International Journal of Engineering Technology Science and Research
IJETSR
www.ijetsr.com
ISSN 2394 – 3386
Volume 4, Issue 11
November 2017

An Outline of Real Time Stream Processing Frameworks in Big Data

Kusumalatha Karre[1]
Sreenidhi Institute of Science and Technology, Hyderabad
Pinjala Srinivasa Rao[2]
ACE Engineering College, Hyderabad

ABSTRACT:
Now-a-days data is growing at a relatively high speed. The data is generated from social media, Health, infrastructure, educational and many more domains. It results in creation of data that is very big i.e Big Data which is generated at high velocity, has huge volume and variety. To handle such data, techniques such as Hadoop and map-reduce are used. But map-reduce can be used for batch processing which cannot be efficient to handle real time streaming data. So, we need other tools and techniques to handle the real stream processing data. In this paper, we discuss some real time stream processing frame works such as Apache s4, STORM ,SAMZA and SAMOA.

KEYWORDS: BIG DATA, STREAM PROCESSING, Apache s4 ,STORM, SAMZA and SAMOA.

1.INTRODUCTION:

Big Data:

Big Data is a new term used to identify the datasets that due to their large size, we cannot manage them with the typical data mining software tools. Instead of defining "Big Data" as datasets of a concrete large size, for example in the order of magnitude of petabytes, the definition is related to the fact that the dataset is too big to be managed without using new algorithms or technologies.[2]

As the growth of Internet, Cloud Computing, Mobile Network and Internet of Things is increasing rapidly, Big Data is becoming a hot-spot in recent years. Big Data Processing is involved in our daily life such as mobile devices, RFID and wireless sensors, which aims at dealing with billions of users' interactive data. At the same time, real-time processing is eagerly needed in integrated system.[1] .

Map –reduce:

The most popular distributed systems used nowadays are based in the map-reduce framework. The map-reduce methodology started in Google, as a way to perform crawling of the web in a faster way. Hadoop is a open-source implementation of map-reduce started in Yahoo! and is being used in many non- streaming big data analysis.The map-reduce model divides algorithms in two main steps: map and reduce, inspired in ideas in functional programming.The input data is split into several datasets and each split is send to a mapper, that will transform the data.The output of the mappers will be combined in reducers ,that will produce the final output of the algorithm. This approach is mainly used for Batch processing.

Stream processing:

Stream processing means processing data record by record as they arrive and incrementally updating all results with each and every new data record. Therefore each updated result is available is available in real-time, typically with a latency of a few milliseconds or less. Stream processing queries run continuously, never ending, processing data as they arrive, usually over time or record-based windows.

Streaming data analysis in real time is becoming the fastest and most efficient way to obtain useful knowledge from what is happening now, allowing organizations to react quickly when problems appear or to detect new

International Journal of Engineering Technology Science and Research
IJETSR
www.ijetsr.com
ISSN 2394 – 3386
Volume 4, Issue 11
November 2017

trends helping to improve their performance. Evolving data streams are contributing to the growth of data created over the last few years. We are creating the same quantity of data every two days, as we created from the dawn of time up until 2003. Such data streams cannot be processed using Hadoop and map-reduce. We need different techniques to handle and process real time streaming data. In the following sections we are discussing the frameworks which are purely meant for stream processing.

2. Apache S4:

Apache S4 : platform for processing continuous data streams. S4 is designed specifically for managing data streams. S4 apps are designed combining streams and processing elements in real time.

S4 stands for SSSS (**S**imple **S**calable **S**treaming **S**ystem). Its characteristics are

- For data processing
- Aims to hide processing complexity
- A platform for data processing development
- Distributed
- Scalable
- Fault Tolerant
- High performance

S4 is a general-purpose, near real-time, distributed, decentralized, scalable, event-driven, modular platform that allows programmers to easily implement applications for processing continuous unbounded streams of data.

