Building a Self-Adaptive Content Distribution Network

Gawesh Jawaheer
Department of Computing, Imperial College London
gawesh@doc.ic.ac.uk

Abstract

In this paper, we propose a framework for building a self-adaptive Content Distribution Network (CDN). Such a CDN will exhibit self-adaptive behaviour at a coarse grained granularity. It will be able to cope with serving heterogeneous resources under unpredictable conditions. We describe the concepts that underpin our framework and we discuss the issues that it raises.

1. Introduction

The phenomenal growth of the WWW is straining its traditional content distribution system whereby browsers are served content from a web server. Nowadays, very popular web sites such as those of portals like Yahoo or large multinational companies like IBM or news sites like CNN are using the Content Distribution Networks (CDNs) provided by commercial companies such as Akamai to host and serve their content [1]. A CDN typically consists of a system of globally distributed web servers which serve content on behalf of a content provider. These web servers, also known as edge servers, are strategically placed on the edge of the Internet in order to be closer to the clients they serve. Thus, CDNs aim to improve user access latency, throughput, reliability and scalability.

Initially, CDNs were used mostly to serve images [10]. However, they are now faced with demands of serving web pages with heterogeneous contents of varying demand characteristics. Already, CDNs face difficulties in provisioning of web sites in flash crowd situations [12]. In the future, we anticipate more acute problems when they will be required to serve personalised content, dynamic content and applications. Providing such capability represents a unique opportunity for CDNs as it provides a competitive advantage over client-side caching technologies [11].

The solution to these problems is to enable CDNs with self-adaptive behaviour. Self-adaptive edge servers can be scattered around the world to sustain the scalability of the system. Taking self-adaptive software as an analogy [3], we define a self-adaptive CDN as a system which evaluates and changes its own behaviour when the evaluation indicates that it is not accomplishing what the system is intended to do, or when better functionality or performance is possible. There are several implications that emerge from such a definition. First, we have to define the behaviour of a CDN. Then, we need a means of evaluating the behaviour against predefined goals of the system. Finally we need a means of implementing the changes in behaviour.

In this paper, we describe a framework for developing a self-adaptive CDN that exhibits self-adaptive behaviour at a coarse grained granularity. We provide ideas for the realisation of such a framework and discuss some issues raised by our approach. We do not discuss implementation issues as these form part of our future work.

2. Background

A CDN consists of globally distributed web sites which host web based applications [2]. Each site is usually made up of a farm of web servers. These web servers, also known as edge servers, are strategically placed on the edge of the Internet. Hence, closer to the clients they serve. CDNs operate by replicating content from an origin server, typically a web server under the control of the organisation paying for the services of the CDN, to several of these edge servers and transparently route requests from clients (web browsers) away from the origin servers onto the edge servers, hence distributing the load. Thus the requested resources are served from the edge servers. With such an arrangement, CDNs aim to improve user access latency, throughput, reliability and scalability. The essential building blocks of a CDN are:

- Origin servers: these hold the latest version of the resources. Typically these resources would be embodied in web pages. The resources are replicated from the origin servers onto edge servers. Ideally, the origin server would serve to users only the web pages which embody the resources.
- Edge servers: these serve resources to users. Typically, user requests to an origin server are transparently redirected to the edge servers. Whenever, a resource is unavailable on an edge server, it is requested from its origin server.
- DNS servers: these form part of the redirection mechanism. The URLs of the resources embodied in the web pages served by the origin server are resolved by these DNS servers. This enables them to redirect users to the edge servers.
- Load balancing mechanisms: these dictate how the resources are replicated on the edge servers. They also dictate to which edge server to redirect a request for a resource.

CDNs serve various types of resources from a varied set of web sites such as corporate, entertainment and news sites. Current CDNs serve text, images, video, audio and streaming media, although images represent the bulk of the traffic. The next generation of CDNs will add to this diversity of resources by supporting the distributed execution of applications and the provisioning of personalized content. All these resources have various characteristics which place requirements, sometimes conflicting, on the CDN. Furthermore, these characteristics, hence requirements change over time. However, the distribution model of a CDN is rigid and does not cope well with such varied and changing requirements. In the following sections, we outline our proposed solution to this problem, namely a framework for developing self-adaptive CDNs. We also analyse a few of the key issues that the latter raises.

3. Self-adaptive CDN

A self-adaptive CDN will serve resources in an intelligent manner. It will be able to reason about a requested resource and in order to satisfy some overall administrative goal will adapt its behaviour to serve that resource. Hence it will be able to cope with heterogeneous content with varied demand characteristics and unpredictable events such as flash crowds. We now present two key concepts for realising such self-adaptive behaviour in a CDN.

