Architectures for scalable and flexible Web personalization services

Claudia Canali
University of Parma
claudia@weblab.ing.unimo.it

Sara Casolari, Riccardo Lancellotti
University of Modena and Reggio Emilia
sara@weblab.ing.unimo.it, lancellotti.riccardo@unimore.it

Abstract

The complexity of services provided through the Web is continuously increasing and issues introduced by both heterogeneous client devices and Web content personalization are becoming a major challenge for the Web. Tailoring Web and multimedia resources to meet the user and client requirements opens two main novel issues in the research area of content delivery. The content adaptation operations may be computationally expensive, requiring high efficiency and scalability in the Web architectures. Moreover, personalization services introduce security and consistency issues for user profile information management. In this paper, we propose a novel distributed architecture, with four variants, for the efficient delivery of personalized service where the nodes are organized in two levels. We discuss how the architectural choices are affected by security and consistency constraints as well as by the access to privileged information of the content provider. Moreover we discuss performance trade-offs of the various choices.

Keywords: Web content adaptation, High performance architectures, Web content delivery

1. Introduction

The current trend in the evolution of the Web is towards an ever increasing complexity and heterogeneity. The growth in Web popularity is related to the diffusion of heterogeneous client devices, such as handheld computers, mobile phones, and other pervasive computing devices. Furthermore, the ever increasing complexity of Web services has led to an heterogeneity of offered services, which in several cases are tailored on the user preferences, leading to the deployment of personalized Web services. The need for personalization led to the introduction of content adaptation services that tailor Web resources to the user preferences and to the capabilities of their clients.

Content adaptation is an expensive task from a computational point of view. As a consequence, much interest is focused towards high performance architectures capable of providing efficient and scalable adaptation services. We can define three players in the game of content adaptation, as shown in Fig. 1: the client which issues requests, the content provider which hosts the Web resources being requested and the adaptation provider which is the intermediary entity that carries out the actual content adaptation.

The idea of having a third-party to carry out content adaptation is not the only feasible solution. Some client side solution for advertisement removal or to increase user information privacy are available (e.g. Mozilla Extensions, like Bug Me Not). An approach that adds all adaptation services to the content provider platform [14] remains a valid solution when the popularity of the content provider is medium-low. However, with the number of clients and device profiles continuously increasing (hundreds of different device profiles already exist [18]), an infrastructure that uses a geographically distributed system of intermediary nodes seems the most practicable solution among the existing alternatives [3, 12] to improve performance and scalability.

We can define two main degrees of freedom in the problem of designing such an infrastructure. Each of them can be identified by a question:

1. which adaptation services are to be offered?
2. which relationship exists between the previously defined players?

The first question is motivated by the plethora of adaptation services that can be deployed: such services can adapt Web content to both the client device (*transcoding*) or to the user preferences (*personalization*). In the latter case, personalization can be carried out on the basis of previously stored information (*state-aware personalization*) or not (*state-less personalization*).

As for the second question, we recognize two possible scenarios depending on whether the adaptation provider has privileged access to the resources of the content providers (including Web content, related meta-data, and databases) or not.

Each of the two questions affects the architectural choices for the design of distributed content adaptation architectures.

Figure 1. Actors in content adaptation

The contribution of this paper is twofold. First, we propose a two-level architecture for scalable content adaptation services. Such architecture covers a wide spectrum of adaptation services that ranges from transcoding to sophisticated state-aware personalization. Furthermore, we propose four variants of the basic two-level architecture which exploit the characteristics of the two levels and the different relationship between adaptation and content provider.

Second, we discuss the performance issues of the proposed architectural choices outlining for each of them which choice can provide better performance and under which circumstances the performance gain is more evident.

The remainder of this paper is organized as follows. Section 2 describes the basic two-level architecture and introduces two variant architectures to address the issues related to the class of content adaptation to be deployed. Section 3 describes the two variants of the basic two-level architecture depending on the relationship between the adaptation and the content providers. Section 4 describes the model used to discuss performance issues of the two-level architectures, while the actual qualitative evaluation of the performance is provided in Section 5. Section 6 presents some related work. Finally Section 7 provides some concluding remarks.