It grounds on the same concepts (partitioning inspired by map-reduce, actors-like distribution model), but with the following objectives:

- cleaner and simpler API
- robust configuration through statically defined modules
- cleaner architecture
- robust codebase
- easier to develop S4 apps, to test, and to use the platform

Features:

- TCP-based communications
- state recovery through a flexible check pointing mechanism
- inter-cluster/app communications through a pub-sub model
- dynamic application deployment
- toolset for easily starting S4 nodes, testing, packaging, deploying and monitoring S4 apps

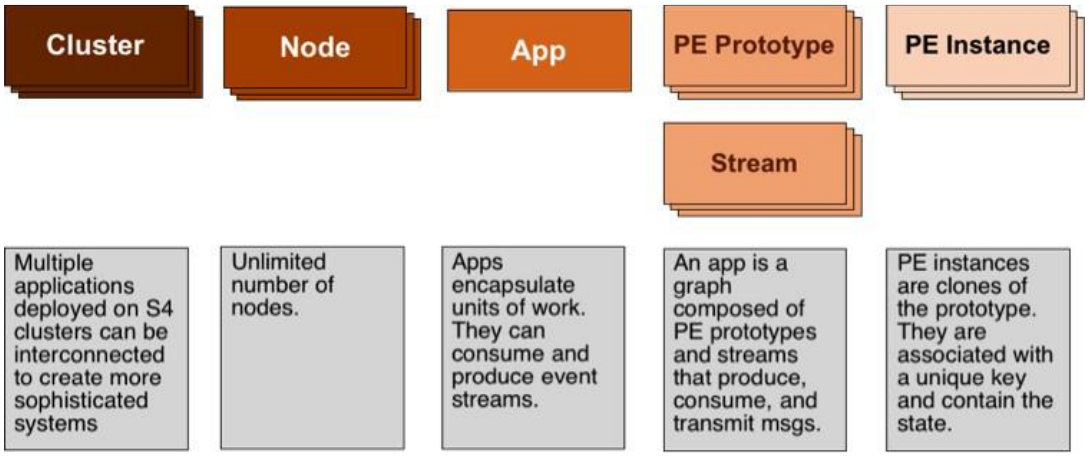

FIG:2.1 Architecture of Apache S4

International Journal of Engineering Technology Science and Research
IJETSR
www.ijetsr.com
ISSN 2394 – 3386
Volume 4, Issue 11
November 2017

In S4, computation is performed by Processing Elements (PEs) and messages are transmitted between them in the form of data events. The state of each PE is inaccessible to other PEs; event emission and consumption is the only mode of interaction between PEs. The framework provides the capability to route events to appropriate PEs and to create new instances of PEs. These aspects of the design provide the properties of encapsulation and location transparency.

3.Apache Storm:

The core part of the Big Data Real-time Processing System is Storm, which is a distributed and fault-tolerance real-time computing system according to Eclipse Public License 1.0. Strom can easily compile and expand complicated real-time computation in a computer cluster, just like what Hadoop does in batch processing. Storm can guarantee that each message will be rapidly disposed. Moreover, any program language can be used for development.

The reason for choosing Storm as our main processing system including several reasons:

- Simple programming model
- A service framework support hot deployment
- High fault-tolerance: Storm will manage the errors happened in working procedure and nodes
- Horizontal scaling: computations are done in parallel
- Reliable message handling
- "Local Mode" simulation

The Storm Cluster is made up of a main node and several working nodes. A daemon process called "Nimbus" is running on main node, in order to allocate codes, arrange tasks and detect errors. Each working node has a daemon process called "Supervisor" to monitor, start and stop working process. The coordination work between Nimbus and Supervisor is handled by "Zookeeper" as shown in Fig.3. Zookeeper is the subproject of Hadoop, and it aims at coordinate works in large-scale distribution system. The Storm Cluster is similar with Hadoop, where Nimbus corresponds to Job Tracker, and Supervisors corresponds to Task Trackers.

