Cosmos: A Wiki Data Management System

Qinyi Wu Calton Pu Danesh Irani

College of Computing
Georgia Institute of Technology
Atlanta, GA
{qxw, calton, danesh}@cc.gatech.edu

ABSTRACT

Wiki applications are becoming increasingly important for knowledge sharing between large numbers of users. To prevent against vandalism and recover from damaging edits, wiki applications need to maintain revision histories of all documents. Due to the large amounts of data and traffic, a Wiki application needs to store the data economically on disk and processes them efficiently. Current wiki data management systems make a trade-off between storage requirement and access time for document update and retrieval. We introduce a new data management system, Cosmos, to balance this trade-off.

Keywords
Version control systems, Wikis

1. INTRODUCTION

Social media applications, such as wikis and blogs, are growing fast and attracting millions of users around the world to share and exchange knowledge. For example, Wikipedia [5] receives over forty thousand browsing requests a second and over thousands of new revision creation requests daily. It is critical that the underlying wiki data management system be efficient.

To detect vandalism behavior and recover from damaging edits, wiki systems have been maintaining the entire revision histories of their documents. Current approaches either use traditional version control systems (VCSs) or database management systems (DBMSs) for their data management. VCSs use disk space economically because they only store the differences between consecutive revisions of a document, but they have limited support for data indexing and replication as required by current wiki systems to sustain a large amount of browsing and update requests [7]. On the other hand, DMBSs provide advanced support for the missing features in VCSs, but they store revisions in their entirety disregarding overlapped content between consecutive revisions. Using a data dump from English Wikipedia [4], our experiments show that the disk space consumption between these two approaches can be more than thirty times.

This paper introduces a new wiki data management system, referred to as Cosmos, which balances the trade-off between storage requirement and access time. Cosmos uses a data structure, called *partial persistent sequence* [8], to represent a document and its revision history. This new data structure creates persistent and unique position identifiers to index characters of a document. Those position identifiers can be used to store revisions economically on disk and process requests for document update and retrieval efficiently.

Next we explain how Cosmos uses PPSs to manage document revision histories and then describe the system architecture. After that, we summarize our experiment results. For more detail, please refer to our technical report [8]

2. PPS AND ITS APPLICATION TO DOCUMENT REPRESENTATION

Conceptually, a partial persistent sequence (PPS) consists of a sequence of characters. Each character is uniquely indexed by a rational number. We call the rational position indexes *position stamps*. For example, the position stamps for "*abc*" are 0, 0.5, and 1 respectively. A PPS never deletes any characters. It has only one operation $INSERT$. For a newly inserted character, the PPS first locates the position stamps that are neighboring to the insertion point. Then it assigns a rational number that falls with the range of these two neighbors. In Figure 1, 0.3 and 0.4 are the neighbors. '*e*' is now indexed by 0.35. Algorithms that are used to compute new position stamps are called *encoding schemes*. In Figure 1, the newly inserted character halves the interval between its two neighbors.

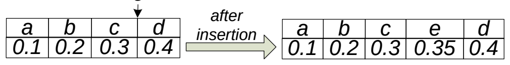

Figure 1: A PPS insertion example

In wikis, a document consists of a sequence of characters. It can be updated by insert and delete operations. A new revision is created when a user commits his modification to the wiki data storage system. A PPS represents a document in two parts: *mapping* and *revision*. The mapping part records the correspondence between position stamps and their corresponding characters. The revision part contains the position stamp information for each revision. Even though the content of a document can be updated by insert and delete operations, the PPS never deletes anything. To correctly construct the content of a revision, the revision maintains an array of position stamps for those characters it contains. Figure 2 illustrates the idea. A document is represented by a PPS, which contains five position stamps. There are two revisions defined on it: Rid_i and Rid_j. Rid_i corresponds to the character sequence "$c_0c_1c_2c_4$", and Rid_j "$c_2c_3c_5$". To obtain a revision, we first obtain the array, and then sequentially concatenate the characters they point to.