2. Content adaptation in a two-level architecture

We now introduce the basic functions of a Web content adaptation system. The main issue in the design of a distributed architecture for content adaptation is the mapping between these functions and the nodes composing the content adaptation infrastructure. Not every mapping is possible and the type of adaptation has a significant impact on the architectural choices, as we will discuss in the following of this section.

We can distinguish between actual adaptation tasks and support functions. For content adaptation we can refer to three main classes of services, namely *transcoding*, *state-less personalization*, and *state-aware personalization*. Transcoding is the services of tailoring Web content to the capabilities of the client device and the network connection. For example, we can reduce the quality factor of an image with the goal of reducing the size of the file to be delivered. The same goal can be achieved by means of compression techniques. Another significant example is scaling down of graphic Web resources to fit small displays.

State-less personalization basically refers to the adaptation services that extract user preferences from the client request without combining them with previously stored information. Some examples of these personalization services include: advertisement removal, virus scanning, and insertion of random banners.

State-aware personalization refers to the personalization services that are based on some stored information. This information is typically contained in some database(s) and can be extracted as a consequence of explicit information coming from the user request or inferred through the run-time or off-line analysis of the user behavior (e.g., through data mining on log files of a Web site). Hence, the user profile typically consists of the parameters for the personalization services as well as additional information such as the recent user history that can be analyzed to auto-adapt the service to the user behavior. The use of stored information allows the infrastructure to integrate the above state-less operations with richer services. A non exhaustive list of such services includes personalization based on previously registered user profile, adaptation to the user navigation style, adaptation to the user interests, content filtering (for kids), location and surrounding-based services that achieve personalization on the basis of the user geographic location and services involving cryptographic digital right management.

Support functions are mainly related to the management of client requests and additional data. This consists in request parsing and extraction of useful information (e.g., user identity, preferred languages, and browser characteristics) and in forwarding the requests to the content adapter which can be either on a local or remote node. The support function also comprises the storage and management of additional information. For example the system may need to handle user profiles describing personalization preferences for each user. Another significant example is related to the caching of Web resources, fragments and meta-data associated with them. However, caching is not a critical task in this context because the personalization of resources hinders the effectiveness of most caching algorithms.

We propose an architecture which is based on a subdivision of nodes in two levels namely *edge* and *internal*, as shown in Fig. 2.

The edge level is characterized by a large amount of nodes, each located close to the network edge. Such

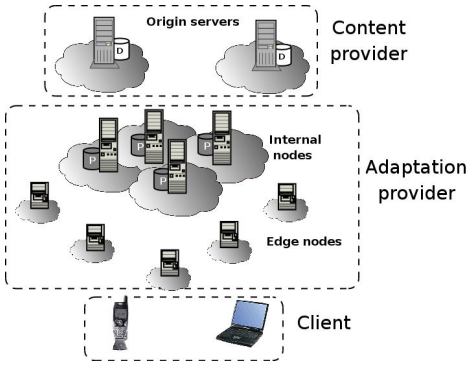

Figure 2. Two-level architecture

nodes are usually placed in the points of presence of ISPs to be as close as possible to clients, as shown in the figure. The internal level of the content adaptation architecture is composed by a smaller number of powerful nodes. Such nodes can be placed in *well connected* locations, which means in Autonomous Systems with an high peering degree, to reduce communication costs, especially with respect to the edge nodes. Fig. 2 also shows the origin servers belonging to the content provider which are not part of the content adaptation infrastructure and host the repository of the Web resources (shown as the white data storage attached to the origin server nodes).

The characteristics of the edge nodes bring issues and limitation for the deployment of adaptation services. First, since the number of edge nodes is high, such nodes must be as simple as possible to reduce management issues. Furthermore, special care should be devoted to consistency in the information stored on the edge nodes, which means that data replication should be avoided whenever possible [11]. The second issue is related to security. Since edge nodes are not under the strict control of the adaptation provider, sensitive information related to the user profile should not be stored in this level of the adaptation infrastructure.

