Cloud Dataflow using Millwheel and Flume Java

Mrs. Dhanamma Jagli[1] Mr. Sachin Bhandari[2] Ms. Vani Ganji[3]
[1]Assistant Professor [2,3]MCA Student
[1,2,3]Department of MCA
[1,2,3]V.E.S. Institute of Technology, Mumbai, India

Abstract— Cloud DataFlow is a stream analysing tool for cloud storage. It is a system for building big and fast data analysis pipelines. Dataflow is based on few of technologies the company has beenusinginternally,including Flume and MillWheel. This article focuses on these technologies as well as the core components of DataFlow. An example of twitter hashtag auto-completion tool is considered and discussed by implementing it using this technology.

Key words: Pipeline; Sdk; Prefix; Stream; Optimize; Graph; Analysis

I. INTRODUCTION

Google has expanded its Cloud Platform with a new managed service called Cloud Dataflow. It is simple, flexible, streaming analysis tool that allows developer to create data pipelines to help them ingest, transform and analyze data. Developers can use the service to work with streaming real-time data and by uploading batches of data to the system.

Cloud Dataflow is Google's descendant to MapReduce, which has been a new App Engine feature for quite a while now. Google is using Java for the first Cloud Dataflow SDK, but it is also providing a dashboard for monitoring these pipelines from the developer console[1].

According to Google, it is to help its users get "actionable insights from your data while lowering operational costs without the hassles of deploying, maintaining or scaling infrastructure."

The way it works is the user writes a sequence of logical data transformations that are easy to write and very intuitive that specifies the analysis to do[3]. Submit this to the data flow service.And it runs it for the user. It's fully managed. So it runs on as many machines as are necessary in order to run for the pipeline.All the configuration, all the parallelism, all of the tuning, it's all taken care of.

II. OVER VIEW OF MILLWHEEL

The stream processing work in Google DataFlow system is carried out by the MillWheel framework. It was firstly implemented and tested for the Google's Zeitgeist*(now replaced with Hot Trends), a feature offered by Google, offering insights from web searches throughout the year. The framework was designed to achieve the prime objectives of fault tolerance and scalability. The framework improves real-time data processing and analysis. MillWheel incorporates stream and batch processing in one framework. In addition the framework offers users a higher level of abstraction, thereby enabling faster implementations with very few lines of code[5].

A. Expectations from the MillWheel framework:

Following are the requirements by Google for a stream processing framework, which are reflected in MillWheel:

1) Data should be available to consumers as soon as it is published (i.e. there are no system-intrinsic barriers to ingesting inputs and providing output data).
2) Persistent state abstractions should be available to user code, and should be integrated into the system's overall consistency model.
3) Out-of-order data should be handled gracefully by the system.
4) A monotonically increasing low watermark of data timestamps should be computed by the system.
5) Latency should stay constant as the system scales to more machines.
6) The system should provide exactly-once delivery of records. [6]

B. MillWheel Overview [5]:

The MillWheel system, at an abstracted level is a user-defined computation graph working on input data and generating output. The input and output data is represented in trinity(key, value, timestamp). [4]

The key is a metadata field with semantic meaning in the system, the value can be an arbitrary byte string, corresponding to the entire record.

The following are the building blocks of MillWheel framework.

1) Computation:

Computations are the user-defined codes forming the application logic. The code is executed on the arrival of input data, flows through a series of processes and outputs the computation results either to the user or forwards it further into the system. The computations are to be coded without considering the distribution of keys among different machines and in context of a single key[4].

2) Keys:

Keys are used for differentiation between the various records in the system.

3) Streams:

Streams are the delivery channel between various computations in the system.

4) Persistent State:

The user may require short term or long term persistence. The user is required to provide the persistence routines if required any.

5) Low Watermarks:

The low watermark at A, is defined as min(oldest event still in A, low watermark amongst all streams sending to A).[5]

6) Timers:

Timers are triggerable components that are executed at a certain wall time or a low watermark value.

7) The components collectively enable MillWheel to achieve the objectives of scalability, fault tolerance and perform better than its predecessor MapReduce.

