium-grain* and *coarse-grain*, that are characterized by increasing levels of granularity in request dispatching.

Fine-grain dispatching

The *fine-grain* dispatching distributes requests among the edge and the core nodes by operating at the level of individual components of a Web resource. If the edge node utilization U_E is below the threshold U_T, all the N components of the Web resource are processed locally. Otherwise, the dispatching mechanism determines the amount of components N_C that should be forwarded to the core node on the basis of the fraction $\frac{U_E-U_T}{1-U_T}$, that indicates how much the edge node utilization is beyond the threshold:

$$N_C \Leftarrow \lfloor N \frac{U_E - U_T}{1 - U_T} \rfloor \quad (1)$$

In this way, a higher value of U_E determines a higher amount of components that are assigned to the core node in the attempt to offload the edge node. We assume that the N_C forwarded components are randomly chosen among the N components of the Web resource.

Medium-grain dispatching

The *medium-grain* dispatching operates at a coarser level of granularity than the *fine-grain* scheme by considering that the Web resource components are grouped into *macrocomponents*. Indeed, in modern Web sites components typically embed other components in a recursive way. Hence, Web resources can be structured as hierarchical documents, where the overall resource represents the first level root node and the single components are the leaf nodes. This is consistent with the fragment-based composition of a Web resource described in [27, 7]. The *medium-grain* dispatching considers a Web resource as composed by M macrocomponents, where a macrocomponent is the set of components belonging to the same sub-tree that starts from the second level of the hierarchy related to the Web resource. If the edge node utilization U_E is below the threshold U_T, all macrocomponents are processed locally. When the edge node utilization is above the threshold, the dispatching mechanism computes the number M_C of macrocomponents that should be forwarded to the core node in a way similar to Equation 1. The M_C forwarded macrocomponents are randomly chosen among the M macrocomponents of the Web resource.

Coarse-grain dispatching

The *coarse-grain* dispatching operates at the level of an entire Web resource. If the edge node utilization U_E is below the threshold U_T, the Web resource is processed locally. Otherwise, the dispatcher assigns the entire Web resource to the core node.

4 Prototype

To evaluate the performance of the request dispatching mechanisms, we have implemented a prototype supporting a Web portal site that provides dynamic and personalized contents.

4.1 Services and workload

Our prototype provides three main services, that are typically offered by Web portals for personalized contents, such as MyYahoo [21]: personalized banner insertion, aggregation of RSS feeds and adaptation of Web content to the user device.

The personalized banner insertion service uses a database to associate banners to one or more keywords: it extracts from the database a list of banners according to the user interests. Then, a set of banners is randomly selected from the list and inserted into the Web resource.

The aggregation of RSS feeds allows us to generate the Web resources through the information collected from multiple sources. The RSS feeds are downloaded off-line and periodically refreshed. Then, the conversion from the RSS-XML code to HTML is carried out dynamically for every request depending on the user preferences.

The adaptation of Web content to the user device aims to transform static HTML code and embedded images on the basis of the information stored in the user profile. For example, the user may select a transcoding service to adapt images to the client display size and color depth. Furthermore, the user may select a data compression function to reduce the bandwidth consumption related to HTML and Javascript code download.

The described services are characterized by increasing computational requirements that may involve service times of different orders of magnitude. Indeed, a banner insertion typically requires few milliseconds, a feed aggregation task may be in the order of tens of milliseconds, while the adaptation of an embedded image may take from hundreds of milliseconds up to 1 second [8]. We define three workload models, namely *WL1*, *WL2* and *WL3*, that are characterized by a different mix of services. The exact percentages are reported in Table 1. In the workload WL1, the majority of the requests refers to the same average service (Feed aggregation) and consequently have similar computational requirements; in the workload WL2, the requests are evenly distributed among the three offered services; in the workload WL3, the majority of the requests is evenly distributed among the two services with the lowest and the highest computational requirements (Banner insertion and Content adaptation, respectively), while a little percentage of requests refers to the third average service (Feed aggregation).

Table 1. Workload composition

Workload	Service		
	Banner insertion	Feed aggregation	Content adaptation
WL1	5%	90%	5%
WL2	33%	34%	33%
WL3	45%	10%	45%

To exercise our prototype we use synthetically generated traces since none of the existing benchmarking models (e.g., Spec-Web, TPC-W) includes features for highly personalized Web contents. The requests in the traces are divided into sessions, each related to a different user. Sessions are initiated at the rate of 5 sessions per second and are evenly distributed among the edge nodes. Each session contains requests for 5 Web resources on average. A Web resource typically consists of a template and 10 components, each of them requiring one of the previously described services. We use traces generating a workload intensity that is lower than the overall system capacity, because our study aims to evaluate the impact of the dispatching granularity on the performance of a system that is in non-saturated conditions. To handle flash crowd events, the system may be integrated with already existing mechanisms for preventing overloads, such as policies of admission control [16].