Due to the limited replication of the internal nodes, we can guarantee higher security standards for them. Hence, these nodes are more suitable for adaptation tasks requiring sensitive information stored in the user profile. For this reason, Fig. 2 shows the user profile storage as a gray data storage attached to the internal nodes. Furthermore, computational power in these nodes is not an issue because, thanks to the reduced degree of replication, we do not have the management problems of the edge nodes. Sophisticated local replication strategies can provide the amount of computation power required. An example of local replication is clustering [4] in which a Web switch is placed in front of the system to transparently distribute requests evenly among the nodes of the cluster. With such an approach the only interface to the system is the switch, which means that computational power can be seamlessly improved by adding nodes to the cluster.

The final issue to be addressed is related to guaranteeing adequate data consistency. The most common approach is to reduce data replication. Literature shows that data consistency can hardly be provided over more than 10 nodes and the bound of synchronous updates is to be relaxed if more replication is needed [11]. Our approach to reduce data replication is based on hashing mechanisms that *partition* the space of user profiles. The user profile is replicated only on one internal node (or few nodes, depending on the hash algorithm used and on whether backups are kept) to avoid replica consistency issues.

We define two different variants of the base architecture to map adaptation functions on the two levels. The first, namely *strict* two-level architecture, forces state-less personalization and transcoding services to be deployed on edge nodes, while state-aware personalization are to be carried out on the internal nodes.

A more flexible and sophisticated architecture variant, namely *relaxed* architecture, could replicate the portion of the user profile not containing sensitive information to the edge nodes to move as much adaptation tasks as possible close to the clients. This additional feature can be implemented at the cost of only one additional profile replica because the user is typically connected only to one edge server at once which represents the entry point to the content adaptation infrastructure. Whenever the user changes its access point, the profile is moved to the new edge node according to the user migration.

The development of this partial profile replication requires a support for a finer control in the user information management system. Each entry in the user profile must be enriched with attributes to describe whether the entry contains sensitive information or not, these attributes can also be used to export some parameters in a read-only mode. These visibility and write mode attributes can be further exploited to create the so called group profiles [2]. A group profile is related to a set of users instead of a single user. A group leader is a special user which has a read/write access to the profile, while other members of the group can access the group profile in a read-only mode.

3. Relationship between adaptation and content providers

In our study we take into account two different scenarios depending on whether the content adaptation

architecture has privileged access to content provider information or not. We propose two variants of the base two-level architecture, namely *independent* and *custom* architecture. The two variants exploit the different relationship between the Web content and the adaptation providers (that is, the answer to the second question outlined in the beginning of the paper).

In the independent architecture the (possibly paying) customers of the Web content adaptation service are the end users. The content adaptation provider is typically an ISP or some network operator which provides its customer with the service of tailoring *any* Web site accessed to their needs and preferences. The adaptation service provider tailors the content delivery service on the end-user requirements. An example of such service is the AvantGo company `http://www.avantgo.com/` which enables Web access to PDAs by tailoring the Web page layout and the embedded object size to devices with small displays. The service provided by AvantGo is mainly related to state-less personalization and transcoding, however it is a real-world example of independent architecture. Such service can be provided to both single users as well as institutions or companies willing to enable ubiquitous Web access for their employees.

In this architecture there is no preferential access from the adaptation provider to the origin Web servers. As a consequence the adaptation provider has no way to know the semantic of the Web applications if we exclude what can be inferred by analyzing the user interactions. The lack of this knowledge hinders the development of sophisticated adaptation services which could take advantage from the knowledge about the Web page content. A typical example is provided by the insertion of context-sensitive advertisement banners: if the personalization system has no idea about the page being sent to the user, the banners can be only related to a previously stored user interests list, but cannot be tailored on the user context. Furthermore user preferences cannot be easily extracted from the user behavior, and cannot be automatically tuned without recurring to computational expensive data mining to be carried out off-line.

Nevertheless, this architecture still allows the deployment of interesting adaptation services. State-less personalization and transcoding tasks can be easily carried out. In a similar way, state-aware personalization tasks which are not context sensitive (such as social navigation) can be easily provided.

