What Your CDN Won't Tell You: Optimizing a News Website for Speed and Stability

Julian Dunn
SecondMarket Holdings, Inc.
jdunn@aquezada.com

Blake Crosby
Canadian Broadcasting Corporation
me@blakecrosby.com

Abstract

In this paper, the authors discuss their experiences implementing, operating, and optimizing a content delivery network for Canada's largest news website, that of the Canadian Broadcasting Corporation (CBC). The site receives over one million unique visitors per day. Although CBC uses the Akamai Aqua Platform (formerly EdgeSuite) as its content delivery network of choice, the lessons described here are generally applicable to any infrastructure fronted by a CDN.

1. Introduction

Content Delivery Networks[1] (CDNs) have become increasingly popular over the last decade. With the explosion in users on the Internet, it has become impractical for companies with any significant amount of traffic to invest solely in their own infrastructure to deliver information to their end users.

Nowhere is this truer than in the online media business, where day-to-day traffic volumes can deviate wildly based on the news of the day. For example, the 9/11 attacks on the World Trade Center towers caused site outages for many major media organizations when servers failed under the extreme traffic load[3]. This event — and many others since then, like earthquakes, tsunamis, and the 2008 United States presidential election — repeatedly taught media companies a lesson: they could no longer afford to only invest in capital equipment like hundreds of racks of servers, simply to handle the one or two major news events that might cause traffic spikes of ten or one hundred fold. Some external buffer, like a CDN, is necessary.

Many Internet system administrators are already familiar with the technical design of a CDN. Very briefly, a CDN operates on the premise that long-haul Internet backbones are slow or unreliable and have a long propagation time, whereas access to one's Internet Service Provider (ISP)'s data center is fast, since it is often only one or two hops away. By building a private WAN with proprietary, optimized routing algorithms over the public Internet between ISPs and placing acceleration/caching services in ISP data centers, CDNs can accelerate the speed of content delivery to the end-user. Also, by operating on the assumption that many users will access the same piece of content, CDNs can serve as a distributed cache to further increase the speed of content delivery.

In practice, CDNs are most useful for mostly static content; dynamic, real-time applications, such as Facebook or Twitter, do not benefit as much from CDNs, although those organizations still use CDNs for delivery of static assets such as graphics. Furthermore, increases in speed and response times of even the public Internet backbones over the last decade have diminished the speed advantages of CDNs. They are however, an excellent fit for media organizations like CBC, where relatively-infrequently changing content is coupled with enormous variability in traffic volumes.

2. What does "Tuning a CDN" mean?

Tuning a CDN is a sophisticated process requiring an in-depth understanding of one's overall business and technical requirements. It is more than just establishing and configuring technical parameters, such as Time-To-Live (TTL) policies on content. Although TTLs are an important component of effective CDN operation, they are not the only parameter worth examining — a mistake that CBC initially made.

It's important to develop a detailed understanding of one's content delivery infrastructure prior to embarking on a tuning exercise. Some of the elements worth examining are:

- The architecture of the web-serving infrastructure from which the CDN retrieves content, also known as the origin
- Amount of personalization and content targeting on the site
- Frequency of content change
- User access patterns
- Business requirements governing what metrics should be optimized (cost, content freshness, reliability, complexity)

These elements all have a material impact on how well one's site integrates with a CDN, and ultimately what rules should be employed when.

CDNs vary wildly in the ability for system administrators to tune the operation of their algorithms. In general, performance tuning a CDN comes down to several key parameters:

- The frequency at which the CDN should retrieve content from your origin, and on what basis (Time-To-Live, freshness checks);
- The tradeoff between maximizing the proportion of objects served from the CDN rather than the origin (known as the origin offload percentage) versus content freshness;
- The relative cost of bandwidth at the edge (CDN to end-user) versus the origin. Also important to consider: the so-called midgress bandwidth costs, which is the price that the CDN charges for internal CDN communication between cache hierarchies;
- Whether it is acceptable to serve expired (dirty) objects from the CDN;
- What HTTP optimizations should be implemented between the origin and the CDN to maximize performance.

