An Improved Stream Processing Access

T.Swathi, N.Kasiviswanath, M.Padma

Abstract— Migration of Legacy applications into modern Cloud, IOT architecture are challenging tasks and many researchers are showing interest to build modern Real time cloud and IOT based applications like smart cities, Video mining, Health care, Industrial event monitoring and many more for modern human life. Such applications should require efficient online data streaming techniques to process large amount of unstructured online data streams instead of offline. Modern customer centric applications with different verticals are looking for distributed and horizontal data streaming approaches. Many real time streaming approaches are emerging to utilize or process large real-time data by replacing legacy centralized scenarios which are causing more memory utilization, delay and fault tolerance. In this paper we present common models and architectures for real time utilization of cloud and IoT based application stream processing. Utilization of the real-time data of IoT/Cloud applications are possible with collective streaming techniques of network, data processing. In this paper we are focusing on improving stream processing techniques, limitations and future research directions for real-time stream processing.

Keywords— Big Data, Big Data Processing, Stream Processing, IoT.

I. INTRODUCTION

Big Data has become a popular term which is used to describe the exponential growth and availability of data. As a reason there is a demand for large-scale data processing and data analysis applications to handle large amount of data. NIST defines big data as “The inability of traditional data architectures to efficiently handle the new datasets”.

The Characteristics of Big Data according to NIST are defined as

- Volume (the size of the dataset)
- Variety (data from multiple repositories, domains)
- Velocity (rate of flow)
- Variability (the change in other characteristics)

Big data is classified on five aspects

- Data sources : The generation of data from different sources.
- Content format : Format of data like image, text, mp3 etc
- Data stores: The storage of data in different file systems
- Data staging: An intermediate stage for processing data.
- Data Processing: where data is processing for further usage.

Revised Manuscript Received on September 10, 2019.

T.Swathi, Assistant Professor, CSE Dept., G Pulla Reddy Engineering College, Kurnool, Andhra Pradesh, India.
(E-mail: swa1041@gmail.com)
N.Kasiviswanath, Assistant Professor, CSE Dept., G Pulla Reddy Engineering College, Kurnool, Andhra Pradesh, India.
(E-mail: hodcse@gprec.ac.in)
M.Padma, Professor & HOD, CSE Dept., G Pulla Reddy Engineering College, Kurnool, Andhra Pradesh, India.
(E-mail: padma.gprec@gmail.com)

The rest of the research survey paper organized into three sections, Section II explains the background and survey work of the research topic in detail. The Survey summary and performance evaluation in III. The Proposed architecture in IV. The conclusion and future directions of the work in section V.

II. BACKGROUND AND SURVEY

Big data is of different types basically dependent on how they are generated and how they are used. The processing types can be as [3]

I. Batch processing
II. Stream processing
III. Graph processing
IV. Machine learning processing

Difference between batch and stream processing can be given as follows[8]

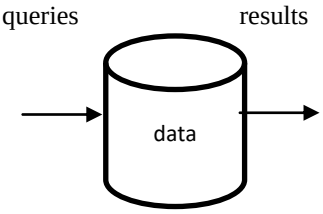


fig:1.1 Batch processing system

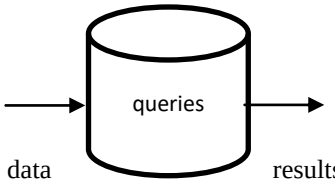


fig:1.2 Stream processing system

- In batch processing data is stored and indexed and then processed but stream processing takes inbound data while it is in flight.
- Stream processing is designed to analyze and act on real-time data .
- In batch data is collected stored and analyzed. In this users are in active state and data analytics and processing systems is in passive state.
- In stream the users are passive and processing systems are active.

Data streams can differ across domains, but in general, it is commonly considered as input data that arrives at a high rate, and continuously at different speeds. As a result there is a great demand for communication and computing infrastructure. The data types in a stream may vary according to the applications, including discrete signals,

event logs, monitoring information, time series data, audio, video, among others. It is also important to distinguish between streaming data when it arrives at the processing system for instance, a log or queuing system, and intermediate streams of tuples resulting from the processing by system elements.

