

# DYCE: Model-based Emulation of Dynamic Web Content

Weisong Shi, Eli Collins, and Vijay Karamcheti  
Department of Computer Science  
Courant Institute of Mathematical Sciences  
New York University  
{*weisong, vijayk*}@cs.nyu.edu

## ABSTRACT

Requests for dynamic and personalized content increasingly dominate current-day Internet traffic. To efficiently serve this trend, several server-side and cache-side techniques have recently been proposed. Although such techniques, which exploit different forms of reuse at the sub-document level, appear promising, a significant impediment to evaluating their effectiveness on general workloads is the lack of synthetic content generators. More fundamental and the underlying reason for this shortcoming, is the absence of good models describing characteristics of dynamic web content.

This paper describes our ongoing research, which attempts to address both of these shortcomings. By analyzing the content of several web sites that provide dynamic content, we derive a set of models that capture the characteristics of such content in terms of independent parameters such as the distributions of object sizes and their freshness times. These models in turn drive a Java-based dynamic content emulator, DYCE, which generates edge side include (ESI) based dynamic content and serves requests for both whole documents and separate objects. DYCE is available for public use and is being used internally to evaluate design choices for the CONCA edge-server architecture, which attempts to improve the cacheability of dynamic and personalized web content.

## Keywords:

Object Characteristics, Dynamic Web Content, Dynamic Content Emulator

## 1 Introduction

Researchers have recently proposed several server-side and cache-side mechanisms to improve the generation and serving of dynamic web content. Server-side techniques, exemplified by techniques such as delta encoding [8], data update propagation [4], fragment-based page generation [5, 6], reduce the load on the server by allowing reuse of previously generated content to serve new requests. Cache-side techniques, exemplified by systems such as Active Cache [3], Gemini [9], CONCA [11], and the content assembly technique proposed by Wills et al. [13], attempt to reduce the latency of dynamic content delivery by moving some functionality to the edge of network. Similar trends are also visible in commercial caching and edge server products, most notably IBM's WebSphere [7] and Akamai's Edgesuite [1]. Despite their difference in focus, both server-side and cache-side approaches share the same rationale, specifically that it is possible to view the document in terms of a quasi-static *template* (expressed using formatting languages such as XSL-FO [14] or edge-side include (ESI) [12]), which is filled out with multiple individually cacheable and/or uncacheable *objects*. This object composition assumption enables surrogates and downstream proxy caches to reuse templates and cached objects to efficiently serve subsequent requests and additionally reduce server load, bandwidth requirements, and user-perceived latencies by allowing only the unavailable objects to be fetched.

Although the above techniques appear promising, it is difficult to predict to what extent their stated benefits apply to a workload different than the one they were evaluated on. As an example of the challenges, consider the following questions that we were faced with when trying to come up with general policies for our CONCA architecture [11]: At what granularity must objects be cached to achieve sufficient reuse? Is there a sharp threshold for choosing this granularity, or is it the case that the benefits are continuously varying? Can we estimate likely freshness times of objects from the duration they have been present in the cache? One way of answering these questions is to try out various policies using a synthetic content generator. However, the state-of-the-art is such that neither a template-based dynamic content generator, nor, more fundamentally, models describing the characteristics of such content, exist.

This paper describes our efforts on addressing both of these shortcomings. By analyzing the content of several web sites that serve dynamic content, we derive a set of models that characterizes this content in terms of a small number of independent parameters. Using these models, we then design and implement an effective Java-based dynamic content emulator (DYCE).

## 2 Characterizing Dynamic Content

From the perspective of surrogates and proxy caches that are attempting to improve delivery of dynamic content, there are several metrics of interest: *number of objects* that make up a document, their *size distribution*, and their *freshness times*. If one has access to the web server and/or application server of content providers, it is a relatively straightforward exercise to instrument these to obtain the above metrics of interest. Unfortunately, many commercial web sites which create dynamic content are proprietary. Therefore, we adopt an alternative approach based on the analysis of dynamic content (an HTML file) downloaded from various web sites.

The biggest difficulty that must be overcome in our chosen approach is the lack of an explicit template associated with the document, which can help identify its component objects. To work around this difficulty, we make the assumption that *the document template can be expressed as a nested table*, an assumption that is true of pretty much every web site we have examined.

Given this assumption, our approach first extracts the template associated with the document, then identifies the (logical) objects that fill out this template (grouping neighboring objects that exhibit the same freshness time characteristics), and finally aggregates logical objects into physical objects that serve as the granularity at which document characteristics are modeled. To cope with the absence of an explicit template in the documents, we inferred both the template and the component objects using parameterized *level-based* and *size-based* splitting techniques described in additional detail in a technical report [10]

## 2.1 Analysis of Dynamic Content Characteristics

We analyzed traces collected from six representative Web sites, with frequently changing dynamic content: three news sites (`www.cnn.com`, `dailynews.yahoo.com`, `www.nytimes.com`), two e-commerce sites (`www.amazon.com`, `www.barnesnoble.com`), and an entertainment site (`www.windowsmedia.com`). The traces consisted of downloads of the main page at each of the sites, every ten minutes over a two-week interval.