Fig. 3 shows the independent architecture. The structure is similar to the base two-level architecture described in Fig. 2, with a significant amount of distributed edge nodes and few internal nodes. When a client request is received by an edge node (Step 1 in Fig. 3), the edge node decides based on the requested adaptation service if the content adaptation is to be carried out locally (Step 2) or if the request is to be forwarded to an internal node (Step 3) for processing (Step 4). In both cases, since we assume that no caching is to be adopted, a fetch operation from the origin server (Steps 5a or 5b) has to be carried out.

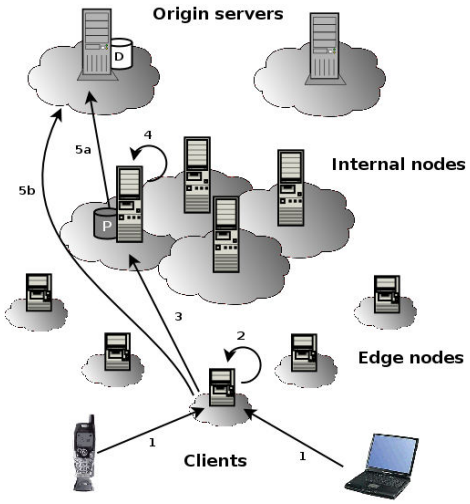

Figure 3. Independent architecture

In the custom architecture, the customer of the adaptation service is the Web content provider. In this case we have a strict collaboration between the content and the adaptation service providers. This case is common in the Content Delivery Network model, when a content provider outsources the delivery of Web content and Web services to a third-party operating a distributed infrastructure. Most CDNs focus on the delivery of static content, although some research has been devoted to exploit intermediary nodes for the generation of dynamic documents [15]. In this case, it is a natural evolution to provide also content adaptation in the intermediary nodes of the CDN.

The strict relationship between the content provider and the adaptation provider allows a close interaction of the adaptation and delivery services with resource replication strategies and the semantics of the Web-based services. Besides replications governed by client requests, it is possible to have pro-active replications in which the content provider pushes its content on the internal nodes of the infrastructure. The replication is not limited to standard Web content but also includes meta-data on how content must be adapted. This allows the deployment of *server-directed* adaptation similar to what suggested by Knutsson *et al.* [13] for

transcoding. The rich content replication also enables generation of Web contents based on the assembly of fragments of Web resources, using mechanisms similar to the ESI system [8]. The joint use of content-assembly functions and server-directed adaptation, united to the access to content-providers databases, allows the deployment of complex state-aware adaptation services which are not feasible in the independent architecture, such as the creation of information portals which aggregate multiple sources according to the user preferences. On the other hand, the deployment of simpler adaptation services, such as state-less personalization and transcoding, that do not require previously stored information other than that directly supplied by the client can take full advantage of the highly distributed and replicated nature of edge nodes.

In Fig. 4 we present the custom architecture. The main difference between the custom and the base two-level architecture is the presence of a replica (full or partial) of the origin server on the same network of the internal nodes. The asynchronous push-based update mechanism of such replica is shown in Fig. 4 and is marked with the label A.

The dynamic process of client request service is similar to that described in Fig. 3: client requests (Step 1) can be either processed on edge nodes (Step 2) or forwarded to internal nodes (Steps 3 and 4). Unlike the previously described architecture, the fetch process (Steps 5a and 5b) involves the origin server replica instead of the origin server itself.

In both cases of full or partial origin server content replication, however, the problem of consistency arises. Indeed, the custom architecture demands special care for information consistency because we must preserve it both in the user profiles and in the Web content replicas. Multiple approaches have been proposed to preserve consistency in Web caching [20, 19], but such approaches cannot be easily adapted to the consistency and security issues related to the delivery of personalized contents. We hence prefer to preserve consistency by avoiding replication as much as possible by means of hashing, as described in the previous section. There is a clear trade-off between consistency of the two types of information: a strict hashing in the user identity requires the replication of Web content on every internal node, while an hash calculated only on the basis of the requested resource would lead to an unacceptable replication of user profiles. It seems then promising a more sophisticated hashing scheme that uses both the user ID and the resource ID to select the right internal node. Such an algorithm selects a group of internal nodes based on one information (e.g., the user ID) and then selects the right node within the group based on the other (in the example the resource ID).

