A New Plug-in System Supporting Very Large Digital Library

Jin Wang, Yong Zhang, Yang Gao, and Chunxiao Xing

Research Institute of Information Technology
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
{wangjin12,yang-gao10}@mails.tsinghua.edu.cn,
{zhangyong05,xingcx}@tsinghua.edu.cn

Abstract. In the era of Big Data, actual demands of collecting large volumes of complex digital information have brought new challenges to digital library software. This scenario calls for the construction of Very Large Digital Library (VLDL). New approaches and technologies are needed to deal with the various issues in designing and developing VLDL. In this paper, we design a plug-in system—PuntStore as a general solution to very large digital library. PuntStore supports different kinds of storage engines and index engines to deal with the problem of storing and retrieving data efficiently. We also design a new index engine pLSM in PuntStore to meet the specific needs in digital libraries. The successful adoption of PuntStore in the project of the Digital Library on History of Science and Technology in China (DLHSTC) shows that PuntStore can function effectively in supporting VLDL systems.

Keywords: VLDL, plug-in system, PuntStore, storage engine, index.

1 Introduction

In the era of Big Data, the actual demands of data collection and repository have brought new challenges to digital library software. The past few years have witnessed the soaring volume as well as complex kinds of digital information to be collected and stored. Thus a variety of measures should be taken to deal with various data forms, large volumes, metadata heterogeneous, strictly demands of the data exchange and sharing, widely distributed content and long-term preservation. This calls for the construction of Very Large Digital Library (VLDL), which has raised many new issues.

Nevertheless, the definition of VLDL still remains to be an open problem. It cannot be considered as very large databases storing digital contents. As described in [2], digital library systems should deal with architecture management, functionality, user management besides content management. Similar to Big Data, the matter of "very largeness" includes the features of volume, velocity and variety. Digital Libraries becomes "very large" when any of these aspects reaches a magnitude that requires specialized methods [4]. Thus a variety of new approaches and technologies are needed in VLDL's data management, including data integration, data storage and access, and data ranking. Some existing DL systems have made some adaption to

S.R. Urs, J.-C. Na, and G. Buchanan (Eds.): ICADL 2013, LNCS 8279, pp. 45–52, 2013.

VLDLs. For example, J. Thompson has made an attempt to improve the performance of Greenstone in National Library of New Zealand's PapersPast digital library by applying parallel processing [7]. The Europeana project has also taken some measures in metadata management in the background of VLDL, such as storing the metadata of contents, using the Semantic Element as its data model [9].

In order to satisfy the specific needs of VLDL, it is crucial to have an efficient way of storing and retrieving contents with different formats or even different structures. We have designed a plug-in system PuntStore, as a general solution. PuntStore can fulfill the task of supporting heterogeneous application by dividing this task into specific steps and providing a variety of plugs for different steps. As a plug-in system, PuntStore provides a universal interface through which storage and index mechanisms of other DBMS could be integrated into the PuntDB, the database system offered by PuntStore. PuntStore also makes optimization in storage, distribution, scalability, heterogeneity and security. To reach the goal of efficiently retrieving heterogeneous data, we design a new index engine pLSM for PuntStore.

The paper is organized as follows: Section 2 describes an overview of PuntStore. Section 3 presents its adoption in the practical project of Digital Library on History of Science and Technology in China (DLHSTC). Section 4 provides the detail of storage engine and index engine in PuntStore and the design of pLSM index engine. Section 5 is concerned with two experiments: one is about multiple storage engines on different datasets, another is pLSM index, showing how it outperforms state-of-art index engines. Finally, we conclude in Section 6.

2 System Overview

The main problem that PuntStore is faced with is to deal with massive, complex, heterogeneous and continuous changing digital information. Yet there also exists a large amount of structured data which should be managed by relational DBMS. PuntStore also has to satisfy various kinds of needs for information retrieving. Based on the above consideration, we design the three-layer architecture for PuntStore.

