Wikipedia Workload Analysis

Guido Urdaneta Guillaume Pierre
Maarten van Steen
Dept. of Computer Science, Vrije Universiteit, Amsterdam
Email: {guidou,gpierre,steen}@cs.vu.nl

September 17, 2007

Technical report IR-CS-041, Vrije Universiteit, September 2007.

Abstract

We study an access trace containing a sample of Wikipedia's traffic over a 107-day period. We perform a global analysis of the whole trace, and a detailed analysis of the requests directed to the English edition of Wikipedia. In our study, we classify client requests and examine aspects such as the number of read and save operations, flash crowds, and requests for nonexisting pages. We also outline strategies for improving Wikipedia performance in a decentralized hosting environment.

Keywords: Workload analysis, Wikipedia.

1 Introduction

Despite numerous pessimistic predictions, Wikipedia is a blatant success. As of August 1st 2007, it contains approximately 7.9 million articles in 253 languages, and is considered one of the ten most visited web sites on the Internet [1]. Its uninterrupted popularity growth has forced its operators to upgrade the hosting architecture from a single server to a distributed architecture with more than 250 servers at three locations on different continents. The current architecture, however, is still subject to scalability issues since it has centralized components such as a database for each language edition.

For Wikipedia it is particularly important to find economical ways to make their system more scalable, since its operation depends essentially on donations and the work of volunteers. There are basically three techniques that can be used to improve the scalability of any distributed system: distribution, replication and caching. In contrast with the current architecture which relies on database replication and Web page caching, we recently proposed an alternative architecture based on data distribution [19]. In this architecture, each server is responsible for hosting only a small fraction of the whole Wikipedia content. We believe that this would allow the system capacity to scale linearly with the number of hosting resources. However, the extent to which such an architecture may work in practice greatly depends on the characteristics of the workload addressed to it.

To gain a better understanding of the Wikipedia workload, we obtained and studied an access trace containing a 10% sample of the total Wikipedia traffic over a 107-day period. Studying the workload as a whole and classifying the different request types handled by the site allows us to validate assumptions on the parts of the load that are critical for overall performance. We also analyzed the workload of the English-language Wikipedia on a per-document basis. This analysis allows us to understand the properties of individual documents and the interaction between the web front end and the storage system.

Unlike previous workload analysis of systems such as e-commerce sites [4], P2P file sharing systems [11], static Web sites [7, 5, 2] and multimedia delivery systems [12, 16, 9], our study gives insight on the functioning of a collaborative web site, where most of the content is created and updated by external users and not the operators.

Figure 1: Wikipedia Architecture

2 Wikipedia Operation

Wikipedia is composed of a number of wikis [14]. Each wiki is typically associated with a different language edition and has a separate DNS name. For example, `en.wikipedia.org` refers to the English-language edition of Wikipedia, and `fr.wikipedia.org` to the French-language edition. In addition to Wikipedia, the Wikimedia Foundation, which is responsible for the hosting of Wikipedia, uses the same infrastructure to host other related wiki projects, such as Wiktionary (a dictionary) and WikiNews (a news site).

As is shown in Figure 1, the functionality of Wikipedia can be divided into three parts: page management, control and search. The page management part is the most important since most of the information provided by Wikipedia such as encyclopedic articles, user information, and discussions is in the form of wiki pages. Each page has a unique identifier consisting of a character string and an integer representing a name space. Pages can be created, read, and modified by any user. However, a page update does not result in the modification of an existing database record, but in the creation of a new record next to the previous version. It is therefore straightforward for a user to get a list of all editions of a page, read old versions as well as reverting a page to a previous state. Privileged users have the option to rename, delete, and protect pages from being edited. Part of the load gen-

erated by page read operations issued by anonymous (not logged-in) users is handled by a group of external cache servers.

