Analysis and design of an efficient NoSQL data storage schema for an interlinked social notification feed

Research thesis proposal

Florian Dejonckheere

Abstract

The advent of dynamic Web 2.0 web applications has led to a massive increase in the amount of social web activity. The need for performant storage systems to keep up with this vast deluge of data is directly tied to this. The generated data is characterized by its size and inherently structured nature: most types of social networks can be modeled using a simple graph topology.
In this paper, commonly used techniques in the industry related to the storage of non-critical notifications in small to medium-size web applications will be presented. An analysis of the structured and linked data provided by the Open Webslides (2017) project will be made, and an expandable abstraction will be designed in this context. This design will then be matched to a stable, scalable and maintainable storage system. A concrete implementation plan and roadmap for developing and rolling out the mentioned abstraction mechanism will also be presented.

Keywords
Webapplications — Social feed — Databases — Cloud — NoSQL — Social Graph — Big Data

Contact: florian@floriandejonckheere.be

Contents

1. Introduction

The Open Webslides (2017) project provides a user-friendly platform to collaborate on webslides - slides made with modern web technologies such as HTML, CSS and JavaScript. One of the core features this application provides is *co-creation*. The co-creation aspect manifests itself in several forms within the application; annotations on slides and a change suggesting system resembling GitHub's pull request feature are the main mechanisms. Because of the inherent social nature of co-creation, a basic notifications feed was also implemented. This feed is tailored to the user, and reflects the most recent changes, additions and comments relevant to the slide decks the user is interested in.

However, at the moment the functionality implemented in the system contains only the bare necessities. The module will be expanded in the future, and doing so requires a structural and conceptual rethinking of how the notifications are generated, stored and queried. The size of the dataset is also expected to grow linearly with user activity, therefore scalability is a requirement as well. This paper has two concrete goals.

First, it aims to analyze and summarize the existing frameworks and software packages currently used in the industry to store structured non-relational graph or document data. In order to achieve this, an introduction into NoSQL and the different types of NoSQL database management systems will be given first, followed by a comparative study between different vendors of NoSQL database management systems.

Secondly, the linked structure and social graph provided by the Open Webslides' notification feed will be examined, and a maintainable and expandable abstraction will be designed. This allows any future development to easily add notification types into the system. This data schema will then be evaluated in the context of the comparative study. Finally, an implementation plan will be composed for the concrete development of the described data storage schema.

2. Use case

This thesis is a study of NoSQL data storage techniques applied to the Open Webslides (2017) project. The online, interactive platform that the project provides includes a list of

notifications in reverse chronological order, tailored to the user. This feature is called the *social feed*. It enumerates the most recent social activity on the platform. For example, if a user updates a slide deck, the user's friends would be able to find a change notification in their respective social feeds.

From the perspective of the application code that generates the social feed, the user, slide deck and notification should be considered separate entities. The notification itself has relations to the other entities: the author (the *subject*), the slide deck (the *object*), along with the *predicate* property that provides information on what operation was performed (for example creating or updating a slide deck).

This data model is also characterized by the *write-once read-many* nature of the information; once the notification has been generated, it does not need to be modified again. It will also only be queried in a very specific way: the application server will always attempt to retrieve the most recent notifications starting from the user entity. This principle is an important aspect to take into consideration in the choice of data storage mechanism.

3. State-of-the-art

In current literature, studies such as Moniruzzaman and Hossain (2013, 4), Nayak, Poriya, and Poojary (2013) and Dayne Hammes and Mitchell (2014) have already analyzed the disparity between traditional relational database systems and NoSQL stores. However, the conceptual and technical difference between these data storage models will not be scrutinized any further, since this paper presents a data storage solution applied to the social notification feed of the Open Webslides (2017) project.

There are already many existing free and commercial products for the storage of NoSQL data, such as Redis (Sanfilippo, 2009), CouchDB (Apache Software Foundation, 2005a), MongoDB (Inc., 2009) and Neo4j (Technology, 2007). Finding the right database model for this use case (section 2) is one of the hurdles this paper intends to handle. Zhao (2015) describes the development of a messaging system for astrophysical transient event notifications. Part of this paper is a qualitative comparison between document-based NoSQL storage solutions fit for this particular use case. We expect this paper to provide a solid base of reasoning in order to find a scalable and efficient solution for resolving similar computational challenges.

The goal of this paper is to provide a performant, scalable and maintainable data storage schema for the Open Webslides (2017) platform, regarding the linked social graph that powers the *Social Feed* functionality present in the platform.

