Computer Networks 53 (2009) 1830–1845

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Wikipedia workload analysis for decentralized hosting ☆

Guido Urdaneta *,[1], Guillaume Pierre, Maarten van Steen

VU University, Dept. of Computer Science, De Boelelaan 1083, 1081HV Amsterdam, The Netherlands

ARTICLE INFO

Article history:
Received 17 July 2008
Received in revised form 23 December 2008
Accepted 22 February 2009
Available online 6 March 2009

Responsible Editor: C. Westphal

Keywords:
Workload analysis
Wikipedia
Decentralized hosting
P2P

ABSTRACT

We study an access trace containing a sample of Wikipedia's traffic over a 107-day period aiming to identify appropriate replication and distribution strategies in a fully decentralized hosting environment. 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, significant load variations and requests for nonexisting pages. We also review proposed decentralized wiki architectures and discuss how they would handle Wikipedia's workload. We conclude that decentralized architectures must focus on applying techniques to efficiently handle read operations while maintaining consistency and dealing with typical issues on decentralized systems such as churn, unbalanced loads and malicious participating nodes.

1. Introduction

Despite numerous pessimistic predictions, Wikipedia is a blatant success. As of December 2007, Wikipedia contains approximately 9.25 million articles in 253 languages, and is considered one of the ten most visited web sites on the Internet [3]. Its uninterrupted popularity growth has forced its operators to upgrade the hosting architecture from a single server to a distributed architecture with more than 350 servers at three locations on different continents [9]. Although Wikipedia is by far the largest system of its kind, many other wikis are being developed as a support for various collaborative tasks [20].

The popularity of wiki-based applications, together with their fundamentally collaborative nature, has driven several research efforts to host them in a peer-to-peer fashion [11,21,22,27,32,35]. Unlike a traditional centralized solution, in a decentralized system the web site is hosted by a dispersed set of nodes where there is no central control point that has full knowledge about the entire system. The motivations for hosting a wiki in a decentralized fashion vary from scalability and reduced operational costs considerations to the desire to provide additional functionality such as offline page edition.

We observe, however, that none of these efforts had the opportunity to extensively study the proposed system behavior under a realistic workload. Although wikis conceptually provide very simple functionality, the read/write nature of the traffic addressed to them and certain advanced functionalities make it remarkably difficult to host wikis in a decentralized fashion. Achieving good performance and scalability in this type of environment relies upon the appropriate combination of techniques such as distribution, replication and caching [23]. The optimal application of these techniques depends greatly on certain properties of each individual hosted item. For example, the popularity distribution of the hosted documents gives an indication of which are more important and thus better candidates for massive replication or placement in powerful servers; the frequency of save operations for a document or, more properly, its save/read ratio, may be used

☆ A preliminary version of this paper was published in the local conference of the Dutch ASCI Graduate School [33]. The version presented here uses a more recent data set, includes a validation of the data set, and several additional analyses.

* Corresponding author. Tel.: +31 20 5987754.
E-mail addresses: g.urdaneta@few.vu.nl (G. Urdaneta), gpierre@cs.vu.nl (G. Pierre), steen@cs.vu.nl (M. van Steen).

[1] Supported by the Programme Alban, the European Union Programme of High Level Scholarships for Latin America, scholarship No. E05D052447VE.

1389-1286/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2009.02.019

in the calculation of an appropriate replication factor; the presence of unexpected load variations related to real-world news or phenomena like the Slashdot effect [2] may indicate the need for special algorithms to handle such situations; requests for nonexisting items may require special measures since the node handling such a request in a decentralized environment does not necessarily know whether the item exists or not.

To illustrate the difficulty of decentralized wiki hosting, and to provide guidelines for future research on the topic, we obtained and studied an access trace containing a 10% sample of the total Wikipedia.org 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.

We analyze all these generic properties that affect the performance of a decentralized system, but we also take into account the effect of features specific to wikis, such as the fact that wiki pages can be read in different formats; that older versions of pages must remain available; and the use of mechanisms such as transclusion, which require reading multiple pages from the storage system to produce the final page in the format requested by the user.

The contribution of this work is twofold. First, to our knowledge, our study is the first to give insight on the functioning of a collaborative web site, where almost all of the content is created and updated by external users and not the operators. Previous Web workload analyses focused on other types of Web sites such as e-commerce sites [6], P2P file sharing systems [15], static Web sites [8,7,4] and multimedia delivery systems [16,31,12]. Second, we derive guidelines from our study that highlight a few do's and don'ts in the design of architectures for hosting some of the large wikis on the Internet.

The rest of the paper is organized as follows. Section 2 gives an overview of the Wikipedia operation. Section 3 describes the information available in the trace. Section 4 presents a general analysis of Wikipedia as a whole. Section 5 presents a detailed analysis of the English Wikipedia. Section 6 discusses how previously-proposed decentralized wiki systems would behave under a Wikipedia-like workload. Finally, Section 7 discusses other related work, and Section 8 serves as our conclusion.

2. Wikipedia operation

Wikipedia is composed of a number of wikis [20]. 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).

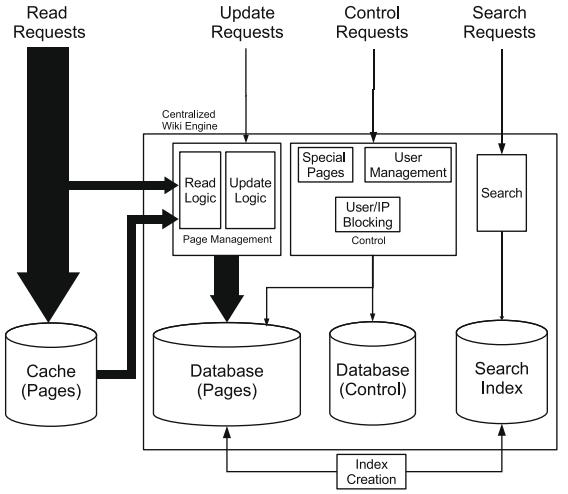

Fig. 1. Wikipedia architecture. The width of arrows roughly denotes the fraction of requests of each type.

As is shown in Fig. 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 generated by page read operations issued by anonymous (not logged-in) users is handled by a group of external cache servers, with a reported hit-rate of 85% for wiki pages and 98% for media files [9].

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, similar to a symbolic link. Redirection is implemented by rendering the content of the target page when the master page is requested. 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 applica-

tion in that it does not access the centralized database, but instead accesses a separate index file generated periodically from the text of the pages.

Finally, the control part groups the rest of the functionalities. It encompasses features such as 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; user/IP address blocking, which allows administrators to prevent page updates from certain IP addresses or user accounts; and 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 all requests directed to all the wiki projects they operate. The sample used in our study was generated by Wikipedia's frontend proxy caches, and contains 20.6 billion HTTP requests corresponding to the period from September 19th, 2007 to January 2nd, 2008. Each request in our trace is characterized by a unique ID, a timestamp, the requested URL, and a field that indicates if the request resulted in a save operation. Each Wikipedia request in the studied period has a 10% probability of being included in our trace. As a consequence, no bias is introduced at the request level. However, aggregate information related to specific pages may be inaccurate due to the sampling and user actions such as page removal and creation. We quantify the impact of this inaccuracy in Section 5, where we analyze the English Wikipedia at the page level.

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 focusses on server-side information. The only client-side data available is update information present in public snapshots of the Wikipedia database. For some of our analyses, we used a snapshot of the English Wikipedia database, dated January 3rd, 2008 [14].

4. Global analysis

From the URL included in the trace it is possible 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 reference multiple uploaded images and static files, these two types of requests account for more than 64% of all requests. We can also see that page editions (at 0.03%) are very infrequent compared to page reads, and image uploads are even less frequent (0.002%). 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 the trace has references to more than 800 wikis with more than 2000 requests in our sample (many of them nonexisting), almost half of the total traffic is directed to the English Wikipedia, and 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 and thus have an effect on how strategies should be selected to improve performance.