Pages are written using a markup language called "wikitext." One important aspect of this language is that it allows for the creation of parameterized pages called templates, and the inclusion of one page into another. For example, there is a template with an information box for soccer players that takes parameters such as name, nationality and current club. This template is rarely requested by end users directly, but it is included in the articles of many soccer players and is thus frequently requested in an indirect way. A page can also be configured to redirect all its read requests to another page, similarly to a symbolic link. One consequence of these features is that there is not a one-to-one correspondence between the HTML pages that users typically read and the wikitext pages stored in the database.

The search part allows users to enter keywords and receive lists of links to related wiki pages as a result. This part of the system is isolated from the rest of the application in that it does not access the centralized database, but instead a separate index file generated periodically from the text of the pages.

Finally, the control part groups the rest of the functionalities. It encompasses: (i) user management, which allows users to authenticate to the system and have their user names stored in public page history logs instead of their IP addresses; (ii) user/IP address blocking, which allows administrators to prevent page updates from certain IP addresses or user accounts; and (iii) special pages, which are not created by users, but generated by the execution of server-side logic and provide information about the database or specific functions such as uploading static files to be referenced in wiki pages.

3 Wikipedia Traces

To conduct our study of the Wikipedia workload, we were provided by the Wikimedia Foundation with a sample of 10% of the total traffic directed to all the wiki projects they operate. The sample used in our study contains 14.9 billion HTTP requests and corresponds to the period February 1st, 2007 to May 16th, 2007. Each request is characterized by a unique ID, a timestamp, and the requested URL. Starting from March 30th one extra field was added to indicate if a page request resulted in a save operation.

For privacy reasons, the trace given to us by the Wikimedia Foundation does not contain any direct or indirect means to identify users, such as client

IP address or session cookie. Our study therefore considers only server-side information.

For some of the analyses we used a publicly available snapshot of the English Wikipedia database [10], dated May 27th, 2007.

4 Global Analysis

From the URL included in the trace it is possible in most cases to determine the targeted wiki project and the type of operation issued by the client. Table 1 shows the different types of requests addressed to Wikipedia, and their relative frequency.

We can see that most of the traffic is generated by the action of end users issuing read operations to wiki pages. Since it is common for pages to include multiple uploaded images and static files, these two types of requests account for more than 70% of all requests. We can also see that page editions (at 0.04%) are very infrequent compared to page reads, and image uploads are even less frequent (0.005%). It is thus clear that a high degree of caching or replication can be used to improve performance. It should also be noted that a nontrivial number of page requests are for formats different from the default HTML, which suggests that in some cases replicating the wikitext instead of, or in addition to the final HTML would produce further performance improvements.

Not all wikis in Wikipedia are equally popular or equally used. Table 2 shows the distribution of request load by the wiki projects as well as the ratio of HTML read requests to save requests. Although Wikipedia contains more than 800 wikis which received more than 2000 requests in our sample, almost half of the total traffic is directed to the English Wikipedia. About 90% of the traffic is concentrated in the 10 most popular wikis. This shows that a strategy of using a separate database for each wiki cannot efficiently solve the scalability issues since there is a large imbalance in the load. This also justifies a more comprehensive study of the English Wikipedia in order to gain a deeper understanding of the issues that affect global Wikipedia performance. We can also see that the read/save ratio varies significantly for the different language editions of Wikipedia. This shows that factors such as culture, size, and geographical distribution of the user base influence the workload an thus have an effect on how strategies should be selected to improve performance.