2.1 The Architecture of PuntStore

As is shown in Figure 1, PuntStore consists of three layers:

- Storage Layer: Storage Layer is the core of PuntStore. The main function of Storage Layer is to store and manage data, create and maintain indexes. The Storage Layer consists of many PuntDBs, which consists of many PuntTables. There is a record library and an index library in each PuntTable. PuntDB and PuntTable will be described in details in 2.2. To manage metadata in PuntStore, we design a structure named Punt Digital Object (PDO), which will be described in 2.3.
- Management Layer: Management Layer is responsible for parsing requests submitted by Service Layer. Then it would finish the practical manipulations on PuntTable and PuntDB according to the interfaces offered by Storage Layer. In order to transmit messages between different layers, we design Message Object (MO), which will also be described in 2.3.

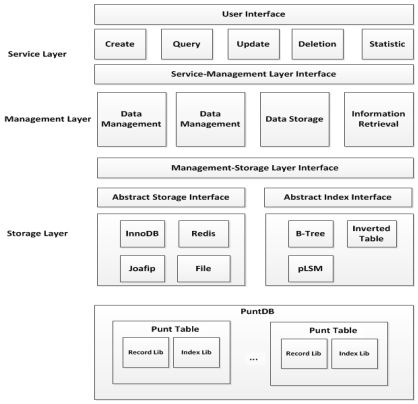


Fig. 1. The Overall Architecture of PuntStore

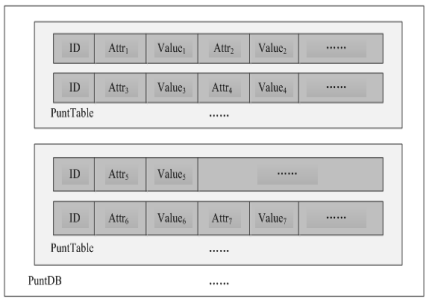


Fig. 2. The Data Model of PuntDB

- Service Layer: Service Layer offers different services to users according to data and index management interface provided by Management Layer. These services include creating database, table and index, manipulating insertion, query, updating, deletion and analysis on data.

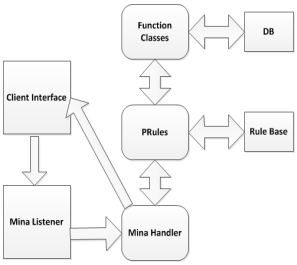
2.2 PuntDB

PuntDB is an optimized NoSQL database to support the storage and analysis of massive data. Just like the SAP HANA database in the SAP HANA Application [3], high-level applications in PuntStore are based on the storage functions provided by PuntDB. The data model of PuntDB is shown in Figure 2.

Every PuntDB can include many PuntTables, which are the actual "repositories" in the PuntStore system. A PuntTable encapsulates multiple storage systems and provides a uniform data management interface for the applications. Applications can choose its storage strategy or make combination strategies without modifying their data operations. Each PuntTable in PuntDB has multiple records. Every record has an ID as the unique identifier, just like the primary key in relational database. The rest part of a line consists of a key and the corresponding value, just like the property and its value in relational database. What is different from relational database is that PuntStore does not have a global schema, the property and number of elements in each line can be different from those of other lines. Moreover, because of trading off ACID properties, PuntDB could be easily scaled up. Thus the scalability of PuntStore could be ensured.

2.3 Introduction of MO and PDO

As is shown in Figure 3, MO is the object to transfer information between different layers. It could also be regarded as a service model. MO could be used to provide different services, such as file, log, user, tag and metadata service. We could encapsulate the services inside it and provide a single interface to the outside world. With the features of low coupling and naturally distributing, the design goal of distributed storage could be reached by implementing MO in the Storage Layer.



