January 2016 IJMIE Volume 6, Issue 1 ISSN: 2249-0558

BIG DATA PROCESSING MODELS DEVELOPED BY GOOGLE INC.: AT A GLANCE

Saatchi Nandwani*

Akshat Kumar**

ABSTRACT:

Whether it's fine-tuning supply chains, monitoring shop floor operations, gauging consumer sentiment, or any number of other large-scale analytic challenges, big data is having a tremendous impact on the enterprise. The amount of business data that is generated has risen steadily every year and more and more types of information are being stored in digital formats. This paper presents a review of various data processing solutions developed by Google Inc. from 2002-2015 necessary for handling such large data set. These models define various methods implemented to handle Big Data, also in the paper are listed various frameworks wherein these processing models have been incorporated.

Keywords: Big Data, Data Processing, Google Inc.

* Computer Engineering, HMR Institute of Technology and Management, Guru Gobind Singh Indraprastha University, New Delhi

** Information and Communication Technology, Manipal Institute of Technology, Manipal University, Karnataka

Introduction

Big data analytics refers to the process of collecting, organizing and analyzing large sets of data to discover patterns and other useful information. It is characterized by 4V's:

Volume:Volume represents how the data is large. Size or the Volume of data now is more than Petabytes. The grand scale and rise of data outstrips traditional store and analysis techniques.

Variety: Variety points to the number of sources or incoming vectors leading to your databases. That might be embedded sensor data, phone conversations, documents, video uploads or feeds, social media, and much more.

Velocity: Velocity or speed refers to how fast the data is coming in, but also to how fast you need to be able to analyze and utilize it.

Veracity: If you can't trust the data itself, the source of the data, or the processes you are using to identify which data points are important, you have a veracity problem.

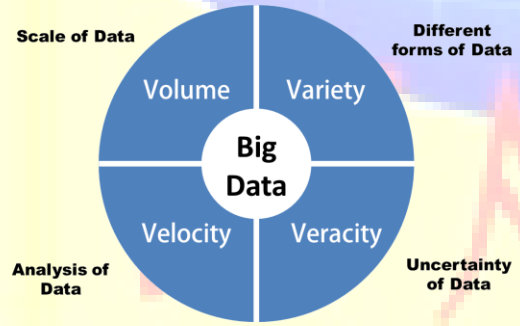
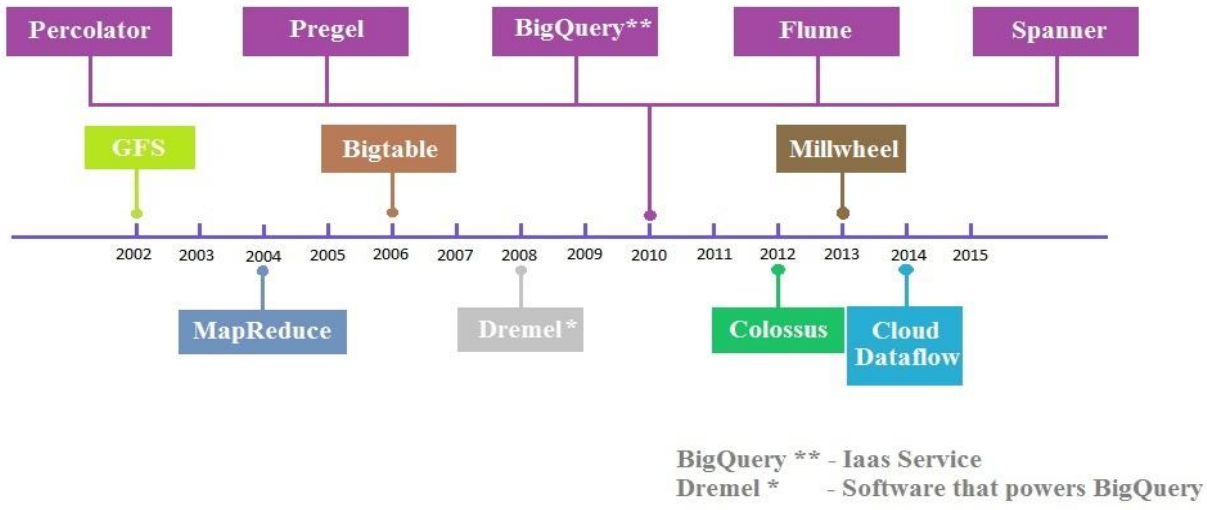


Fig 1 : Introduction

Big data requires exceptional technologies to efficiently process large quantities of data within tolerable elapsed times. Suitable technologies include A/B testing, crowdsourcing, data fusion and integration, genetic algorithms, machine learning, natural language processing. Additional technologies being applied to big data include massively parallel-processing (MPP) databases, search-based applications, data mining, distributed file systems, distributed databases, cloud-based infrastructure (applications, storage and computing resources) and the Internet.