Our next analysis examines the usage of the most popular wikis. Figure 2 shows the request rate for the four most popular Wikipedias during a two-

Request Type	Description	Frequency
Wiki Page Current HTML Read	Request for the current version of a wiki page using the normal HTML rendering.	15.01%
Wiki Page Current Other Read	Request for the current version of a wiki page using a different format.	6.87%
Wiki Page Save Submit	Request that results in a page update or creation.	0.04%
Wiki Page History Read	Request for the edition history of a page.	0.07%
Wiki Page Version Read	Request for a specific version of a page. It may be a diff operation, in which case two versions are compared.	0.07%
Wiki Page Other	Request in which a page name is specified, but the information retrieved is independent from the page's wikitext. For example, Wikipedia allows obtaining the Javascript or CSS used in the rendering of a specified page without obtaining the page itself.	4.24%
Binary File Normal Read	Request for a user uploaded binary file, typically an image.	12.71%
Image Thumbnail Read	Request for a thumbnail of a user uploaded image.	27.40%
Special Upload	Request for the possible upload of a binary file. It can be a new file, an update of an existing file or the retrieval of a form that allows the user to send an actual upload request. It is impossible to determine the name of the uploaded file from our traces nor if it is an actual upload.	0.005%
Special Full Text Search	Keyword search request handled by the search component of the wiki engine.	1.11%
Special Other	Request for a special page other than a full text search or upload. Some of these special pages result in the retrieval of wiki pages. However, the names of the wiki pages involved cannot be obtained from the trace.	0.70%
Static File	Request for a static file. These are usually files used in the rendering of wiki pages, such as CSS and Javascript files as well as generic images such as bullets.	30.04%
API	Requests directed to a web service API. Most of these requests result in the retrieval of wiki pages. It is often not possible to determine the names of the pages involved.	0.05%
Static Wiki Page	A static HTML version of Wikipedia is available as an alternative to the normal one. The static version is updated periodically but is never guaranteed to have the most current version of a page.	0.009%
OpenSearch	Requests that produce search results in a standardized format.	1.03%
Unidentified	Other types of request.	0.64%

Table 1: Wikipedia request types, and their frequencies.

Wiki	Frequency	HTML Read/Save Ratio
English Wikipedia	45.5%	480.0
Wikipedia Commons	16.9%	-
German Wikipedia	6.3%	504.7
Japanese Wikipedia	6.0%	1081.2
Spanish Wikipedia	4.8%	458.8
French Wikipedia	3.1%	228.1
Mathematical formulas	2.3%	-
Italian Wikipedia	1.8%	216.2
Portuguese Wikipedia	1.8%	346.5
Polish Wikipedia	1.8%	363.5
Dutch Wikipedia	1.1%	258.7
Others (< 1% each)	8.6%	-

Table 2: Distribution of load and read/save ratios across different wiki projects

week period. The workload follows typical time-of-day and day-of-week patterns. However, the load variations differ for each wiki. For example, within a single day the request rate is expected to change by a factor of about 3.5 in the English Wikipedia and by a factor that can be as high as 30 in the German Wikipedia. On the other hand, we did not observe any flash crowds that may affect the normal daily behavior.

5 English Wikipedia

We now focus on the English edition of Wikipedia to conduct a more in-depth workload analysis. Figure 3 shows the popularity distribution of all referenced pages in the trace. Pages are ordered by decreasing number of HTTP requests. As reported for other types of websites, the page popularity distribution approximately follows a Zipf distribution [4, 5, 13]. However, the five most requested pages show a popularity two orders of magnitude higher than any other page. This is an artifact of the structure of Wikipedia pages: four of these pages contain CSS or Javascript code included in many other HTML wikipedia pages and the other is the main page, which is the most popular HTML-rendered page for obvious reasons.

(a) English (b) German

(c) Japanese (d) Spanish

Figure 2: Usage of the four most popular Wikipedias over a two-weeks period.

Figure 3: Page popularity distribution

Figure 4: Correlation between HTTP reads and HTML reads

Most requests performed by ordinary users are read operations that normally result in a default HTML rendering of the requested page. Wikipedia implements caches for these HTML renderings to reduce the centralized database load. However, a page can be rendered in many different formats. Figure 4 shows the correlation between all HTTP page reads, and reads that result in the default HTML rendering. Each point represents a page, and pages where all HTTP reads are in the default HTML format appear in the diagonal. For readability reasons the graph shows a random sample of 10% of all pages. It can be seen that for a significant number of pages, especially among the less popular pages, the number of read operations in a non-default format exceeds the number of default HTML reads. More specifically, this is the case for 21.8% of all referenced pages in the trace with at least one read operation involving the most recent version of the page. This suggests that it is important to consider the caching or replication of wikitext as an alternative or in addition to HTML caching and replication.