III. OVERVIEW OF FLUME JAVA

FlumeJava is merely a Java library that works in collaboration with the MillWheel framework. It provides programming abstraction and optimization, it allows the programmer to form pipelines to the millwheel system. The intermediate routines for building pipelines across various MapReduce systems required a lot of code. FlumeJava reduces these codes and enables easier programming code for the beginners of parallel Programming. This also gives an impression of working on a single system with concrete data sets. The user is exempted from the tedious complications of handling a parallel system[6].

The following are the basic core operations provided by FlumeJava:

1) parallelDo() : executes a parallel operation (i.e. mapping)
2) groupByKey() : groups <key,value> pairs by the "key"
3) combineValues() : combine values to a key(<key, group(values)>
4) flatten() : combines multiple <key, groups> into one data set
5) The following are few derived operations consisting of core operations:
6) count() : number of times certain key appears
7) join() : join two tables on the same key <k,V1> U <k,V2> = tuple<V1,V2>.
8) top() : takes in a comparison function and gives the top result of comparison

Flumejava provides a feature namely Deferred evaluation and Optimization . Once these operations are defined, the library checks whether it has to be carried out locally or need to add the mapReduce functionality to it. FlumeJava analyses the series of operations used and creates a directed graph for it. Then it further organizes and aligns the code to achieve optimized results. Later, the optimizer forms a MapShuffleCombineReduce (MSCR) generalization of MapReduce which it then executes.

The differed Evaluation provides a real time look and feel and yields debugging benefits and the best optimizations.

IV. CLOUD DATAFLOW COMPONENTS

A set of SDKs that you use to define data processing jobs. The Dataflow SDKs feature a unique programming model that simplifies the mechanics of large-scale cloud data processing. You can define your data processing jobs by writing programs using the Dataflow SDKs, such as the Cloud Dataflow Java SDK.[1]

A Google Cloud Platform managed service. The Dataflow service ties together and fully manages several different Google Cloud Platform technologies, such as Google Compute Engine, Google Cloud Storage, and BigQuery to execute data processing jobs on Google Cloud Platform resources.[1]

Let's look into each of these components in brief.

A. Dataflow java sdk:

The java sdk allows the user to program in a simple and elegant way using the pipelines as the programming model .It provides the basic primitives for handling data processing, reading, writing and transforming the data. The user can use the available primitives, pipelines and methods for programming the application.

B. Google Compute Engine:

Google provides a service called compute engine that provides a platform for using virtual machines. It provides scaling, performance, power and value of using up to 1000 CPUs thus providing a base for strong computing environment . Like all cloud services it uses the pay as you go model for charging the user.

C. The main features are:

Creation of Virtual Machines with variety of configurations
Maintenance and storage of data in block storage
Maintaining the Network access of the Virtual Machines
Availability of tools and Oauth2.0 authentication for your Virtual Machines

D. Google Data Storage:

Google provides a cloud storage platform for maintaining and storing your data on the cloud. It provides a reliable, highly available, and greater performance. It allows the user to pick from any of the 3 available models as per their requirements.

E. Standard Storage:

It provides the highest level of availability and durability among all the three storage services. It is specifically designed for the applications where low latency and frequent data access is required

F. Durable Reduced Availability Storage:

DRA has the same durability and performance as the Standard storage but is lower in terms of cost and availability.

G. Nearline Storage:

Nearline storage is designed for storing backup data, archived files and disaster recovery. Unlike other time consuming storage devices for archived data, nearline storage provides a faster access time

V. BIGQUERY

Querying massive data sets can be quite a challenge. This task becomes very efficient with google Bigquery. Since querying against big datasets can be quite a task, and will require appropriate hardware and the infrastructure, these requirements are fulfilled by google's infrastructure. The user needs to move the data into bigquery, and the rest is handled by them[11]. The user can access the data using the Web UI or a command line tool

Google tells BigQuery is complementary to Dataflow. Developers can use Dataflow as a part of the data ingestions into BigQuery, for example, by preparing or filtering the data for BigQuery. Once the data is cleaned, it can be written to BigQuery, where it becomes immediately accessible. At the same time, though, Dataflow can be used to read from BigQuery in case you want to join data from

your database with other data sources. And to complete the cycle, you can then write all of this back to BigQuery, too, of course. [3]

VI. IMPLEMENTATION WITH EXAMPLE

A feature that from many websites out there is auto completing words. Google does this. As you start typing a search, we auto complete and give search suggestions. And also Twitter does this. If you start typing in a hash tag, Twitter will automatically start suggesting hashtags that you might actually be trying to type. So what does it take to build an application like this? And let's focus now on this Twitter example of auto completing hash tags.