Data stream is a collection of unbounded data elements (data packets) generated in time. Modern society are monitored or manage by various devices like sensors, actuator, cameras which generates real time high volume of environmental data and send as data streams to Cloud / IoT Platforms. Once the data received by stream processing system the streaming system then do the validations like[9]:

- What type of data received by stream ex: .CSV, JSON, HDFS, …?
- How to process the stream valuable data?
- How to utilize the real-time data for multiple applications through various channels.

Robust streaming techniques has to process real time data without delay and should be able to forecast at application layer. IoT applications like the business specific and content curation will vary application to application. Designing and implementation of such streaming engine should have capabilities like stream partitioning, Query mobility, availability, storage. In this paper we discussed about the various existing steaming platforms and their capabilities.

A stream processing architecture can comprise multiple tiers of collecting and processing data, with the connection between these tiers. The following architecture [6]is a modern stream processing architecture with different layers like data collection from different layers and analysis at top layer.

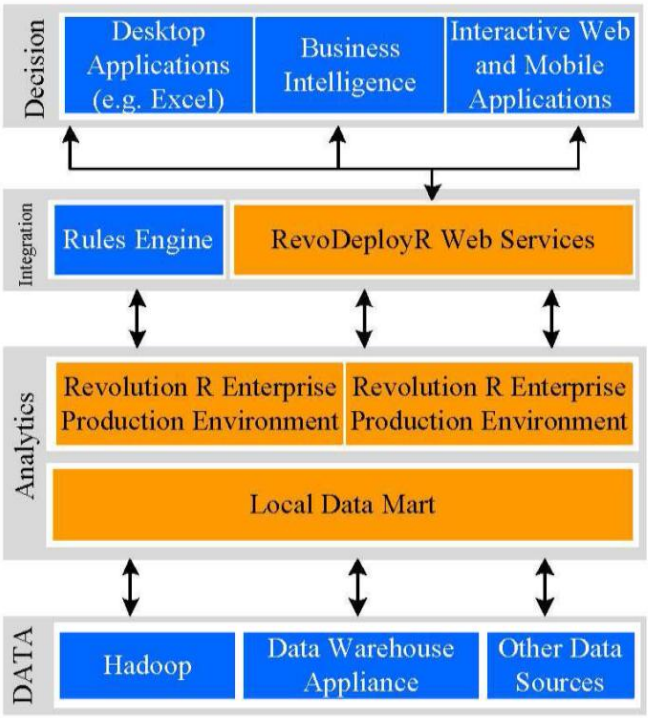
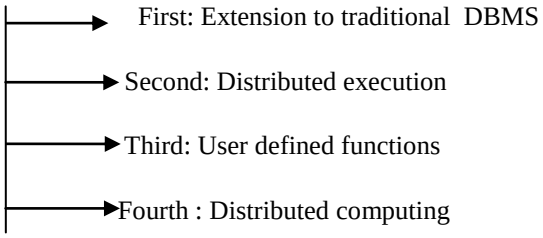


fig: 2.1 Framework of stream processing system

The generations of stream processing engines are summarized as follows. At present the fourth generation techniques are following[8].

Generations of stream processing engines



MapReduce [16] process the larger data sets and also works like a programming model by itself. Lagging distributed space to main scalability of the connected devices is one of the demerits of this system. MIT and Brown University jointly implemented Aurora [17] handles single site location data streaming, whereas Aurora seamlessly working with medusa to fulfill distributed streaming capabilities. SPADE [14] fulfill the stream scheduling and synchronization process for distributed systems. StreamIt [16] is programming language provides robust API (Filters, Connecting Filters, re-initialization, latency) to increase programmer productivity towards efficient streaming. Borealis [20] is the enhance version of Aurora supports distributed stream processing with core modules of both medusa and aurora. It allows to generate dynamic query results and make modifications. Modern distributed streaming systems supports internet-based cloud / IoT applications, Apache S4, Apache Storm [11], Apache Samza [14], Spark Streaming [13], Twitter's Heron [14], Neptune [15]. Some of the Commercial stream engines are developed by Microsoft and Amazon, such as Google Millwheel [16], Azure Stream Analytics [17], The typical architecture of modern distributed streaming engines illustrated in following figure[20].