Big data processing can be broadly classified under the below given 3 sub-categories:

1. **Batch processing**: Data processing under this category is full power, full table scan with significant (from minutes to hours) lag.
2. **Stream processing**: A stream of data is processed at real time in parallel. The slice of data being processed at any moment in an aggregate function is demarcated by a sliding window.
3. **Fast/Real-time processing**: Performing ad hoc analysis on a full big data set within seconds lags. Working effectively as a big data OLAP (Online Analytical Processing) system. The amount of data processed per operation in OLAP is substantially more massive than data processed in OLTP. Image how daunting the challenge is!



Google is one of the big power players on the World Wide Web and beyond. The company relies on a **distributed computing system** to provide users with the infrastructure they need to access, create and alter data. Google has one of the largest and most advanced computer networks. Google's backbone network has thousands of miles of fiber optic cable, uses advanced software-defined networking and has edge caching services to deliver fast, consistent and scalable performance.

History of Bigdata at Google

Fig 2: History of Bigdata

Google has led the industry with innovations in software infrastructure such as MapReduce, BigTable and Dremel. Today, Google is pushing the next generation of innovation with products such as Spanner and Flume. Also we have Google Cloud Services that lets you get access to Google's technology innovations faster.

The following paper reviews the various data processing solutions developed by Google Inc. The paper has been described as follows. Section I: Introduction about Big data and the various types of data processing models. Section II: describes briefly different data processing solutions developed by Google Inc. Section III: describes the architecture of those solutions. Section IV: describes where these solutions have been deployed.

Section I

1. Google FileSystem

Google File System is a scalable distributed file system for large distributed data-intensive applications. It provides fault tolerance while running on inexpensive commodity hardware, and it delivers high aggregate performance to a large number of clients.

2. Map Reduce

MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a *map* function that processes a key/value pair to generate a set of intermediate key/value pairs, and a *reduce* function that merges all intermediate values associated with the same intermediate key.

3. Big Table

Bigtable is a distributed storage system for managing structured data that is designed to scale to a very large size: petabytes of data across thousands of commodity servers. It is built on Google File System, Chubby Lock Service, SSTable (log-structured storage like LevelDB) and a few other Google technologies.

4. Dremel

Dremel is a scalable, interactive ad-hoc query system for analysis of read-only nested data. By combining multi-level execution trees and columnar data layout, it is capable of running aggregation queries over trillion-row tables in seconds. The system scales to thousands of CPUs and petabytes of data, and has thousands of users at Google.

5. FlumeJava

It is a system that aims to support the development of data-parallel pipelines. FlumeJava is a Java library centered around a few classes that represent parallel collections. Parallel collections support a modest number of parallel

operations which are composed to implement data-parallel computations. An entire pipeline, or even multiple pipelines, can be implemented in a single Java program using the FlumeJava abstractions; there is no need to break up the logical computation into separate programs for each stage.

6. Percolator

Percolator is a system for incrementally processing updates to a large data set that creates the Google web search index. By replacing a batch-based indexing system with an indexing system based on incremental processing using Percolator, we process the same number of documents per day, while reducing the average age of documents in Google search results by 50%.

7. Pregel

It is a scalable infrastructure to mine a wide range of graphs. In Pregel, programs are expressed as a sequence of iterations. In each iteration, a vertex can, independently of other vertices, receive messages sent to it in the previous iteration, send messages to other vertices, modify its own and its outgoing edges' states

8. Spanner

Spanner is Google's globally distributed NewSQL database, the successor to BigTable. Google describes Spanner as a not pure relational database system because each table must have a primary key column. Described as a NewSQL platform, Spanner is used internally within

Google's infrastructure as part of the Google platform. Spanner uses the Paxos algorithm as part of its operation to shard data across hundreds of datacenters.

9. Colossus

Colossus(or GFS2) is a proprietary distributed fil*e* system from Google. It is the evolutionary development of GFS. Optimized for operation with the big data sets, Colossus is a scalable, high-accessible and fault tolerant system allowing the data to be stored reliably.At the same time, Colossus eliminates some bottlenecks of the predecessor.

10. Millwheel

Millwheel is a framework for building low-latency data-processing applications that is widely used at Google. Users specify a directed computation graph and application code for individual nodes, and the system manages persistent state and the continuous flow of records, all within the envelope of the framework's fault-tolerance guarantees.