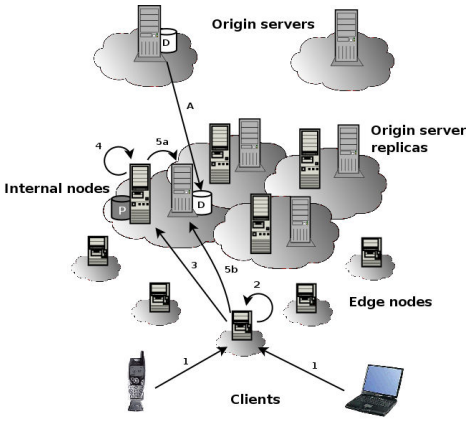

Figure 4. Custom architecture

4. Performance model

We now introduce the model that will be used to evaluate the performance of the proposed architectures. The model does not aim to be exact from a quantitative point of view: we use a simplified model to draw qualitative considerations on the performance impact of several architectural choices.

$$T_r = \begin{cases} T_{r_{Edge}} & \text{if adaptation on edge node} \\ T_{r_{Inner}} & \text{otherwise} \end{cases} \quad (1)$$

$$T_{r_{Edge}} = T_{Net_{C-E}} + T_{Adapt} + T_{Fetch} \quad (2)$$

$$T_{r_{Inner}} = T_{Net_{C-E}} + T_{Net_{E-I}} + T_{Adapt} + T_{Fetch} \quad (3)$$

The main performance parameter that we will use to describe performance is the user-perceived response time of the system T_r defined in Equation 1. In the case when the client request is serviced by the edge layer $T_{r_{Edge}}$, the response time can be expressed as in Equation 2, as the sum of the contribution due to network time from the edge node to the client $T_{Net_{C-E}}$, and the time for the actual adaptation time T_{Adapt} and to the time due to the interaction with the origin Web server T_{Fetch}. The case where the content adaptation is carried out by the internal layer is modeled by Equation 3. The response time $T_{r_{Inner}}$ is similar to the previously described case of $T_{r_{Edge}}$, with the only significant difference that the response time has an additional term which represents the network delay due to the communication between the two layers of the adaptation infrastructure $T_{Net_{E-I}}$.

The terms that contribute to the response time can be further explained to better understand how they can affect the system performance.

$$T_{Net} = T_{Latency} + \frac{Size}{BW} \quad (4)$$

Network time contribution can be defined as the sum of latency and actual transfer time. The latency $T_{Latency}$ depends only on network characteristics and status, while transfer time depends on both the available bandwidth BW and on the amount of data being transferred $Size$, as shown in Equation 4. We report in Table 1 some common values for both latency and and bandwidth for typical network settings, that allow to calculate the bounds of the contribution of network-related delays. Another contribution to the response time which deserves further inspection is the adaptation time T_{Adapt}. This value is characterized by high variability. Previous studies show that content adaptation can take from few milliseconds to seconds. We carried out a set of simple experiments and compared our results with the literature. Our tests replicate the critical operations of content adaptation and personalization and measure the time taken by the task. The node used for those experiments is equipped with a 1 GHz Pentium CPU and 1Gb of RAM. Transcoding services, in particular the ones involving compression and image manipulation are highly CPU intensive. For an image resizing on banner images the time T_{Adapt} is in the order of 100 ms, while larger image (such as photos) may take up to 1 s for transcoding. This is consistent with previous studies on content adaptation which report that the transcoding time is in the range of hundreds of milliseconds or even seconds [5]. On the other hand less intensive adaptation tasks such as advertisement/link removal or banner insertion, which are typical for state-less operation require only simple string manipulation. Our experiments show that string manipulation on out experimental testbed take a time in the order of 10 ms even for large HTML files. Furthermore, caching can be effective in reducing the cost of transcoding and state-less personalization if the adaptation service can be described in terms of providing few standard versions of each Web resource. On the other hand operations such as virus scanning are more intensive than simple HTML code manipulation. In this case time up to 100 ms can be taken to scan a single file.