Akamai's Aqua Dynamic Site Accelerator[4] product is aimed at generic website workloads, so many parameters are tunable via basic configuration. For experienced system administrators, "extended metadata" can be added to a basic configuration, to be able to tune parameters such as content time-to-live (TTL) values, logging and reporting, and content compression. Extended metadata also allows the administrator to implement extra features such as HTTP pipelining, origin forwarding, large object optimization, failover rules, and Edge-Side-Include[5] processing. There are also additional features unavailable to administrators through the control panel, but which can be implemented by Akamai solutions engineers, such as complex redirects using regular expressions, or adjusting the cache key for an HTTP object based on a cookie or User-Agent.

With all these tuning parameters available to the average system administrator, it can be difficult to know what to alter. In the authors' experience, CDN vendors do not provide a lot of guidance, primarily because the answers to which parameters to modify and what values to set vary greatly based on the customer's workload, requirements, and business context. Design and optimization of the entire system, including the architecture of the origin infrastructure, is generally beyond the scope of even a professional services engagement. The most important takeaway before beginning an optimization activity is to understand the existing content access patterns and delivery requirements.

3. Content Access Patterns in Media

Online news sites like CBC's have several characteristics distinct from many other websites.

3.1. Unpredictable Traffic Patterns

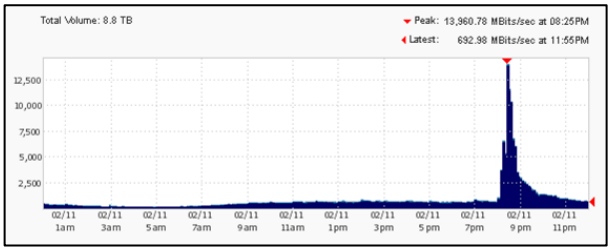

Fig. 1: Example of a traffic patterns seen during a breaking news event, where site traffic (seen here in MBits/sec) increases an order of magnitude in a matter of minutes.

Flash news events, such as natural disasters or political uprisings, can change daily traffic patterns by several orders of magnitude in a matter of minutes, as seen in the above figure, which illustrates traffic seen during the evening of February 11, 2012, when pop singer Whitney Houston died.

3.2. Content Frequently Updated Within First 24 Hours, Then Almost Not At All

Breaking news stories are revised frequently within the first 24 hours, sometimes dozens of times within the first few hours, and then very infrequently after 48 hours. If a story has new developments, reporters will often write a new story linking to the original story, rather than updating the old one. Traffic to the story has a corresponding exponential fall-off, but the long tail is infinite and may peak if there are new developments, even years later, in a story. For example, if a crime is committed on Monday and someone is arrested on Friday, the original crime story may see a burst of traffic as a result of users clicking on it to read more information about the arrest.

3.3. User Accesses Outpace Content Updates

Each news story may see at most several dozen revisions. Between updates, hundreds of thousands of users could see that revision of the story.

4. Content Delivery Requirements in Media

Optimization of content delivery in an online media environment tries to balance several sometimes-conflicting requirements:

Content freshness: The need to deliver the most up-to-date breaking news stories and to disseminate, as quickly as possible, updates to those stories;

Stability and reliability: The need to maintain as near a 100% system uptime as possible even under extreme traffic load;

Minimize system complexity: The business rules used to configure the content delivery infrastructure should be broad, yet simple enough that they can cover most operating scenarios, even in the face of extreme site traffic. Notwithstanding the foregoing, if rules need to be changed, their effects should be instantaneous, since they are generally in response to an incident. Finally, the rules should be understandable by even junior operations staff;

Cost minimization: The need to keep operating costs within budget parameters even in the face of extreme site traffic caused by high-impact breaking news, e.g. Arab Spring or the 2011 Japanese tsunami & nuclear power plant crisis. CBC is also a publicly funded organization and thus vigilant management of both operating and capital costs is vital, since the corporation is subject to extra levels of scrutiny not usually attached to private corporations.[6]

5. Existing CDN Deployment and Problems

Between 1995 and 2006, CBC's website was driven by a family of J2EE applications that rendered news content from a relational database. The design was understandable, given the stated business requirement for content freshness as a top priority. However, a dynamic application was not a good fit for CBC, given the content access patterns described in section 3: the application unnecessarily re-rendered stories for each and every user request, and resulted in many outages under high load.

