Review: BigTable

Louis Rabiet

October 16, 2013

What problem did the paper address? Who is the intended audience?

The paper is trying to describe a NoSQL database that could be used for Web storage. The goal of the system is to store a massive number of pages that will contain the web page content and of course several others metadata (to be able to construct a search engine and to provide information for other Google services). While Google is making the paper and the architecture description public, the community has build a system HBase that is closely related to BigTable paper but that HBase is not necessarily corresponding to the actual BigTable architecture. This architecture is here to respond to very specific need (indexing the web) and the intended public could be other concurrent of Google or users of Hadoop.

Is it important/interesting? What was the context for the paper?

The whole NoSQL that emerged in the 00' is a very interesting development. People are trying to invent new technique that will allow more efficient requests when dealing with non-structured data. The main interest for me personally is to see if people are able to reconstruct or at least to have some of the theoretical properties that have been developed in the relational database field. A criticism of the NoSQL database in general is that most people (industrial groups) who are using it are trying to solve practical problems and that they are not caring about nice properties but about cost efficiency. For example very people are actually using a 3NF database. Of course NoSQL have some property that are easy to exploit in today's architectures, like easier horizontal scaling.

What is the approach used to solve the problem?

The papers is describing the column oriented database and describing some of the optimization that are put in place.

What are the possible inefficiencies in this approach?

- Why someone would like to use the Paxos algorithm ?
  Even if consensus is know to be a difficult problem (impossible in asynchronous systems for example[3]), Paxos as described in [2] seems really inefficient by letting several nodes to be the coordinator. An other problem I see is the constraint imposed on the failure type in the Paxos consensus. One could implement [1]

- Finding a tablet location seems too difficult.
  BigTable is using a tree structure to store the tablet. I would have prefer a more deterministic way of finding the place (like hashing), because finding a tablet should no require any request or exchange of messages.

- The number of tablets seems to be a problem.
  The optimization like compaction is not a elegant way of dealing with a huge number of tablets (there is a threshold to fix and it is difficult to know what is the correct value). I would prefer a system that allow aggregation of data one-by-one, meaning that if a the number of tablets is too high new data should be incorporated to existent tablets.

- GFS is probably struggling for providing a huge number of accesses.
  QFS[5] is a more efficient filesystem that should provide much better performance (it is beating Hadoop FS).

How does the paper support or otherwise justify the conclusions it reaches?

Contrary to the previous papers I reviewed, the article is trying to give some number on the efficiency of the system but there is still no comparison (maybe due to lack of similar systems at the time of publication).

What problems are explicitly or implicitly left as future research questions?

- One important question that we should answer is: Are the relational DB not scalable ?
  An interesting database from this point view is the VoltDB `[[http://voltdb.com][http://voltdb.com]]` that is able to scale and to stay in the SQL world. But VoltDB can still be criticized because RAM is not so cheap anymore (meaning that in clusters what is lacking is always memory so putting pressure on the RAM is obviously speeding up the database but it is due to the nature of the storage not depending on software optimization.

- An interesting development in the NoSQL movement is the RDF databases (for example 4store[4]) that are providing some standard for data model, high-level query language and such.

References

[1] Nicolas Braud-Santoni, Rachid Guerraoui, and Florian Huc. Fast byzantine agreement. In *Proceedings of the 2013 ACM symposium on Principles of distributed computing*, PODC '13, pages 57–64, New York, NY, USA, 2013. ACM.

[2] Tushar D. Chandra, Robert Griesemer, and Joshua Redstone. Paxos made live: an engineering perspective. In *Proceedings of the twenty-sixth annual ACM symposium on Principles of distributed computing*, PODC '07, pages 398–407, New York, NY, USA, 2007. ACM.

[3] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, April 1985.

[4] Steve Harris, Nick Lamb, and Nigel Shadbolt. 4store: The design and implementation of a clustered rdf store). 2009.

[5] Michael Ovsiannikov and al. The quantcast file system. VLDB '13, 2013.