SmallTable

Siddharth Bhatia
BITS, Pilani
siddharthbhatia2003@gmail.com

ABSTRACT

Google BigTable is a distributed storage system used across multiple Google services. It is flexible and a high performance solution for Google scale products. But no open source framework is available for a normal user. We design and implement a minimal version of the Google BigTable called SmallTable.

KEYWORDS

BigTable, Distributed computing

1 INTRODUCTION

BigTable[2] is a compressed, high performance, and proprietary data storage system built on Google File System[4], Chubby Lock Service[1], SSTable (log- structured storage like LevelDB) and a few other Google technologies. It is a sparse, distributed, persistent and multidimensional sorted map. BigTable is scalable, self-managing and fault-tolerant. Therefore it has been widely used across Google services like Google Analytics, Google Earth and their personalized search. We have designed and implemented an open source implementation of BigTable called SmallTable.

2 EXPERIMENTAL SETUP

A single node runs master server and is responsible for operations on the tables. Each datanode in the cluster runs a tablet server and is responsible for serving the table rows. Both master and the tablet server handle requests concurrently using threads. HDFS replication factor for the hadoop cluster is set to 3. Client can be run from outside the cluster as well. We used HDFS Java API, TCP sockets for communication between the client and the cluster and JSON for serialization and deserialization of the messages and data storage.

3 STORAGE

3.1 Storage in Tablet Server

Each tablet server has an in memory table (memtable) serving rows from different tables. Memtables work as long as we can fit all the data in the memory. We demonstrate this feature on the number of records. When the number of entries in the memtable exceed a certain threshold, the table is written to the HDFS. Each Tablet server stores files in its own directory in the HDFS-$HOME/*NodeName*. Data for each tablet in a single file (*TableName@FileNumber*.tablet) where *FileNumber* describes how recently the file was created. Tablets file contains the table records in JSON format. The tablet files on the disk are immutable like SSTables in BigTable. To find a particular key, we check the key in the memtable and then the tables in fireverse chronological order using the first value.

3.2 Storage in Master Server

Master server handles requests for creating, updating, opening and deleting tables. Master stores the files in its own directory in the HDFS - $HOME/master. For each table, the master creates a file - *TableName*.smalltable.

4 FORMAT

4.1 Table Format

```
TableName.smalltable
{
        tableName: webtable,
        families: ["lang", "content", "anchor"],
        tablets: {
                "A,Z" : "tabletserver1 : port",
                "a,z" : "tabletserver2 : port"
        }
}
```

4.2 Table Map Format

```
TableName@FileNumber.tablet
{
        "key1": {
                "family1": {
                        "field": {
                                5 : "value3",
                        }
                },
                "family2": {
                        "field2": {
                                19 : "value1",
                                10 : "value2",
                                5 : "value3",
                        }
                }
        },
        "key2": {
                "family1": {
                        "field": {
                                5 : "value3",
                        }
                }
        }
}
```

5 OPERATIONS AND API

We currently support the following Operations and APIs in our framework.

(1) Create Table
    (a) Client sends a create request to the master.

(b) Master creates a table file on the HDFS.
(c) SmallTable table = new SmallTable("simpletable");

(2) View Table
(a) Client sends an open table request to the master.
(b) Master reads the table contents from HDFS and returns the table data to the client
(c) table.open()

(3) Add Row
(a) Column families of the rows are validated with those in the table.
(b) Client sends an add row request with the row data to the tablet server.
(c) Tablet server adds the row in the memtable.
(d) table.addRow(row);

(4) Read Row
(a) Client sends a read row request to the tablet server.
(b) Tablet server searches the memtable. If key is not found, it searches the files on disk and returns the row data to the client.
(c) table.getRow(key);

(5) Update Row
(a) Column families of the rows are validated with those in table.
(b) Client sends an update row request with the row data to the tablet server.
(c) Tablet server updates the row in the memtable.
(d) table.updateRow(key, row);

(6) Delete Row
(a) Client sends a delete row request with the row data to the tablet server.
(b) Tablet server deletes the row from the memtable.
(c) table.deleteRow(key);

(7) Delete Table
(a) Client sends delete table request to the master.
(b) Master deletes the table file from HDFS.

(8) Row Operations:
(a) Create row object: SmallRow row = new SmallRow();
(b) Add column value with timestamp: row.setColumn("familyname:foo", "hello", 5);
(c) Add column value: row.setColumn("familyname:foo", "hello");
(d) Get column value with latest timestamp: row.getColumn("familyname:foo");
(e) Get column value with specific timestamp: value = row.getColumn("familyname:foo", 5);
(f) Get value with latest timestamp: row.getValue("familyname:foo");
(g) Get value with specific timestamp: row.getValue("familyname:foo", 5);
(h) Get column family value: row.getFamily("familyname");

REFERENCES

[1] M. Burrows. The chubby lock service for loosely-coupled distributed systems. In *Proceedings of the 7th symposium on Operating systems design and implementation*, pages 335–350. USENIX Association, 2006.

[2] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS)*, 26(2):4, 2008.

[3] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1): 107–113, 2008.

[4] S. Ghemawat, H. Gobioff, and S.-T. Leung. The google file system. In *ACM SIGOPS operating systems review*, volume 37, pages 29–43. ACM, 2003.