4.2 System

The prototype is implemented through open sources technologies for multi-tier Web systems on the core and the edge nodes. The core node of the prototype is a Web cluster with a front-end switch, HTTP-servers, application servers and back-end servers. The software handling dynamic generation and adaptation services on the application servers is based on the Perl scripting language. The back-end servers on the core node run a MySQL DBMS. The database is replicated on each back-end server and contains the data required by the offered services, for example, a list of banner images used by the banner insertion service. Furthermore, the DBMS handles the user profiles. Each edge node is composed by two servers: a front-end server running an Apache Web server that is equipped to carry out the request dispatching process by means of additional modules, and an application server, that hosts a query cache to accelerate queries frequently issued to the back-end of the core node. The query cache is similar to the caching module proposed in other middleware systems [13, 22].

The prototype for the experiments is deployed on 17 physical servers with the same capacity, where 9 servers compose the core node and 8 physical servers are used for the edge nodes. The core node consists of 1 Web switch, 2 HTTP servers, 4 application servers, and 2 back-end servers, while each edge node consists of 1 front-end server and 1 application server. All the servers are connected through a fast Ethernet network. Since the generation and adaptation tasks considered in our experiments are characterized by a significant computational cost, the system load mostly lies on the application servers. Hence, we choose to utilize the same number of application servers on the core and the edge nodes to provide the same computational capacity for a fair comparison among the different dispatching mechanisms, with no influence due to the specific architectural choices. The CPU utilization of the application servers is collected through the System Activity Reporter (SAR).

4.3 Network

To evaluate the impact of network delays on the performance of the dispatching mechanisms, we emulate Wide Area Network effects on the links connecting the edge and the core nodes. We do not emulate WAN effects on the link connecting the client to the edge nodes since it does not affect the results of the comparison of the dispatching mechanisms. We use the *netem* packet scheduler (part of the Linux kernel) to emulate network delays on the links between the edge and the core nodes. The netem scheduler introduces packet delay and loss to mimic WAN effects. We add also a token bucket filter scheduler to add bandwidth limitation effects in order to achieve a full WAN emulation. We consider three scenarios where the mean value for the delay on the edge-to-core links is set to 10, 40, 100 ms. The emulated WAN effects include also packet loss (set to 1%) and bandwidth limitation (10Mbit/s). Round trip delays, loss rates and bandwidth considered in the network scenarios are consistent with the datasets from real-world measurements used in [28].

5 Experimental results

In this section we present the performance evaluation of the request dispatching mechanisms for different workload and network scenarios. We consider as the main performance metric the Web resource response time, that is measured as the time between the client request and the arrival of all the components of the Web resource (see Figure 2 for the service of a Web resource).

5.1 Performance comparison

Figure 3 shows the cumulative distribution of the Web resource response time for the *coarse-grain*, *medium-grain* and *fine-grain* dispatching mechanisms under the workloads WL1, WL2, and WL3 described in Section 4. We note that the impact of request dispatching granularity on performance strongly depends on the characteristics of the workload. The *fine-grain* dispatching provides slightly worse performance than that of the other two dispatching alternatives for the workload WL1, where all the requests have similar computational requirements, while it outperforms both the *medium-grain* and *coarse-grain* dispatching schemes for the workloads WL2 and WL3, characterized by highly variable computational requirements. We now analyze the performance of the three dispatching mechanisms for each workload.

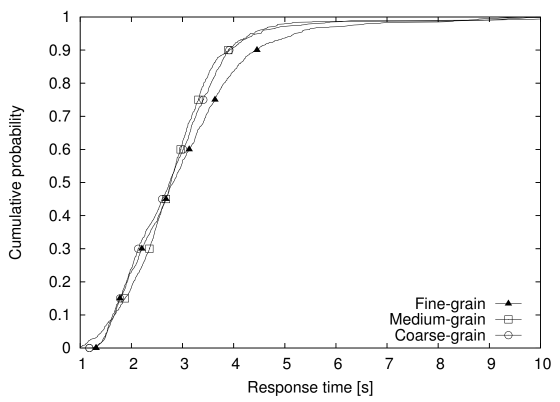
Figure 3(a) shows the performance of the dispatching mechanisms for the workload WL1. We see that the *medium-grain* and *coarse-grain* dispatching schemes achieve almost identical performance, while the *fine-grain* dispatching achieves slightly higher response time. This result demonstrate that, when all requests have similar computational requirements, a fine granularity at the level of single Web resource components is not able to improve the system performance with respect to dispatching mechanisms that operate on classes of components or entire Web resources. Moreover, we should consider that a *fine-grain* dispatching causes a higher overhead on the system if compared to medium or coarse dispatching, since it requires operations on the user profile information and Web resource template at the beginning of each Web resource request.