5.1 Save Operations

Now we turn our attention to save operations. Figure 5 shows pages ranked by the number of save operations. We can see that the popularity of save operations also follows a Zipf-like distribution. We note that only 2.5% of the pages referenced in the trace have at least one save operation. However, these pages represent 56.2% of all HTTP page requests. This forces us to

Figure 5: Distribution of page save operations

make a distinction between maintained pages, which have at least one save operation, and unmaintained pages, which present no save operations in the trace. Unmaintained pages are effectively read-only and can be treated as if they were static files. Maintained pages, on the other hand, require more attention as they are more difficult to host in a scalable way and represent more than half of all page requests.

Figure 6 shows the correlation between the number of read and save operations. Each point represents a page. All pages with at least one read and one save operation are represented in the graph. The line represents the average number of save operations for pages with a given read popularity. We observe some correlation between the number of read and save operations per page, so a popular page will be updated more frequently than an unpopular one. There is however a lot of variation from page to page.

Now we consider the ratio between the number of save and read operations for a given page. This metric is important because it can help determine the appropriate replication degree for a maintained page. Figure 7 shows the correlation between the overall popularity of a page and its save/read ratio. Each scattered point represents a page, and the line represents the average save/read ratio for pages with a given read popularity. All pages with at least one read and one save operation are represented in the graph.

Figure 6: Correlation between the numbers of page read and save operations

Figure 7: Correlation between read requests and save/read ratio

Figure 8: Correlation between save requests and save/read ratio

The save/read ratio clearly decreases when the page popularity increases. This is a good property, as a low save/read ratio means that replication and caching become easier. The most popular pages can thus be aggressively replicated, while the least popular pages will rather benefit from distribution, where very few servers are responsible for each page.

Figure 8 shows the same save/read page ratio, but this time as a function of the total number of saves of the page. Clearly, the number of page saves cannot be used as a predictor for the save/read ratio, and thus cannot be used by itself to define caching and replication strategies.

5.2 Flash crowds

As we saw previously, Wikipedia as a whole does not seem to exhibit flash crowd behavior. However, individual pages may present spikes of popularity. Such events can potentially have an important impact on a decentralized hosting infrastructure, since each server may host only a handful of documents.

We analyzed occurrences where the number of requests that a page receives in a whole day represents at least a 10-fold increase or decrease with respect to the previous day. We ignored cases where the daily number of requests is less than 100 in our trace for both the previous and actual day of the event. Table 3 shows the number of sudden increases and decreases that we observed.

Request Type	Number of events	Number of Pages
Increases	16523	14913
Decreases	8895	8549

Table 3: Number of sudden increases and decreases in daily number of requests by a factor of at least 10

Figure 9: Distribution of flash crowds vs. page popularity

As we can see, page-level flash crowd events are relatively common, as we noticed on average 157.4 sudden increases per day. Furthermore, it is uncommon for a page to experience multiple crowds in the studied period.

Figures 9 and 10 relate flash crowd behavior to popularity. In Figure 9 the top curve shows the number of HTTP read requests for all pages ranked by popularity, while the bottom curve ranks only pages that exhibit at least one flash crowd event in its daily request rate. We see that flash crowds commonly happen to pages with a wide range of popularity. Figure 10 shows that the fraction of pages that experience flash crowds is relatively independent of the initial page popularity. Therefore, popularity cannot be used to predict flash crowds.