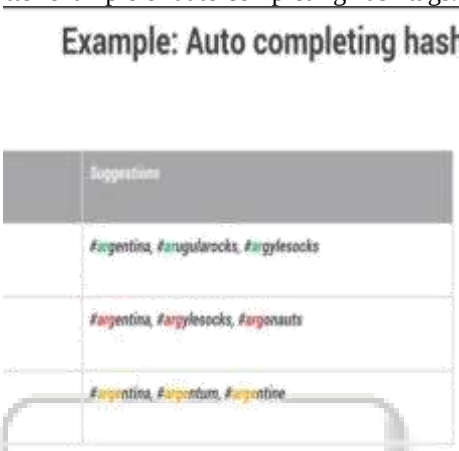

Fig. 1: Auto Completing Hashtag

A front end displays the above image and it is all about displaying it well and intuitively and making it easy for the user to use. So if the user has typed in "ar," "arg," "arge." then we show these suggestions for the above prefixes. And as the user types in more and more characters we've colored the prefix that they've typed, the suggestions keep refining, and this is the model we want to build.

So what we need to do, is build a prefix model that will show the users prefixes of hashtags that they want until they've typed the entire hashtag in an increasing prefix way.

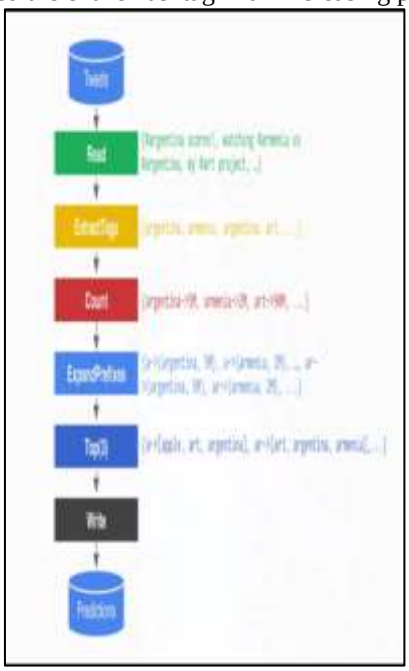

Fig. 2: Dataflow Pattern

The point where you want to run this on your actual big data at scale that's where Dataflow comes in. So you take your code, and now instead of running it locally on your machine, you submit it to the Google Cloud Dataflow Service[6]. Once it's submitted to the Google Cloud Dataflow Service, it starts running and processing all the data.

The Dataflow service also gives a monitoring console, which allows monitoring this data while it's processing. It lets you see progress of your data and helps debug problems in your pipeline as they happen.This is a simple and intuitive monitoring console that matches one to one to the lines of code you wrote.

Fig. 3: Hashtag Count

It doesn't show you these big blocks saying, something is wrong in your pipeline[3]. It rather shows you, what's happening with your extract tags, what's happening with your account, what's happening with your expand prefixes and so on.

The optimizer takes in the code you've written, determines which lines of code can actually run together as one group, rearranges things a little bit into something that can run efficiently, and produces this optimized program on the right here to actually execute. Older systems actually didn't provide an optimizer. So when you use those systems, you usually had to hand optimise your code, for example Mapreduce.

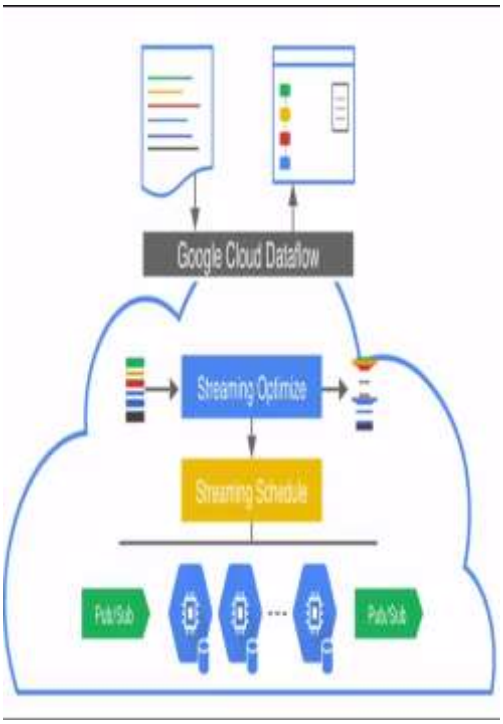

Fig. 4: Dataflow Architecture

To read it from a streaming source providing tweets we use Google Cloud Pub/Sub , it is a new product from Google that just went into limited preview that provides a reliable many to many message delivery channel[2]. And here it integrates with Dataflow in order to constantly stream data in and out of a pipeline, essentially to create a data flow pipeline, a data analysis pipeline that never finishes, an infinite pipeline tracking trends in real time.