For the workloads WL2 and WL3 the *fine-grain* dispatching achieves better performance than that of *medium-grain* and *coarse-grain* alternatives, as shown in Figures 3(b) and 3(c).

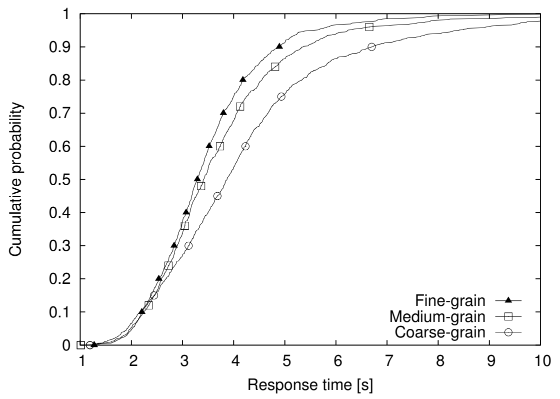
In Figure 3(b), we see that for the workload WL2 the *fine-grain* dispatching achieves slightly lower response time if compared to the *medium-grain* dispatching, but it significantly ouperforms the *coarse-grain* alternative, with a difference of 29.7% on the 90-percentile of the response time. These results demonstrate that, in a scenario where the computational requirements of the Web resource components may be heterogeneous, a coarse granularity does not provide the request dispatching with the flexibility necessary to efficiently control the load on the edge nodes. On the other hand, a finer granularity at the level of component classes or individual components allows to better exploit the edge nodes capabilities and, consequently, to process a higher number of resource components on these nodes without overloading them.

If we now consider the performance of the dispatching mechanisms under the workload WL3, shown in Figure 3(c), we see that the *fine-grain* dispatching allows significantly lower response time if compared to the other two dispatching mechanisms, which achieve similar performance. This result shows that, in a scenario where the computational requirements of single Web resource components are very heterogeneous, request dispatching needs to operate with a very fine granularity at the level of individual components to fully exploit the edge nodes capabilities.

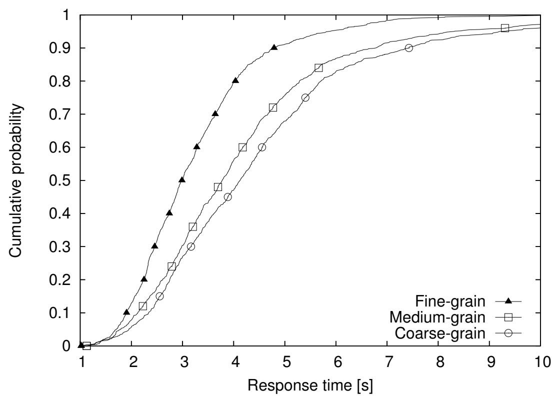
To summarize the main results of our analysis, we evidence the importance of choosing the correct level of granularity on the basis of the characteristics of the expected workload. Indeed, when the computational requirements are similar for any request, all the considered dispatching mechanisms achieve comparable performance, but a coarse granularity is preferable since it places less overhead on the



(a) WL1 workload



(b) WL2 workload



(c) WL3 workload

Figure 3. Cumulative distribution of response time

system. On the other hand, in the case of very heterogeneous workloads a fine dispatching granularity is necessary to efficiently control the nodes load.

5.2 Wide Area Network effects

Due to the distributed nature of the considered Web system, the user-perceived response time may be affected by network delays. Indeed, every Web resource component processed on the core node has to pay an additional network delay if compared to the components that are processed on the edge nodes. It seems interesting to investigate the impact of WAN effects on the overall performance of the dispatching mechanisms. Table 2 shows the 90-percentile of the response time for the dispatching mechanisms as a function of the mean delay on the edge-to-core link. The reported results are referred to the workload WL3. The results of experiments for the workloads WL1 and WL2 lead to similar conclusions.

