International Journal of Latest Trends in Engineering and Technology
Vol.(8)Issue(3), pp.356-360
DOI: http://dx.doi.org/10.21172/1.83.051
e-ISSN:2278-621X

HYBRID DATA PROCESSING MODEL FOR BIG DATA – A REVIEW

Jigisha Purohit[1]

Abstract- Big data programming models are widely used in large scale data processing and data analytics. In today's scenario large volumes of data is being generated from smart phones, wireless sensors, and enterprises. A common processing technique which fits single enterprise scope is not sufficient for such huge amounts of data. Its function varies with the type of data being handled, whose range varies from historical data to real time live streaming. The two broad views of data handling being Batch data processing and Real time data processing techniques. Hybrid approach for data processing which handles both batch data and real time data to solve specific set of business requirements. This paper focuses on study and issues of existing hybrid data processing model to solve big data issues [1].

Keywords - Big data, Hybrid Processing, Lambda Architecture, Summingbird

I. INTRODUCTION

Big data is a popular term used to describe the exponential growth and availability of data, both structured and unstructured. Large data sets now need to be processed at close to real-time speeds. Batch processing tries to gather more data before processing it. It consumes fewer resources but at the cost of higher latency. The MapReduce model and its counterpart open source implementation Hadoop, has proven itself as the de facto solution to big data processing. Hadoop is inherently designed for batch and high throughput processing jobs. Stream processing treats the incoming data as a stream and processes it through a processing pipeline as soon as the stream is gathered. It is more computationally intensive but grants lower latency. The problem with these approaches is that business requirements are both historic and real-time—simultaneously. One solution to this problem is hybrid data processing model. Hybrid—is becoming dominant. Hybrid is the future of big data because users increasingly realize that no single type of analytic platform is always best for all requirements. In the following section we will be analyzing different hybrid data processing models and its architecture.

II. BIG DATA

Big data is where the data volume, acquisition velocity, or data representation limits the ability to perform effective analysis using traditional approaches or requires the use of significant horizontal scaling for efficient processing. There is three characteristics define big data: volume (Terabytes -> Zettabytes), Variety (Structured -> Semi-structured -> Unstructured) and Velocity (Batch -> Streaming Data).

Volume: Increase in the amount of data generated on the Internet in recent years by the social media, increase in amount of sensors and system to system communication and records of transaction over long period of time has all been part of the contribution to Big data.
Velocity: Torrents of data being pushed in near real time for processing from social websites like Twitter and Facebook, RFID tags, sensors must be quickly dealt with is considered Velocity of Big data.
Variety: All the data being produced may not be of same type, rather they are never of same type. Some are financial transactions, email, video, text, audio and so on. Managing all these and merging is a big task, this aspect of Big data is viewed as Variety. It may also be like data from log files, media files and app info, all need to be processed.

[1] *Bhagwan Mahavir College of Management (MCA), Bhagwan Mahavir Education Foundation, Surat, Gujarat, India*

A broad division of processing Big data can be seen as Batch data processing and Real time data processing. Choosing between which kind of processing to be used is purely based on the nature of data to be handled. We now look into two major divisions of data processing models.

III. BATCH PROCESSING

Batch processing can be seen as the traditional Big Data - it revolves around the MapReduce paradigm where an input file is split into a number of smaller input files (the map step), an algorithm is executed in parallel batch mode on each of the input files across possibly many commodity computers, and the outputs from each step are then combined into a single output (the reduce step). The Apache Hadoop project implements this paradigm; this open source implementation has been adopted and modified to a various degree by a number of derivative distributions, most notably Cloudera CDH, Hortonworks Data Platform (HDP), MapR M series, and Amazon Elastic MapReduce (EMR). Utilizing the Hadoop framework requires an application to be developed using the MapReduce paradigm and is typically recommended when the input data is large (terabytes or larger) and diverse. Hadoop is accompanied with its own file system, Hadoop Distributed File System (HDFS), which provides storage and data redundancy on top of commodity hardware. HDFS is not a POSIX compliant file system and any data being operated on via Hadoop is required to be imported into HDFS. Upon completion of the data analysis, the data is exported onto a POSIX file system to be accessed by other tools.