State-aware services are extremely heterogeneous and can require different operation ranging from the user preferences-based advertisement insertion to the cryptographic operation of a digital right management system.

Table 1 summarizes the typical order of magnitude for the adaptation time depending on the type of content adaptation carried out.

Parameter		Minimum value	Maximum value
$T_{Net_{C-E}}$			
	$T_{Latency}$	10 ms	10^2 ms
	BW	9 Kbit/s	2 Mbit/s
$T_{Net_{E-I}}$			
	$T_{Latency}$	10 ms	10^2 ms
	BW	1 Mbit/s	50 Mbit/s
T_{Adapt}			
	$T_{Transcoding}$	10^2 ms	10^3 ms
	$T_{State-less}$	10 ms	10^2 ms
	$T_{State-aware}$	10 ms	10^3 ms
$T_{Net_{*-O}}$			
	$T_{Latency}$	10 ms	10^2 ms
	BW	1 Mbit/s	50 Mbit/s
T_{Server}		1 ms	10^2 ms

Table 1. Parameters of the model

$$T_{Fetch} = T_{Net_{*-O}} + T_{Server} \quad (5)$$

The final contribution to the response time is related to the time taken for the interaction with the origin Web server, which we describe as the *fetch time*, T_{Fetch}. Equation 5 shows that the fetch time is composed by the sum of the time due to network data transfers, that is $T_{Net_{*-O}}$ and the time taken by the origin Web server to respond T_{Server}. While we already discussed the characteristics of Wide Area Network delays, the contribution of T_{Server} requires some more explanation. Servicing a static Web page is a simple task which usually takes less than 1 ms. On the other hand, modern Web systems often involve dynamic generation of Web contents which can require up to hundreds of milliseconds, as reported in literature [6].

5. Architecture evaluation

From the previously described model we can draw a qualitative evaluation of the architectural choices sketched in Sections 2 and 3. We are interested in understanding which architectural choices provide better performance and under which circumstances the difference is more evident.

We start our analysis by focusing on the different content adaptation functions. From Section 2 we recall that in the strict architecture state-aware personalization is carried out on the internal nodes, while other forms of content adaption are carried out on the edge nodes. On the other hand the relaxed architecture tries to shift also non-critical state-aware personalization on the edge level.

Since we cannot place any constraint on $T_{Net_{C-E}}$ nor on T_{Fetch}, we focus on the remaining contribution