Table 2. 90-percentile of response time [s]

Dispatching	Mean delay on edge-to-core link		
mechanism	10 ms	40 ms	100 ms
Fine-grain	5.38	5.71	6.38
Medium-grain	5.75	6.28	7.36
Coarse-grain	6.46	7.14	8.32

From Table 2 we observe that network delays have different impact on the performance of the dispatching mechanisms. As the mean delay on the edge-to-core link passes from 10 to 100 ms, the increment of the 90-percentile of the response time is 0.95 s for the *fine-grain* dispatching, while it reaches almost 2 s for the *medium-grain* and *coarse-grain* alternatives. This result can be explained if we consider that the *fine-grain* dispatching allows a more efficient utilization of the edge node computational power if compared to other dispatching mechanisms that operate with a less fine granularity. The *fine-grain* dispatching permits to assign a higher number of components to the edge node without overloading it, thus limiting the amount of edge-to-core communications that negatively affects the resource response time. We conclude that a fine granularity of request dispatching makes the system performance less sensitive to network delays on the links connecting edge and core nodes.

6 Related work

Much interest has been focused on request dispatching at the level of cluster-based Web systems [5], where context-aware dispatching mechanisms have been exploited to allow a distribution at the granularity of individual requests among the nodes of a cluster. Fine-grain dispatching has been proved to provide better performance with respect to a coarse-grain dispatching thanks to a good load balance among the cluster nodes, although it introduces a significant overhead at the Web switch level that may limit Web cluster scalability [4]. Solutions for a scalable fine-grain dispatching have been proposed in [9, 26], however, these studies focus on cluster-based systems distributed in a local area, while our approach targets geographically distributed Web systems.

Request dispatching is one of the main issues addressed by geographically distributed systems, such as multi-clusters or Content Delivery Networks [19, 25]. The majority of the studies on distributed systems mainly focuses on static or streaming contents, while we consider a highly dynamic and personalized scenario. Kangasharju *et al.* [18] found that, in a static scenario, full redirection schemes using coarse-grain dispatching achieve better performance with respect to fine-grain schemes that split requests for Web resource components among several servers. Only more recently, the research community and the industrial world have investigated the concept of edge computing, that pushes the computation on the edge of the network for better efficiency and performance. Rabinovich *et al.* [24] proposes to move content generation on the edge servers in a CDN-like environment. The proposal is similar to the Akamai EdgeComputing platform presented by Davis *et al.* in [12]. However, both these proposals rely on DNS-based approaches for request distributions, hence, they do not consider a fine-grain dispatching at the level of single Web page components, as we do in this paper.

To perform a fine-grain dispatching at the level of single Web page components, we exploit the composition of Web pages based on fragments. In the last few years, the generation of Web pages from fragments has become increasingly common, as testified by the proposals in [7, 20] of fragment-based publishing systems. Many research studies have exploited a fragment-based approach to improve Web content delivery. The ESI proposal [27] is an attempt to develop a standard protocol for managing fragments and assembling them at the network edge in response to a client request. Datta *et al.* [11] proposes a proxy cache that stores static fragments remotely and performs assembly of a response. Algorithms for cache management that include fragment invalidation and prefetching have been proposed in [6] to improve the delivery of dynamic Web contents. The fundamental difference with these studies is that our proposal replicates the computation to generate and/or adapt Web contents on the network edge, while the other studies handle only responses and leave the computation on the origin server.

7 Conclusions

In this paper we investigate the impact of request dispatching granularity on the performance of distributed Web systems for personalized contents. We consider three dispatching mechanisms, namely *fine-grain*, *medium-grain* and *coarse-grain*, that operate with different levels of granularity. Through a prototype, we compare the performance of the three dispatching mechanisms for different workload and network scenarios. We found that the choice of the best dispatching granularity strongly depends on the workload characteristics. When the requests have similar computational requirements, any dispatching granularity provides similar performance: in this case, a coarse granularity allows the deployment of simple dispatching mechanisms that cause a lower overhead with respect to fine-grain dispatching. On the other hand, when the workload is highly heterogeneous, fine-grain dispatching allows a more effective load sharing, that provides better performance if compared to a coarse-grain dispatching.

References

[1] T. Abdelzaher, K. Shin, and N. Bhatti. Performance guarantees for Web server End-Systems: a control-theoretical approach. *IEEE Transactions on Parallel and Distributed Systems*, 13:80–96, Jan. 2002.

[2] A. Arlitt, B. Krishnamurthy, and J. Rolia. Characterizing the scalability of a large Web-based shopping system. *ACM Transactions on Internet Technology*, 1(1):44–69, 2001.

[3] M. Aron, P. Druschel, and W. Zwaenepoel. Efficient support for P-HTTP in cluster-based Web servers. In *Proceedings of the USENIX 1999*, Monterey, CA, Jun. 1999.

[4] M. Aron, D. Sanders, P. Druschel, and W. Zwaenepoel. Scalable context-aware request distribution in cluster-based network servers. In *Proceedings of the USENIX 2000*, San Diego, CA, Jun. 2000.