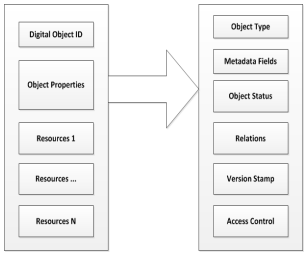


Fig. 3. The Architecture of Message Object

Fig. 4. The Architecture of Punt Digital Object

PDO is a digital object model to deal with metadata management in Storage Layer. We could see in Figure 4 that PDO has three components: Digital Object ID, Object Properties and a list of resources. One PDO could contain different kinds of data resources. Each resource has an identifier and its own contents. With the design of PDO, we could provide a uniform model for heterogeneous data. In this way, we could offer a concrete solution to metadata management in PuntStore to meet the needs of VLDL system.

3 Application

PuntStore has been deployed in the practical project of Digital Library on History of Science and Technology in China (DLHSTC) . Figure 5 shows its overall architecture. Like Bonolo [6], DLHSTC is a project to facilitate ubiquitous access to knowledge. The DLHSTC project aims at building an open digital library to the public and researchers in specific fields. Its main task is to collect, categorize and rescue the relics of science and technology history in China. By the end of 2012, the DLHSTC had included the parts of engineering, mathematics, mechanics, and water conservancy history of China. The DLHSTC system has 10 sub-systems, 85 programs, and 10 sub-databases, and its scale of data has increased to 900GB, including about 250,000 full text data. It is an information-rich, open resource library. A large scale of comprehensive resources are integrated in DLHSTC, including full text of historical documents, microscopic images, synopsis, catalogues, manuscripts, pictures, audios, videos, and animations. Nowadays, DLHSTC has become the largest comprehensive scientific data center in China.

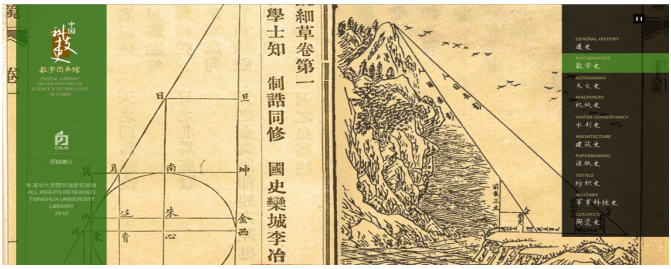


Fig. 5. The Project of DLHSTC

4 Storage and Index Engines

4.1 Storage Engine Overview

The structure of a record library is shown in Figure 6. Since PuntStore is faced with data with different structures, various kinds of storage mechanisms are needed in the record library to satisfy the needs for different applications. So PuntStore has a plug-in storage engine. Storage mechanisms in other DBMS could be easily implemented into PuntStore. Moreover, PuntStore may also support user defined storage mechanisms. The four main kinds of storage mechanisms in PuntStore are InnoDB, Redis, Joafip and File. To improve the efficiency of reading and writing files, optimizations have been made to file storage mechanism in PuntDB.

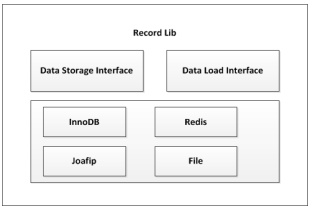


Fig. 6. The Record Library of PuntStore

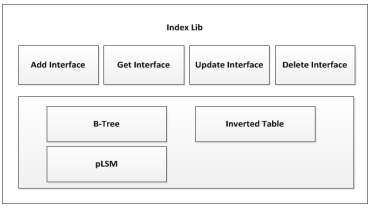


Fig. 7. The Index Library of PuntStore

4.2 Index Engine Overview

Index is a data structure that improves the speed of data retrieval operations on a database table at the cost of slower writes and the use of more storage space [10]. A highly efficient index could improve the speed of data retrieval operations on a database table. To accelerate information retrieval in PuntDB, PuntStore offers efficient index library and provide a plug-in index engine. Users could create or drop some kinds of indexes according to specific requirements. As is shown in Figure 7, PuntDB provides three kinds of indices: B-Tree, Inverted Table and pLSM—a new index engine we developed. With the collaboration of Inverted Table and B-Tree, PuntDB could also support union query between different fields and keywords.