VII. FEATURES

A. Unified Programming Model

Cloud Dataflow provides unified programming primitives for both batch and stream-based data analysis. Powerful windowing semantics enable intuitive temporal processing patterns that address a wide range of data processing scenarios, like session analysis, anomaly detection, and funnel analysis.

B. Managed Scaling

As a managed service, Cloud Dataflow fully manages the lifecycle of required compute resources, in order to reduce the burden related to resource management and cluster operations. Cloud Dataflow can horizontally auto-scale compute resources to achieve the needed throughput level and can automatically re-partition work to optimize resource utilization.

C. Reliable & Consistent Processing

Cloud Dataflow provides built-in support for fault-tolerant execution that is consistent and correct regardless of data size, cluster size, processing pattern or pipeline complexity[8]. Developers can focus on writing business logic instead of handling control plane exceptions from hardware and network failures, or tuning execution to accomodate inputs[12].

D. Open Source

Google has made the Java-based Cloud Dataflow SDK available as open source. This SDK allows the Cloud Dataflow programming model to be widely used, so that all developers can benefit from the productivity of writing simple and extensible data processing pipelines that can describe both stream and batch processing tasks.

E. Built For the Cloud

From the ground up, Cloud Dataflow is built on and for the cloud. Cloud Dataflow workers run on stock Google Compute Engineinstances, providing developers with an operationally familiar and cost-effective compute environment. Cloud Dataflow integrates with Cloud Storage, Cloud Pub/Sub and BigQuery for seamless data processing[12].

F. Monitoring

Integrated into the Google Developers Console, Cloud Dataflow provides statistics such as pipeline throughput and lag, as well as worker log inspection—all in near-real time. The monitoring console mirrors the processing logic of the pipeline, enabling developers to easily understand pipeline execution

VIII. CONCLUSION

Usually the challenge to develop a real time streaming application is the amount of data that one needs to deal with. But now this is all possible because of technologies like DataFlow . We are talking about doing true streaming analysis, not micro batch or being limited by specific amounts of data[11]. We can actually go to very large data as much as you like with this solution so with that said, if you haven't been thinking about big data before, maybe it's time to think bigger. And it's time to think of the Cloud and how it's limitless and makes things a lot easier. And because they are much easier, naturally it's much faster to develop and also, you could start dealing with data types you were not naturally using before because of the complexity of the technology. But now it makes things a lot easier to include it in your application and collaborate innovation into the application.

REFERENCES

[1] https://cloud.google.com/dataflow/java-sdk/what-is-cloud-dataflow-java-sdk
[2] https://umnbigdata2012.wordpress.com/2012/09/20/flumejava-by-google/
[3] http://techcrunch.com/2014/06/25/google-launches-cloud-dataflow-a-managed-data-processing-service/
[4] MillWheel: Fault-Tolerant Stream Processing at Internet Scale. url: http://static.googleusercontent.com/media/research.google.com/en//pubs/archive/41378.pdf
[5] http://the-paper-trail.org/blog/paper-notes-stream-processing-at-google-with-millwheel/
[6] MillWheel: Fault-Tolerant Stream Processing at Internet Scale. url: http://static.googleusercontent.com/media/research.google.com/en//pubs/archive/41378.pdf
[7] https://cloud.google.com/storage/
[8] https://cloud.google.com/compute/docs/

[9] https://cloud.google.com/bigquery/what-is-bigquery
[10] http://www.theregister.co.uk/2014/06/25/google_cloud_platform/
[11] http://texploration.wordpress.com/2014/06/26/google-dataflow-service-to-fight-against-amazon-kinesis/
[12] https://cloud.google.com/dataflow/model/programming-model