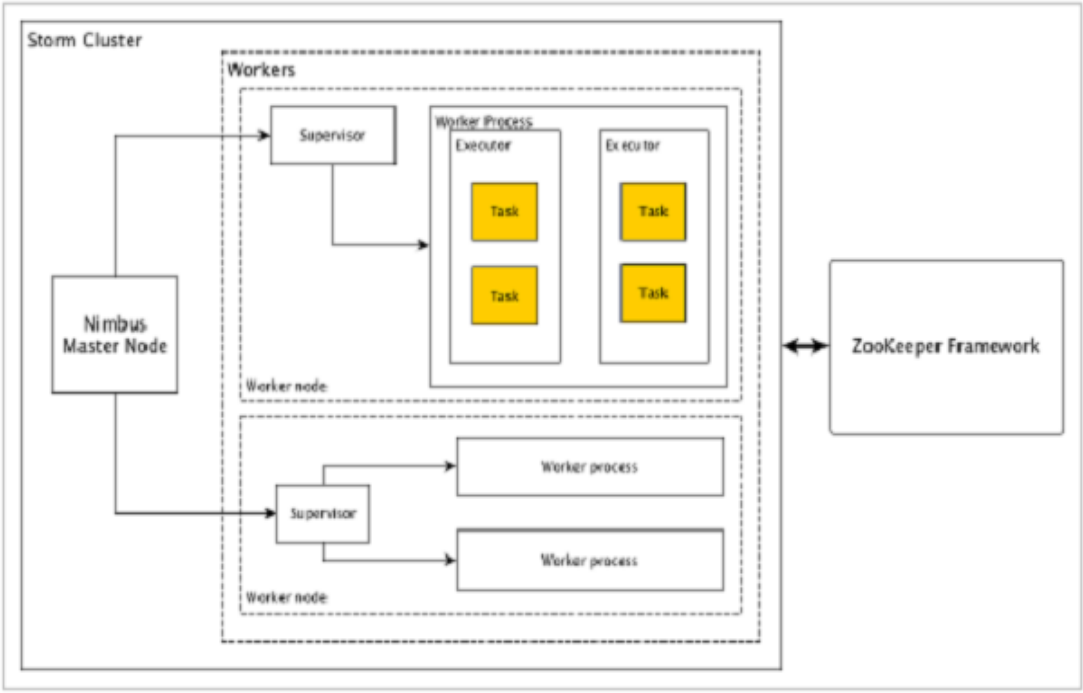

Fig:3.1 Architecture of STORM

International Journal of Engineering Technology Science and Research
IJETSR
www.ijetsr.com
ISSN 2394 – 3386
Volume 4, Issue 11
November 2017

unbounded sequence of tuples is a "Stream" as in Fig.4, which is the basement of the transmission in Storm. The source where streams come from is a "Spout", just like a water tap. The Spouts can talk with queues, web logs, API calls and event data.

In order to realize real-time computation on Storm, "topologies" should be created inside of it. Topology is a graph of computation consists of processing logic nodes and Spouts, and the way how nodes and Spouts are connected indicates how data should be passed around between these nodes. The processing logic nodes are called "Bolts", and there is an example of one topology in Fig.5. Data streams are generated from Spouts, and bolts consumes any number of streams, do some processing settled by user, or export some new streams. The links between the Bolts and Spouts show how streams are passing around, and to decide topology is to set up what the system will do in actual environment.

4. Apache Samza:

Apache Samza is another distributed stream processing framework. Samza is built on Apache Kafka for messaging and YARN for cluster resource management. Its website provides the following overview of Samza:

- Simple API: Unlike most low-level messaging system APIs, Samza provides a very simple callback-based "process message" API comparable to MapReduce.
- Managed state: Samza manages snapshotting and restoration of a stream processor's state. When the processor is restarted, Samza restores its state to a consistent snapshot. Samza is built to handle large amounts of state (many gigabytes per partition).
- Fault tolerance: Whenever a machine in the cluster fails, Samza works with YARN to transparently migrate your tasks to another machine.
- Durability: Samza uses Kafka to guarantee that messages are processed in the order they were written to a partition, and that no messages are ever lost.
- Scalability: Samza is partitioned and distributed at every level. Kafka provides ordered, partitioned, replayable, fault-tolerant streams. YARN provides a distributed environment for Samza containers to run in.
- Pluggable: Though Samza works out of the box with Kafka and YARN, Samza provides a pluggable API that lets you run Samza with other messaging systems and execution environments.
- Processor isolation: Samza works with Apache YARN, which supports Hadoop's security model, and resource isolation through Linux CGroups.

