

Towards a Fully Decentralized and Collaborative Hosting Infrastructure for Wikipedia

Guido Urdaneta
VU University
De Boelelaan 1083a
1081HV Amsterdam
The Netherlands
g.urdaneta@few.vu.nl

Guillaume Pierre
VU University
De Boelelaan 1083a
1081HV Amsterdam
The Netherlands
gpierre@cs.vu.nl

Maarten van Steen
VU University
De Boelelaan 1083a
1081HV Amsterdam
The Netherlands
steen@cs.vu.nl

1. INTRODUCTION

Since its creation, Wikipedia has experienced an uninterrupted popularity growth that has made it one of the ten most visited web sites on the Internet[1]. This has forced its operator, the Wikimedia Foundation, to upgrade the hosting architecture from a single server to a distributed architecture of more than 250 servers at three locations on different continents. This architecture, however, is still subject to scalability issues since it relies on 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, and its popularity growth does not necessarily generate the extra income that can compensate for the corresponding increase in hosting costs.

We believe that an economical way to fix Wikipedia's scalability issues is to use a collaborative and decentralized hosting infrastructure. We expect that a significant number of Wikipedia supporters will be willing to share some of their personal computing resources to help host the system, similarly to the way they contribute by donating money or editing content.

2. PROPOSED ARCHITECTURE

Before motivating our design decisions, we first need insight on the actual functioning of Wikipedia. To do this, we have studied a very large sample of Wikipedia traffic provided to us by the Wikimedia Foundation [4]. The results of this study show in particular that page management constitutes 96% of Wikipedia's load in number of requests. Page management includes requests involving reads or updates of wiki pages as well as its embedded elements such as user-uploaded images, and javascript and CSS files.

These results make it clear that our architecture must focus

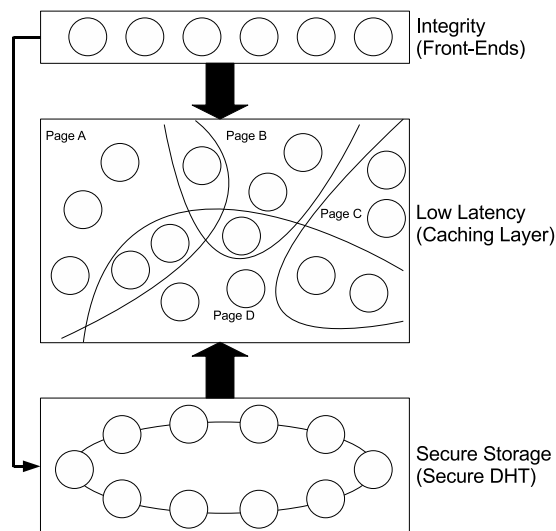


Figure 1: Proposed architecture for decentralizing page management. The tail of the arrow indicates who starts communication.

on providing a scalable decentralized solution to page management. To solve this problem, we need decentralized algorithms to satisfy functional Wikipedia requirements such as transitive inclusion, categories and broken link detection, but also non-functional requirements such as low latency to clients, load balancing, fault-tolerance and privacy. The most difficult challenge we face is solving these problems in an environment where mutually untrusted parties try to cooperate, and where a fraction of the participants may exhibit malicious behavior.

Figure 1 shows our proposed architecture for decentralizing Wikipedia's page management functionality. We use a distributed hash table (DHT) as a reliable mechanism to store pages. DHTs provide a scalable fully decentralized lookup service for $(key, value)$ pairs, and typically guarantee logarithmic lookup time and memory consumption with respect to the number of participating nodes in the system. Wikipedia pages are identified by a unique name provided in most requests, which matches the model provided by DHTs. DHTs are known to be vulnerable to malicious nodes, but many techniques have been proposed to address this problem, such as data replication, constrained routing tables,

redundant routing and certified node identifiers [3].

Apart from secure storage of data, it is also necessary to guarantee that clients always receive correct responses from the system. Since end-users use regular web browsers, we cannot assume any verification capabilities on the client side. We must make sure that clients do not access the system through malicious participants, since they could provide wrong responses or ignore updates sent by clients. Therefore, clients should read pages only through front-end nodes they trust (e.g., nodes operated by their ISP or by the Wikipedia operator). Moreover, update operations must be handled by nodes trusted by both the clients and the Wikipedia operator, since these operations modify the state of the system. Digital signatures can be used to allow participating nodes to verify that updates come from trusted sources.