4. Methodology

First, a range of industry-standard NoSQL database management systems such as MongoDB (Inc., 2009), HBase (Apache Software Foundation, 2005b) and Neo4j (Technology, 2007) will be qualitatively analyzed. Three of the five types of NoSQL database types (Nayak et al., 2013) will be included in the study: column-oriented, document based and graph databases. Criteria for comparison include how the database management system concretely stores its data on disk, the query format and specific programming language bindings. Another important aspect is the distributed nature of many NoSQL databases. Using Brewer's conjecture (Gilbert & Lynch, 2002, 2) – often called the CAP theorem – the existing types of data storage systems will be examined and summarized. There is also a practical factor present in the research; this includes the license of the project, its active maintainability and future prospects. Common types of NoSQL databases include key-value store, column-oriented, document store and graph databases (Nayak et al., 2013). This paper will provide a short introduction to these types, before proceeding with the type that fits our use case the most.

Second, the data model specific to the Open Webslides project will be examined. We will start from the data model that is already implemented in the current iteration of the platform. At the time of writing, the existing base implementation of the social notification feed only contains two types of notifications. This paper will try to extrapolate this concept into a more generalized, abstract system in which developers can easily plug additional notification types. The physical properties of the data model will also be taken into account: the data will be written to the data storage only once, but read many times. It is also highly interlinked information, as a notification will always relate to one or more users as a subject, and a target object as well – most likely a slide deck or collection of slide decks. These links need to be maintained, and efficiently reconstructed when queried.

Finally, a sample dataset will be constructed using the aforementioned detailed analysis. Empirical testing will be conducted against multiple database management systems, and the results will be summarized and interpreted. Various information flows will be tested; however, the most important process remains efficiently querying the stored data.

Using the comparative study of storage engines, data model analysis and the empirical results an implementation plan will be constructed. This plan will serve as a recommendation for future development.

5. Expected results and conclusions

The NoSQL ecosystem, unlike relational databases, is headed towards specialization, so different solutions are headed in different directions (Maroo, 2013). In this paper, we expect to find one type of NoSQL database that is a better fit for the Open Webslides use case, in clear contrast with the other types of storage engines. Due to the inherently highly interlinked nature of the stored data, we suspect a graph-based database management system to provide most advantages, and generally the most performant experience.

This expectation is amplified by the availability and good community support of Ruby bindings to the most popular graph database management systems.

Since the platform being discussed only caters to a small to medium user base, we do not expect the need to scale horizontally beyond one instance. However, the vertical scalability is still a topic for discussion, and we expect to determine the computational order of magnitude in order to efficiently query the given dataset during this study.

Finally, the implementation plan should describe a concrete roadmap, stretching over a development period with a baseline expectation of one to three months. Roll-out of this mechanism should also be included in this plan.

References

Apache Software Foundation. (2005a). Couchdb. Retrieved from https://couchdb.apache.org/

Apache Software Foundation. (2005b). Neo4j. Retrieved from https://hbase.apache.org/

Dayne Hammes, H. M. & Mitchell, H. (2014). *Comparison of nosql and sql databases in the cloud*. Southern Association for Information Systems.

Gilbert, S. & Lynch, N. (2002). Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *ACM SIGACT News*, *33*, 51–59.

Inc., M. (2009). Mongodb. Retrieved from https://www.mongodb.com/

Kaur, K. & Rani, R. (2013). Modeling and querying data in nosql databases. In *2013 ieee international conference on big data* (pp. 1–7). doi:10.1109/BigData.2013.6691765

Maroo, T. (2013). *Handling with dynamic, large data sets - nosql a buzzword or savior?* JECRC Foundation.

Moniruzzaman, A. B. M. & Hossain, S. A. (2013). Nosql database: New era of databases for big data analytics - classification, characteristics and comparison. *International Journal of Database Theory and Application*, *6*.

Nayak, A., Poriya, A., & Poojary, D. (2013, March). Type of nosql databases and its comparison with relational databases. *International Journal of Applied Information Systems (IJAIS)*, *5*(4).

Open Webslides. (2017). Open Webslides. Retrieved from http://openwebslides.github.io/

Sanfilippo, S. (2009). Redis. Retrieved from https://redis.io/

Technology, N. (2007). Neo4j. Retrieved from https://neo4j.com

Zhao, Y. (2015). *Event based transient notification architecture and nosql solution for astronomical data management* (Doctoral dissertation, Massey University).