The original CDN implementation was perceived as an insurance policy against origin failures. However, in so doing, the set of business rules required to implement impose static object caching on a dynamic system quickly grew out of control. The rules were not only difficult to maintain, but often did not meet business requirements: objects were frequently updated before their TTL expired, causing many urgent phone calls to operations staff to purge the CDN cache so users could see the new content. These purge requests would take up to fifteen minutes to be propagated across the CDN's extensive server network of 60,000 servers)[7].

The first step in properly re-implementing the content delivery architecture was to ensure a stable origin upon which to build. Because of the content access patterns described above and lack of content targeting, user sessions or personalization on the site, it was adequate to convert the origin to a largely static delivery system. As such, CBC's news stories today are effectively static HTML documents. The origin consists of an Apache server farm with no dynamic modules. Stories generated from the CMS are converted into HTML fragments containing the story body, headline, associated links, and other display metadata for the user. These fragments are then wrapped by a "story wrapper" template, which uses Server-Side Includes (SSI) to inject story variables into the appropriate areas for display to the end user. (See Figs. 2 and 3.)

```
<!DOCTYPE html>
<!--#config errmsg="" -->
<!--#if expr="${QUERY_STRING}" -->
<!--#include virtual="${QUERY_STRING}" --
>
<!--#set var="newsStory" value="true"-->
<!--#endif -->
<!--#if expr="!${storyBody}" -->
<html>
<head>
<meta name="robots" content="noindex, no-
follow"/>
<meta http-equiv='refresh' con-
tent='0;url=http://www.cbc.ca/404.html'
/>
</head>
<body>Page Does Not Exist - 404</body>
</html>
<!--#else -->
<html>
<body>
<div id="storyhead">
<!--#include
virtual="/cms/html/head/${storyHead}" -->
</div>
<div id=”storybody”>
<!--#include
virtual=”/cms/html/body/${storyBody}” -->
</div>
</body>
</html>
<!--#endif -->
```

Fig. 2: Simplified example of story "wrapper" that includes a story headline & body included by way of an HTTP query string. Rewrite rules (see Fig. 3) ensure that the query string is hidden and remapped to a pseudo-filesystem path.

```
RewriteRule ^/news/story/(.*)
/i/news/v10/storytemplates/default.html?/
cms/html/story/$1 [L]
```

Fig. 3: Example of rewrite rule illustrating how news stories that appear to be static HTML documents under /news/story are actually remapped for processing by the story wrapper mechanism.

While readers may criticize SSI as a primitive technology, lacking features available in more complex languages like PHP or Ruby, SSI is lightweight and fast, runs within Apache, scales extraordinarily well under

high load, and provides good security, provided that the ability to execute arbitrary scripts is disabled.[8] The limited feature set also makes it very difficult for developers to make mistakes by introducing external dependencies, e.g. XML transformation, database connections, etc. Finally, the simple feature set has an unintended, but desirable side-effect: it mandates the use of the content management system as the canonical source of all information associated with a news story. Reporters cannot manipulate the content displayed to end-users without entering it in the CMS, which has an audit trail for legal purposes, such as libel defense.

6. The Origin's Stable; What Do We Do With the CDN?

Once the origin was stable, the team was able to carefully consider the complex nest of origin server and CDN tuning rules.

The authors decided to begin by conducting a thought experiment. What would be the consequences if there was no tuning at all and the origin was exposed directly to the CDN? Obviously, this would expose the origin to high risk under high traffic and defeat the purpose of having a CDN, but it would significantly simplify the configuration. With this thought experiment as a foundation, the team began to really think about the company's conflicting requirements from section 4 and prioritize them. Should all other factors really be sacrificed for the sake of site freshness, even for a news organization? The authors agreed that it shouldn't, and decided upon the following order of priorities:

- Origin stability
- Content freshness
- System complexity
- Cost

In other words, under no circumstances should the origin fail, even under a crushing traffic load. After all, content delivery rules that optimize for content freshness are pointless if one's website is down. Next, the tuning rules must be implemented in such a way that provides the freshest content possible. Third, the business rules must be simple enough that even a junior operator can understand – and if absolutely necessary, adjust them, should a major news event strike in the middle of the night and the origin require extra protection.