Our next analysis examines the usage of the most popular wikis. Fig. 2 shows the request rate for the four most popular Wikipedias during a two-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 2.3 in the English Wikipedia and by a factor that can be as high as 19 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. The data we consider in this section includes all the requests in the trace directed to a wiki page in the English Wikipedia. In this study requests for uploaded images, special pages, static files, or other types of objects are not included. Requests to pages that specify an invalid page name are included and analyzed in Section 5.5.

There are two main reasons why we focus on wiki pages. First, they can be updated by ordinary users at any time, so they introduce a nontrivial consistency problem in a decentralized replicated scenario. Second, they are directly or indirectly responsible for the vast majority of the Wikipedia traffic. As we have seen in Table 1, static files and uploaded images are requested more frequently than pages, but this is explained by the fact that wiki pages often reference several images and static files. Static files are rarely updated, if ever, so they do not represent a special difficulty in a decentralized environment. Uploaded media files can be updated by users, but in practice this occurs very rarely, so they can, in general, be regarded as static files or as read-only wiki pages.

Table 1
Wikipedia request types, and their frequencies expressed in fractions of the total number of requests (numbers do not add up to 100% due to rounding error).

Description	Frequency (%)
Requests for static files. 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	24.04
Requests for user-uploaded media files, typically images	21.88
Requests for thumbnails of user-uploaded images	18.70
Requests for the current version of a wiki page using the default HTML rendering	13.15
Requests for the current version of a wiki page using a different format, such as printer-friendly versions or raw wikitext	8.50
Requests related to cache maintenance	5.18
Requests 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	4.49
Requests that produce search results in a standardized format known as OpenSearch	1.31
Requests for special pages other than 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	0.88
Keyword search requests handled by the search component of the wiki engine	0.81
Requests directed to a web service API. Most of these requests result in the retrieval of wiki pages, but in many cases it is not possible to determine the names of the pages involved	0.14
Requests for the edition history of a page	0.06
Requests for a specific version of a page. It may be a diff operation, in which case two versions are compared	0.06
Requests that result in a page update or creation	0.03
Requests for a static HTML version of Wikipedia that 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	0.01
Requests for possible uploads of media files. They can refer to new files, updates of existing files or 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	0.002
Other types of request	0.75

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

Wiki	Frequency (%)	HTML read/save ratio
English Wikipedia	45.05	480.0
Wikipedia commons	14.33	N/A
German Wikipedia	7.07	504.7
Japanese Wikipedia	6.19	1081.2
Spanish Wikipedia	4.87	458.8
French Wikipedia	3.42	228.1
Mathematical formulas	2.45	N/A
Italian Wikipedia	2.03	216.2
Portuguese Wikipedia	1.84	346.5
Polish Wikipedia	1.70	363.5
Dutch Wikipedia	1.1	258.7
Others (<1% each)	9.95	Unknown

In this section, unless explicitly stated, when we refer to the total number of pages, we mean all the requested page names, including invalid ones. When we refer to existing pages, we refer to pages that were registered as existing in the database at some point during the studied period. This includes pages that were deleted or renamed. When we denote a number as coming from the database (e.g. number of updates in the database), we are using data from the database snapshot and not from the trace.

5.1. Trace validation

Our first analysis studies the validity of our trace. We claim that the trace is unbiased at the request level, since all requests have the same probability to appear in the trace. However, when requests are aggregated to produce information about specific pages, the sampling and certain events during the study period may produce inaccuracies. For example, low-traffic pages may not be represented at all in the trace, while pages deleted during the study period may appear as having a certain popularity that no longer applies.

Fortunately, the trace is not our only source of information. We also use a publicly available snapshot of the English Wikipedia database with accurate information about events such as page creation, deletion and renaming, as well as the full history of updates for all the valid pages at the time the snapshot was created.[2] Since the snapshot was taken just one day after the end of the study period, the information we derive from it is very accurate with respect to the trace.

Table 3 summarizes the classes of pages that may deserve special attention, and their importance in terms of the fraction of pages they represent with respect to the total number of existing pages, and the fraction of requests they represent with respect to the total number of page requests in the trace. It should be noted that we identify pages by their name, so our page-level analyses are actually at the page-name level.

We can see that the impact of these events on our page-level analyses is limited. Note that, despite their specificities, these pages do provide accurate information for many of our analyses. Our analyses measure some variables that characterize the set of all pages, and allow us to classify them. The most important variables we study at the page level are the distribution of popularity in terms of number of requests and number of save operations, the format in which pages are read, and the ratio between save and read operations.

Deleted and newly created pages present no problem regarding any of these variables apart from the fact that individual page popularity rankings cannot be used as a predictor for future popularity.

[2] The snapshot does not include the update history for deleted pages.

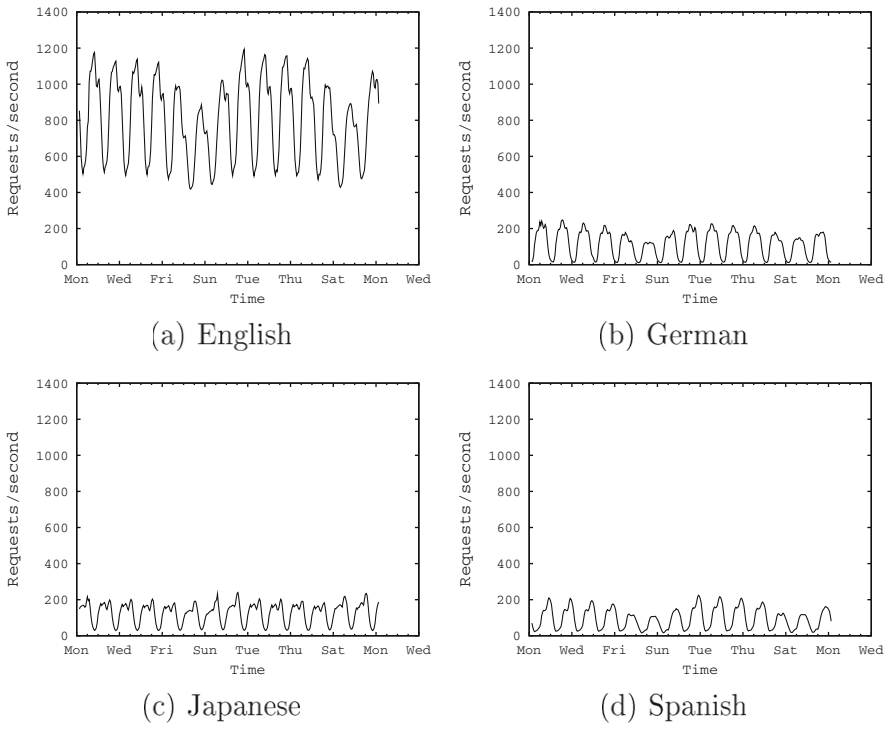

Fig. 2. Usage of the four most popular Wikipedias in the first two weeks of the studied period.

Table 3
Pages with events during the study period that may influence the accuracy of results.

Category	Fraction of pages (%)	Fraction of requests (%)
Pages that were created during the study period. The popularity rank derived from the number of requests during the study period cannot be used as an estimate of the future popularity rank for these pages	10.94	0.87
Pages that were deleted during the study period. Popularity rank in the studied period cannot be used to predict future popularity, which is expected to be very low or null	5.03	2.98
Pages that were renamed during the study period. In this case, we consider the page as two different pages	0.62	0.52
Pages that were renamed during the study period, with a new redirect page created with the old name. In this case, we consider the old page name a single page, and the new name as a new page. This is inaccurate, because the history of save operations stays with the new name, and the nature of the load for the old name may change significantly. For example, if a frequently updated page is renamed with a redirect, the new redirect page will probably have fewer updates. The result is that the save/read ratio for the old name will be calculated using two different workloads	0.11 (old name) 0.15 (new name)	0.21 (old name) 0.52 (new name)
Total	16.85	5.1

Renamed pages represent a negligible portion of the dataset. However, the little information they provide can still be useful. First, they contribute to correctly determine a popularity distribution of page names, which is relevant in a decentralized setting, where a distributed lookup service is needed. Second, what could be perceived as inaccuracies in the determination of save/read ratios can be considered as part of the load variations that pages are subject to. For example, renaming a page with a redirect is not very different from simply modifying it to redirect requests to another page.

