ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology
Vol. 2, Issue 4, April 2013

Flexible Data Streaming In Stream Cloud

J.Rethna Virgil Jeny[1], Chetan Anil Joshi[2]

Associate Professor, Dept. of IT, AVCOE, Sangamner,University of Pune, Maharashtra, India[1]

Student of M.E.(IT), AVCOE, Sangamner, University of Pune, Maharashtra, India[2]

Abstract: Most of the applications in some special domains such as Telecommunication systems, Share market, Fraud detection and network security which required online processing of incoming data. They produce very high incoming load which needs to process by multiple nodes. The current system is on single node bottleneck and with the static configuration and hence it is not able to scale with the input load. So in this paper we present stream cloud a high flexible data stream processing engine for processing bulky data stream. This is particularly suitable for applications in online transactions, monitoring financial data processing and fraud detection systems that require timely processing of continuous data. Stream cloud using novel parallelization technique which splits one query into sub queries which are independently allocates to individual nodes for execution. It is having some elastic protocols which is used to dynamic resource management and load balancing of incoming load.

Keywords: Data stream, Load balancing, Elasticity, Scalability, Flexibility

I. INTRODUCTION

Number of applications where large amount of data need to be processed in quasi-real-time process which is having the limitations of store then process paradigm[1]. So many applications which require real time processing of high volume data streams pushing the limits of traditional data processing. These stream based applications include market feed processing and electronic trading for share market, network and infrastructure monitoring, fraud detection in banking sector for detecting any unauthorized user and command and control in military environments for controlling the drivers and adopter of wireless sensor network technologies[2], which is useful for investigating putting vital-signs monitors on all soldiers. In addition, there is also a Global Positioning System presented in many military vehicles, but it is not connected yet into a closed-loop system. So with this technology, the army people can monitor the exact positions of all vehicles in real time.

Traditional data stream processing technology using a novel computing paradigm for some specific applications scenario where massive amount of requested data should be processed with the minimum delay and correct output. SPE process tuple on the fly. i.e. Queries which are executing in SPE's are defined as “continuous” or “uninterrupted” since they are continuously standing through the streaming data. SPE continuously handling a streams of tuple just as traditional database systems handle different relations in database system. The system features higher scalability, elasticity, and load balancing which useful for fast execution of the query with the minimum delay [3]. A stream is nothing but the infinite sequence of tuples which is sharing a given schema denoted as (A1,A2,…. An). Each tuple have a time stamp attribute which is set at the data sources [4]. When the clock synchronization is not feasible, each tuple is time stamped at the entry point of the data streaming. Here in this, query is nothing but the acyclic graph where each node is an operator and edges define data flows path [5]. The query having two types of operator's such as stateless or stateful.

Stateful operators are like (Aggregate, Join, and Cartesian Product (CP)) perform operations on continuous sequences of incoming tuple. In streaming system because of the infinite nature of the continuous incoming data stream, stateful operators perform their operations on sliding windows of tuple for a fix periods of time or fix number of tuple. Stateless operators are like (Map, Union, and Filter) do not keep any state or sessions across the multiples tuple and perform their operations based on input tuple. In this, compiler takes the query and generates its parallel version which is later on deployed on a different cluster of node. Scalability can be achieved by using the Parallelization Strategy. SC performs stream splitting/dividing and hides the parallelization logic within smart nodes which gives guaranteeing semantic transparency i.e. the output of the parallel execution matches with the output of the centralized system.

The stream cloud can be summarized as,

- Highly scalable and elastic stream process engine for shared nothing clusters.
- A novel parallelization technique which minimizes distribution overheads.
- Elastic resource management using some dynamic algorithms.
- Transparency in query execution via query compiler.
-

II. BACKGROUND

This includes the data streaming basic concepts such as data stream, data streaming operators and continuous queries [6].

ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology
Vol. 2, Issue 4, April 2013

A. Data streaming model

This is continuous stream of tuple by the stream process engine. Here the continuous queries can be defined as direct acyclic graph with some additional input and output edges. Here every node i.e. an operator that takes tuple from one or multiple input streams and after execution produces tuples for one or multiple output streams. Here queries are called as "continuous" or "uninterrupted" because results are computed in continuous online fashion after the input tuple are processed.

1) *Data streaming operators:* This section provides the basic overview related to the data stream operators. These operators are useful for processing the input queries and produce the output tuple. Basically the operators can be classified by two ways whether they are maintaining any state while processing any input query or not. Stateless operators perform every operation one after one. So that they are not maintaining any state between execution of tuple and produces corresponding output. e.g., for stateless operators include Map, Filter and Union operators. Statefull operators maintaining states while executing the tuples. They are Processing multiple input tuple and produces single output. e.g. of Stateful operators include Aggregate, Join, and Cartesian Product (CP).