11. Google Cloud Platform

The Google Cloud Platform offers various fully managed Big Data processing services that remove the operational burden found in traditional data processing systems. They enable you to build applications on a platform that can scale with the growth of your business and drive down data processing latency, all while processing your data efficiently and reliably.

Every day, customers use Google Cloud Platform to execute business-critical big data processing workloads, including: financial fraud detection, genomics analysis, inventory management, click-stream analysis, A/B user interaction testing and cloud-scale ETL.The services are described below:

12.1 BigQuery

BigQuery is a RESTful web service that enables interactive analysis of massively large datasets working in conjunction with Google Storage. It is an Infrastructure as a Service (IaaS) that may

be used complementarily with MapReduce.BigQuery (BQ) is reportedly based on Dremel, a scalable, interactive ad hoc query system for analysis of read-only nested data.

12.2 Cloud PubSub

Cloud Pub/Sub -its backend messaging service that makes it easier for developers to pass messages between machines and to gather data from smart devices. It's basically a scalable messaging middleware service in the cloud that allows developers to quickly pass information between applications, no matter where they're hosted.

12.3 Cloud Dataflow

Google Cloud Dataflow is a simple, flexible, and powerful system you can use to perform data processing tasks of any size. Cloud Dataflow can be used for nearly any kind of data processing task, **including both batch and streaming data processing**. The Dataflow SDKs provide a unified data model that can represent any size data set, including an unbounded or infinite data set from a continuously updating data source such as **Google Cloud Pub/Sub**. The Dataflow managed service is capable of running both batch and streaming jobs. Dataflow can also be used for "Extract, Transform, and Load" tasks.

Section 2:Architecture

I. **GFS**

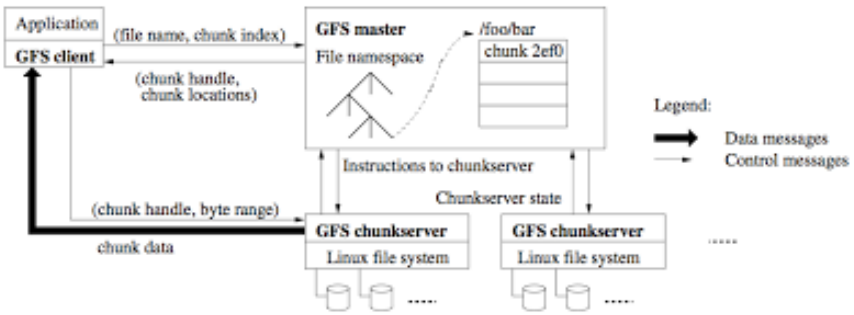


Fig 3: GFS Architecture

A GFS cluster consists of a single *master* and multiple *chunkservers*and is accessed by multiple *clients*, as shown

in Figure 1. Each of these is typically a commodity Linux machine running a user-level server process. It is easy to run both a chunkserver and a client on the same machine, as long as machine resources permit and the lower reliability caused by running possibly flaky application code is acceptable.

Files are divided into fixed-size *chunks*. Each chunk is identified by an immutable and globally unique 64 bit *chunkhandle*assigned by the master at the time of chunk creation. Chunkservers store chunks on local disks as Linux files and read or write chunk data specified by a chunk handle and byte range. For reliability, each chunk is replicated on multiple chunkservers. By default, we store three replicas, though users can designate different replication levels for different regions of the file namespace.

The master maintains all file system metadata. This includes the namespace, access control information, the mapping from files to chunks, and the current locations of chunks. It also controls system-wide activities such as chunk lease management, garbage collection of orphaned chunks, and chunk migration between chunkservers. The master periodically communicates with each chunkserver in Heartbeat messages to give it instructions and collect its state.