Another potential source of inaccuracies is the sampling. Pages with very low traffic may be misrepresented, mainly because they may not be reported at all in the trace, but also because they may be overrepresented due to chance. We can get an indication of how representative the sample is by comparing the save operations reported in the trace, with the save operations stored in the database snapshot.

Fig. 3 shows the correlation between the number of save operations per page reported in the trace and the save operations registered in the database snapshot during the studied period. Each point in the graph represents a page, the thick line represents the median number of saves in the database for a given number of save requests in the trace, and the bars represent the 10th and 90th percentiles. We clearly see that there are only a few outliers, which can be attributed to, for example, manual intervention and

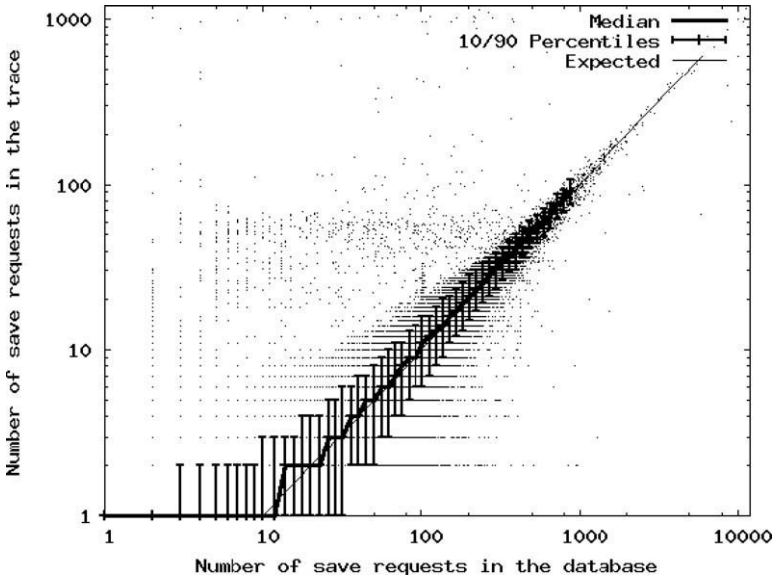

Fig. 3. Correlation between the number of save operations per page reported in the trace and in the database snapshot.

exceptions generated by the backend. The correlation coefficient is 0.81. As expected, the median is a good approximation to a line with slope 10. We can see that the variability is reduced as the number of saves increases. Since read requests are considerably more frequent than save requests, we can expect the sampling to be even more accurate in that case. We conclude that the sample is representative of the real traffic for virtually all pages, and that only pages whose influence on the load is negligible may have a significant probability of being misrepresented.

5.2. Page popularity

Fig. 4 shows the popularity distribution of all referenced pages in the trace. Pages are ordered by decreasing number of requests in the studied period. Unlike other types of websites, where the page popularity distribution closely follows a Zipf distribution [6,7,18], the popularity distribution of Wikipedia pages can be described as having three zones. First, we see that the four most popular pages show a popularity orders of magnitude higher than any other page. This is an artifact of the structure of Wikipedia pages: three of these pages contain CSS or Javascript code included in many other HTML-rendered Wikipedia pages and the other is the main page, which is the most popular HTML-rendered page for obvious reasons. A second part is constituted by approximately the next 20,000 pages in popularity, which roughly follow a Zipf distribution (with an estimated $\beta = 0.53$), indicated in the graph. A third part is constituted by the remaining 28.7 million pages, which deviate from the model by having lower frequencies than predicted by the Zipf distribution. The more rapid decline in pages with 50 or fewer requests might be due to the sampling, as we saw with pages with few save requests in Fig. 3.

In Wikipedia, 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. Fig. 5 shows the correlation between the number of reads per page in any format, and reads that result in the default HTML rendering. Each point represents a page, and pages where all reads are in the default HTML format appear on 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, the number of read operations in a non-default format is considerable. More specifically, 8.1% of the existing pages with at least one read operation in the trace have more than 25% of their read operations in a non-default format. For these pages it would be useful to replicate or cache the wikitext since it can be used to generate all possible formats.

5.3. Save operations

Now we turn our attention to save operations. Fig. 6 shows pages ranked by the number of save operations in the trace. We can see that the popularity of save operations approximately follows a Zipf distribution where we have computed β to be 0.64.

We note that 44% of the existing pages have at least one real (not sampled) save operation during the studied period, and they represent 91% of all page requests in the trace. This forces us to make a distinction between pure read-only pages, which are easy to host, and updated pages, which are more difficult to host in a scalable way and represent the vast majority of page requests. Moreover, pages with only a handful of updates in the study period can be considered unmaintained, and treated similarly to read-only pages.

Fig. 7 shows the correlation between the number of read and save operations. Each point represents a page.

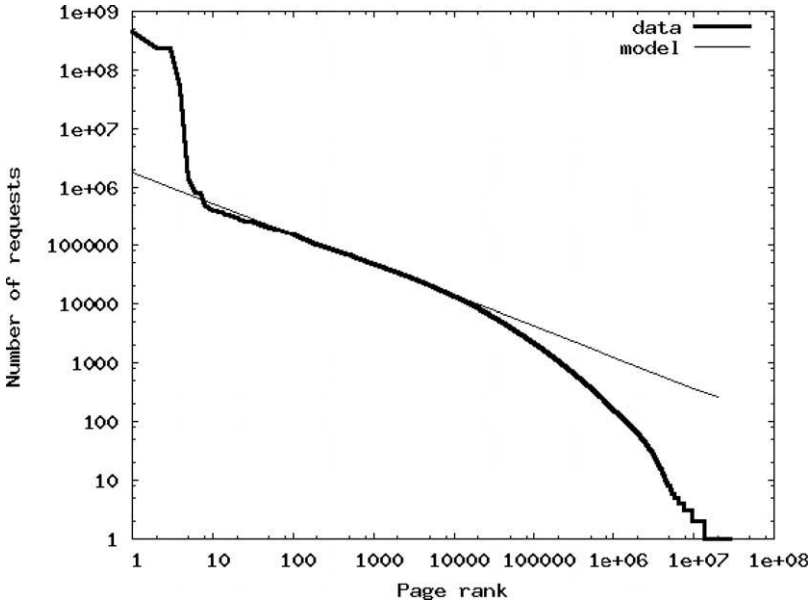

Fig. 4. Page popularity distribution.

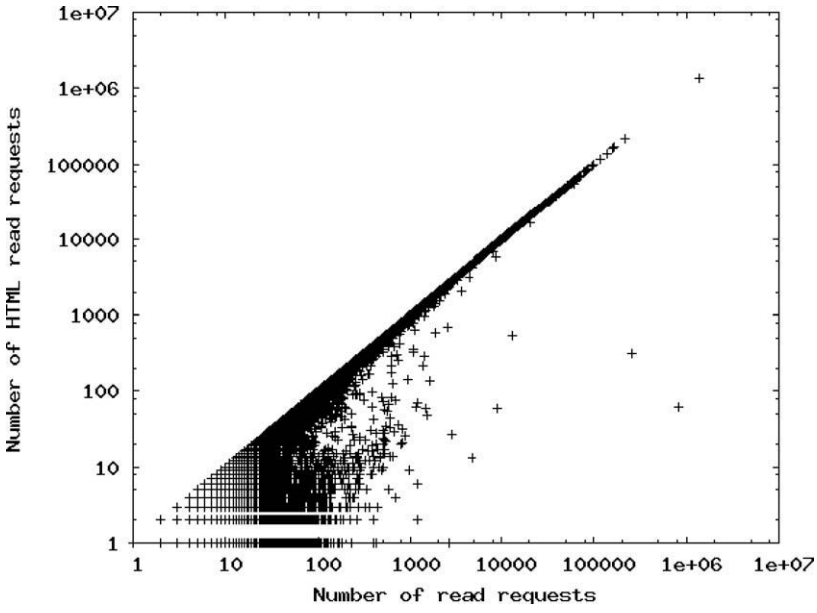

Fig. 5. Correlation between the number of all reads per page and page reads in the default HTML format.