Our trace also shows that 14913 pages experienced at least one flash crowd event of any type, and that 31.9% of these had at least one save operation during the whole period. This fraction is significantly greater than the general fraction of maintained pages with respect to all pages,

Figure 10: Fraction of flash-crowded pages by popularity

which means that a maintained page is more likely to experience a flash crowd than an unmaintained page.

These results demonstrate that page-level flash crowds cannot be ignored when designing a decentralized hosting architecture. On the other hand, predicting which page may experience a flash crowd is difficult, since flash crowds are often related to real-world events external to Wikipedia. One may thus need to apply more sophisticated flash-crowd warning systems, such as [6].

5.3 Nonexisting Pages

4% of Wikipedia requests are addressed to pages that do not exist. Such requests may result from simple typing mistakes by the end users. This could however be problematic when using a decentralized hosting infrastructure, as each request may potentially cause a distributed lookup operation.

Figure 11 shows the "popularity" of requested nonexisting pages. This popularity again closely follows a Zipf distribution. The effects of requests to non-existing pages can therefore be largely reduced by using negative caching techniques where front-end servers cache the information that a page does not exist.

We however also note that certain non-existing pages are requested too frequently to be attributed to typing mistakes alone. A typical URL corresponding to one of the most requested nonexisting pages is the following:

Figure 11: "Popularity" distribution of nonexisting pages

`http://en.wikipedia.org/wiki/404_error//skins-1.5/common/iefixes.js?61//skins-1.5/common/IEFixes.js?61//skins-1.5/common/IEFixes.js?61//skins-1.5/common/IEFixes.js?61//skins-1.5/common/IEFixes.js?61//w/index.php?title=-&action=raw&gen=js//w/index.php?title=-&action=raw&gen=js//skins-1.5/common/wikibits.js?6`

This type of URL is clearly not produced by a normal user. In addition, such pages are among the flash-crowded pages with most requests. For example, the URL just mentioned is the twelfth most frequently requested page in the studied period, and the fifth most requested one among pages with at least one flash-crowd event. Moreover, all the requests for this URL were carried out over a seven-day period. This suggests that these requests are generated by malicious users who attempt to attack the normal operation of Wikipedia. This type of attack is one source of page-level flash crowds.

5.4 Indirect reads and saves

Our last analysis concerns the effect of pages that are read indirectly. As we explained above, Wikipedia allows one page to include the content of another page with the inlusion and redirection features. One important implication of these features is that save operations to included or redirected pages affect the consistency of replicated or cached HTML renderings of their master pages. This means, for example, that an otherwise read-only page

can become a frequently updated page if it includes a frequently updated template page.

Our analysis assumes that all direct requests for pages that reference other pages via inclusion or redirection result in the indirect read of the referenced pages, and that the maps of inclusions and redirections do not change over time with respect to the one available in our database snapshot. These assumptions are unrealistic because requests may be processed by caches thus reducing the number of database accesses. In addition, page updates may result in the modification of the inclusion and redirection maps. This is especially true for newly created pages. However, our main purpose is to find out the impact of update operations on the ability of other pages to be cached or replicated. While our indirect popularity measure does not provide an accurate estimate of indirect database accesses, it does provide an estimate of the popularity of the master pages that would need to be updated or invalidated in case of a save operation to the included page.

Our trace shows that 721,764 pages were read indirectly at least once. Of these, 86.3% were never updated. Figure 12 shows the correlation between the number of indirect reads and save operations for pages that have at least one indirect read and one save operation (these represent 13.7% of indirectly read pages). The graph shows that the most indirectly-read maintained pages, have a small number of updates. The number of update operations increases for less popular indirectly read pages. This may be due to the Wikipedia policy of protecting popular templates from being updated by unprivileged users. Our conclusion is that indirect save operations have an impact on caching and replication of master pages, but with the use of appropriate policies this impact can be reduced significantly in practice.