II. MapReduce

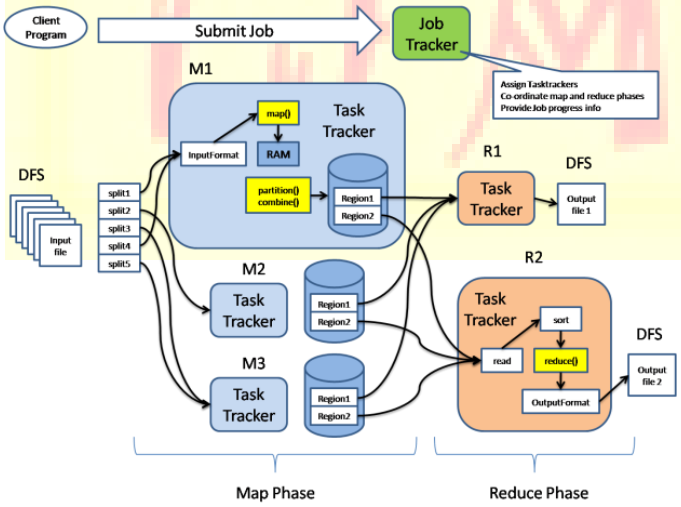
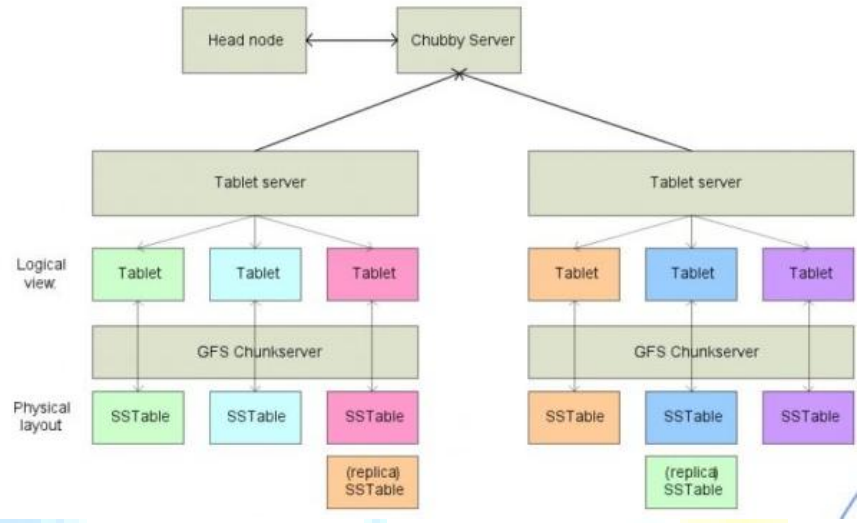


Fig 4: MapReduce Architecture

The job execution starts when the client program submit to the JobTracker a job configuration, which specifies the map, combine and reduce function, as well as the input and output path of data. The JobTracker will first determine the number of splits (each split is configurable,

~16-64MB) from the input path, and select some TaskTracker based on their network proximity to the data sources, then the JobTracker send the task requests to those selected TaskTrackers.

Each TaskTracker will start the map phase processing by extracting the input data from the splits. For each record parsed by the "InputFormat", it invokes the user provided "map" function, which emits a number of key/value pair in the memory buffer. A periodic wakeup process will sort the memory buffer into different reducer node by invoke the "combine" function. The key/value pairs are sorted into one of the R local files (suppose there are R reducer nodes).



When the map task completes (all splits are done), the TaskTracker will notify the JobTracker. When all the TaskTrackers are done, the JobTracker will notify the selected TaskTrackers for the reduce phase.

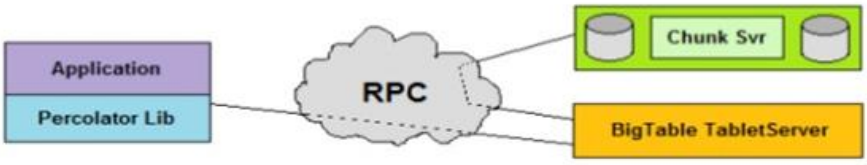
Each TaskTracker will read the region files remotely. It sorts the key/value pairs and for each key, it invoke the "reduce" function, which collects the key/aggregated Value into the output file (one per reducer node).Map/Reduce framework is resilient to crash of any components. The JobTracker keep tracks of the progress of each phases and periodically ping the TaskTracker for their health status. When any of the maps phase TaskTracker crashes, the JobTracker will reassign the map task to a different TaskTracker node, which will rerun all the assigned splits. If the reduce phase TaskTracker crashes, the JobTracker will rerun reduce at a different TaskTracker.

After both phases completes, the JobTracker will unblock the client program.

III. **BigTable**

Fig 5: Bigtable Architecture

The Bigtable implementation has three major components: a library that is linked into every client, one master server, and many tablet servers. Tablet servers can be dynamically added (or removed) from a cluster to accommodate changes in workloads. 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 _les in GFS. In addition, it handles schema changes such as table and column family creations. Each tablet server manages a set of tablets. The tablet server handles read and write requests to the tablets that it has loaded, and also splits tablets that have grown too large. As with many single-master distributed storage systems client data does not move through the master: clients communicate directly with tablet servers for reads and writes. Because Bigtable clients do not rely onthe master for tablet location information, most clientsnever communicate with the master. As a result, the masteris lightly loaded in practice. A Bigtable cluster stores a number of tables. Each tableconsists of a set of tablets, and each tablet containsall data associated with a row range. Initially, each tableconsists of just one tablet. As a table grows, it is automaticallysplit into multiple tablets, each approximately100-200 MB in size by default.