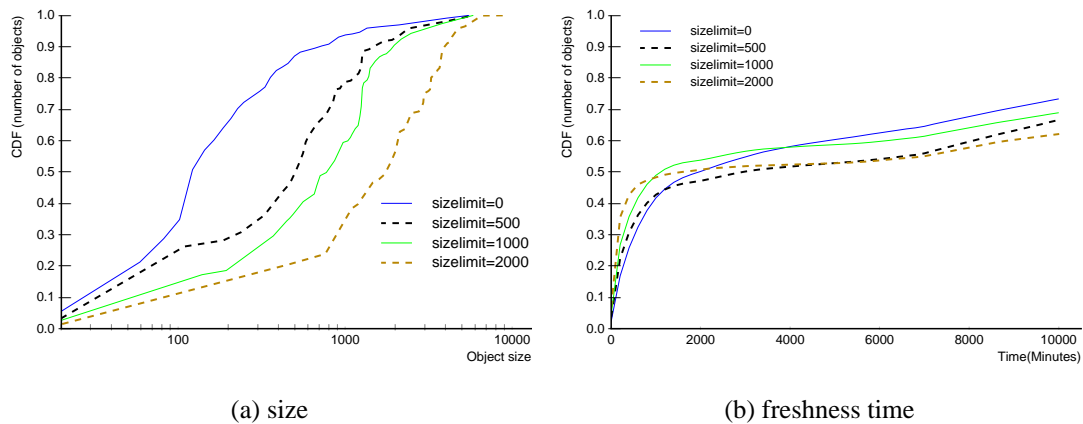


Figure 1: **The measured cumulative distribution of (a) object sizes and (b) freshness times for different size limit settings for traces collected from the `www.cnn.com` site.**

Each of these traces was analyzed using the methodology described earlier [10]. Due to space restrictions, we discuss only the results for the `cnn` trace, which is representative of the others. Figure 1 shows the measured cumulative distribution of object sizes and freshness times for different size-limit settings; the latter parameter denotes the target of the document splitting process outlined earlier. We find that a large number (about 90%) of objects in our documents are of relatively small size (smaller than 500 bytes), and while half of the objects exhibit freshness times that fall between a few minutes and a day, a significant fraction (almost 50%) of the objects are relatively long-lived, remaining essentially unchanged over the duration of the trace.

Looking across all of the traces, our results are summarized in the following general observations:

- The sizes of component objects making up a dynamic web page can be captured very well using an *Exponential* distribution. This is in contrast to how static documents are modeled, usually with a heavy-tailed *pareto* distribution.
- Except for a considerable fraction of objects that change very infrequently, the freshness times of component objects can be captured using a *Weibull* distribution. For two of the sites (`www.amazon.com` and `www.bn.com`), this distribution degenerates into a sharp bimodal one: all of the objects either change on almost every access or change very infrequently.
- Content from all of the six sites demonstrated significant opportunity for reuse across both the temporal and spatial (in linked documents) dimensions. More interesting is the relationship between content reuse and the object granularity: for four of the sites (`www.cnn.com`, `dailynews.yahoo.com`, `www.nytimes.com`, `www.windowsmedia.com`), there was a graceful degradation of reusability with increasing object granularity, while the degradation was much sharper for the other two.

## 3 DYCE: Dynamic Content Emulator

Using the above models, we designed and implemented a dynamic content emulator (DYCE), which generates parameterizable dynamic content that adheres to a subset of the ESI specification. DYCE builds on top of the Tomcat web server from the Jakarta open source initiative [2], and can service requests for both whole documents as well as individual components. DYCE is easy to configure, extend, and use and should prove a useful tool for other researchers in this area. To validate both our models and their use in DYCE, we wrote an idealized cache simulator that works off both the real trace data as well as the output of DYCE. Comparing the outputs of this simulator for

the two cases verifies that DYCE effectively models real content, and at a significant simulation cost advantage. The source code of DYCE is available for public use at <http://www.cs.nyu.edu/~weisong/dyce.html>.

DYCE separates out the functions of content generation and content representation into two modules: the *dynamic content generator* (DCG) and the *dynamic content presenter* (DCP). DCP interfaces with requesters and appears as a traditional web server augmented with additional functionality described later in the section. The content presented to requesters by the DCP is currently packaged as an HTML document augmented with *Edge Side Include (ESI)* directives. (see Figure 2).