3.1. Contextual characteristics

The behaviour of a CDN is expressed by the way it consumes resources like cpu capacity and bandwidth to provide other resources like text, images or video. In our framework, each resource has a contextual characteristic. Our notion of context embodies the application and time domains. Contexts are hierarchical and have inheritance properties. For example, the WEB context is the parent of the NEWS context and the CORPORATE context. And an image with the NEWS contextual characteristics inherits the WEB contextual characteristics. Our definition of context distinguishes among different types of web sites such as corporate, news, personal etc and also among different events in time such as a flash crowd. Within our framework, all the resources consumed by a CDN and the resources provided by an origin server are described using a semantically rich language. The aim is to use such a language to capture several types of relationships:

- within resources consumed
- within resources provided and
- between resources consumed and resources provided.

3.2. Encapsulating the behaviour of a CDN

Having defined the properties of the resources served by a CDN, we now tackle the issue of describing its behaviour. In an open system like a CDN which works under varying operational conditions, describing its behaviour using a description language is unfeasible. Such a language would have had to describe the interactions of all the pertinent components of the CDN. It is also highly likely that not all of those interactions may be known at design time. For example, we cannot predict what will be the most accessed resource on a particular web site. The description language if we were to create one, would also have to describe the various interactions of these components in response to some adaptation. This is an unrealistic approach that is bound to fail in the case of a CDN.

Instead, we describe the architectural constraints of the CDN in terms of its resources. This enables us to succinctly encapsulate the behaviour of the CDN. [4] have shown a similar approach to work in the field of adaptive software. However, this does not resolve the issue of how self-adaptive behaviour will be achieved. The behaviour of a CDN is altered by changing the relationships between its resources. Hence, given the architectural constraints, self-adaptive behaviour can be achieved by solving the constraint satisfaction problem between all the consumed and provided resources. As is the case in the software engineering domain, this problem is best solved through a set of rules which specify how resources can be consumed and provided. It is an approach that has been experimented in a previous work on developing an adaptive web server system where adaptivity is achieved through a set of rules [6] [7]. We advocate the same for realising the self-adaptive CDN. Self-adaptive behaviour will be achieved through a set of rules about resource interaction. These rules, together with the architectural constraint specifications will drive the change in relationships among resources.

The work by [4] has provided the motivation for describing self-adaptive behaviour of a CDN using architectural constraints in terms of its resources. However, we do not carry out adaptation at such high levels of granularity as in [4] and [5]. Indeed, the latter works propose to design and implement adaptive software components, whereas our framework enables adaptation in a CDN at a lower level of granularity and a higher level of abstraction. For example, we do not support unbinding or rebinding of the software components of the CDN at run time. This distinction in the granularity at which adaptation is carried out characterises autonomic systems surveyed in [9]. In the following section, we discuss several issues that our framework raises.

4. Issues in a self-adaptive CDN

These issues can be summed up by the following diagram:

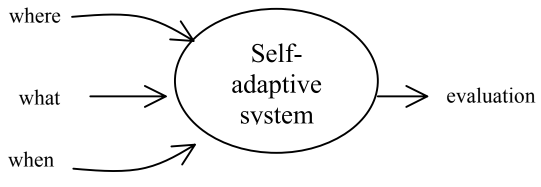

Fig. 1 Issues in a self-adaptive system

The pseudo-first issue of *how* to enable adaptive behaviour formed the crux of this paper and has already been addressed in Section 3. We now discuss the other issues.

4.1. Where do we adapt?

Although our framework does not specify where adaptive behaviour will be enabled, it does make certain implicit assertions. For example, by realising self-adaptive behaviour through architectural constraints in terms of resources, it excludes having any self-adaptive software component per se. We felt that any other approach would not have been able to meet the performance requirements for a CDN. Furthermore, we exclude adaptive behaviour at the network level, an approach that would have been favoured by researchers in active networks. As advocated by [8], it is our design philosophy to push the complex functionalities of a system in its higher layers as much as possible.

4.2. What do we adapt?

In section 3.2, we mentioned that self-adaptive behaviour in our framework would be achieved through a set of rules about resource interaction. Those rules dictate *what* to adapt? For example, a CDN could increase its load in order to improve throughput or decrease its load in order to improve access latency. The elaboration of these rules would be based on general principles about consumption of resources on the WWW and overall administrative goals.