IV. **Dremel**

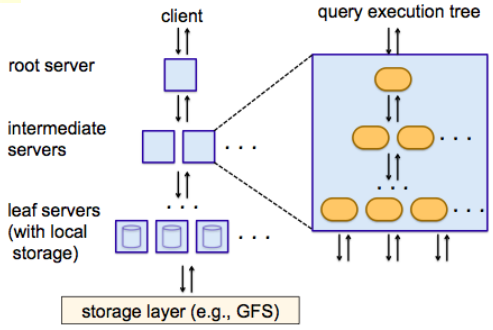


Fig 6: System Architecture and Execution inside a Server Node

One of the challenges Google had in designing Dremel was how to dispatch queries and collect results across tens of thousands of machines in a matter of seconds. The challenge was resolved by using the Tree architecture. The architecture forms a massively parallel distributed tree for pushing down a query to the tree and then aggregating the results from the leaves at a blazingly fast speed.

V. Percolator

Fig 7: Percolator Architecture

A Percolator system consists of three binaries that run on every machine in the cluster: a Percolator worker, a Bigtable tablet server, and a GFSchunkserver.

All observers are linked into the Percolator worker, which scans the Bigtable for changed columns ("notifications") and invokes the corresponding observers as a function call in the worker process. The observers perform transactions by sending read/write RPCs to Bigtable tablet servers, which in turn send read/write RPCs to GFS chunkservers. The system also depends on two small services: the timestamp oracle and the lightweight lock service. The timestamp oracle provides strictly increasing timestamps: a property required for correct operation of the snapshot isolation protocol. Workers use the lightweight lock service to make the search for dirty notifications more efficient.

VI. FlumeJava

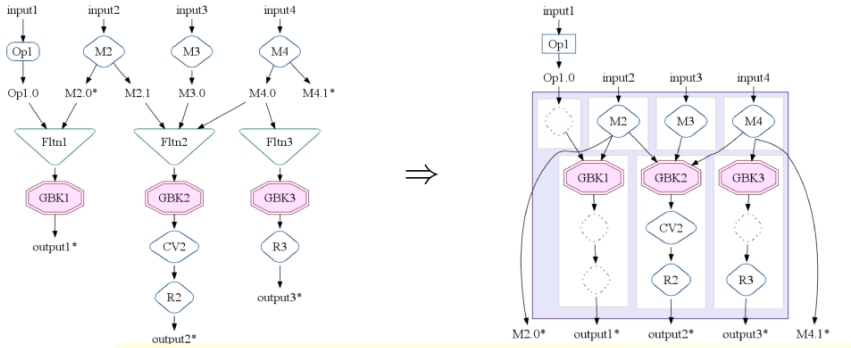


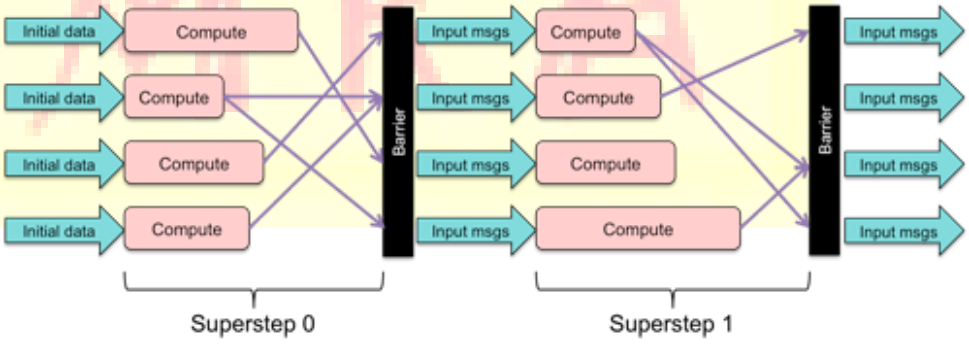
Fig 8: FlumeJava Architecture

FlumeJava, a new system that aims to support the development of data-parallel pipelines.FlumeJava is a Java library centered around a few classes that represent parallel collections. Parallel collections support a modest number of parallel operations which are composed to implement data-parallel computations. An entire pipeline, or even multiple pipelines, can be implemented in a single Java program using the FlumeJava abstractions; FlumeJava's parallel collections abstract away the details of how data is represented

To achieve good performance, FlumeJava internally implements parallel operations using deferred evaluation.