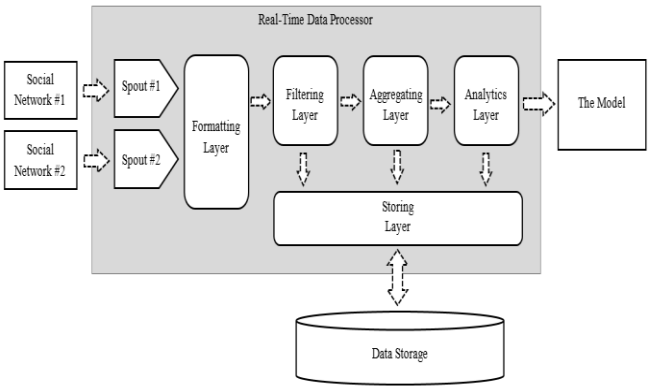


fig:2.2 Distributed stream engine.

Figure.1. Architecture of Distributed Real time Stream Processing Engines

III PERFORMANCE EVALUATION OF STREAM TECHNOLOGIES

Feature / Functions	**Storm**	**Spark**	**Flink**	**Samza**	**SGD(Stochastic Gradient Decent)**	**AWS-AML**	**Jubatus**
Programming	Clojure	Scala	Java	Scala	--	--	
Event Size	Mini-Batch	Micro Batch	Single	Single			Batch
Reliability	Medium but High in Strom+ Trident		High	High	Medium	High	High
Stream File							JSON
Primitives							
State	Record ACK's	Check Points	Distributed Snapshots	Local and Distributed Snapshots (Fault Tolerance)			
Messaging							
Resource	YARN Mesos	YARN Mesos	YARN	YARN			YARN
Fault Tolerance	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Back-Pressure	Low	Low	Yes	Yes			
Latency	Very Low	Medium	Low (Configurable)	Low			Low

We evaluated these modern distributed streaming technologies with various performance metrics and presented their presence and non-presence of activities. Table .1. illustrates the detailed performance evaluation of the streaming technologies as follows:

IV DISCUSSION & RESULTS

The proposed system mainly contains 3 parts, namely Connected devices (Non-TCP devices), Standard devices (Laptops, gateways, etc.,), Cloud environment. Once the stream data entered in to the Cloud environment data is processed in 5 steps, likely ingest, pipelines, Storage and application & presentation layers. Cloud Pub/Sub or commercial messaging system (Kafka) will publish and subscribe the information along with monitoring and logging services. In Pipelines stage cloud dataflow service will be used to process the data, GCP (Google Cloud Platform) offers Cloud Dataflow which is a custom implementation with open source Apache Beam. In the process the next stage is sub categorized as storage and analytics. Cloud storage have 3 services, Cloud storage (Amazon s3) offers BLOB storage type data and Bigtable offer HIVE based data. Bigtable allows to run basic SQL queries. Cloud data store is a SQL system allows to store small amount of information. Stream data analytics is used to visualize the data by Cloud DataProc (library designed by MapReduce and Apache spark). Cloud data lab adopts machine learning approach useful for data analysis which is the last and final stage. Application and presentation engine and Container engine are two different services useful to perform high-end computations for container application offered by Google Cloud Functions.