However, after frequent insertion and updating operations, the logically continuous leaf nodes of B-Tree would not be physically adjacent, thus a large number of random I/O would occur when doing query operations. This problem is known as “aging” problem of B-Tree. To eliminate the aging problem and improve the write throughput, we present pLSM-Tree index as an index engine in PuntStore to replace B-Tree.

4.3 Implementation of pLSM Index

The pLSM index structure is based on the traditional LSM-Tree index framework [5]. There are two components in pLSM index. The in-memory component is a light weight data structure—Skip List, while the external-memory component consists of multiple levels of fractional sorted runs.

The insertion happens only in the in-memory component. When this component is full, it will be merged out to disk. When a sorted run on disk reaches its limited size, it will also be merged into the next smaller run. Under the LSM-Tree framework, pLSM can transform random I/Os into sequential ones by perform batch writing. By implementing the logarithm method [1] on the size of sorted runs on disk, the merging operation would be I/O efficient. In order to improve the read performance of pLSM, we add an auxiliary data structure Bloom Filter to the external component. Bloom Filter is a random data structure with high space efficiency. Another advantage of pLSM is that it supports efficient bulk deletion. This feature can help to efficiently create and drop indices. The specific implementation mechanisms are omitted here due to limited length of the paper. More details can be found in our previous work [8].

4.4 Summary of pLSM

To sum up, since pLSM performs much better in insertion and eliminates the aging problem of B-Tree, it can satisfy the speed requirements for collecting, processing and using entities in VLDLs. When more records are needed to be indexed, we only need to append a new array after the last sorted runs instead of changing the whole index structure. So pLSM has good scalability when the number of records increases sharply. The above features make pLSM a proper choice as the index structure in VLDL systems.

5 Experiment

In order to evaluate the efficiency of our storage engine and pLSM index engine, we perform several experiments. All the following experiments are set up on a server machine with 32GB RAM and a 2.40GHz Intel(R) Xeon E5620 CPU with 2 cores. The environment for our workbench is Windows server 2008. Storage of different structured data

To test the efficiency of our plug-in storage engine, we import three different datasets into PuntStore using different storage mechanisms. Due to the requirement of our system, the insertion cost is usually the bottleneck of our system. So we do the insertion experiment here. The datasets are real workloads from the project of DLHSTC:

A. Dataset Text consists of full text contents. The size is 534MB.
B. Dataset Small Records is part of metadata from the DLHSTC project, which are structured data. The size is 4420MB.
C. Dataset Image consists of image records to be displayed in the digital library. The size is 1530MB.

From Figure 8, we can see that using file as storage has the best time efficiency. This is because file has a much simpler storage mechanism than DBMS. However, file is not able to guarantee usability in some cases since it cannot support data schema and transactions. Redis performs better than MySQL InnoDB for dataset Image but much worse for dataset Small Records, which is structured data. With multiple storage mechanisms implemented, we can make flexible choices according to the characteristics of data, thus enhancing the flexibility of PuntStore.

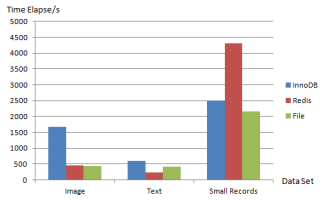


Fig. 8. The Result of Insertion Using Different Storage Engines

5.1 Performance of pLSM

To verify the efficiency of pLSM, we make two experiments comparing pLSM with B-Tree index. The size of each index entry is 64B. The first experiment is insertion test of the two indexes. The result in Figure 9 shows that pLSM has a much better insertion performance than B-Tree.

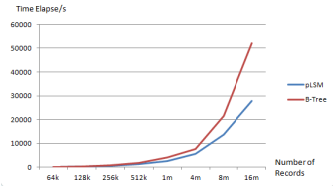


Fig. 9. Result of Insertion Experiment

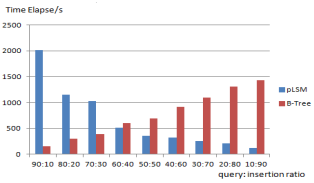


Fig. 10. Result of Mixed Workload Experiment

The second experiment is a mixed workload with the variation of ratio between query and insertion. We added 1 million records to each index in bulk to do the initialization. Figure 10 indicates that with the increasing of insertion ratio, the time elapse of pLSM decreases sharply, and the time elapse of B-Tree increases along with the insertion ratio. From the experiment, we find that when index size is larger than 1 GB, B-Tree's aging problem becomes so serious that it will cost more than one day to continue inserting into B-Tree index. So our experiment stops here. Although B-Tree's query performance is better, it is not practical to use B-tree since insertion time dominates in this situation. Instead, pLSM would be more efficient.

6 Conclusion

Very Large Digital Library has brought new challenges in almost every issue of Digital Library. In term of the system aspect, we provide PuntStore, a plug-in system as a concrete solution to support VLDLs and adopt it into the project of Digital Library on History of Science and Technology in China. In term of the specific aspect, we design a new index method pLSM. The results show that pLSM could overcome the aging problem of B-Tree and perform well in the occasion of VLDL. As the practical result depicted above, our designs are proper to be implemented in VLDL system. For future work, we would like to perform large scale distribution of PuntTable in more

practical projects. There are also many other research topics in VLDL such as architecture model, security, quality of service and data ranking that can be studied.

Acknowledgement. Our work is supported by National Basic Research Program of China (973 Program) No.2011CB302302, Key S&T Projects of Press and Publication under Grant No GXTC-CZ-1015004/02, and Tsinghua University Initiative Scientific Research Program.

References

1. Bentley, J.L.: Decomposable searching problems. Inf. Process. Lett. 8(5) (1979)
2. Candela, L., Athanasopoulos, G., Castelli, D., Raheb, K.E., Innocenti, P., Ioannidis, Y., Katifori, A., Nika, A., Vullo, G., Ross, S.: The Digital Library Reference Model. Deliverable D3.2b, DL.org (April 2011)
3. Farber, F., Cha, S.K., Primsch, J., Bornhovd, C., Sigg, S., Lehner, W.: SAP HANA Database - Data Management for Modern Business Applications. SIGMOD Record (2011)
4. Manghi, P., Pagano, P., Ioannidis, Y.: Second Workshop on Very Large Digital Libraries:in conjunction with the European Conference on Digital Libraries. SIGMOD Rec. 38, 46–48 (2010)
5. O'Neil, P., Cheng, E., Gawlick, D., O'Neil, E.: The log-structured merge-tree (LSM-tree). Acta Informatica 33(4), 351–385 (1996)
6. Phiri, L., Williams, K., Robinson, M., Hammar, S., Suleman, H.: Bonolo: A General Digital Library System for File-Based Collections. In: Chen, H.-H., Chowdhury, G. (eds.) ICADL 2012. LNCS, vol. 7634, pp. 49–58. Springer, Heidelberg (2012)
7. Thompson, J., Bainbridge, D., Suleman, H.: Towards Very Large Scale Digital Library-Building in Greenstone Using Parallel Processing. In: ICADL 2011, pp. 332–341 (2011)
8. Wang, J., Zhang, Y., Gao, Y., Xing, C.: pLSM: A Highly Efficient LSM-Tree Index Supporting Real-Time Big Data Analysis. In: COMPSAC 2013, pp. 240–245 (2013)
9. Union E. Europeana (EB/OL) (2012), `http://www.europeana.eu/portal/`
10. Wikipedia. Database index (EB/OL) (2013), `http://en.wikipedia.org/wiki/Database_index`