Readers may criticize CBC for placing cost last, particularly since the corporation is a public benefit corporation, supported by taxpayers. However, the company mitigates this prioritization in a number of ways:

- Using a local peering exchange (Toronto Internet eXchange[9], or TorIX) to connect to the CDN, whose ingress servers are also at TorIX. In this way, transit bandwidth from the origin to the CDN is effectively free;
- Negotiating with the CDN so that midgress bandwidth is free. That is, bandwidth used to send data on behalf of CBC internally in the CDN's infrastructure is not charged to CBC;
- Choosing a 95th percentile-billing model for HTTP content to mitigate traffic spikes causing high one-time costs;
- Configuring the origin to send appropriate cache-control headers to minimize the amount of content sent to the CDN. (See section 7.)

7. Implementation Parameters

With a general list of priorities in mind, we can discuss the specific parameters that CBC uses to optimize its use of the CDN.

7.1. Set a global site TTL

Using the principle of "sometimes the defaults are good enough", the authors set a global site TTL of 20 seconds for almost all objects except HTML (`Content-Type: text/html`); HTML has a global site TTL of 120 seconds. In conjunction with the other configurations below, the authors have found this to be adequate in handling 99% of all traffic situations. If a major news event occurs, operations staff will carefully monitor the performance of the origin and increase the global TTL if needed. (Such a change could theoretically be automated.)

7.2. Heavily leverage If-Modified-Since

As explained above, after the expiry of a TTL, Akamai edge servers will issue GET requests to the origin with an `If-Modified-Since` (IMS) header set to the time at which it last retrieved the objects in question. The origin is configured to correctly send `HTTP/1.1 304 Not Modified` if the object has not changed, thereby ensuring that object bodies are not sent unnecessarily.

The authors have found this simple configuration to be extremely effective at achieving a high origin offload: approximately 70% of origin requests result in a `HTTP/1.1 304 Not Modified`. (See Fig. 4.)

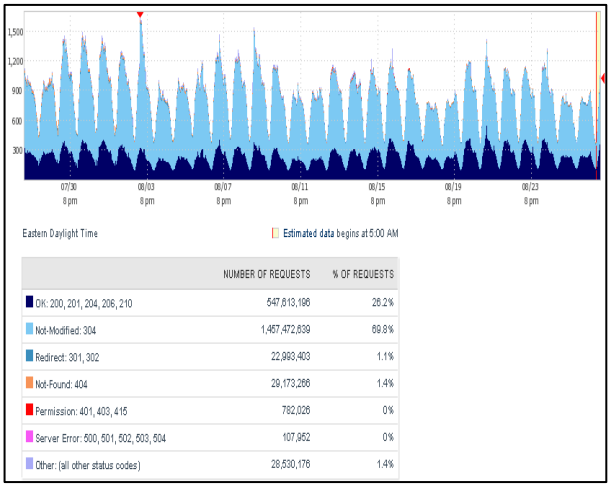

Fig. 4: Distribution of typical HTTP access codes over a four week period, showing that the majority of user requests are serviced by 304 Not Modified responses (in light grey).

7.3. Organize the Filesystem by Object Mutability and Override Global Site TTLs by Content-Type

By organizing the filesystem with top-level directories that correspond approximately to the update frequency of objects inside them, content-type-based overrides can be set by directory. For example, CBC has a top-level includes directory under which global assets like site icons, JavaScript and CSS are deployed through a rigorous change control process. Although CBC could just let Akamai use the global site TTL and issue many unnecessary but harmless `If-Modified-Since` requests, the authors have instead set a TTL by content-type to something relatively high, such as one (1) hour, given that the objects will not change multiple times within that time window.

```
<Location "/includes">
  ExpiresByType "text/css" "access
plus 1 hour"
  ExpiresByType "application/x-
javascript" "access plus 1 hour"
  ExpiresByType "image/gif" "access
plus 1 hour"
</Location>
```

Fig. 5: Illustration of TTLs set by filesystem path and content type.