6 Images

As we have seen in Table 1, read requests for images and other binary files are read more frequently than pages. This is explained by the fact that wiki pages often include several images. We do not extend our detailed analysis to images because they are, on average, updated 7 times less frequently than pages in the English Wikipedia, so they can in general be regarded as static files.

Figure 12: Correlation between indirect reads and saves

7 Implications

Our study of the Wikipedia workload provides important insight relevant for improving the performance of Wikipedia hosting, especially in a decentralized and collaborative setting, which is our final goal. The most important result of our analysis is that we are now able to classify Wikipedia pages according to their workload and we are able to give some guidelines on how to exploit the distribution, replication, and caching techniques to improve performance. We summarize these guidelines in Table 4.

To efficiently handle the workload, a decentralized solution should automatically determine the appropriate replication and caching policies for every page and be prepared to handle operational situations such as flash crowds. Classifying pages and determining per-page replication strategies is a solvable, yet nontrivial problem [15]. Such a solution must also satisfy a number of functional requirements such as efficiently detecting if a page exists or not and implementing relationships among pages such as redirection, inclusion and edition protection; as well as extra-functional issues such as load balancing, security, and privacy.

The problem of flash crowds has been studied and several solutions have been proposed [6, 17, 3, 18, 8]. Some of these solutions may need to be adapted to a decentralized environment and they must take into account the daily and weekly access patterns.

Type of page	Fraction of requests in trace	Commentary
Pages that are read-only in practice and are mostly read (>75%) in default HTML format	29%	HTML caching or replication seems to be the most appropriate strategy. The degree of replication should depend on the popularity of the page.
Pages that are almost read-only and have a significant fraction (≥25%) of reads in alternate formats	11%	Wikitext replication in combination with HTML caching is a good choice in this case. The degree of replication should depend on the popularity of the page.
Maintained pages that are mostly read (>75%) in default HTML format	48%	HTML replication with a replication factor controlled by the popularity of the page.
Maintained pages that have a significant fraction (≥25%) of reads in alternate formats	8%	Wikitext replication with a replication factor controlled by the popularity of the page. HTML caching can be considered if the read/save ratio is considerably high.
Nonexisting pages	4%	Negative caching combined with attack detection techniques.

Table 4: Wikipedia page characterization according to the workload. The first four rows of the table consider only pages marked as existent in the database snapshot.

The problem of implementing relationships among pages in a decentralized way is complicated by the fact that the relationships must be kept in a consistent state in the presence of updates and partial failures. For example, if a page is modified so that it is protected from editions and the administrator chooses a cascading option for this protection, then all included pages must be notified of this change. Solving this problem in a decentralized environment is similar to implementing consistency for replicas, which is a problem that has to be solved both to improve performance and to achieve fault tolerance.

The problem of detecting if a page exists or not, given its name, is crucial for two reasons. First, Wikipedia functionality requires links to nonexisting pages to be rendered in a different color than links to valid pages. Second, this is, with little modification, the same problem as locating the node where a page resides in order to forward requests to it.

However, the most difficult challenge we face when implementing a decentralized and collaborative system for hosting Wikipedia is solving all the aforementioned problems in an environment where mutually untrusted parties participate while at the same time guaranteeing fair resource usage and privacy for regular Wikipedia users who have nothing to do with the hosting of the system.

Acknowledgments

We wish to thank the Wikimedia Foundation, especially Gerard Meijssen and Tim Starling, for making their access trace available for our study.

This work is in part supported by the Programme Alban, the European Union Programme of High Level Scholarships for Latin America, scholarship No.E05D052447VE.

References

[1] Alexa Internet. Alexa web search - top 500, 2007. `http://www.alexa.com/site/ds/top_sites?ts_mode=global`.

[2] Virgílio Almeida, Azer Bestavros, Mark Crovella, and Adriana de Oliveira. Characterizing reference locality in the WWW. In *Proceedings of the IEEE Conference on Parallel and Distributed Information Systems (PDIS)*, Miami Beach, FL, 1996.