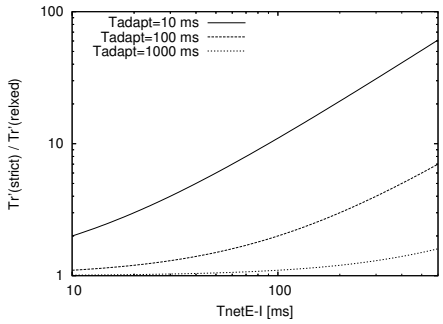

Figure 5. Comparison of strict and relaxed architectures

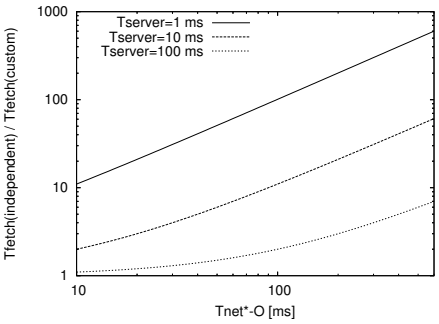

Figure 6. Comparison of independent and custom architectures

to the response time as defined in the previous section. In particular we focus our analysis on the case of state-aware content adaptation in which the two architectures provides different performance. Let $T'_r(strict) = T_{Adapt} + T_{Net_{E-I}}$ and $T'_r(relaxed) = T_{Adapt}$. Fig. 5 shows the ratio $\frac{T'_r(strict)}{T'_r(relaxed)}$ for the two architectures as a function of the network delay between edge and internal nodes ($T_{Net_{E-I}}$) and adaptation time (T_{Adapt}). The graph can be easily understood by considering Equation 3. If $T_{Adapt} >> T_{Net_{E-I}}$, then the penalty (in the form the network delay) to be paid to carry out content adaptation on the internal node is negligible if compared with the time taken by the actual content adaptation task. Hence, in the case of computational-intensive personalization, the level that actually carries out content adaptation is not an issue and both strict and relaxed architectures are viable solutions. On the other hand, in the case of light personalization, such as banner insertion/removal, T_{Adapt} is comparable with $T_{Net_{E-I}}$ (if not lower). Hence the choice of using edge nodes for the adaptation brings significant performance gain. Obviously the lighter is the adaptation service, the bigger is the benefit from using the edge node for its deployment.

We now compare the performance of the independent and custom variants of the two-level architecture. The main difference between independent and custom architectures lies in the T_{Fetch} expression: for the custom architecture the network contribution $T_{Net_{*-O}}$ is negligible and T_{Fetch} is related only to the server time T_{Server}.

The performance difference between the two variants of the architecture is more evident in the case of resources where T_{Server} is negligible, as in the case of static resources, while the performance difference is less significant when the origin server generates dynamic resources with time consuming calculations. Fig. 6 shows the performance gain in T_{Fetch} depending on the origin server time and on the network delays. If we consider a $T_{Server} = 100ms$, we observe that the ratio between the fetch time of the independent and the custom architectures is not highly affected by network delay and is close to 1, while for very fast Web server responses ($T_{Server} = 1ms$) the custom architecture provides much better performance.

6. Related Work

Content adaptation has been an interesting topic in Web-related literature of the last years. We can ascribe most contribution to two large groups of research topics: proposal of adaptation services and proposal of efficient and scalable architectures. Most studies on content adaptation services propose novel sophisticated adaptation systems. Examples include text-to-speech conversion [1] directed to single users as well as collaborative Web browsing [2]. Even more interest has been devoted to the proposal of transcoding services aiming to enable ubiquitous Web access from heterogeneous client devices [7, 3]. However, most of these studies does not take into account performance issues and the only proposals to improve system scalability are related to the introduction of caching [17] or locally distributed clusters [9].

Studies proposing scalable content adaptation and delivery systems, evaluate distributed architectures of collaborative intermediary nodes. Different architectures has been evaluated ranging from flat and hierarchical schemes [5]. However most of these studies only focus on transcoding services, which does not introduce significant security and consistency problems. A noteworthy proposal is a peer-to-peer content adaptation system [16] called Tuxedo which allows ubiquitous Web access providing both personalization and transcoding services. However, the study does not eval-

uate in deep detail security issues arising from the distribution of sensitive information among untrustworthy nodes. The importance of providing security and consistency by controlling information replication has been pointed out recently in the field of distributed Web systems [10]. Although the study does not focus on Web content adaptation but is more directed towards generation of dynamic Web content, this study confirms our concern for security and consistency guarantee which imposes bounds that must be taken into account in the design of scalable distributed architectures. Other studies [20, 19] propose algorithms and protocols to ensure Web cache consistency, however, such solutions, while reducing the cost of cache update, do not address the issues related to data privacy and cannot be easily adapted to the case of personalized Web content.

7. Conclusions

In this paper we proposed a novel distributed architecture and four variants of the base two-level model. We discuss the security and consistency issues that must be addressed to provide sophisticated state-aware personalization and we show how these issues are addressed by our architecture.

We proposed two architectural variants (strict and relaxed) based on whether state-aware adaptation is to be carried out only on the internal nodes or can be deployed on the edge nodes. Furthermore we proposed two additional architecture variants (independent and custom) based on the possibility of replication of content provider information on the internal nodes.