Map The Map operator is a generalized projection operator which is used to transform the whole schema of the input tuples.

The map can be defined as,

$$M\{F'_1 \longleftarrow f_1(t_{in}),\ldots\ldots,F'_n \leftarrow f_n(t_{in})\}(I,O)$$

Where I and O represent the input and output streams respectively, F'1 and F'n are the schema of output tuple.

Filter The Filter is a generalized selection operator used to discard or to route tuple from one single input request stream to multiple output streams.

The filter can be defined as,

$$F\{P_1,\ldots,P_m\}(I,O_1,\ldots,O_m[O_{m+1}])$$

Where I is the input stream, O1…. Om, Om+1 is an ordered set of output streams and P1. Pm is an ordered set of predicates.

Union The Union operator is used to merge multiples tuples from multiple input streams into a single output stream.

This can be defined as,

$$U\{\}(I_{1,\ldots\ldots,}I_{n,}O)$$

Where I1…In is a set of input strea1ms and O is the output stream.

Join and Cartesian Product These two stateful operators are used to match different tuple from two separate input streams, a separate window is maintained for each input stream.

This can be defined as,

$$J\{P,WT_{ype}[,ts],Size\}(S_l,Sr,O)$$

$$CP\{P,WT_{ype[,ts]},Size\}(S_{l,}S_{r,}O)$$

B. Parallel stream processing

This section includes different alternative Strategies to parallelize queries in stream cloud. This section also shows how the stream cloud operators hide the parallelization logic and guarantee parallelization transparency. When moving from a centralized system to a parallel execution, the challenge is in guaranteeing semantic transparency. i.e. the output of the parallel execution should be always same to the output of the centralized systems, while minimizing overhead. In parallel stream processing specific attention must be given to stateful operators, So here as we must ensure that all tuples that must be perform aggregated or joined operations together are always processed by the same node For e.g. If An Aggregate operation running on a particular node and it calculates the daily average expense of mobile phone users that must process all tuple belonging to the same user in order to produce the correct output result. In stream cloud the parallelization strategies can be characterized by spectrum. The position of each strategy presented in a spectrum is depends on the granularity or depth of the parallelization unit. So here, basically in one context we have a parallelization strategy which keeps the complete query as its parallelization unit. At the other context, we have a parallelization strategy that uses individual operators as parallelization units. In query parallelization there are two main factors for the distribution cost and these are,

1. The number of hops performed by each tuple from one node to another node.
2. The communication fan out of each node with the other nodes.

Here, We illustrate three different parallelization strategies for abstract query in the following figure and its deployment is on different clusters of nodes. Here in this example, the query is composed by two stateful operators. i.e. Join (J) and an Aggregate (Ag) and four stateless once. i.e. two Maps (M) and two Filters (F).

ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology
Vol. 2, Issue 4, April 2013

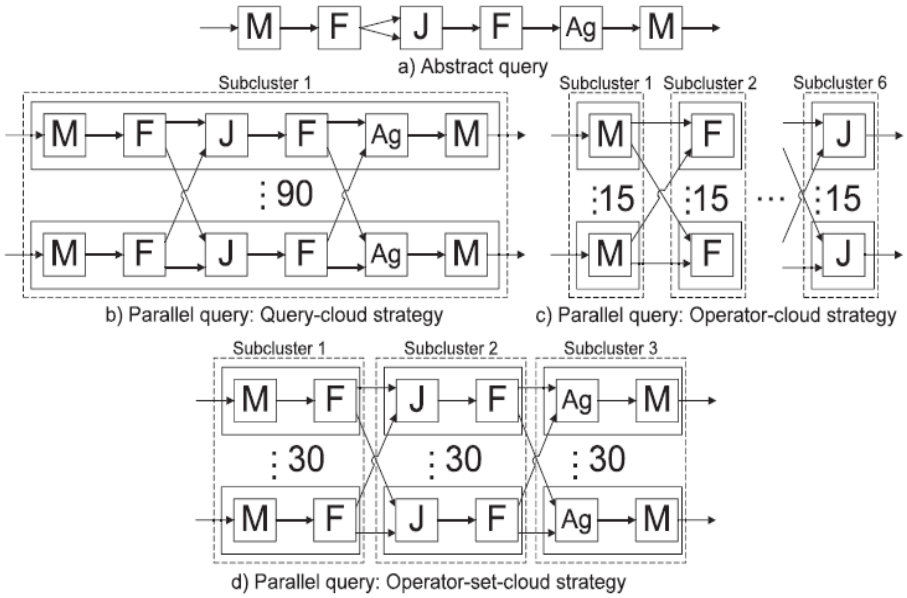

Fig. 1 Query parallelization strategies

There are three types of alternative strategies for parallel stream processing which are as follows

1) *Query-cloud strategy (QC):* If the parallelization unit is the whole query, then the deployment is on each node. Semantic transparency is achieved by continuously redistributing tuple just before each stateful operator. So here the joined and aggregation operations can be done by the same node only. So, In this The number of hops of each tuple is equal to the number of stateful operators and the total fan out for each node is equal to the number of operators minus one.