Given in the figure is the

The core of the FlumeJava optimizer transforms combinations of ParallelDo, GroupByKey, CombineValues, and Flatten operations into single MapReduces. To help bridge the gap betweenthese two abstraction levels, the FlumeJava optimizer includes an intermediate-level operation, the MapShuffleCombineReduce (MSCR) operation. An MSCR operation has M input channels (each performing a map operation) and R output channels (each optionally performing a shuffle, an optional combine, and reduce). Each input channel m takes a PCollection<Tm>asinput and performs an R-output ParallelDo "map" operation (which defaults to the identity operation) on that input to produce R outputs of type PTable<Kr,Vr>s; the input channel can choose to emit only to one or a few of its possible output channels.



Each output channel r Flattens itsM inputs and then either (a) performs a GroupByKey "shuffle", an optional CombineValues"combine", and a Or-output ParallelDo "reduce" (which de-faults to

the identity operation), and then writes the results to OroutputPCollections, or (b) writes its input directly as its output.

VII. Pregel

Fig 9: Pregal Architecture

Pregel's programming model fundamentally uses **message passing between vertices in a graph**. Pregel organizes this message passing into a sequence of iterations called *supersteps*. At the beginning of each superstep, the framework runs a **user-specified compute function on each vertex**, passing it all the messages sent to that vertex during the last superstep. The compute function has the opportunity to process these messages and then send messages to other vertices, which will be received in the next superstep. It also can "vote to halt," deactivating the vertex it's running on until the vertex receives a message. Once all vertices have voted to halt, the job terminates.

VIII. Spanner

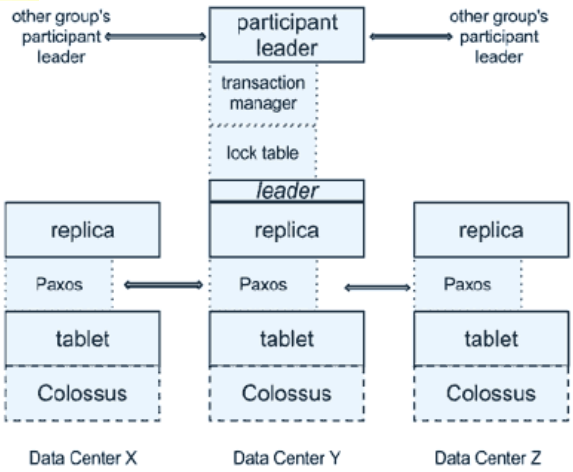


Fig 10: Spanner Architecture

Each spanserver manages 10 to 1000 data structures called tablet. A tablet has a concept similar to a tablet of BigTable. It can store multiple mappings in the form of (key: string, timestamp:int64) string.

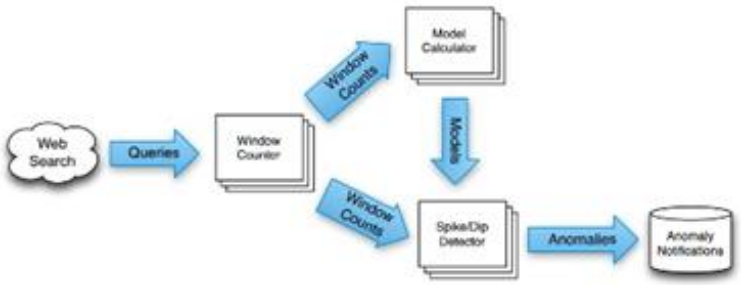
The status of a tablet is stored in **Colossus Distributed File System** (the successor of Google File System) in the form of B-tree file and write-ahead log (WAL).

Spanner uses Paxos state machine to support data replication among spanservers.

A spanserver has the transaction manager to support distributed transactions. The transaction manager is not involved in a transaction performed in a single Paxos group, but when a transaction is performed across multiple Paxos groups, one of the participant leaders is selected as coordinator leader, and performs coordination to enable phase-2 commit.

