Bigtable: A Distributed Storage System for Structured Data

CSE706: Distributed Computing Systems

Group 12

Istiaq Mohammad - 22266025

Mirza Md. Nazmus Sakib - 22166017

Irfana Afifa - 22166047

Contents

Introduction

- Bigtable is a distributed data storage system
- Can scale to thousands of machines and petabytes of data
- Used in various Google products such as Analytics, Earth, and more
- Shares many similarities with databases but provides a different interface
- Treats data as uninterpreted strings, allowing user defined schemas

Data Model

- A Bigtable is a sparse, distributed, persistent multidimensional sorted map
- Map indexed by a row key, column key, and a timestamp

  (row:string, column:string, time:int64) ! string

- Row range is dynamically partitioned
- Columns are grouped into families for access control and compression
- Timestamps help to differentiate and manage version control

API

- Bigtable API allows creating and deleting tables and column families
- Bigtable supports single-row transactions
- Bigtable supports the execution of client-supplied scripts
- Bigtable can be used both as an input source and as an output target for MapReduce jobs

Building Blocks

- Bigtable uses the distributed Google File System to store log and data files
- A Bigtable cluster typically operates in a shared pool of machines so a cluster management system is required
- Google SSTable file format is used internally to store Bigtable data which provides a persistent, ordered immutable map from keys to values.
- Bigtable relies on a highly-available and persistent distributed lock service called Chubby which works using five active replicas to ensure atomic read-write access.

Implementation

- Three major components: a library that is linked into every client, one master server, and many dynamic tablet servers.
- The master is responsible for assigning tablets to tablet servers, detecting the addition and expiration of tablet servers, balancing tablet-server load, and garbage collection of files in GFS.
- Clients communicate directly with tablet servers for read-writes, making the master lightly loaded in practice.
- As tables grow, they get split across multiple tablets

Refinements

- To improve the performance, the following refinements were made:
- Locality groups
- Compression
- Caching for read performance
- Bloom filters
- Commit-log implementation
- Speeding up tablet recovery
- Exploiting immutability

Performance Evaluation

- A Bigtable cluster with N tablet servers to measure the performance and scalability of Bigtable as N is varied was set up
- R was the distinct number of Biqtable row keys involved in the test.

Experiment	# of Tablet Servers			
	1	50	250	500
random reads	1212	593	479	241
random reads (mem)	10811	8511	8000	6250
random writes	8850	3745	3425	2000
sequential reads	4425	2463	2625	2469
sequential writes	8547	3623	2451	1905
scans	15385	10526	9524	7843

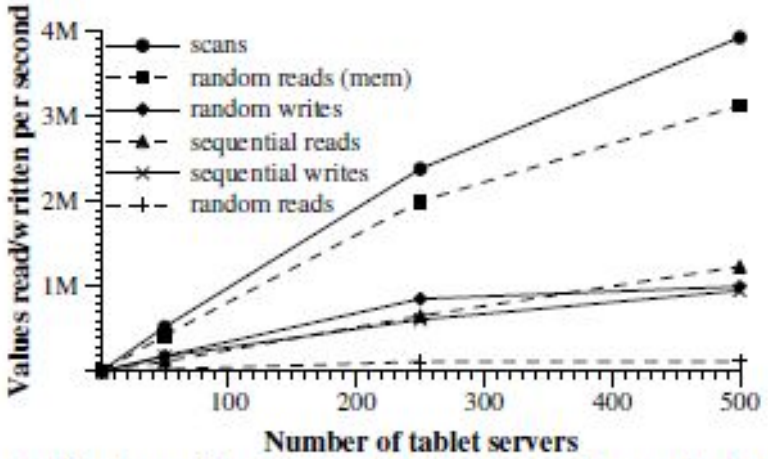

Figure 6: Number of 1000-byte values read/written per second. The table shows the rate per tablet server; the graph shows the aggregate rate.

Real Applications

- Bigtable is used to improve the performance of the following Google services:
- Google Analytics
- Google Earth
- Google Personalized Search

Lessons

- Large distributed systems are vulnerable to many types of failures
- It is important to delay adding new features until it is clear how the new features will be used
- The importance of proper system-level monitoring
- The value of simple designs and the importance of code and design clarity

Related Work

- The Boxwood project has components that overlap in some ways with Chubby, GFS, and Bigtable.
- CAN, Chord, Tapestry, and Pastry have tackled the problem of providing distributed storage or higher-level services over wide area networks
- Oracle's Real Application Cluster database is a parallel database that can store large volumes of data, but supports a complete relational model

Conclusion

- Bigtable clusters have been in production use since April 2005
- Allows for flexibility, scalability, and easy integration into Google's infrastructure
- Developing their own data model provided Google a greater degree of control, flexibility and optimization for their services.