2) *Operator-cloud strategy (OC):* In this strategy, if the parallelization unit is a single operator. Then deployment is on different subset of nodes (called sub cluster). In this context, communication happens from every node of one sub cluster to all its peers in the next sub cluster. So here, The total number of hops is equal to the number of nodes or operators minus one and the fan out for every node is given by the size of the following sub cluster.

3) *Operator-set-cloud strategy (SC):* Operator-set-cloud strategies minimize the number of tuple hops. Here in this strategy every query is splits/divides into the multiple sub queries as stateful operators, plus one as a additional operator. If the starting of query is stateless operators then the sub query having stateful operator followed by all the stateless operators connected to its output. If the starting of the query is the stateless operators then the first sub query having all of the stateless operators.

III. PROPOSED SYSTEM

In this paper, we have focused on a new computing paradigm technique which is based on Stream Processing Engines (SPEs). The stream processing engines are the computing systems which are initially designed to process the continuous stream of data with the minimum delay time. In the new system, data streams are not stored but they are processed on the fly which is using the continuous queries. In this system the queries are continuously standing over the tuple and produces continuously output. So here, SPEs are able to distribute different queries among a multiple cluster of nodes which are also called as inter query parallelism or even it distributes different operators of a query across different nodes which is also called as inter operator parallelism [3]. For the fast data streaming SPE needs to use one or more self-joins which is also using the complex predicates. Fast streaming applications call for more scalable SPEs that should be able to aggregate or join the computing power of thousands of cores to process millions of tuples per second and that can be achieved by avoiding single-node bottlenecks of current SPEs lies in architecting a parallel-distributed SPE with intraoperator parallelism [7].

In this paper we have focused the stream cloud scalability with the efficiently using load balancing techniques and scalability/flexibility with the help of dynamic resource management whenever any failure occurs. This system also include Distributed stream processing for Distributed Stream processing engine such as Borealis allow deployment of each operator to a different node, Load balancing which is used to balance the incoming stream tuple load across several nodes, Elasticity which gives the resource manager allocates resources to its application, on-demand. It is also called as dynamic resource management of the stream cloud and Parallel stream processing like Aurora [8] and Flux [9] for parallel stream processing[10].

ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology
Vol. 2, Issue 4, April 2013

IV. ARCHITECTURE

Architecture defines an elastic resource management and dynamic load balancing of stream cloud. Figure shows a sample configuration with elastic management components. Stream Cloud architecture includes an Elastic Manager (EM), Resource Manager (RM), and Local Managers (LMs). Each SC instance runs an LM which continuously monitors resource utilization and incoming stream load, and is able to reconfigure the local LB i.e. update to its (BIM). In this context, Each LM periodically reports monitoring information to the EM that aggregates it on a per-sub cluster basis. Based on the collected data by EM, it decide to reconfigure the system either to balance the load to decommission instances. If sometimes the reconfiguration is needs to be done then reconfiguration decisions are taken and executed independently for each sub cluster.