All pages with at least one read and one save operation in the trace are represented in the graph. The line represents the median number of save operations for pages with a given read popularity. We observe that read and save operations per page are correlated, so popular pages are more likely to be frequently updated than unpopular pages. Moreover, we can see that for pages with at least 1000 read requests in the trace, the median read/save ratio is essentially constant and approximately equal to 1000, with a slight tendency to grow for the most popular pages. The variability is significant and there are many exceptions, but the order of magnitude of the variability is also constant.

These results suggest that less popular pages would benefit more from distribution than replication, and that replication is, in principle, a good strategy for the more popular pages. Moreover, the essentially constant median read/save ratio could be used to establish initial default

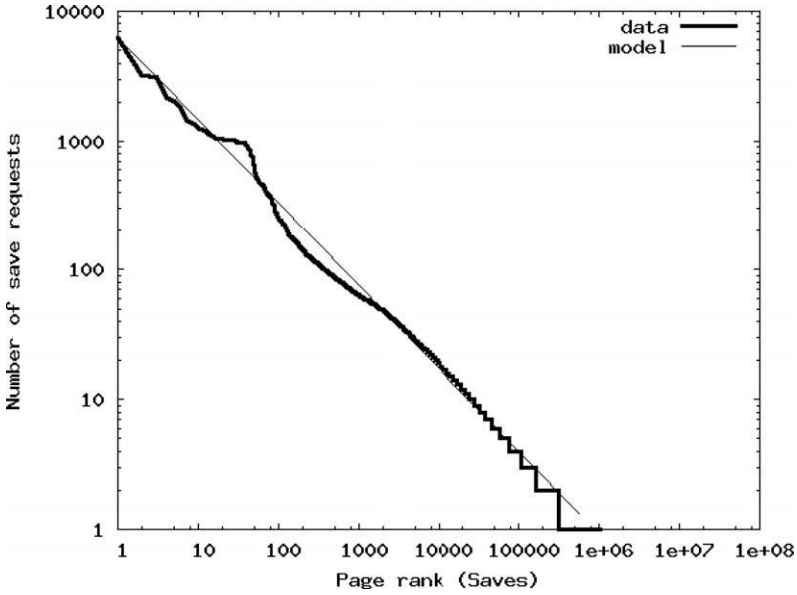

Fig. 6. Distribution of page save operations in the trace.

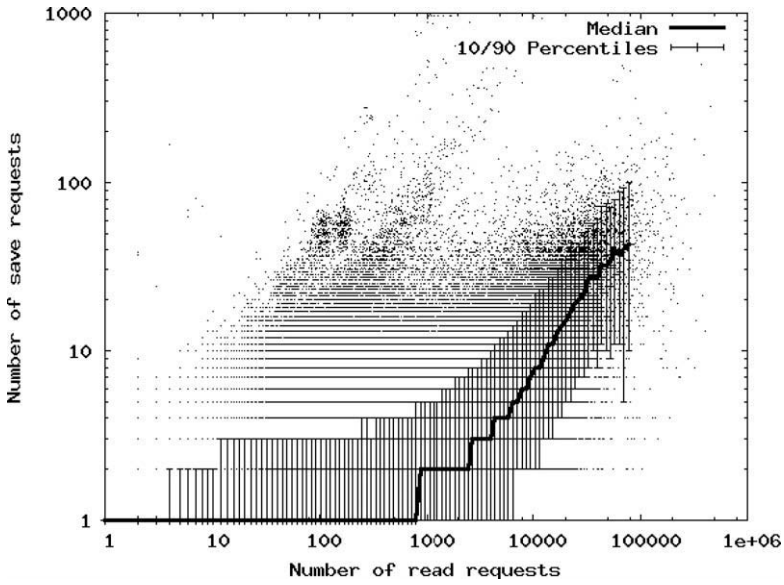

Fig. 7. Correlation between the numbers of page read and save operations.

policies for newly created pages. However, as we will see in Section 5.6, these considerations do not necessarily apply to replication of HTML-rendered pages.

5.4. Load variations

As we saw previously, Wikipedia as a whole does not seem to exhibit significant load variations apart from normal time-of-day and day-of-week patterns. 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 and thus be affected by such local variation.

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. In a decentralized system, such pages would most probably need to adjust their replication factor or migrate to more appropriate servers. 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 4 shows the number of significant load variations that we observed.

Table 4
Number of increases and decreases in daily number of requests per page by a factor of at least 10.

Request type	Number of events	Number of pages
Increases	14,404	13,649
Decreases	8177	7847

As we can see, significant variations of the popularity of pages are relatively common, as we noticed on average 133.6 increases per day (14,404 increases in 108 days). Furthermore, it was uncommon for a page to experience multiple events of this type in the studied period.

Fig. 8 relates page-level load variations to popularity and shows the fraction of pages with load variations given the popularity. We can see that the fraction of pages that experience load variations is essentially independent of the initial page popularity and that they cover a wide range of popularities. Therefore, current popularity cannot be used to predict future load variations.

5.5. Nonexisting pages

About 3.5% of page requests are addressed to pages that do not exist in the database. Such requests may result from simple typing mistakes by end-users. This could, however, be problematic when using a decentralized hosting infrastructure, as each request may potentially cause a distributed lookup operation.

Fig. 9 shows the "popularity" of requested nonexisting pages. This popularity again approximately follows a Zipf distribution (with an estimated $\beta = 0.68$).

The effects of requests to nonexisting pages can therefore be largely reduced by using negative caching techniques where frontend servers cache the information that a page does not exist.

We however also note that certain nonexisting pages are requested too frequently to be attributed to typing mistakes alone. One of the most requested nonexisting page names is the following (truncated to 255 characters during the analysis):

```
Skins-1.5/common/skins-1.5/common/skins-
1.5/common/skins-1.5/common/skins-1.5/com-
mon/skins-1.5/common/skins-1.5/common/
skins-1.5/common/skins-1.5/common/skins-
1.5/common/skins-1.5/common/skins-1.5/com-
mon/skins-1.5/common/skins-1.5/common/
skins-1.5/common/
```

This type of page name is clearly not produced by a normal user. It is possible that requests of this type are generated by faulty processes or malicious users trying to disrupt the normal operation of Wikipedia.

5.6. Indirect save operations

Our next analysis concerns the effect of pages that are included in other pages. As we explained above, Wikipedia allows one page to include the content of another page with the inclusion 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 that an otherwise read-only page can become a frequently updated page if it includes a frequently up-

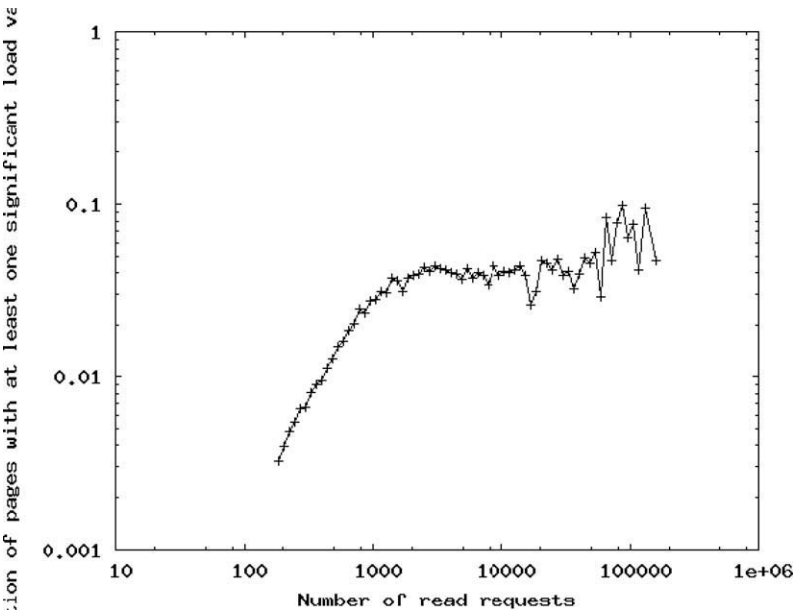

Fig. 8. Fraction of pages with load variations ranked by popularity.