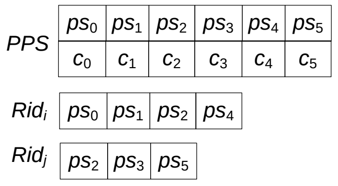
PPS
ps_0 ps_1 ps_2 ps_3 ps_4 ps_5
c_0 c_1 c_2 c_3 c_4 c_5
Rid_i
ps_0 ps_1 ps_2 ps_4
Rid_j
ps_2 ps_3 ps_5

Figure 2: A document's PPS representation and two of its revisions

3. COSMOS SYSTEM ARCHITECTURE

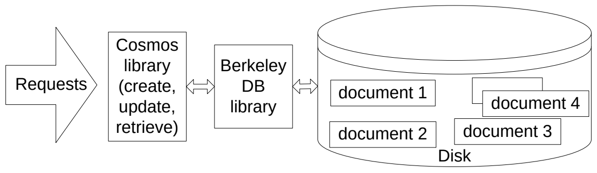

Figure 3: Architecture of Cosmos

Figure 3 shows the system architecture of Cosmos. It provides a library for wiki data management. Cosmos uses Berkeley DB [6] as its backend storage system. Berkeley DB is a general-purpose database engine that supports efficient data management on key/data pairs. In our implementation, all data are constructed in the form of key/data pairs.

A document is represented by two tables: *mapping* table and *revision* table. For a browsing request, Cosmos first looks up the revision table to obtain the position stamps of a revision, de-references those position stamps to their corresponding characters by the mapping table, and finally concatenates them as the returning result. For an update request to the latest revision of a document, Cosmos uses a diff utility [1] to first locate modification places. It then assigns new position stamps for insertion or remove position stamps from the latest revision. Finally Cosmos creates a new revision and updates both tables. When storing a PPS on disk, Cosmos does not explicitly store all position stamps and their mappings. Instead, it stores consecutively inserted characters in a compact record. This compact record will be further divided into sub-records if the characters are modified. Since the compact record in general takes much less disk space than storing the characters directly, Cosmos is able to manage document revisions economically on disk.

4. EXPERIMENT RESULTS

We compare the performance of Cosmos with two other wiki data management systems. One is TWiki [3], which uses a VCS-based approach. The other is MediaWiki [2], which uses a DBMS-based approach.

Hardware configuration All experiments are conducted on a 64-bit GNU/Linux machine with Intel Core 2.83GHz CPU, 4GB RAM, and 1-Terabyte SATA hard disks.

Software configuration We used MediaWiki 1.13.5 with MySQL version 5.1.30 and TWiki with RCS 5.7.

Data set Wikipedia provides full-text access to all documents and their revision histories. We use a dump of the English Wikipedia (enwiki-20080103-pages-meta-history.xml.7z [4]) which is around 850GB in size. We do a random sampling to obtain 10 percent of documents and store their revisions in different systems. In the sampled data set, there are 20,039 documents and 3,053,829 revisions.

Experiment result summary Our experiments show Cosmos consumes one-fifth of the disk-space and achieves an order of magnitude speed-up in document retrieval when compared to MediaWiki (stored on the MySQL relational database). When compared to Twiki, although Cosmos consumes five times as much disk-space, it decreases the sequential revision access time by a factor of five. We note that the document update time for Cosmos is higher than that of MediaWiki and TWiki, but at about 100ms it is well within the human reaction time.

5. CONCLUSION

We introduce a new wiki data management system, Cosmos, to achieve a balance between low disk-space consumption and efficient document retrieval and update. We present its design and implementation, based on partial persistent sequences, as well as demonstrate its performance using a representative sample of Wikipedia data. As a next step, we plan to import the whole data dump of Wikipedia and study its scalability in large settings.

6. REFERENCES

[1] GNU wdiff. `http://www.gnu.org/software/wdiff/`.
[2] MediaWiki. `http://www.mediawiki.org/wiki/MediaWiki`.
[3] TWiki. `http://twiki.org/`.
[4] Wikipedia data dump download. `http://download.wikimedia.org/enwiki/`.
[5] Wikipedia the free encyclopedia. `http://en.wikipedia.org/wiki/Main_Page`.
[6] Michael A. Olson, Keith Bostic, and Margo I. Seltzer. Berkeley db. In *USENIX Annual Technical Conference, FREENIX Track*, pages 183–191, 1999.
[7] GÍęrald Oster, Pascal Molli, Sergiu Dumitriu, and RubÍęn MondÍęjar. UniWiki: A Reliable and Scalable Peer-to-Peer System for Distributing Wiki Applications. Research Report RR-6848, LORIA – INRIA Nancy Grand Est, Feb 2009.
[8] Qinyi Wu, Calton Pu, and Danesh Irani. Cosmos: A wiki data management system. Research Report GIT-CERCS-09-07, Georgia Institute of Technology, 2009.