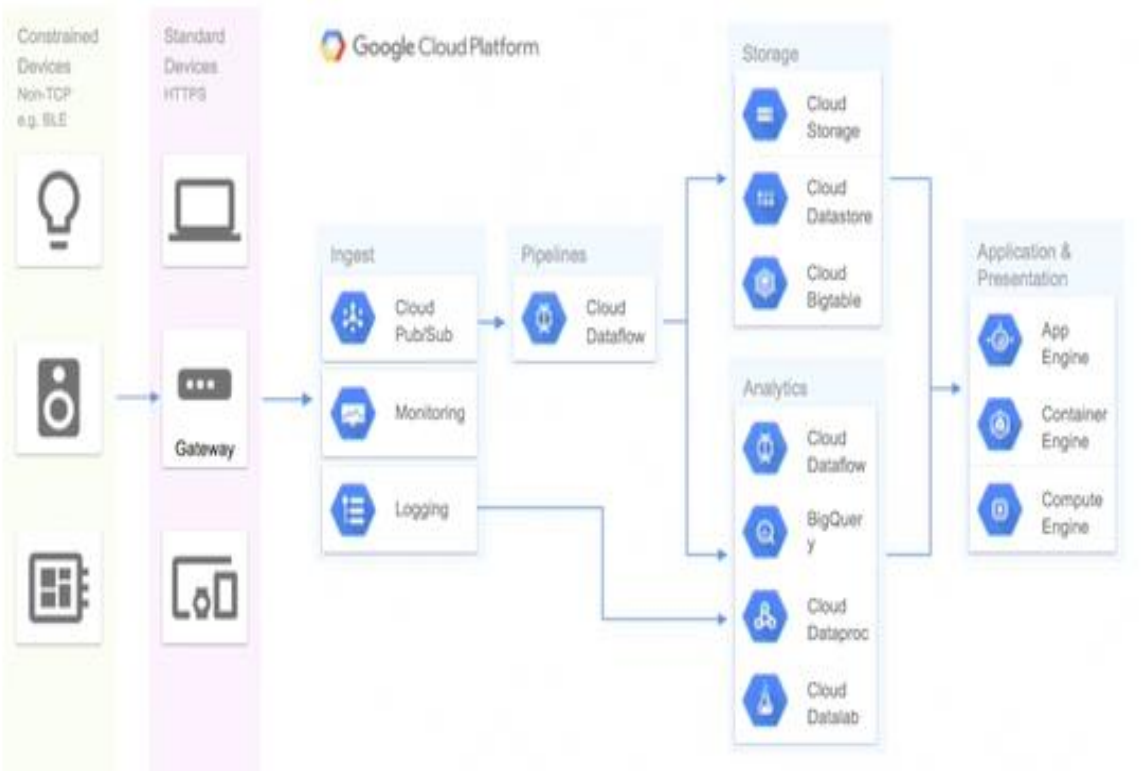


Fig.4.1 A new stream processing system

V CONCLUSION & OPEN RESEARCH CHALLENGES

In this paper we identified, understand and examined the modern distributed stream processing technologies for processing real-time streams of Cloud and IoT based applications. We examined message system, how seamlessly work with streaming technologies/platforms. The legacy or offline streaming techniques can't able to handle massive data generated by trillions of devices connect with large scale IoT Cloud based applications. On designing and developing a new stream processing technique is useful to save the delay in processing, rollback, scheduling, portioning, etc. Novel distributed stream processing techniques are capable to achieve good throughput, i.e. around 1.2 million of messages per second /node with 75% of average utilization of CPU Cost. From the survey we identified that the existing streaming platform are not offering full-fledged API to survey better for data driven society in terms of real time data utilization. Some of the API's are lagging in terms of backpressure, flow control and time stamps, fault tolerance, state and memory utilization, stream partitioning and so on. Implementation of such API are interesting to build modern enterprise landscapes.

Open Research Challenges

Each distributed stream processing has unique capabilities, early and modern distributed systems fulfil the requirement such as scheduling, synchronization, clustering, partitioning, mobility, State, Stream formats, Latency, etc. Samza, Flink, Millwheel and Lambda resolves many issue and some of the open challenges are:

1. Handling massive site(Sources) along with Parallel processing
2. CPU Cost and Resource allocation/Utilization.
3. Design and implementation of On-Demand Profiling(ODP) to increase service capability.
4. Asynchronous Processing and Multithreading.
5. Event based CDC (Change data capture) to address Resource contention.