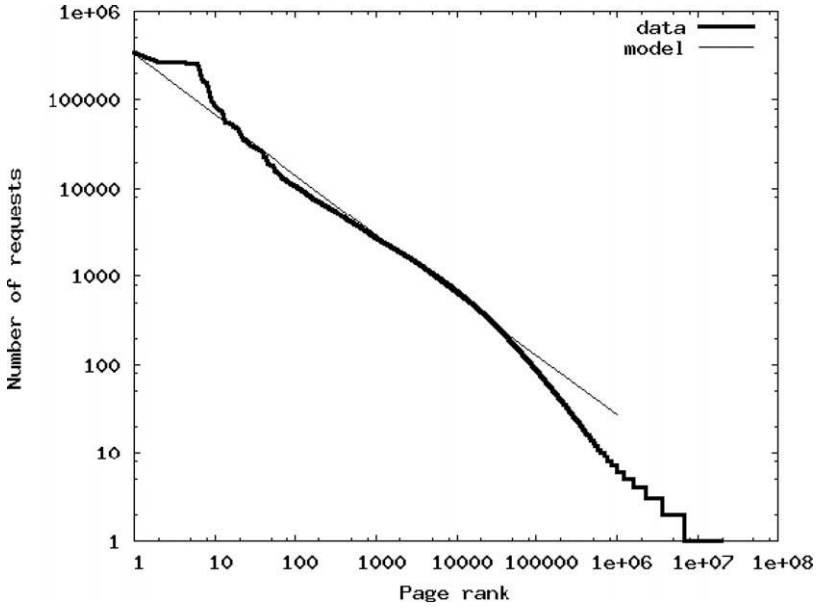

Fig. 9. "Popularity" distribution of nonexisting pages.

dated template page. For example, in an extreme case, the user discussion page "User_talk:Hu12" has a read/save ratio of 8.5 if we consider only direct save operations. However, if we take into account indirect save operations, this ratio drops to just 0.14, meaning that it is updated more often than read.

We try to determine the impact of indirect update operations on the ability of pages to be cached or replicated. To do this, we examine the inclusion and redirection maps available in the database snapshot and determine the indirect number of saves on the master pages. This method is not fully accurate, since the inclusion and redirection maps can change as the result of save operations. However, inclusion and redirection maps are not expected to change significantly during a 3.5-month period.

To quantify the impact of indirect saves, we compute the cache hitrate of an infinite size cache for HTML versions of pages. The cache hitrate is defined as the quotient between the number of requests that successfully read the cache versus the number of requests that either use the cache successfully or cause it to be reloaded. In our case, we use the expression $\frac{H-1}{H+D+I}$, where H is the number of read operations in HTML format, D is the number of direct updates and I the number of indirect updates.

Fig. 10 shows pages ranked by their HTML cache hitrate both taking and without taking into account the effect of

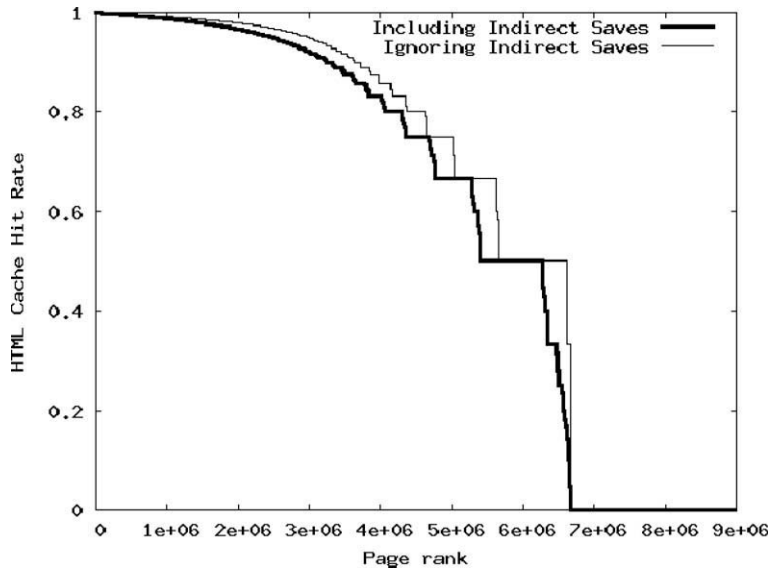

Fig. 10. Distribution of HTML cache hit-rate.

indirect saves. We see that if indirect saves are ignored, approximately 41% of pages have an HTML cache hitrate of 90% or more. Indirect updates make this fraction drop to approximately 37%.

Our conclusion is that indirect save operations can have an impact on caching and replication of master pages. Thus, it is important to minimize this impact by implementing policies such as protecting popular templates from being updated by unprivileged users, and by adopting separate policies for replicating wikitext and rendered versions of pages.

5.7. Old page versions

From the point of view of data storage, a page consists of the current version of its content, the content of all previous versions, and some metadata such as update restrictions. From the database snapshot, we estimate that old versions account for approximately 95% of the total storage space. However, they are rarely accessed, as shown by Table 1, and are read-only by definition.

This definition of a page is not convenient for load balancing, since the cost of moving a page would increase every time the page receives an update, to the point that after a certain number of updates, load balancing would stop being a viable solution.

A possible solution to this problem would be to assume that old versions are separate documents from the current version of the page, and that every time a page is updated, a new unpopular read-only document is introduced into the system. Thus, old versions of pages are easy to host on computers with low bandwidth and potentially spare disk space, such as ADSL-connected personal computers.

5.8. Discussion

Our study of the Wikipedia workload provides important insights relevant for hosting Wikipedia in a decentralized and collaborative environment. We have centered our detailed analysis on a number of essential page characteristics: frequency of requests, the format in which pages are read, the frequency of direct save operations, the frequency of indirect save operations, and relevant ratios between these variables. These variables together should determine the policies for distributing and replicating pages, which includes both the original wikitext source format and rendered HTML formats.

There are several factors that make it difficult to automatically determine policies. First, one must decide whether to replicate pages, or to try to place them on the most appropriate nodes. Second, one may need to select separate policies for the wikitext and rendered versions of a page, since direct save operations affect the consistency of both versions, while indirect updates affect only rendered versions. Third, all variables should be considered in combination to decide the policies, which may result in difficult tradeoffs. For example, an unmaintained page that is very popular in HTML mode and receives many indirect updates is in the situation where it should benefit from HTML replication, but this replication may introduce scalability problems due to the high indirect update rate. At the same time, the wikitext source could be easily replicated, and this could be used to generate the HTML, but at a significant cost in terms of performance with respect to a previously rendered replica.

In addition, a decentralized system must be ready to take emergency action under unexpected load variations on specific pages that may result from real-world events external to the system, and should efficiently handle invalid requests, including some that might try to disrupt the normal operation of the system.

In Table 5 we attempt to classify Wikipedia pages according to their workload, and give some guidelines for setting policies for each type of page. We classify pages according to three metrics: HTML cache hitrate, direct save/read ratio, and fraction of reads in HTML format. Together with page popularity, we believe that these metrics are the most relevant to select appropriate hosting policies at the page level. For simplicity, we classify pages as having a "high" or "low" values for each metric. Although the cutoff values are chosen rather arbitrarily, we believe our classification is sufficient for providing general guidelines.

Table 5
Wikipedia page characterization according to the workload. The cutoff values we use are: 85% or more is considered a high cache hit ratio, 15% or more is a high save/read ratio, and 75% or more reads in HTML is a high HTML fraction. Popularity is used to determine the strength of the policies.