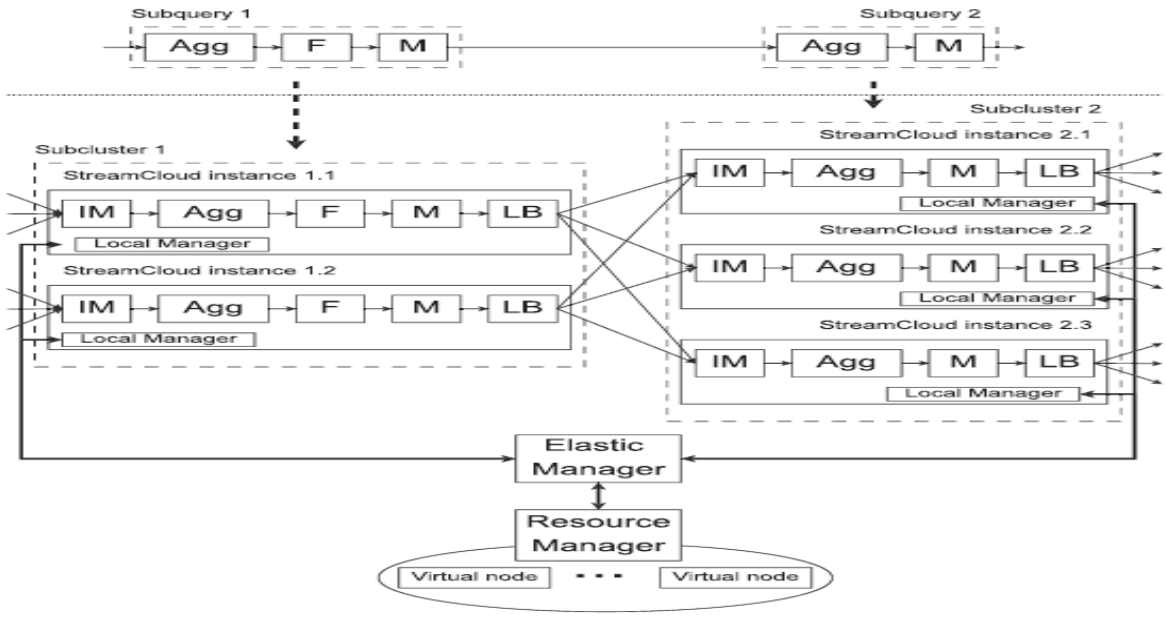

Fig. 2 Elastic management architecture

If sometimes any instances must be decommissioned then the EM interacts with the RM. Once the EM receives a new instance, the sub query is then deployed and the new instance can be added to the sub cluster that was about to saturate. There are so many protocols are presented for improving elasticity of stream cloud such as,

- Elastic reconfiguration protocols.
- Elasticity Protocol.

Whenever need to reconfigure a sub cluster it means that transferring the ownership of one or more buckets from one instance (old owner) to another (new owner) in the same sub cluster.

V. CONCLUSION

In this paper, We have presented Stream Cloud, a highly flexible and elastic data streaming system. Stream cloud provides transparent parallelization that preserves the syntax and semantics of centralized queries. Flexibility and scalability is attained by means of a novel parallelization strategy that minimizes the distribution overhead. Stream cloud flexibility and dynamic load balancing gives efficiency with minimizing the number of resources used for coping with unstable workloads. The evaluation demonstrates the large flexibility, scalability and effectiveness of elasticity of Stream Cloud.

REFERENCES

[1] M. Stonebraker, U. C¸ etintemel, and S.B. Zdonik, "The 8 Requirements of Real-Time Stream Processing," SIGMOD Record, vol. 34, no. 4, pp. 42-47, 2005.

[2] S. Chandrasekaran, O. Cooper, A. Deshpande, M.J. Franklin, J.M. Hellerstein, W. Hong, S. Krishnamurthy, S. Madden, V. Raman, F. Reiss, and M.A. Shah, "Telegraphcq: Continuous Dataflow Processing for an Uncertain World," Proc. First Biennial Conf. Innovative Data Systems Research (CIDR), 2003.

[3] D.J. Abadi, Y. Ahmad, M. Balazinska, U. C¸ etintemel, M. Cherniack, J.-H. Hwang, W. Lindner, A. Maskey, A. Rasin, E. Ryvkina, N. Tatbul, Y. Xing, and S.B. Zdonik, "The Design of the Borealis Stream Processing Engine," Proc. Second Biennial Conf. Innovative Data Systems Research (CIDR), pp. 277-289, 2005.

[4] N. Tatbul, U. C¸ etintemel, and S.B. Zdonik, "Staying Fit: Efficient Load Shedding Techniques for Distributed Stream Processing," Proc. Int'l Conf. Very Large Data Bases (VLDB), pp. 159-170, 2007.

[5] D.J. Abadi, D. Carney, U. C¸ etintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S.B. Zdonik, "Aurora: A New Model and Architecture for Data Stream Management," VLDB J., vol. 12, no. 2, pp. 120-139, 2003.

[6] StreamCloud: A Large Scale Data Streaming System Vincenzo Gulisano1, Ricardo Jimenez-Peris1, Marta Patino-Martinez1, Patrick Valduriez2.

[7] M.T. O zsu and P. Valduriez, Principles of Distributed Database Systems, third ed. Springer, 2011.

[8] M. Cherniack, H. Balakrishnan, M. Balazinska, D. Carney, U. C¸ etintemel, Y. Xing, and S.B. Zdonik, "Scalable Distributed Stream Processing," Proc. Biennial Conf. Innovative Data Systems Research (CIDR), 2003.

[9] M.A. Shah, J.M. Hellerstein, S. Chandrasekaran, and M.J. Franklin, "Flux: An Adaptive Partiti oning Operator for Continuous Query Systems," Proc. IEEE Int'l Conf. Data Eng. (ICDE), pp. 25-36, 2003.

[10] Y. