論文 / 著書情報
Article / Book Information

論題(和文)	An evaluation of Fat-Btree based and HDFS based Storage Systems for Small File I/O Applications
Title(English)	An evaluation of Fat-Btree based and HDFS based Storage Systems for Small File I/O Applications
著者(和文)	羅敏, 横田治夫
Authors(English)	Min Luo, Haruo Yokota
出典(和文)	, , ,
Citation(English)	, , ,
発行日 / Pub. date	2011, 3
権利情報 / Copyright	ここに掲載した著作物の利用に関する注意: 本著作物の著作権は（社）情報処理学会に帰属します。本著作物は著作権者である情報処理学会の許可のもとに掲載するものです。ご利用に当たっては「著作権法」ならびに「情報処理学会倫理綱領」に従うことをお願いいたします。 The copyright of this material is retained by the Information Processing Society of Japan (IPSJ). This material is published on this web site with the agreement of the author (s) and the IPSJ. Please be complied with Copyright Law of Japan and the Code of Ethics of the IPSJ if any users wish to reproduce, make derivative work, distribute or make available to the public any part or whole thereof.

An evaluation of Fat-Btree based and HDFS based Storage Systems for Small File I/O Applications

Min LUO[†] Haruo YOKOTA[†,‡]

[†]Department of Computer Science, Graduate School of Information Science and Engineering

Tokyo Institute of Technology

Email: [†] luomin@de.cs.titech.ac.jp, [‡] yokota@cs.titech.ac.jp

1. Introduction

Parallel database systems [1] exploit multiprocessor computer architectures in order to build high-performance database servers at a much lower price than equivalent mainframe computers. The hardware limitation in the mainframe database is overcome in parallel databases by scaling systems with more processors and storage disks that work in parallel. Therefore, scalability is one of the core features in parallel database systems.

Among the three most prominent parallel database architectures, shared-nothing architecture provides the best scalability out of the shared-memory and shared-disk architectures [1]. Value-range partition strategy in shared-nothing may efficiently support both point and range queries, which are inefficiently supported by the other two partition strategies, hash and round-robin. However, the lack of efficient parallel B-tree index for the range-partitioning data greatly limits the scalability of range-partitioned shared-nothing database systems.

In this paper, we address the data access efficiency and scalability by introducing a first trial of combining parallel B-tree structures with open-source database management systems. Different from previous parallel B-tree index, the Fat-Btree index we adopt has high cash hit rates and low update synchronization cost; therefore, the proposed parallel database has a good scalability and data accessing efficiency. To evaluate the scalability in our proposed system, we compare it with HBase, a distribute DBMS built on top of Hadoop, which is famous for the efficient and scalable data accessing performance. Experimental results on a 100-nodes cluster system verify that our parallel database greatly outperform HBase in both system scalability and throughput.

2. A Parallel DBMS based on Fat-Btree

2.1 Fat-Btree Index

B-tree based parallel indexing with value-range based data partitioning schema, is proposed for high throughput and efficient range query. However, it suffers high index structure synchronization costs. To reduce these costs, an update-conscious parallel B-tree structure, a Fat-Btree, has been proposed [2]. An example of a four-PE Fat-Btree is given in Fig. 1, where multiple copies of index nodes close to the root node with relatively low update frequency are replicated on several PEs, while leaf nodes with relatively high update frequency are distributed across the PEs. Thus, the maintenance cost of the Fat-Btree is much lower than that of other parallel Btree structures. In addition, Fat-Btree has a higher cache hit rate [2] and more efficient concurrency control protocols than other methods [3].

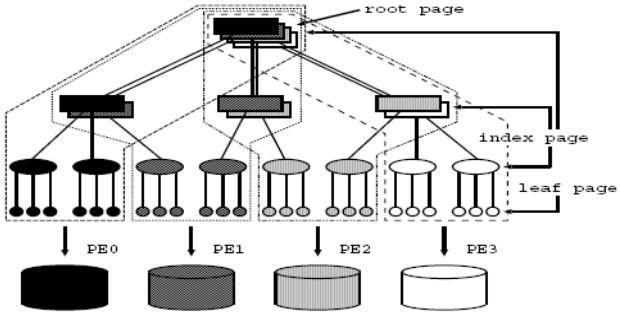

Fig.1 Fat-Btree

2.2 System Implementation

Because PostgreSQL [4] is a most famous and widely adopted open source DBMS in academic society, we choose it as the database layer on each PE in our implementation. In this system, data are stored as table tuples indexed by the local sub-Fat-Btree indexes on each independent PostgreSQL instances. As described in Sec. 2.1, because the replicated intermediate index-nodes have pointers to their child index-nodes in the neighbor PE, the intermediate paths are formed from the root index-node to every leafnodes located any PEs. A tuple retrieval request may transfer between PEs by following these intermediate paths.

In our implementation, we build the Fat-Btree index as an independent process outside PostgreSQL. Fig. 2 shows the query process by using the Fat-Btree index in our system. Details of the additional components in our system are described below.

1. Backends: “Postmaster” creates “backend(BEs)” to serve client requests. The remote BE retrieve data that may be stored at remote PE by using tuples’ PID and TID returned by “FBT Mgr.|”.

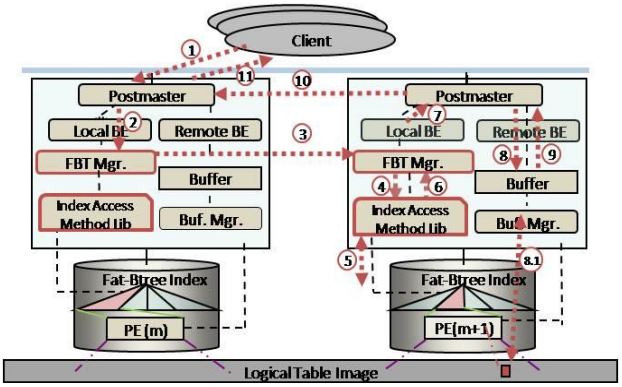

Fig.2 System Architecture of our Parallel Database

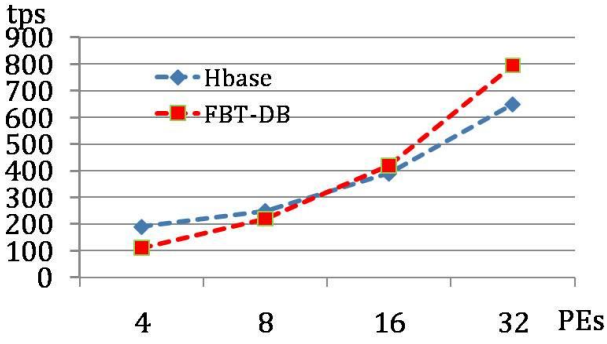

Fig.3 Scalability in HBase and Fat-Btree DBMS

2. Fat-Btree Manager: "FBT Mgr." traverses local Fat-Btrees by using the functions in "Index Access Method Library". It also provides remote connection interface between Fat-Btree on different PEs

3. Index Access Method Library: This library provides functions to traverse or update Fat-Btree structure. Tuples' ID (TID) and their host PEs' ID (PID) that stored with their partitioning attribute in Fat-Btree are used to fetch tuples at different PEs.

4. Buffer Manager: "Buf. Mgr." buffers recently accessed data by Fat-Btree retrieval, and search target data in the local buffer before traversing Fat-Btree. It also maintains buffered data consistency.

3. Experimental Study

To evaluate our system, we compare it with HBase [5], a key-value store on top of Hadoop Distributed File System (HDFS), under developed by an open source project, named Hadoop [6]. It is famous for the more efficient and scalable data accessing than that of Hadoop's HDFS based Map-Reduce system. Although HBase is designed for superior unstructured data retrieval in key-value pairs, it is still meaningful to compare its scalability with our row-based relational parallel DBMS, because both systems should efficiently support point queries.

In this experiment, we deploy HBase-0.20.2 and our Fat-Btree based parallel database systems on our cluster system in Table 1. We adopt the default settings of HBase and HDFS [5, 6], and use a dataset that contains 10K tuples, each row with 4 KB of data in a two columns schema, to evaluate the throughput of both systems for random data access.

We focus on the small-file application, because both systems are not designed for storing huge-files. And row sizes in previous HBase evaluation work, both in simulation and real application datasets [7, 8], are in a scale of several kilo-bytes of data

Table 1 Experimental Environment

Blade server:	Sun Fire B200x Blade Server
CPU:	AMD Athlon XP-M 1800+ (1.53 GHz)
Memory:	PC2100 DDR SDRAM 1 GB
Network:	1000BASE-T
Ethernet Switch:	Catalyst 6505 (720GB/s backbone)
Hard Drives:	TOSHIBA (30 GB, 5400 rpm, 2.5 inch)
OS:	Linux 2.4.20
Java VM:	Sun J2SE SDK 1.6.0 18 Server VM

Fig. 3 shows that the key-value store HBase system has better performance for a small number of nodes; however, our Fat-Btree based parallel database outperforms it when the number of PEs increases. This result illustrates that our proposed Fat-Btree index based parallel database owns even better scalability than the hash-base key-value store HBase, which is famous for its scalability in cloud systems.

4. Conclusion

In this paper, we presented the implementation of a parallel database based on Fat-Btree index. We evaluate its efficiency for small file I/O applications by comparing with a famous scalable key-value store HBase. Experimental results shows our Fat-Btree based database provide high scalability for the range-partitioned data and outperforms the key-value store HBase due to this scalability.

5. Acknowledgements

Part of this research was sponsored by MEXT via Grants-in-Aid #19024028 and #22240005.

References

[1]. David DeWitt and Jim Gray. Parallel database systems: the future of high performance database systems. Commun. ACM 35, 6 (June 1992), 85-98

[2]. H. Yokota, Y. Kanemasa, and J. Miyazaki, "Fat-Btree: An update conscious parallel directory structure," in ICDE '99. pp. 448-457, Mar 1999.

[3]. T. Yoshihara, D. Kobayashi, and H. Yokota, "Mark-opt: A concurrency control protocol for parallel B-tree structures to reduce the cost of SMOs," IEICE Trans. Inf. Syst., vol.90, no.8, pp.1213-1224, 2007.

[4]. PostgreSQL. http://www.postgresql.org/, 2010.

[5]. HBase, http://hbase.apache.org/, 2010

[6]. Hadoop, http://hadoop.apache.org/, 2010.

[7]. Carstoiu, D.; Cernian, A.; Olteanu, A.; , "Hadoop Hbase-0.20.2 performance evaluation," New Trends in Information Science and Service Science (NISS), 2010 , 4th International Conference on, pp.84-87, 2010

[8]. http://research.yahoo.com/files/HBaseAtRapleaf.pdf