7.4. Enable Last Mile Acceleration and Origin Compression

Last mile acceleration means that the CDN's edge servers will attempt to use gzip compression to send content to end-users. Origin compression means that the origin will send compressed objects to the CDN. Both are enabled; this way, the CDN merely needs to send payloads directly to the end-user without needing to recompress them.

7.5. Enable HTTP persistent connections and set appropriate timeouts

Some CDNs will attempt to keep a pool of connections open to the origin to avoid incurring the overhead of connection setup and teardown on each user request that results in a cache miss. This requires not only the use of of persistent connections (pconns) in the origin webserver, but also careful tuning of pconn timeouts to match those configured at the CDN.

The Akamai configuration documentation[10] emphasizes that the timeout configured in Dynamic Site Accelerator (DSA) must be shorter than the maximum timeout configured at the origin, to avoid an in-progress pconn being terminated. Since the default value in DSA is 300 seconds, CBC sets the `KeepAliveTimeout` value in Apache to exactly 301 seconds.

7.6. Understand and properly configure all cache-control headers

It's critical to understand and properly configure all cache control headers when interacting with a CDN. The following are critical:

Cache-Control

The global time-to-live value is specified in the `max-age` parameter. `public` should also be specified to ensure that downstream caches cache the content.

Expires

The date the object expires. This may be used by a CDN in preference to the `Cache-Control` header. Generally its value should be the current time (value of the `Date:` HTTP header) plus the TTL.

Last-Modified

As described above, the `Last-Modified` header is sent with all objects with the exception of dynamic content and server-side HTML.

ETag

Entity Tags[11], or ETags, are a hash of the file's inode and last modified time, and can often be an excellent hint to a CDN as to whether or not a cached object is dirty. They should be used with care[12], however, for the following reasons:

- They are incompatible with SSI documents. ETag will be set to the "hash" of the SHTML wrapper even though SSI-included objects might have changed.
- ETags work perfectly for non-SSI documents in a deployment scenario where all webservers are connected to shared storage, thus ensuring consistent object inodes across the cluster, so

long as the webserver's operating system respects the shared storage system's inode scheme. However, in a scenario where each webserver serves objects from its own local disk with different inodes between machines, ETags are not suitable.

8. General Lessons Learned

The authors offer the following general advice by way of summarizing the specific, technical implementation details outlined in section 7.

8.1. Keep caching rules simple

Sometimes the defaults are good enough. For example, many origin HTTP servers already set headers (`Cache-Control`, `Expires`, `ETag`) that are usable hints for downstream content delivery networks, caches, or proxies. The default behavior of these headers is often adequate, but should be verified by system administrators to ensure suitability.

8.2. Tune at the origin first rather than at the CDN

Many CDNs provide the ability to tune content freshness at the "edge", or in the vendor's delivery network. The authors shied away from this approach, at least until origin tuning options were exhausted. There are several reasons for this, aside from the obvious downside of vendor lock-in. First, implementing tuning features directly in the vendor's network increases the propagation time for any changes to those features. For example, Akamai stores customer configurations in XML files ("metadata") that have to be carefully managed and go through a rigorous workflow before they can be deployed to Akamai's entire network of 60,000 servers. This is an appropriate level of paranoia for many features, but excessive in an emergency situation when TTLs need to change immediately due to high traffic. Making TTL changes at the origin and then signaling to the CDN that future requests should be treated differently is the fastest way to propagate those changes across the CDN's network.

8.3. Understand and categorize content before tuning

When tuning – in the form of custom TTLs – is necessary, it's most efficient to group that content together in some way so that one or two rules may suffice. For example, the majority of the TTL rules implemented at CBC are by MIME (content) type, in preference to rules based on filesystem paths.

8.4. Understand what "TTL" actually means

There is often a major misconception among system administrators and users as to the meaning of "TTL", at least insofar as it applies to Akamai Dynamic Site Delivery. "TTL" does not mean the interval at which the entire object body will be re-fetched from the origin; it simply means the interval at which the CDN cache will check the origin for object freshness. If the cache determines, from origin HTTP headers, that the object has not been modified from the copy it has in cache, it will not re-fetch the object body and instead reset the TTL timer. Therefore, it is actually safe to set the site TTL quite low for a site on which objects change relatively infrequently; it can be some fraction of the average modification time of objects.

9. Tools Used To Debug and Improve Cache Performance

CBC has found the following resources useful in examining system behavior, debugging problems, and providing statistics on how performance can be improved:

- The Akamai Luna Control Center (formerly EdgeControl) where system administrators can generate reports in real-time as well as run ad-hoc reports on past traffic.
- The "origin OK volume per URL" report is handy in determining which objects are being fetched from the origin.
- Error reports: To improve performance, the "top URLs, by number of errors" report is useful for tracking down and identifying the top broken links that cause origin requests.

System administrators and developers also use a number of Firefox plugins to troubleshoot CDN behavior by examining and/or modifying HTTP headers:

- Firebug[13] allows users to view HTTP request and response headers to troubleshoot caching problems.
- Tamper Data[14] can also be used to inspect and modify HTTP/HTTPS headers.
- Akamai Headers is a plugin that system administrators have used in the past to decipher Akamai specific information such as the cache key and which edge server responded to your request. However, this plugin is no longer actively maintained by Akamai and may not be suitable for general use.

10. Outcomes

CBC has greatly increased the performance and maintainability of its content delivery network by redesigning its origin architecture and simplifying the configuration of both its origin servers and the CDN. The business requirements are achieved with fewer than 25 lines of tuning in Apache. All business rules are implemented at the origin, meaning that any changes are instantaneously propagated to the CDN.

The entire solution is extremely cost effective from a capital expenditure viewpoint. CBC ran its entire origin with only six webservers between 2003-2010, and today the origin consists of a nine-server cluster of commodity servers with only 40% CPU utilization across all machines.

11. Future Work

Dynamic features, are slowly being added to CBC's website, requiring a careful rethinking of the origin architecture to ensure that we do not return to the mistakes of the past. Some dynamic functionality has been implemented by using Edge Side Includes, which provides a more expressive and powerful processing language over SSI.

At the present time, ESI processing is done by the CDN, but CBC is considering implementing an origin ESI processor and cache, such as Varnish[15], to be able to achieve even more control over caching rules and tuning.

12. Acknowledgements

The authors would like to thank their colleagues in the Digital Operations group at CBC, and in particular David Raso for designing and implementing the Story Wrappers system.

13. References

[1] Dilley, J. et al. Globally distributed content delivery. *IEEE Internet Computing*, September-October 2002,
[3] Noam, E. M. Straining communications systems to the limit. *The New York Times*, September 24, 2011, pp. 4-C.4.
[4] Akamai Dynamic Site Accelerator. Web, September 2012. http://www.akamai.com/html/solutions/dynamic_site_accelerator.html
[5] Nottingham, M., et. al. Edge Side Includes Language Specification. Web, September 2012. http://www.w3.org/TR/esi-lang
[6] Government of Canada. Access to Information Act. Web, September 2012. http://laws-lois.justice.gc.ca/eng/acts/A-1/index.html
[7] Economou, G. How Akamai Maps the Net: An Industry Perspective. Web, September 2012. http://www.akamai.com/dl/akamai/economu_mapping_the_internet.pdf
[8] Security Tips – Server Side Includes. Web, September 2012. https://httpd.apache.org/docs/current/misc/security_tips.html#ssi
[9]About TorIX. Web, September 2012. http://www.torix.ca/about.php
[10] Akamai Edge Server Configuration Guide. Proprietary information of Akamai Technologies. May 2012.
[11] Fielding, R., et. al. RFC 2616: Hypertext Transfer Protocol (HTTP/1.1). Web, September 2012. http://www.w3.org/Protocols/rfc2616/rfc2616.html
[12] Yahoo. Best Practices for Speeding Up Your Website: ETags. Web, September 2012: http://developer.yahoo.com/performance/rules.html#etags
[13] FireBug. Web, September 2012. https://getfirebug.com/
[14] Tamper Data. Web, September 2012. http://tamperdata.mozdev.org/
[15] Varnish Cache. Web, September 2012. https://www.varnish-cache.org/