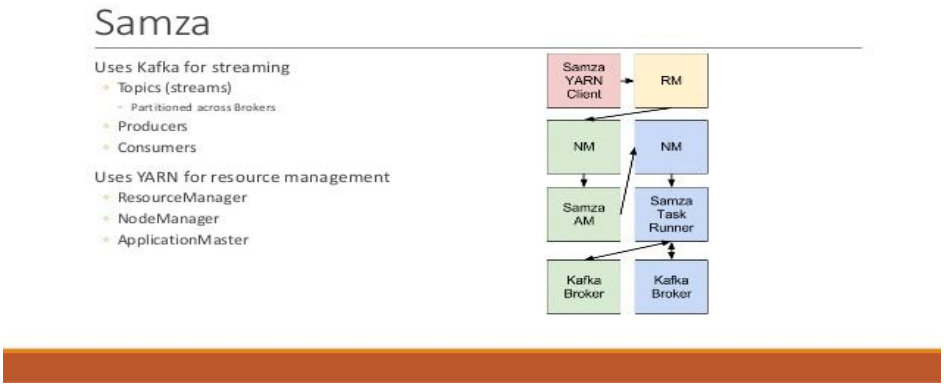

Fig 4.1: Architecture of SAMZA

International Journal of Engineering Technology Science and Research
IJETSR
www.ijetsr.com
ISSN 2394 – 3386
Volume 4, Issue 11
November 2017

5.SAMOA:

SAMOA is both a platform and a library of algorithms for big data stream mining and machine learning. It features a pluggable architecture that allows to run on top of several distributed SPEs. This capability is achieved by designing a minimal API that captures the essence of modern distributed SPEs. Initial support is provided for S4 and Storm, but bindings for new systems can be added easily. samoa takes care of hiding all the differences of the underlying SPEs in terms of API, model of computation and deployment. samoa supports the most common machine learning tasks such as classification and clustering. It includes distributed versions of classical streaming algorithms such as Hoeffding decision trees and k-means-based clustering. samoa also provides a simplified API for the algorithm developer.

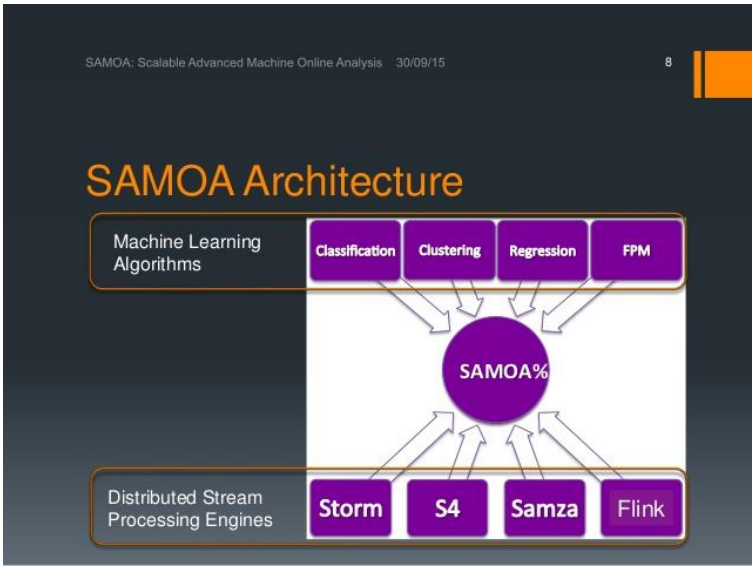

Fig5.1: Architecture of SAMOA

6.CONCLUSION:

Data is growing at a high speed thereby resulting in Big Data which is of huge volume generated at a high velocity and comprising of various types of information. To handle such data we need stream processing. A few frameworks for stream processing are discussed and we can use any one of them based on our need and methodology to handle and analyze the data.

REFERENCES:

[1] **Big Data Real-time Processing Based on Storm by** Wenjie Yang, Xingang Liu* and Lan Zhang and Laurence T. Yang in 2013 at 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications.

[2] **Mining big data: current status, and forecast to the future** W Fan, A Bifet - ACM sIGKDD Explorations Newsletter, 2013 - dl.acm.org

[3] **SAMOA: A platform for mining big data streams** G De Francisci Morales - … of the 22nd International Conference on World …, 2013 - dl.acm.org

[4] **S4: Distributed stream computing platform** L Neumeyer, B Robbins, A Nair… - Data Mining Workshops …, 2010 - ieeexplore.ieee.org

[5] M. A. Beyer and D. Laney, **"The Importance of 'Big Data'**: A Definition," Jun. 2012.

[6] L. Neumeyer, B. Robbins, A. Nair, and A. Kesari. **S4:Distributed Stream Computing Platform**. In ICDM Workshops, pages 170–177, 2010.

[7] R. K. Karmani, A. Shali, and G. Agha, "Actor **frameworks for the JVM platform: a comparative analysis,"** in PPPJ '09: Proceedings of the 7th International Conference on Principles and Practice of Programming in Java. New York, NY, USA: ACM, 2009, pp. 11–20.