IV. REAL – TIME / STREAMING BIG DATA PROCESSING

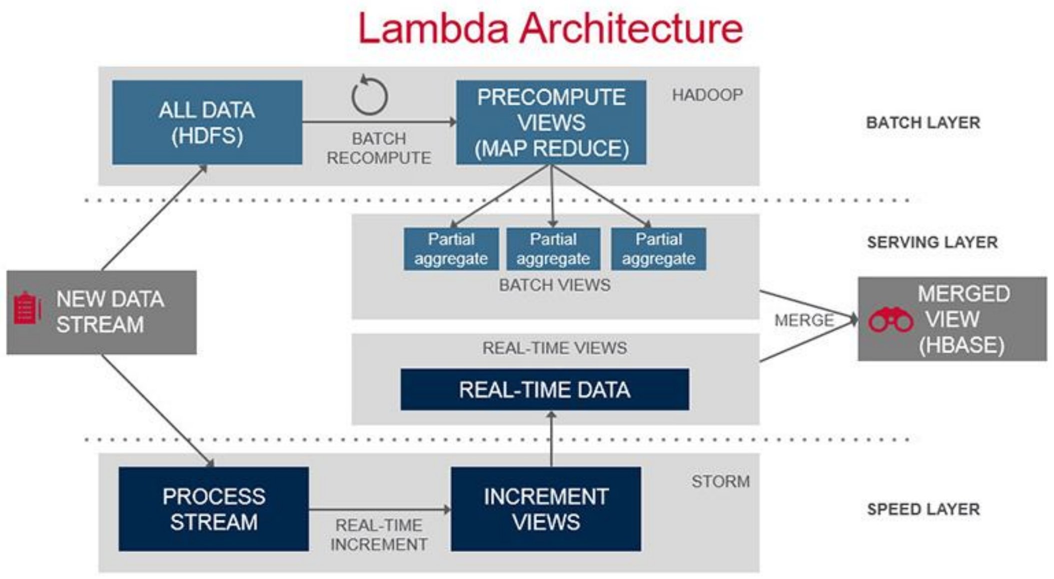
Real time data processing involves a continual input, process and output of data. Data must be processed in a small time period (or near real time), whereas “complex event processing” (CEP) utilizes event-by-event processing and aggregation (e.g. on potentially out-of-order events from a variety of sources – often with large numbers of rules or business logic). For this reason multiple types of event processing have evolved, described as queries, rules and procedural approaches (to event pattern detection). Essential to stream processing is Streaming Analytics, or the ability to continuously calculate mathematical or statistical analytics on the fly within the stream. Stream processing solutions are designed to handle high volume in real time with a scalable, highly available and fault tolerant architecture. This enables analysis of data in motion. In contrast to the traditional database model where data is first stored and indexed and then subsequently processed by queries, stream processing takes the inbound data while it is in flight, as it streams through the server. Stream processing also connects to external data sources, enabling applications to incorporate selected data into the application flow, or to update an external database with processed information. So many tools are available for streaming big data processing like Apache Storm, Apache Spark.

V. HYBRID BIG DATA PROCESSING

Batch Processing on Big Data solve the problem of large volume of data and real-time processing is more computationally intensive and deal with big data property velocity but grants lower latency. The problem with these approaches is that business requirements are both historic and real-time—simultaneously. One solution to this problem is hybrid data processing model. Hybrid—is becoming dominant. Users increasingly realize that no single type of analytic platform is always best for all requirements.

VI. LAMBADA ARCHITECTURE

Summingbird is also one of the first openly available Lambda Architecture compliant systems. The characteristics of the Lambda Architecture - immutable master dataset and the combination of batch, serving, and speed layer - enables people to build robust large-scale data processing systems that can deal with both batch and stream processing and has use cases from social media platforms (such as Twitter, LinkedIn, etc.) over the Internet of Things (smart city, wearables, manufacturing, etc.) to the financial sector (fraud detection, recommendations).[11]



The Lambda Architecture as seen in the picture has three major components.

1. Batch layer that provides the following functionality
   i. managing the master dataset, an immutable, append-only set of raw data
   ii. pre-computing arbitrary query functions, called batch views.
2. Serving layer—This layer indexes the batch views so that they can be queried in ad hoc with low latency.