Optimizing Reprocessing overhead

VI. REFERENCES

1. Wu, Dongyao, et al. "HDM: A Composable Framework for Big Data Processing." IEEE Transactions on Big Data 4.2 (2018): 150-163.
2. de Assuncao, Marcos Dias, Alexandre da Silva Veith, and Rajkumar Buyya. "Distributed data stream processing and edge computing: A survey on resource elasticity and future directions." Journal of Network and Computer Applications103 (2018): 1-17.
3. Sarnovsky, Martin, Peter Bednar, and Miroslav Smatana. "Big Data Processing and Analytics Platform Architecture for Process Industry Factories." Big Data and Cognitive Computing 2.1 (2018): 3.
4. Gao, Kun, and Yiwei Zhu. "Deep Data Stream Analysis Model and Algorithm With Memory Mechanism." IEEE Access 5 (2017): 84-93.
5. Yu, Weiren, et al. "Ring: Real-time emerging anomaly monitoring system over text streams." IEEE Transactions on Big Data (2017).
6. Ounacer, Soumaya, et al. "A New Architecture for Real Time Data Stream Processing." INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS8.11 (2017): 44-51.
7. Krawczyk, Bartosz, et al. "Ensemble learning for data stream analysis: A survey." Information Fusion 37 (2017): 132-156.
8. Batyuk, Anatoliy, and Volodymyr Voityshyn. "Apache storm based on topology for real-time processing of streaming data from social networks." Data Stream Mining & Processing (DSMP), IEEE First International Conference on. IEEE, 2016.
9. Ishizuka, Yuji, Wuhui Chen, and Incheon Paik. "Workflow Transformation for Real-Time Big Data Processing." Big Data (BigData Congress), 2016 IEEE International Congress on. IEEE, 2016.
10. Namiot, Dmitry. "On big data stream processing." International Journal of Open Information Technologies 3.8 (2015).
11. Zheng, Zhigao, et al. "Real-time big data processing framework: challenges and solutions." Applied Mathematics & Information Sciences 9.6 (2015): 3169.
12. Ranjitha, P. "Streaming analytics over real-time big data." Global Journal of Computer Science and Technology (2015).
13. Krempl, Georg, et al. "Open challenges for data stream mining research." ACM SIGKDD explorations newsletter 16.1 (2014): 1-10.
14. Abadi, Daniel J., et al. "The design of the borealis stream processing engine." Cidr. Vol. 5. No. 2005. 2005.
15. Arasu, Arvind, et al. "STREAM: the stanford stream data manager (demonstration description)." Proceedings of the 2003 ACM SIGMOD international conference on Management of data. ACM, 2003.
16. Chandrasekaran, Sirish, et al. "TelegraphCQ: continuous dataflow processing." Proceedings of the 2003 ACM SIGMOD international conference on Management of data. ACM, 2003.
17. Abadi, Daniel J., et al. "Aurora: a new model and architecture for data stream management." the VLDB Journal 12.2 (2003): 120-139.
18. Hashem, Ibrahim Abaker Targio, Ibrar Yaqoob, Nor Badrul Anuar, Salimah Mokhtar, Abdullah Gani, and Samee Ullah Khan. "The rise of “big data” on cloud computing: Review and open research issues." Information Systems47 (2015): 98-115.
19. Yadranjiaghdam, Babak, Nathan Pool, and Nasseh Tabrizi. "A Survey on Real-Time Big Data Analytics: Applications and Tools." Computational Science and Computational Intelligence (CSCI), 2016 International Conference on. IEEE, 2016.
20. Zheng, Zhigao, et al. "Real-time big data processing framework: challenges and solutions." Applied Mathematics & Information Sciences 9.6 (2015): 3169.