The paper provides an evaluation of the four architecture variants. We show that the relaxed architecture outperforms the strict architecture. Such difference is more evident for light personalization services when the adaptation time is comparable with network delays: in such case the relaxed architecture is more than 10 times faster than the strict architecture. A similar performance comparison is provided also for independent and custom architectures, with custom architecture outperforming the independent architecture especially in the case when the origin Web server is much faster with respect to network delays.

References

[1] M. Barra, R. Grieco, D. Malandrino, A. Negro, and V. Scarano. Texttospeech: a heavy-weight edge service. In *Poster Proc. of 12th WWW Conference*, Budapest, HU, 2003.

[2] M. Bonnet. Personalization of Web services: opportunities and changes. *Ariadne*, (28), Jun. 2001.

[3] M. Butler, F. Giannetti, R. Gimson, and T. Wiley. Device independence and the Web. *IEEE Internet Computing*, 6(5):81–86, Sept./Oct. 2002.

[4] 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.

[5] V. Cardellini, M. Colajanni, R. Lancellotti, and P. S. Yu. A distributed architecture of edge proxy servers for cooperative transcoding. In *Proc. of 3rd IEEE Workshop on Internet Applications*, June 2003.

[6] E. Cecchet, A. Chanda, S. Elnikety, J. Marguerite, and W. Zwaenepoel. Performance comparison of middleware architectures for generating dynamic web content. In *Proc. of 4th Middleware Conference*, Jun 2003.

[7] C. S. Chandra, S. Ellis and A. Vahdat. Application-level differentiated multimedia Web services using quality aware transcoding. *IEEE J. on Selected Areas in Communication*, 18(12):2544–2465, Dec. 2000.

[8] Edge Side Includes, 2002. `http://www.esi.org/`.

[9] A. Fox, S. D. Gribble, Y. Chawathe, E. A. Brewer, and P. Gauthier. Cluster-based scalable network services. In *Proc. of 16th ACM SOSP*, pages 78–91, Oct. 1997.

[10] L. Gao, M. Dahlin, A. Nayate, J. Zheng, and A. Iyengar. Application specific data replication for edge services. In *Proc. of 12th WWW Conference*, Budapest, HU, 2003.

[11] J. Gray, P. Helland, P. E. O'Neil, and D. Shasha. The dangers of replication and a solution. In *Proc. of the 1996 ACM SIGMOD International Conference on Management of Data*, Jun. 1996.

[12] A. Joshi. On proxy agents, mobility, and Web access. *Mobile Networks and Applications*, 5(4):233–241, 2000.

[13] B. Knutsson, H. Lu, and J. Mogul. Architecture and performance of server-directed transcoding. *ACM Trans. on Internet Technology*, 3(4):392–424, Nov. 2003.

[14] R. Mohan, J. R. Smith, and C.-S. Li. Adapting multimedia Internet content for universal access. *IEEE Trans. on Multimedia*, 1(1):104–114, Mar. 1999.

[15] M. Rabinovich, Z. Xiao, and A. Aggarwal. Computing on the edge: A platform for replicating internet applications. In *Proc. of 8th Int'l Workshop on Web Content and Distribution*, Hawthorne, NY, Sept. 2003.

[16] W. Shi, K. Shah, Y. Mao, and V. Chaudhary. Tuxedo: a peer-to-peer caching system. In *Proc. of PDPTA03*, Las Vegas, NV, June 2003.

[17] A. Singh, A. Trivedi, K. Ramamritham, and P. Shenoy. PTC: Proxies that transcode and cache in heterogeneous Web client environments. *World Wide Web*, 7(1):7–28, Jan./Mar. 2004.

[18] G. Singh. Guest editor's introduction: Content repurposing. *IEEE Multimedia*, 11(1):20–21, Mar. 2004.

[19] R. Tewari, T. Niranjan, and S. Ramamurthy. WCDP: a protocol for Web cache consistency. In *Proc. of 7th Int'l Workshop on Web Content Caching and Distribution (WCW)*, Boulder, CO, aug 2002.

[20] J. Yin, L. Alvisi, M. Dahlin, and A. Iyengar. Engineering web cache consistency. *ACM Trans. on Internet Technology*, 2(3):224–259, Aug. 2002.