While DHT security techniques can help guarantee data integrity and availability, they do so at the price of significantly increasing the latency of lookups, which is unacceptable in an interactive web application such as Wikipedia. Furthermore, the majority of page requests are read operations in a default HTML format that involves not only the requested page, but also other related pages such as transcluded or redirected pages, which increases latency even more. However, this does not mean that we should not use a secure DHT to store pages. The secure DHT guarantees that a number of replicas of each page can always be easily located even under churn or attacks by malicious nodes. What we need is an additional mechanism to improve the latency of most operations.

The most straightforward way to reduce client-perceived latency is to cache pages and related files, which is what Wikipedia's current architecture does. Our workload study shows that 96% of page requests are directed to pages that are highly cacheable.

To properly take advantage of the potentially high number of collaborators, our caching solution should use decentralized algorithms to replicate cached HTML-rendered wiki pages. Such a solution faces a number of challenges. First, we must decide when and where to create, replicate and destroy cached pages. Second, we must maintain consistency when the cached page or one of its related pages (e.g., a transcluded page) are updated. Third, we must give front-end nodes a way to find cached pages.

We propose to implement a caching algorithm based on the notion of invalidation groups. In this scheme, each page is cached by a group of nodes. Each node may decide to expand the group by replicating the page to other nodes, or to leave the group by deleting the page, according to a heuristic that tries to maximize global performance, but using local information such as page popularity and read/save ratio. Updates are propagated through the appropriate groups using gossiping algorithms based on anti-entropy [2]. A large number of groups can be supported by taking advantage of the overlapping in group membership.

The proposed invalidation scheme also serves to implement page relationships that affect page rendering, such as category membership, redirection, transitive inclusion, links and

backlinks. For example, if page A includes pages B and C, then all members of the group corresponding to page A will also be members of the groups for pages B and C. DHT nodes holding the master copies of a page also have a partial view of the corresponding caching group in order to be able to propagate updates. This allows front-end nodes to locate page caches by querying the DHT. Moreover, once a front-end node knows about a page cache, it can locate more copies directly from the group members. Thus, front-end nodes can maintain dynamic views of caching groups and perform load balancing by distributing requests over the nodes in these views. Front-end nodes can periodically verify the validity of page caches by requesting from the DHT the pages used to build the cache.

We plan to evaluate our algorithms with simulations using synthetic inputs based on the actual Wikipedia workload mentioned above. We also plan to develop analytical models that allow us to estimate the performance of our proposed architecture in terms of global metrics such as client-perceived performance and, load distribution. These analytical models are to be constructed taking into account both the system's characteristics and Wikipedia's workload.

Finally, once our design is tested using simulations and analytical models, we will develop a prototype implementation that will be tested using our sample of Wikipedia's workload under diverse conditions.

3. CONTRIBUTIONS

The main contribution of our research is a solution for the collaborative hosting of a high-traffic dynamic and collaborative web site like Wikipedia. This solution meets two objectives. First, it solves scalability issues present in the current Wikipedia design based on a centralized database. Second, it allows the Wikipedia community to support the project not only with money donations and edition of articles, but also with the donation of unused computing and networking resources, effectively extending Wikipedia's collaborative nature.

4. ACKNOWLEDGMENTS

Guido Urdaneta is supported by the Programme Alban, the European Union Programme of High Level Scholarships for Latin America, scholarship No.E05D052447VE

5. REFERENCES

- [1] Alexa Internet. Alexa web search - top 500, 2008. http://www.alexa.com/site/ds/top_sites?ts_mode=global.
- [2] K. Petersen, M. Spreitzer, D. Terry, M. Theimer, and A. Demers. Flexible Update Propagation for Weakly Consistent Replication. In *Proc. SOSP Conf.*, 1997.
- [3] G. Urdaneta, G. Pierre, and M. van Steen. A Survey of DHT Security Techniques. *Submitted for publication*, 2008.
- [4] G. Urdaneta, G. Pierre, and M. van Steen. Wikipedia Workload Analysis for Decentralized Hosting. Technical Report IR-CS-041, VU University, Amsterdam, The Netherlands, 2008. http://www.globule.org/publi/WWA_ircs041.html.