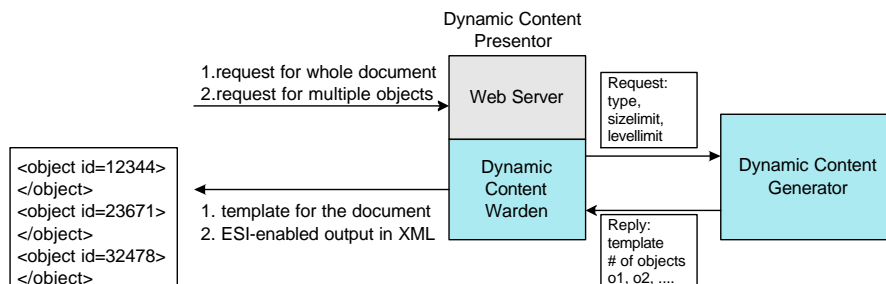


Figure 2: **The architecture of dynamic content emulator, which includes two components: a dynamic content generator and a web server-like dynamic content presenter. The extra functionality of dynamic web presenter is provided by a module called the dynamic content warden.**

To simplify its use, DCP supports a fairly simple interface, servicing two kinds of requests, either for the *document template* or for a set of *selected objects*. The DCG takes three parameters as input to create the corresponding document template and individual objects: the *dynamic content type*, the *size limit* and the *level limit*, two parameters that stem from our proposed splitting schemes.

The DYCE implementation extends the ESI specification in two ways that enables (1) combining of requests for multiple objects from the same document into a single request and response message, and (2) association of per-individual object freshness times. These extensions permit efficient implementation of object-level coherence between a proxy cache or surrogate and a web server.

## 4 Conclusions and Future Work

In this paper, we have modeled the characteristics of dynamic web content by analyzing document traces from dynamic web servers. Using these models, we have designed and implemented a tool, the Dynamic Content Emulator (DYCE), which emulates a web server serving dynamic content. Our future work consists of using DYCE to evaluate and further refine the design of our CONCA prototype [11], which incorporates a novel design for efficient caching of dynamic and personalized content.

## References

- [1] Akamai Technologies Inc. Edgesuite services, [http://www.akamai.com/html/en/sv/edgesuite\\_over.html](http://www.akamai.com/html/en/sv/edgesuite_over.html).
- [2] Apache Jakarta Project, <http://jakarta.apache.org>.
- [3] P. Cao, J. Zhang, and K. Beach. Active cache: Caching dynamic contents on the web. *Proc. of IFIP Int'l Conf. Dist. Sys. Platforms and Open Dist. Processing*, 1998.
- [4] J. Challenger, A. Iyengar, and P. Dantzig. A scalable system for consistently caching dynamic web data. *Proceedings of Infocom'99*, Mar. 1999.
- [5] J. Challenger, A. Iyengar, K. Witting, C. Ferstat, and P. Reed. A publishing system for efficiently creating dynamic web content. *Proc. of the IEEE Conference on Computer Communications (INFOCOM'00)*, Mar. 2000.
- [6] F. Douglass, A. Haro, and M. Rabinovich. HPP:HTML macro-pre-processing to support dynamic document caching. *Proc. of the 1st USENIX Symposium on Internet Technologies and Systems (USITS'97)*, Dec. 1997.
- [7] IBM Corp. Websphere platform, <http://www.ibm.com/websphere>.
- [8] J. C. Mogul, F. Douglass, a. Feldmann, and B. Krishnamurthy. Potential Benefits of Delta-Encoding and Data Compression for HTTP. *Proc. of the 13th ACM SIGCOMM'97*, Sept. 1997.
- [9] A. Myers, J. Chuang, U. Hengartner, Y. Xie, W. Zhang, and H. Zhang. A secure and publisher-centric web caching infrastructure. *Proc. of the IEEE Conference on Computer Communications (INFOCOM'01)*, Apr. 2001.
- [10] W. Shi, E. Collins, and V. Karamcheti. Modeling object characteristics of dynamic web content. Tech. Rep. TR2001-822, Computer Science Department, New York University, Nov. 2001, <http://www.cs.nyu.edu/~weisong/papers/tr2001-822.pdf>.
- [11] W. Shi and V. Karamcheti. CONCA: An architecture for consistent nomadic content access. *Workshop on Cache, Coherence, and Consistency(WC3'01)*, June 2001.
- [12] M. Tsimelzon, B. Wehl, and L. Jacobs. ESI language specification 1.0, 2000, <http://www.esi.org>.
- [13] C. E. Wills and M. Mikhailov. Studying the impact of more complete server information on web caching. *Proc. of the 5th International Workshop on Web Caching and Content Distribution (WCW'00)*, 2000.
- [14] W3C XSL Working Group, <http://www.w3.org/Style/XSL/>.