4.3. When to adapt?

Our framework does not specify *when* to execute self-adaptive behaviour. We may provide execution of self-adaptive behaviour through an event trigger. Otherwise, such execution may be user-triggered. In the latter case, one may argue that this defeats the purpose of a self-adaptive system. However, having such flexibility in our framework enables us to concentrate on the pseudo-first issue of *how* to enable self-adaptive behaviour rather than investigating event detection mechanisms on the WWW. For example, developing an accurate flash crowd detector is a non-trivial task. Furthermore, the flexibility in our framework makes our development of the self-adaptive CDN independent of progress on works in event detection.

4.4. Evaluation

There are no clear metrics by which to evaluate self-adaptive systems. One can think of measures like latency or QoS. However, a lack of definite metrics will lead to adhoc evaluations which make comparison of different self-adaptive systems within the same application domain impossible. [9] discuss this issue further.

5. Related work

The need for moving away from the traditional design of CDNs has been recognised. [11] proposed a CDN for serving applications. They describe the architecture and algorithms for such a CDN. They also carry out a preliminary performance study. We note that the latter was based only on serving one type of resource, namely Internet applications. Hence, no testing with a heterogeneous group of resources was carried out. Different resource types have different characteristics which influence the design of a CDN.
The emergence of CDNs as platforms for transforming content is highlighted by [13]. The latter propose a framework for transforming XML data based upon environment descriptions. Their emphasis is on serving multimedia content and being able to transform the content based on user characteristics. They provide an application scenario for carrying out such adaptation on a CDN. Their work is strong on implementation details. And as such may be useful to us when we come to the implementation stage. Similar to our framework, they propose to have rules to enable adaptation of the content. However, they fail to capture the behaviour of the CDN as we do by defining the architectural constraints. This limitation prevents an implementation of self-adaptive behaviour.

6. Conclusion

This paper describes a framework for developing a self-adaptive CDN. This will enable a CDN to cope with the heterogeneity of resources on the WWW, the emergence of new demands and unpredictable events.

We presented the main concepts of our framework that enables reasoning about resources and encapsulation of the behaviour of the CDN. We also described how to enable self-adaptive behaviour. We then discussed several issues raised by our framework.

Our future work will be to carry out an implementation of our framework. We would like to explore various resource delivery scenarios. In particular we would like to see how the system copes with a flash crowd and the delivery of applications.

References

[1] Akamai, www.akamai.com
[2] Verma D, Content Distribution Networks: An Engineering Approach, John Wiley & Sons Inc, 2002
[3] DARPA Broad Agency Announcement on Self-adaptive Software (BAA-98-12), 1997 http://www.darpa.mil/ito/Solicitations/PIP_9812.html
[4] Georgiadis I, Magee J, Kramer J, Self-organising software architectures for distributed systems, ACM SIGSOFT Workshop on Self-Healing Systems, WOSS 02, Charleston, South Carolina, Nov 2002
[5] Waewsawangwong P, A constraint architectural description approach to self-organising component-based software systems, To appear in the Doctoral Symposium, International Conference on Software Engineering ICSE 2004, Edinburgh May 2004
[6] McCann J, Jawaheer G, Sun L, "Patia: Adaptive Distributed Webserver(a Position Paper)", IEEE Proceedings for The Sixth International Symposium on Autonomous Decentralized Systems, ISADS 2003, Pisa, Italy, Apr 2003
[7] McCann J, Jawaheer G, "Experiences in Building the Patia Autonomic Webserver", IEEE Proceedings of 14th International Workshop on Database and Expert Systems Applications (DEXA), Prague, Czech Republic, Sep 2003
[8] Saltzer J, Reed D, Clark D D, End-To-End Arguments In System Design, ACM Transactions on Computer Systems, V.2, N.4, p. 277-88. 1984
[9] McCann J, Huebscher M, Evaluation issues in autonomic computing, 2004 (submitted for publication)
[10] Krishnamurthy B, Wills C, Zhang Y, On the use and performance of content distribution networks ACM SIGCOMM Internet Measurement Workshop 2001
[11] Rabinovich M, Xiao Z, Aggarwal A, Computing on the edge: a platform for replicating applications, Eighth International

Workshop on Web Content Caching and Distribution, 2003

[12] Jung J, Krishnamurthy B, and Rabinovich M, Flash crowds and denial of service attacks: Characterization and implications for CDNs and web sites, Proceeding of 11th World Wide Web conference, 2002

[13] Kinno A, Yonemoto Y, Morioka M, Etoh M, Environment adaptive XML transformation and its application to content delivery, Proceedings of the 2003 Symposium on Applications and the Internet (SAINT'03)