HTML cache hit-rate	S/R ratio	HTML fraction	%Pages	% Requests	Commentary
Low	Low	Low	6.5	0.2	Replication of wikitext should be the preferred policy for these pages. Popularity should determine the replication degree
Low	Low	High	47.3	0.8	This type of page benefits mainly from load balancing of HTML replicas. Popularity should determine how aggressive the system should try to find appropriate nodes for these pages
Low	High	Low	1.5	0.0	These pages benefit mainly from load balancing. Replication of wikitext should be only for fault tolerance, and HTML replicas do should not be used. Popularity should indicate how aggressively the system should try to find appropriate nodes for these pages
Low	High	High	1.7	0.0	These pages benefit mainly from load balancing. Replication should be only for fault tolerance. HTML caches could be colocated with the wikitext replicas to make consistency easier. Popularity should indicate how aggressively the system should try to find appropriate nodes
High	Low	Low	0.3	20.2	These pages benefit from a high degree of replication of wikitext or alternate formats. There may be little need for HTML replicas, but they may exist and be colocated with wikitext replicas to make consistency easier. Popularity should determine the replication degree
High	Low	High	42.8	75.2	These pages benefit from a high degree of HTML caching. Popularity should determine the replication degree for rendered replicas. Wikitext replication may be used, but mostly for fault tolerance

We can distinguish two important sets of pages. The most important is by far the set of cacheable pages, which represent 43.1% of all pages and 95.7% of requests. These pages are relatively easy to host since they can be easily replicated in the appropriate format. Popularity can be used to determine the appropriate replication degree. The second important group of pages consists of pages for which caching makes no sense due to a low number of read operations with respect to the number of updates. In these cases, the system should rely on a load-balancing algorithm to place each page in the most appropriate node, with replication only for fault tolerance. These pages represent almost half of all pages, but only a small fraction of the requests. Therefore, they are good candidates for hosting in personal computers with ADSL connections.

Determining how much to rely on replication and how much on load balancing is complicated by the fact that the studied variables can take many more than just two values and may require nontrivial monitoring strategies that take into account the cyclic nature of the traffic. A decentralized solution should solve these problems and automatically determine the appropriate parameters for each page. Classifying pages and determining per-page strategies has already been shown to be a solvable, yet nontrivial problem [26].

A decentralized 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 categories, which allow encyclopedic articles to specify topics they cover, and edition protection, which allows an administrator to protect a page and its included pages from being updated. In addition, it should deal with extra-functional issues such as security and privacy.

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.

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. Solving this problem in a decentralized environment is similar to implementing consistency for replicas, which is a problem that must be solved both to improve performance and to achieve fault tolerance.

However, the most difficult challenge faced by 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 for participants, and privacy for regular Wikipedia users who have nothing to do with the hosting of the system.

6. Implications for decentralized wikis

Several decentralized wiki engines have been proposed recently with a varied set of goals that include scalability, cost reduction and offline operation, among others. However, none of them has been tested with a workload from a real-world site. Although not all of these systems were designed to sustain such a demanding workload as Wikipedia, we discuss these systems here, together with the expected performance they may have under a Wikipedia-like workload.

6.1. DistriWiki

One of the earliest decentralized wiki engines is DistriWiki [21]. DistriWiki is based on the JXTA platform, which provides generic peer-to-peer functionality such as searching, caching and discovery. In this system each node maintains a cache of document meta-data. Each time a document is created or updated, an advertisement is broadcast through the whole system. These cached meta-data facilitate the search for a node that contains the requested page.

As discussed previously, caching is an essential feature of a hosting system for Wikipedia, whose workload consists mostly of cacheable requests. However, it is unclear if DistriWiki's caching algorithms allow for caching of the pages themselves. This means that the node(s) responsible for some of the most requested pages of Wikipedia would have to serve each request, in addition to the background workload such as processing advertisements for each page update in the system. Concurrent updates are handled by timestamping all updates and letting users handle conflicts manually. Similarly, no algorithm is provided for efficiently handling churn, page relationships such as transclusion, or requests to nonexisting pages.

Finally, the main characteristic that precludes this system from being suitable for hosting Wikipedia or any other large-scale web site, is that it requires all users, even those who only read pages, to participate in the P2P network and use a specialized user interface application instead of a standard web browser.

6.2. XWiki Concerto

The XWiki Concerto system [11] is a P2P version of XWiki, an open-source wiki engine. In this system, the whole set of wiki pages is fully replicated to all participating nodes. Updates are propagated throughout the system using a gossiping protocol [13], and merged using the Woot algorithm [25]. Woot has many characteristics suitable for a P2P environment: it is fully decentralized, scalable and can guarantee that all replicas eventually converge to the same state.

XWiki Concerto aims at supporting three use cases: (1) massive collaboration where pages can be edited by large numbers of people, similarly to Wikipedia; (2) disconnected work where any participating node should continue offering the Wiki, including update functionality; and (3) opportunistic collaboration where a set of disconnected nodes should be able to create on-the-fly ad-hoc networks by connecting themselves.

Although the design choice of full state replication across all nodes is essential to support disconnected operation, it has important consequences in supporting the Wikipedia-like use-case: replicating large data sets in a dy-

namic environment where nodes can join and leave has been shown to have scalability problems [10]. Creating a replica of Wikipedia's data set (including the full history of previous operations addressed to each page) to a newly joined peer would require dozens of gigabytes of data transfer before the replica becomes usable. Similarly, decentralized update propagation is an interesting idea, but it implies that expensive merge operations will be issued in bursts to any particular node. Wikipedia receives several page updates per second which, if processed in bursts, may have a significant performance impact on the peers.

On the other hand, full page replication makes it easy to deal with challenges such as unequal page popularity, page relationships and nonexisting pages, which can be difficult to solve in architectures based on data distribution.

Although XWiki Concerto may struggle to handle a demanding workload such as Wikipedia, it looks more suited to corporate environments that may require alternative operation modes such as offline operation, with moderately sized data sets that can be fully replicated in a reasonable time.

A number of extensions to XWiki Concerto have been proposed, such as Wooki [35] and Swooki [30]. These systems propose improved algorithms to merge conflicting concurrent updates. As such, these systems provide interesting features such as concurrency awareness, which allows users to know whether a page is the result of an automatic merge by the system or not, and highlights the parts of a page that are subject to concurrency mismatches [5]. However, these systems use a similar architecture as XWiki Concerto, so they would exhibit similar strengths and weaknesses as XWiki Concerto for hosting a Wikipedia-like workload.

6.3. Piki

Piki [22] is a P2P wiki engine based on the FreePastry DHT [1]. In Piki, each wiki page falls under the responsibility of the DHT node whose ID in the DHT ID space is numerically closest to a hash of the page's name. To provide fault tolerance, each page is replicated to other numerically close nodes in the ID space. Piki handles concurrent updates using a master-slave approach where all updates to a page are handled by the primary owner of that page. This allows implementing concurrency semantics equivalent to those of Wikipedia. Piki improves these semantics by sending the issuer of a conflicting update not just the content of the newer version, as Wikipedia does, but the results of a merge of its update with the newer version. Piki also supports full-text search by storing inverted indexes in the DHT, and a form of semantic linking between pages.

Storing wiki pages in a DHT with replication for availability but no form of caching may create a number of difficulties under a Wikipedia-like workload. First, if the number of nodes is moderately large, the latency introduced by the logarithmic DHT lookups may be too high for an interactive application such as a web site. This may be particularly true for pages that make heavy use of transclusion, where a distinct distributed lookup is necessary for each transcluded element of a given page. Second, under churn, multiple nodes in the DHT may temporarily but simultaneously consider themselves the master for any particular page, possibly leading to inconsistent states. Third, the unbalanced page popularity distribution will most likely result in the nodes responsible for the most popular pages being saturated with requests while the rest of the system observes a low load. Note that load balancing within a DHT is considered a difficult problem for which a number of specific solutions exist [19]. Finally, similarly to DistriWiki, Piki requires a special user interface application to read pages from the system, which discards its application for a popular web site like Wikipedia.