[3] Ismail Ari, Bo Hong, Ethan L. Miller, Scott A. Brandt, and Darrell D. E. Long. Managing flash crowds on the internet. *mascots*, 00:246, 2003.

[4] Martin F. Arlitt, Diwakar Krishnamurthy, and Jerry Rolia. Characterizing the scalability of a large web-based shopping system. *ACM Transactions on Internet Technology*, 1(1):44–69, 2001.

[5] Martin F. Arlitt and Carey L. Williamson. Web server workload characterization: the search for invariants. In *SIGMETRICS '96: Proceedings of the 1996 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 126–137, New York, NY, USA, 1996. ACM Press.

[6] Yuliy Baryshnikov, Ed G. Coffman, Guillaume Pierre, Dan Rubenstein, Mark Squillante, and Teddy Yimwadsana. Predictability of web-server traffic congestion. In *Proceedings of the Tenth IEEE International Workshop on Web Content Caching and Distribution*, pages 97–103, September 2005.

[7] L. Bent, M. Rabinovich, G. M. Voelker, and Z. Xiao. Characterization of a large web site population with implications for content delivery. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 522–533, New York, NY, USA, 2004. ACM Press.

[8] Xuan Chen and John Heidemann. Flash crowd mitigation via adaptive admission control based on application-level observations. *ACM Trans. Inter. Tech.*, 5(3):532–569, 2005.

[9] Ludmila Cherkasova and Minaxi Gupta. Analysis of enterprise media server workloads: access patterns, locality, content evolution, and rates of change. *IEEE/ACM Transactions on Networking*, 12(5):781–794, 2004.

[10] Wikimedia Foundation. Wikimedia dump service, 2007. `http://download.wikimedia.org/`.

[11] Krishna P. Gummadi, Richard J. Dunn, Stefan Saroiu, Steven D. Gribble, Henry M. Levy, and John Zahorjan. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 314–329, New York, NY, USA, 2003. ACM Press.

[12] Lei Guo, Songqing Chen, Zhen Xiao, and Xiaodong Zhang. Analysis of multimedia workloads with implications for internet streaming. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 519–528, New York, NY, USA, 2005. ACM Press.

[13] Frank T. Johnsen, Trude Hafsoe, and Carsten Griwodz. Analysis of server workload and client interactions in a news-on-demand streaming system. In *ISM '06: Proceedings of the Eighth IEEE International Symposium on Multimedia*, pages 724–727, Washington, DC, USA, 2006. IEEE Computer Society.

[14] Bo Leuf and Ward Cunningham. *The Wiki Way: Collaboration and Sharing on the Internet*. Addison-Wesley Professional, April 2001.

[15] Guillaume Pierre, Maarten van Steen, and Andrew S. Tanenbaum. Dynamically selecting optimal distribution strategies for Web documents. *IEEE Transactions on Computers*, 51(6):637–651, June 2002.

[16] Kunwadee Sripanidkulchai, Bruce Maggs, and Hui Zhang. An analysis of live streaming workloads on the internet. In *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 41–54, New York, NY, USA, 2004. ACM Press.

[17] Angelos Stavrou, Dan Rubenstein, and Sambit Sahu. A lightweight, robust p2p system to handle flash crowds. *Selected Areas in Communications*, 22(1):6–17, 2004.

[18] Zhihong Tian, Binxing Fang, and Xiaochun Yun. Defending against flash crowds and malicious traffic attacks with an auction-based method. In *WI '04: Proceedings of the 2004 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 24–28, Washington, DC, USA, 2004. IEEE Computer Society.

[19] Guido Urdaneta, Guillaume Pierre, and Maarten van Steen. A decentralized wiki engine for collaborative wikipedia hosting. In *Proceedings of the 3rd International Conference on Web Information Systems and Technologies*, pages 156–163, March 2007.