❖ Speed layer—This layer accommodates all requests that are subject to low latency requirements. Using fast and incremental algorithms, the speed layer deals with recent data only. [13]

VII. SUMMINGBIRD

Hadoop is the standard for batch analytics, although it is mostly accessed via higher-level abstractions such as Scalding and Pig. For on-line processing and real time analytics, Storm has emerged as the standard execution framework. It is clear that Hadoop and Storm both have their place, but merely standardizing on the two processing frameworks does not solve the problem of a developer needing to write everything twice. It would be desirable to have an abstraction for expressing analytical queries that is agnostic to batch or on-line processing, and have a system automatically generate Hadoop jobs or storm topologies as appropriate. Summingbird does exactly this. Summingbird is an open source domain specific language which is implemented with Scala in Twitter to integrate on-line and batch MapReduc e-computations. Summingbird programs are written using data-flow abstractions such as sources, sink and stores. They can run on different execution platforms like Scalding (batch), Storm (on-line) and can also operate in hybrid processing mode.[10]

Summingbird job will produce two types of data: stream and snapshot. Stream contains all of the history of the data, Store is a snapshot of the system at the specified time. Summingbird core through a large number of components to achieve:

- **Producer**- Producer is the Summingbird data stream abstraction, to be passed to a specific MapReduce to do Platform flow compilation. After building up desired MapReduce workflow, hand over Producer to an instance of Platform to compile the MapReduce workflow down to that particular Platform’s notion of a “plan”.

- **Platform-** Platform instance can be used for any stream MapReduce library implementation,that knows how to make sense of the operations one can perform on a Producer. The Summingbird library contains the Storm for Platform, Scalding and memory processing support.
- **Source**- Source represents a source of data, each system has its own definition of the data source, such as Memory Platform defines a Source[T] to be any TraversableOnce[T].
- **Store**- A store represents an abstract model of a key–value store. In the online setting, a store receives partial results that are combined with its present state, and might be backed by memcached, MySQL, or HBase; reasonably low-latency reads and writes are a requirement. In the batch setting, a store contains a snapshot of the aggregated value for each of its keys, which is usually materialized to disk.
- **Sink-**Unlike the Store, the Sink allows you to materialize an unaggregated "stream" representation of the Producer's values. A Sink is a stream, not a snapshot. In Storm and Memory, a Sink is just a function call. In Storm that that function call might populate a log stream, or another realtime queue that a further Summingbird topology could pull in as a Source.
- **Service**- A Service allows the user to perform a "lookup join", or leftJoin, against the current values within a Producer's stream. The joined values can come from another Store's snapshot, another Sink's stream, or even some other asynchronous function call that requires non-negligible time to compute. A Service can also provide access to data from a Store that is being materialized earlier in the job, or in a dependent job. On the Memory or Storm platforms, a Service is a key-value store against which the stream performs lookups. Before a service join, a producer must have type Producer[P, (K, V)].

  The Scalding platform allows you to join against the stream output by another Summingbird job's sink. The join is implemented in a way that prevents any key on the left side of the join from seeing the right side's value at a future point in time.
- **Plan**- The plan is the final representation of the MapReduce flow produced by a Platform after a call to platform.plan(producer). For Storm, the plan is a StormTopology instance that the user can execute using Storm's supplied methods. For the Memory platform, the plan is an in-memory Stream[T] containing the output of the supplied producer. Once you have a plan, you can jump back out to the APIs backing the specific Platform for job customization and submission.[12]

VIII. COMPARISON

Consider the below comparison table. This table compares briefly about the processing models we have mentioned in this paper. Dimension points out exactly what kind of property will this particular solution support. The design column mentions the brief design and programming paradigm behind the data processing technique. Which is mainly, based on the idea which makes the model stand out from the rest. Considering the fact that each model has its own advantages and is mainly designed to handle particular data environment, the last column mentions the advantages of each solution. The Hybrid approach in Big data processing deals with both volume and velocity dimensions of Big data. This type of processing technique helps in getting best out of both worlds. This helps in achieving low latency needed for real time data processing but this may contain errors which can corrected using offline processing of same data. Sparks main advantage over other systems is that it is very fast due to its in-memory computing capabilities. One may argue that this design may not be feasible for Big data analytics with large dataset that may not fit into traditional memory storage systems and need for unrealistic very large memory for its implementation. Spark streaming provides the capability to process streams of data using the same Spark engine. Thus same stack can be made use to perform both types of processing with very low latency. Summingbird library lets you execute same programs on both modes to process streaming data with very low latency using real time mode and provide fault tolerance grantees using batch mode. Summingbird provides high level of reliability by combining batch and associativity technique for production class systems.