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

[6] J. Challenger, P. Dantzig, A. Iyengar, M. Squillante, and L. Zhang. Efficiently serving dynamic data at highly accessed Web sites. *IEEE/ACM Transaction on Networking*, 12(2):233–246, 2004.

[7] J. Challenger, P. Dantzig, A. Iyengar, and K. Witting. A fragment-based approach for efficiently creating dynamic Web content. *ACM Transaction on Internet Technology (TOIT)*, 5(2):359–389, 2005.

[8] S. Chandra. Content adaptation and transcoding. *Practical Handbook of Internet Computing*, 2004. (Munindar P. Singh ed.), Chapman Hall & CRC Press.

[9] L. Cherkasova and M. Karlsson. Scalable Web server cluster design with WARD. In *Proceedings of the 3rd International Workshop on Advenced Issues of E-commerce and Web-based Information Systems*, pages 212–221, Los Alamitos, CA, June 2001.

[10] M. Colajanni, R. Lancellotti, and P. Yu. Distributed architectures for Web content adaptation and delivery. *Web content delivery*, 2005. (Tang, Xu, Chanson eds.), Springer.

[11] A. Datta, K. Dutta, H. Thomas, D. Vandermeer, and K. Ramamritham. Proxy-based acceleration of dynamically generated content on the World Wide Web: an approach and implementation. *ACM Transactions on Database Systems*, 29(2):403–443, 2004.

[12] A. Davis, J. Parikh, and W. E. Weihl. EdgeComputing: extending enterprise applications to the edge of the Internet. In *Proceedings of WWW'04 on Alternate Track Papers & Posters*, pages 180–187, 2004.

[13] L. Degenaro, A. Iyengar, I. Lipkind, and I. Rouvellou. A middleware system which intelligently caches query results. In *Proceedings of Middleware'00*, pages 24–44, New York, USA, 2000.

[14] M. D. Dikaiakos. Intermediary infrastructures for the World Wide Web. *Computer Networks*, 45(4):421–447, Jul. 2004.

[15] M. Eiriniaki and M. Vazirgiannis. Web mining for Web personalization. *ACM Transaction on Internet Technology*, 3(1), 2003.

[16] S. Elnikety, E. Nahum, J. Tracey, and W. Zwaenepoel. A method for transparent admission control and request scheduling in E-commerce Web sites. In *Proceedings of WWW'04*, New York, NY, May 2004.

[17] A. Iyengar, L. Ramaswamy, and B. Schroeder. Techniques for efficiently serving and caching dynamic Web content. *Web content delivery*, 2005. (Tang, Xu, Chanson eds.), Springer.

[18] J. Kangasharju, K. W. Ross, and J. W. Roberts. Performance evaluation of redirection schemes in content distribution networks. *Computer Communications*, 24(2):207–214, 2001.

[19] M. Karlsson. Replica placement and request routing. *Web content delivery*, 2005. (Tang, Xu, Chanson eds.), Springer.

[20] P. Mohapatra and H. Chen. A framework for managing QoS and improving performance of dynamic Web content. In *Proceedings of IEEE GLOBECOM 2001*, San Antonio, USA, Nov. 2001.

[21] MyYahoo, 2007. `http://my.yahoo.com`.

[22] C. Olston, A. Manjhi, C. Garrod, A. Ailamaki, B. Maggs, and T. Mowry. A scalability service for dynamic Web applications. In *Proceedings of CIDR 2005*, Asilomar, CA, Jan 2005.

[23] M. Rabinovich and O. Spatscheck. *Web Caching and Replication*. Addison Wesley, 2002.

[24] M. Rabinovich, Z. Xiao, and A. Aggarwal. Computing on the edge: A platform for replicating Internet applications. In *Proceedings of 8th International Workshop on Web Content and Distribution*, Hawthorne, NY, Sept. 2003.

[25] S. Sivasubramanian, M. Szymaniak, G. Pierre, and M. van Steen. Replication for Web hosting systems. *ACM Computing Surveys*, 36(3):291–334, 2004.

[26] J. Song, A. Iyengar, E. Levy-Abegnoli, and D. Dias. Architecture of a Web server accelerator. *Computer Networks*, 38(1):75–97, 2002.

[27] L. Tsimelzon, B. Weihl, and L. Jacobs. Edge Side Includes language specification 1.0. `http://www.esi.org/language_spec_1-0.html`.

[28] R. Zhang, C. Hu, X. Lin, and S. Fahmy. A hierarchical approach to Internet distance prediction. In *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems (ICDCS'06)*, Washington, DC, USA, Jul. 2006.