6.4. Plantikow et al.

Plantikow et al. propose using a transactional replicated DHT with range query capabilities to implement a P2P wiki [27]. In this system, each page is stored in a cell composed by a number of DHT nodes according to the hash value of the page. The system uses techniques such as hybrid optimistic concurrency control, two-phase commit and read-only multiversioning to guarantee ACID properties even under churn and concurrent conflicting updates. This allows the system to support concurrency semantics essentially equivalent to Wikipedia's. In addition, this system exploits the range query capabilities of their DHT [28] to support relationships among pages.

The system, as defined at this point, is essentially a backend that, if coupled with a naive frontend system, may have problems dealing with the unbalanced load. The underlying DHT will automatically balance load but, as discussed in Section 5.4, significant load variations on individual pages may create continuous shifting of page data from one cell to the other.

In [29], the authors use a caching web server to render pages. They notice that the bottleneck is this frontend, rather than the backend, and suggest using multiple frontends to alleviate this problem, which is consistent with the findings of our workload analysis. However, they do not discuss any mechanisms to keep these caches consistent under direct and indirect update operations.

We note that this system is the only one to explicitly address churn, by storing data within cells composed of multiple nodes. However, while this creates stability in the DHT structure even in the case of churn, the data stored in the DHT still has to be replicated to new nodes each time they (re-)join the system.

6.5. Urdaneta et al.

We recently proposed a decentralized architecture for collaborative Wikipedia hosting [32]. In this design, pages are distributed over a network of collaborators that contribute their computing and networking resources to help host Wikipedia. A load-balancing algorithm tries to place pages on the most appropriate nodes and a distributed hash table is used to locate pages by name. Regular Wikipedia users use a normal web browser and do not necessarily participate in the collaborative hosting system.

Pages are replicated for fault tolerance and we use a gossiping anti-entropy protocol to maintain consistency. Concurrent updates are handled using a simple tie-breaking rule such as last writer wins, which violates Wikipedia's concurrency semantics, but is expected to work well in most cases in a read-mostly system like Wikipedia.

Regular users access the system through trusted nodes to prevent bogus responses from malicious nodes. Updates are signed by trusted frontends to ensure that malicious nodes cannot change their contents without honest nodes noticing.

The system has good resistance to churn as the DHT is used only as an index, and not as a storage system, while the load-balancing algorithm ensures that pages are copied only to nodes that can handle the copy operation without getting overloaded, which is better than in a DHT, where data must be migrated to nodes at specific locations in the ID space regardless of that node's load and capacity.

This system has two important weaknesses with regard to the Wikipedia workload. First, frontend nodes do not have any caching capability and use a DHT to locate wiki pages, which may result in unacceptable high latency for rendering pages. Second, it does not define any mechanism to efficiently support relationships among pages; hence, they must be solved with separate DHT lookups, which, while deterministic and logarithmic, may be too slow for interactive applications, especially if security techniques such as redundant routing are used.

A further weakness of this system is that, while it defines a security policy based on digital signatures and a restricted role for untrusted nodes, it does not define algorithms to prevent malicious nodes from lying in the load balancing protocol.

The advantages of this proposal are its load balancing approach, and its extreme scalability while maintaining (weak) consistency. As mentioned, it can also effectively handle churn.

7. Related work on workload analysis

Many previous studies have characterized different types of web workloads. However, to the best of our knowledge our study is the first one to study the workload of a major collaborative web site such as Wikipedia. Almeida et al. [4] analyze the workload of four web sites and study temporal and spatial locality. They show that temporal locality can be characterized using the stack distance metric, while spatial locality can be modeled using the notion of self-similarity. Arlitt and Williamson [7] study the workload of six web sites and identify a number of invariants that apply to all the studied data sets such as lookup success rate, mean transfer size, size distribution, inter-reference times, and concentration of reference, among others. Bent et al. [8] study the properties of a large number of web sites hosted by a major ISP. They find that the workload contains a high degree of uncacheable requests, related to the widespread use of cookies; that most sites do not use HTTP 1.1 cache-control features, and that most sites would benefit from the use of a content delivery network. Arlitt et al. [6] analyze a five-day workload from a large e-commerce site. They characterize user requests and sessions and determine their impact on scalability. They find that horizontal scalability is not always an adequate mechanism for scaling the system, but that system-level and application-level QoS mechanisms are required in overload conditions. Note that these workloads differ from Wikipedia's in that almost all Wikipedia content is dynamic and updated by end-users.

Various analyses have been conducted to study Wikipedia from publicly available database dumps. Voß [34] studied four language editions of Wikipedia and measured numerous statistics such as size distribution of articles, number of distinct authors per article, number of articles per author, distribution of links among pages, and growth in several variables such as database size and number of articles. Ortega and Gonzalez Barahona [24] analyzed user contributions in the English Wikipedia and found that a small percentage of authors are responsible for most of the contributions. Wilkinson and Huberman [36] studied the relationship between quality and cooperation in Wikipedia articles and found that high-quality articles (denoted as "featured" in Wikipedia) are distinguished from the rest by a larger number of edits and distinct editors, following a pattern where edits begat edits. Hu et al. [17] propose three models for automatically deriving article quality in Wikipedia. The models are based on interaction data between articles and their contributors. While these studies are important to understand the update behavior of Wikipedia users, one cannot ignore the fact that most Wikipedia users simply read the encyclopedia without editing it. Our study differs from these in that we use a unique trace that contains a sample of the full Wikipedia traffic including read requests in addition to the information available in database dumps.

8. Conclusion

Our study of the Wikipedia workload has given us important insight for the design of a peer-to-peer architecture for hosting Wikipedia. The main issues such an architecture has to deal with are a large and heterogeneous data set with an unbalanced read-mostly load, page relationships that make updates to a page affect the rendering of other pages, churn, malicious nodes, and, to a lesser extent, concurrent conflicting updates.

Our analysis has shown that the focus of any architecture for handling the Wikipedia workload must be on optimizing how frontend nodes efficiently handle read requests for wiki pages and their embedded files, which represent more than 90% of all requests (95% if we ignore cache maintenance requests), with a theoretical cache hit ratio of more than 98% for wiki page content and virtually 100% for embedded media and static files. However, current decentralized proposals tend to focus on efficiently decentralizing the backend using distributed hash tables or other mechanisms, or handling concurrent updates, which represent no more than 0.03% of the load. In this sense, the existing Wikipedia architecture based on distributed web caching and invalidations is better suited

for the job than the current proposed decentralized schemes.

We conclude that future decentralized wiki engines will have to concentrate on automatically selecting replication, caching and distribution policies for each document with consistency protocols that take into account page relationships. They will also have to be prepared to deal with churn, unexpected load changes per document, and malicious nodes in order to be considered as serious alternatives for hosting large wikis like Wikipedia.