Table I – Comparison of difference hybrid programming models

Big Data Solution	**Spark / Spark Streaming**	**Shark**	**Summingbird**
Developer	UC Berkley	UC Berkley	Twitter
Dimension	Volume, Velocity	Volume	Volume or Variety

Programming model	InMemory Computing / Resilient Distributed Datasets, Discretized Streams	In-memory Caching of queries	Batches + Associativity
Advantages	Very Fast Cluster computing for Iterative Jobs and Interactive analysis, Single stack for both modes of processing	Very fast compared to Hive and can process petabytes of data	Same programs can be run on both modes of processing, Reliability

REFERENCES

[1]. Balasubramaniyan Sundaresan, Dinesh Kandavel "Big Languages for Big data A study and comparison of current trend in data processing techniques for Big data" Conf. Design and Implementation of Porgramming Languages Seminar, At TU Darmstadt, (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (1) , 2014, 538-541, [December 05, 2015]

[2]. Oscar Boykin, Sam Ritchie, Ian O'Connell, and Jimmy Lin "Summingbird: A Framework for Integrating Batch and Online MapReduce Computations"Proceedings of the VLDB Endowment VLDB Endowment Hompage archive,San Francisco, California, Volume 7 Issue 13, August 2014 ,Pages 1441-1451, [January 16, 2016]

[3]. Tomislav Lipic , Karolj Skala , Enis Afgan, "Deciphering Big Data Stacks: An Overview of Big Data Tools" pages1-2, 2014, [December 19, 2015]

[4]. Abdul Ghaffar Shoro & Tariq Rahim Soomro, "Big Data Analysis: Ap Spark Perspective" Global Journal of Computer Science and Technology : C Software & Data Engineering Volume 15 Issue 1 Version 1.0 Year 2015 Type: Double Blind Peer Reviewed International Research Journal Publisher: Global Journals Inc. (USA) Online ISSN: 0975-4172 & Print ISSN: 0975-4350, [January 05, 2016]

[5]. Samza Spark Streaming Internet :

https://samza.apache.org/learn/documentation/0.10/comparisons/spark-streaming.html, [January 20, 2016]

[6]. Spark Streaming Programming Guide, Internet : https://spark.apache.org/docs/0.7.3/streaming-programming-guide.html [January 12, 2016]

[7]. Sumit Gupta, "Learning Real-time Processing with Spark Streaming", September 2015. Page no: 162-164, [December 28, 2015]

[8]. Cliff Engle, Antonia Lupher, Reynold Xin, Matei Zaharia, Michael J. Franklin, Scott Shenker, Lon Stoica, "Shark: fast data analysis using coarse-grained distributed memory", SIGMOD '12 Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, ACM New York, NY, USA ©2012, Pages 689-692, ISBN: 978-1-4503-1247-9, [January 12, 2016].

[9]. Antonio Lupher , "Shark: SQL and Analytics with Cost-Based Query Optimization on Coarse-Grained Distributed Memory", Electrical Engineering and Computer Sciences University of California at Berkeley, January 13, 2014, [January 08, 2016]

[10]. Johan Fogelstr ̈om and Remzi Can Aksoy , "Using Summingbird for aggregating eye tracking data to find patterns in images in a multi-user environment" School of Computer Science and Communication (CSC), Royal Institute of Technology KTH, Stockholm, Sweden, [January 16, 2016]

[11]. Michael Hausenblas**,** "Twitter Open-Sources its MapReduce Streaming Framework Summingbird" http://www.infoq.com/news/2014/01/twitter-summingbird, January 16,2014 [January 16,2016]

[12]. Andy Schlaikjer, "Core Concept" of Summingbird, https://github.com/twitter/summingbird/wiki/Core-concepts, October 13,2014, [January 16, 2016]

[13]. Developer Central, https://www.mapr.com/developercentral/lambda-architecture, [January 10, 2016]

[14]. Jimmy Lin, "Monoidify! Monoids as a Design Principle for Efficient MapReduce Algorithms", Cornell University Library, April 29, 2013, [December 10, 2015]

[15]. Rupali Y. Behare #1, Prof. S.S.Dandge**,** "A Database Hadoop Hybrid Approach of Big Data", International Journal of Innovative Research in Computer and Communication Engineering, Vol. 3, Issue 5, May 2015, [December 08, 2015]