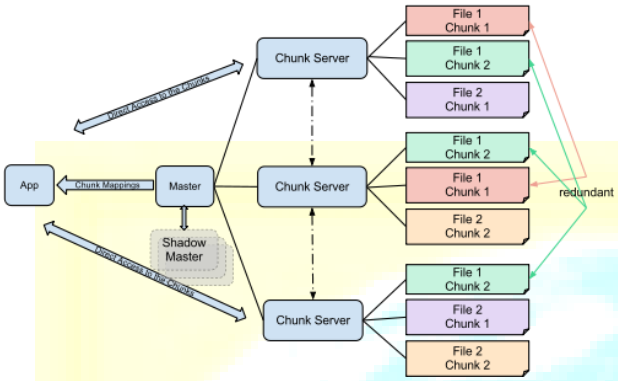
Millwheel

Fig 11: Milwheel Architecture



MillWheel deployments run as distributed systems on a dynamic set of host servers. Each computation in a pipeline runs on one or and persisted work in the system. When aggregating per-processupdates, it tracks the completeness of its information for each computation by building an interval map of low watermark values for the computation. If any interval is missing, then the low watermarkcorresponds to the last known value for the missing interval until it reports a new value. The authority then broadcasts low watermark values for all computations in the system.To maintain consistency at the central authority, we attach sequencersto all low watermark updates. In a similar manner to our single-writer scheme for local updates to key interval state, these sequencers ensure that only the latest owner of a given key interval can update its low watermark value. For scalability, the authority can be shared across multiple machines, with one or more computations on each worker. Empirically, this can scale to 500,000 key intervals with no loss in performance.

IX. Colossus



GFS has been improved and re-written to give rise to Colossus. The underlying architecture remains the same with the following changes being brought about:

Elegant Master Failover-No more 2s delays

Chunk size is now 1MB-likely to improve latency for serving data other than indexing (for eg-Gmail)

Master can store more chunk metadata-Thus more chunks addressable also giving rise to more chunk servers.

Fig 12: Colossus Architecture

X. Google Cloud Platform

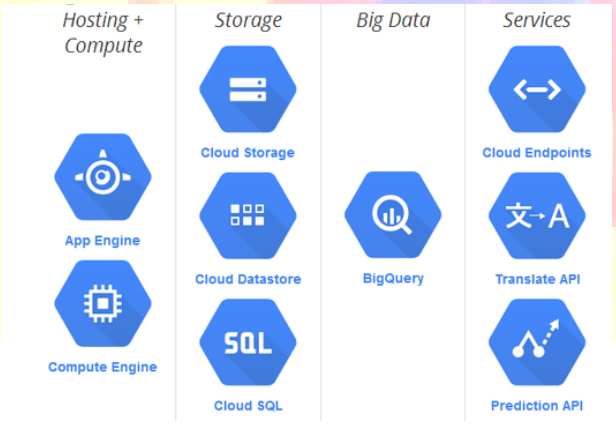


Fig 13: Google Cloud Platform

a) BigQuery

BigQuery makes it possible for public to utilize the power of Dremel for their Big Data processing requirements. BigQuery provides the core set of features available in Dremel to third party developers. It does so via a REST API, command line interface, Web UI, access control, data schema management and the integration with Google Cloud Storage. BigQuery and Dremel share the same underlying architecture and performance characteristics. Users can

fully utilize the power of Dremel by using BigQuery to take advantage of Google's massive computational infrastructure. Thus, including valuable benefits like multiple replications across regions and high data center scalability and most importantly, this infrastructure requires no management by the developer.

b) Cloud PubSub

Cloud Pub/Sub is accessed through a set of REST API's. Small, but important details are the resource names for topics and subscriptions, which must follow the following format: projects/project-identifier/collection/resource-name wherecollection can be topics or subscriptions.

The response to the above format is a simple JSON structure. PubSub can give response in various other formats depending on the API call.

c) Cloud Dataflow

All datasets are represented in PCollections ("parallel collections") in Dataflow. It includes a "rich" library of PTransforms (parallel transforms), including ParDo (similar to Map and Reduce functions and WHERE in SQL), and GroupByKey (similar to the shuffle step of MapReduce and GROUPBY and JOIN in SQL). A starter set of these transforms can be used out of the box, including Top, Count and Mean.

Google Solution/Product	Type	Release Date	Application
Google File System	Distributed file system	2002	Large distributed data intensive applications like Google Search
MapReduce	Programming model	2004	Distributed pattern based searching,distributedsorting,web link graph reversal,machinelearning,etc.
BigTable	Cloud Storage	2006	Web Indexing,GoogleMaps,Google Book Search,Youtube,Gmail
Dremel	Programming model	2008	Powers Google's BigQuery service
Pregel	Programming model	2010	Usage oriented towards graph based algorithms.,e.g, to compute the pagerank
FlumeJava	Programming model	2010	Used in optimizing several pipelined MapReduce jobs
Percolator	Programming model	2010	Google web search Index
BigQuery	Cloud Service	2010	Interctive analysis of massively large data sets
Spanner	NewSQL database	2010	Google's advertisement platform-F1
Colossus	Distributed file system	2012	Replacement for GFS; specifically built for realtime services
Millwheel	Framework	2013	Stream processing system powering Google's Zeitgeist service among other applications.
Cloud Pub/Sub	Cloud Service	2015	Reliable and real time messaging
Cloud Dataflow	Cloud Service	2015	ETL, batch computation, and continuous computation

Table 1: Time Line of Google Big Data Technologies

Conclusion:

Undoubtedly one can say that if we are in a data revolution right now, the computational advance that made it possible is the realization that these computing systems can and should be built from relatively cheap, shared-nothing machines.

