Reading Summary of Mesa: Geo-Replicated, Near Real-Time, Scalable Data Warehousing T11-Pang Cloud Tianyang Liu, Su Pu, Yicheng Wang

link: https://docs.google.com/document/d/1WuXqs6MwaGTXYlmpsVfFmyU345G5INqe8qqQF64dbaU/edit

This work describes the internal implementation of Mesa [1] which is an analytical data warehouse of Google of the latest generation. Mesa scales across multiple data centers, thus having the potential to process data as big as PB level. The system can respond to queries commands within a second, while maintaining ACID attributes. Mesa was designed servicing Google advertisement scenarios. And according to its description, following the development of the advertisement platform, customers proposed multiple demands on advertisement visibility respectively. Specific and more precise data requirements conversely boosted the scaling of the data. Mesa was therefore built to process the growing quantity of data, meantime providing consistency as well as real-time update throughput. Mesa fulfills seven requirements: atomic updates which avoid the status of update being partially in effect, consistency and correctness which can adapt to law, availability which eliminates single point failures, real-time update throughput which completes updates within several minutes across different views and datacenters, query performance which controls 99 percent of the point search latency within hundreds of milliseconds, scalability which supports data in PB level, online data and metadata transformation which posts almost no influence on normal search and updates.

Back then Google technologies could not fulfill all requirements listed above. BigTable [2] could not provide atomic updates and the strong consistency. Technologies like Megastore [3], Spanners [4] and F1 [5] although could give consistency to multi-location data visits, they could not support the peak update throughput which was required by Mesa clients. Anyway, Mesa still in its different basic frameworks took advantage of multiple Google technical components. BigTable was used to store all permanent metadata. Colossus [6] the Google distribution file system was used to store data files. In addition, MapReduce was used to process continuous data. Mesa data model, in its concept is almost the same as the traditional relationship database. All data are stored in a table. One table could also be the object view of a second one, and each table has an appointed structured mode. The interesting part of Mesa is the way of processing updates. Mesa stores data in multiple versions, which enables it to provide consistent data when new updates are being processed. Usually every few minutes, high level system will process a batch task to complete all new updates. That is how it is working.

Important lessons are concluded from this work. First be well prepared before making a large scale architecture into real projects. Events of small possibilities may happen when the scale is large, which could result in a total crush. Leveled design is playing an important part in reducing the system complexity, and should be adopted even though performance loss. Keep codes clean. Do frequent unit tests. Documented normal operations. Top researches today are more likely to be done in groups: this article has 19 coauthors and 6 acknowledgements. In the era of data, research fruits are more dependent of big projects. Big data means continuously generated lots of data, and the biggest challenge is real-time processing: via special hardwires and programming models like MapReduce.

Three strong points are:
1. It can save a lot of time by using MapReduce since MapReduce can save a lot of computation time. The Mesa parallelizing and linked schema is a very good method to save storage because it saves a lot of space to add computation on every query path.
2. The resume key is very special part of Mesa's query servers, Mesa attach the resume key to each block and store the information there, when query server becomes unresponsive, the client won't need to wait and just switch to another query server and resume the query from the resume key instead of doing the whole process again.
3. The Mesa design the structure think of many aspects. In the distribution part, mesa can deal with large data since it stick to rules of distribution, parallelism and cloud computing, it also have lower layer architecture component such as Colossus and BigTable, that make the construction process easier. Mesa also take care of the potential data corruption and component failures, however, there are still some problems remain.

Three weak points are:
1. Though the Paxos deals with the problem of synchronism on datacenter (mainly on metadata), it only makes sure that most of instance is replicated consistently. If the Paxos fails, so would the data on detadata to Mesa instance.
2. During the online checking for data validation, the data is checked very update and query, which means lots of cost and latency for the whole system.
3. Since the controllers assign different tasks to different workers, if lots of tasks are working at the same time, the timers would add lots of work to the controllers.

New research ideas are: The Mesa warehouse processes analytical queries within a single data center, which will lead to a large bandwidth. If there is a way to avoid that while still geo-replicates for fault tolerance, the system will be much efficient.

Reference

[1].https://xduan7.com/2016/02/23/paper-review-mesa-geo-replicated-near-real-time-scalable-data-warehousing/
[2].Vulimiri, Ashish, et al. "Global analytics in the face of bandwidth and regulatory constraints." 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15). 2015.
[3].http://blog.csdn.net/colorant/article/details/50788229