Acknowledgments

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

References

[1] FreePastry, <http://www.freepastry.org>.
[2] S. Adler, The Slashdot Effect: An Analysis of Three Internet Publications, <http://ssadler.phy.bnl.gov/adler/SDE/SlashDotEffect.html>.
[3] Alexa Internet, Alexa Web Search – Top 500, 2007, <http://www.alexa.com/site/ds/top_sites?ts_mode=global>.
[4] Virgílio Almeida, Azer Bestavros, Mark Crovella, 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.
[5] Sawsan Alshattnawi, Gérôme Canals, Pascal Molli, Concurrency awareness in a P2P wiki system, in: Proceedings of the International Symposium on Collaborative Technologies and Systems, May 2008.
[6] Martin F. Arlitt, Diwakar Krishnamurthy, Jerry Rolia, Characterizing the scalability of a large web-based shopping system, ACM Transactions on Internet Technology 1 (1) (2001) 44–69.
[7] Martin F. Arlitt, Carey L. Williamson, Web server workload characterization: the search for invariants, in: Proceedings of the 1996 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS), New York, NY, USA, ACM Press, 1996, pp. 126–137.
[8] Leeann Bent, Michael Rabinovich, Geoffrey M. Voelker, Zhen Xiao, Characterization of a large web site population with implications for content delivery, in: Proceedings of the 13th International Conference on World Wide Web (WWW), New York, NY, USA, ACM Press, 2004, pp. 522–533.
[9] Mark Bergsma, Wikimedia Architecture, 2007, <http://www.nedworks.org/~mark/presentations/san/Wikimedia%20architecture.pdf>.
[10] Charles Blake, Rodrigo Rodrigues, High availability, scalable storage, dynamic peer networks: pick two, in: Proceedings of the Ninth Conference on Hot Topics in Operating Systems (HOTOS), Berkeley, CA, USENIX, 2003, pp. 1–6.
[11] Gérôme Canals, Pascal Molli, Julien Maire, Stéphane Laurière, Esther Pacitti, Mounir Tlili, XWiki Concerto: A P2P wiki system supporting disconnected work, in: Proceedings of the Fifth International Conference on Cooperative Design, Visualization and Engineering, September 2008.
[12] Ludmila Cherkasova, Minaxi Gupta, Analysis of enterprise media server workloads: access patterns, locality, content evolution, and rates of change, IEEE/ACM Transactions on Networking 12 (5) (2004) 781–794.
[13] P.Th. Eugster, R. Guerraoui, S.B. Handurukande, P. Kouznetsov, A.-M. Kermarrec, Lightweight probabilistic broadcast, ACM Transactions on Computer Systems (TOCS) 21 (4) (2003) 341–374.
[14] Wikimedia Foundation, Wikimedia Dump Service, 2007, <http://download.wikimedia.org/>.
[15] Krishna P. Gummadi, Richard J. Dunn, Stefan Saroiu, Steven D. Gribble, Henry M. Levy, John Zahorjan, Measurement, modeling, and analysis of a peer-to-peer file-sharing workload, in: Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP), New York, NY, USA, ACM Press, 2003, pp. 314–329.
[16] Lei Guo, Songqing Chen, Zhen Xiao, Xiaodong Zhang, Analysis of multimedia workloads with implications for internet streaming, in: Proceedings of the 14th International Conference on World Wide Web (WWW), New York, NY, USA, ACM, 2005, pp. 519–528.
[17] Meiqun Hu, Ee-Peng Lim, Aixin Sun, Hady Wirawan Lauw, Ba-Quy Vuong, Measuring article quality in Wikipedia: models and evaluation, in: Proceedings of the 16th ACM Conference on Conference on Information and Knowledge Management (CIKM), New York, NY, USA, ACM, 2007, pp. 243–252.
[18] Frank T. Johnsen, Trude Hafsoe, Carsten Griwodz, Analysis of server workload and client interactions in a news-on-demand streaming system, in: Proceedings of the Eighth IEEE International Symposium on Multimedia (ISM), Washington, DC, USA, IEEE Computer Society, 2006, pp. 724–727.
[19] David R. Karger, Matthias Ruhl, Simple efficient load-balancing algorithms for peer-to-peer systems, Theory of Computing Systems 39 (6) (2006) 787–804.
[20] Bo Leuf, Ward Cunningham, The Wiki Way: Collaboration and Sharing on the Internet, Addison-Wesley Professional, 2001. April.
[21] Joseph C. Morris, DistriWiki: a distributed peer-to-peer wiki network, in: Proceedings of the 2007 International Symposium on Wikis (WikiSym), New York, NY, USA, ACM, 2007, pp. 69–74.
[22] Patrick Mukherjee, Christof Leng, Andy Schuerr, Piki – a peer-to-peer based wiki engine, in: Proceedings of the Eighth International Conference on Peer-to-Peer Computing (P2P), Los Alamitos, CA, USA, IEEE Computer Society, 2008, pp. 185–186.
[23] B. Clifford Neuman, Scale in distributed systems, in: T.L. Casavant, M. Singhal (Eds.), Readings in Distributed Computing Systems, Los Alamitos, CA, IEEE Computer Society, 1994, pp. 463–489.
[24] Felipe Ortega, Jesus M. Gonzalez Barahona, Quantitative analysis of the Wikipedia community of users, in: Proceedings of the 2007 International Symposium on Wikis (WikiSym), New York, NY, USA, ACM, 2007, pp. 75–86.
[25] Gérald Oster, Pascal Urso, Pascal Molli, Abdessamad Imine, Data consistency for P2P collaborative editing, in: Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work (CSCW), New York, NY, USA, ACM, 2006, pp. 259–268.
[26] Guillaume Pierre, Maarten van Steen, Andrew S. Tanenbaum, Dynamically selecting optimal distribution strategies for web documents, IEEE Transactions on Computers 51 (6) (2002) 637–651.
[27] Stefan Plantikow, Alexander Reinefeld, Florian Schintke, Transactions for distributed wikis on structured overlays, in: Proceedings of the 18th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, October 2007.
[28] Thorsten Schütt, Florian Schintke, Alexander Reinefeld. Chord#: structured overlay network for non-uniform load distribution, Technical Report ZR-05-40, Zuse Institute Berlin, August 2005.
[29] T. Schntt, M. Moser, S. Plantikow, F. Schintke, A. Reinefeld, A transactional scalable distributed data store: Wikipedia on a DHT, in: First IEEE International Scalable Computing Challenge (SCALE), Los Alamitos, CA, USA, IEE Computer Society, 2008.
[30] Hala Skaf-Molli, Charbel Rahhal, Molli Pascal. Peer-to-peer semantic wiki. Technical Report 6468, INRIA, November 2008.
[31] Kunwadee Sripanidkulchai, Bruce Maggs, Hui Zhang, An analysis of live streaming workloads on the internet, in: Proceedings of the Fourth ACM SIGCOMM Conference on Internet Measurement (IMC), New York, NY, USA, ACM Press, 2004, pp. 41–54.
[32] Guido Urdaneta, Guillaume Pierre, Maarten van Steen, A decentralized wiki engine for collaborative Wikipedia hosting, in: Proceedings of the Third International Conference on Web Information Systems and Technologies (WEBIST), March 2007, pp. 156–163.
[33] Guido Urdaneta, Guillaume Pierre, Maarten van Steen. Wikipedia workload analysis, in: Proceedings of the 14th Annual Conference of the Advanced School for Computing and Imaging (ASCI), Heijen, The Netherlands, June 2008.
[34] Jakob Voß, Measuring Wikipedia, in: Proceedings 10th International Conference of the International Society for Scientometrics and Informetrics, 2005.
[35] Stéphane Weiss, Pascal Urso, Pascal Molli. Wooki: a P2P wiki-based collaborative writing tool, Technical Report 6226, INRIA, June 2007.
[36] Dennis M. Wilkinson, Bernardo A. Huberman, Cooperation and quality in Wikipedia, in: Proceedings of the 2007 International Symposium on Wikis (WikiSym), New York, NY, USA, ACM, 2007, pp. 157–164.

Guido Urdaneta is a PhD student in the Computer Systems group at VU University Amsterdam. His research interests focus on large-scale distributed systems. Urdaneta holds an MSc degree in Applied Computing from the University of Zulia, Venezuela.

Guillaume Pierre is an assistant professor in the Computer Systems group at VU University Amsterdam. His research interests focus on large-scale distributed systems. Pierre has an MSc and a PhD in Computer Science from the University of Evry-val d'Essonne, France. He is the treasurer of EuroSys, the European Professional Society on Computer Systems, and an editorial board member for IEEE DSOnline.

Maarten van Steen is full professor at VU University Amsterdam. His research concentrates on large-scale distributed systems, with an emphasis on decentralized solutions, notably epidemic-inspired peer-to-peer systems. Application areas for such solutions include Web-based systems and large-scale wireless ad-hoc networks. He holds an MSc in Applied Mathematics from Twente University and a PhD in Computer Science from Leiden University, both from The Netherlands.