Confronted with a data explosion, Google engineers Jeff Dean and Sanjay Ghemawatarchitected two seminal systems: the Google File System (GFS) and Google MapReduce (GMR). The former was a brilliantly pragmatic solution to Exabyte-scale data management using commodity hardware.

The latter was an equally brilliant implementation of a long-standing design pattern applied to massively parallel processing of said data on said commodity machines. GFS and GMR became the core of the processing engine used to crawl, analyze, and rank web pages into the giant inverted index that we all use daily at google.com. This was clearly a major advantage for Google.

However with the advent of technologies like Dremel, Cloud dataflow, etc it has been observed that GMR no longer holds such prominence in the Google stack.

Map-Reduce, as implemented, typically has substantial overhead attributable both to its inherent 'batchness', and the need to have a barrier between the map and reduce phases. MapReduce is designed as a batch processing framework, so it's not suitable for ad hoc and trial-and-error data analysis. The turnaround time is too slow, and doesn't allow programmers to perform iterative or one-shot analysis tasks on Big Data. Dremel which is designed as an interactive data analysis tool for large datasets comes into play here.

Thus with the data mining and extraction of useful information from the mined data gaining importance and relevance in our life, Google with its new innovations in this field has become a part of our life. But with changing usage and requirements,technologies arealso evolving thus making some Google tools irrelevant and other Google tools gaining foot hold and importance. Hence we should keep a close watch on the new emerging technologies to gain an insight of what is to be expected next and being updated with new requirements and solution to the requirements.

REFERENCES

[1] Big Data introduction:
(a)https://books.google.co.in/books?id=h3K4LhkK1BgC&pg=PA121&lpg=PA121&dq=Google's+data+processing+models&source=bl&ots=UQYTrnqqeM&sig=SGdwtzTxbtzoTW4tf9HA14ZWuLU&hl=en&sa=X&ved=0CCsQ6AEwAzgKahUKEwjb0LzL--jHAhWDSo4KHTmRClg#v=onepage&q=Google's%20data%20processing%20models&f=false)

(b)https://en.wikipedia.org/wiki/Big_data

(c)http://www.webopedia.com/TERM/B/big_data_analytics.html

[2]Cloud dataflow: (a)https://cloud.google.com/dataflow/what-is-google-cloud-dataflow

(b)https://speakerdeck.com/campoy/mapreduce-is-dead-long-live-cloud-dataflow

[3] Daniel Peng, Frank Dabek; "Large-scale Incremental ProcessingUsing Distributed Transactions and Notifications"; Google Inc.

[4] Jeffrey Dean, Sanjay Ghemawat; "MapReduce: Simpli_ed Data Processing on Large Clusters"; Google Inc.

[5] Joseph McKendrick, "Big Data, Big Challenges, Big Opportunities: 2012 IOUG Big Data Strategies Survey", IOUG, Sept 2012

[6]Katal, A; Wazid, M.; Goudar, R.H., "Big data:issues, challenges, tools and good practices,"Contemporary Computing (IC3), 2013 SixthInternational Conference on , vol., no., pp.404,409,8-10 Aug. 2013.

[7]Kazunori Sato, Solutions Architect, Cloud Solutions Team; "An Inside Look at Google BigQuery", White Paper

[8]Millwheel: http://the-paper-trail.org/blog/paper-notes-stream-processing-at-google-with-millwheel/

[9]Sanjay Ghemawat, Howard Gobioff, Shun-Tak Leung; "The Google File System"; Google Inc.

[10]Sergey Melnik, Andrey Gubarev, Jing Jing Long, Geoffrey Romer, Shiva Shivakumar, Matt Tolton, Theo Vassilakis; "Dremel: Interactive Analysis of WebScaleDatasets"; Google Inc.

[11] Stephen Kaisler, Frank Armour, J. Alberto Espinosa, William Money, "Big data: issues andchallenges moving forward", IEEE, 46th HawaiiInternational Conference on System Sciences, 2013.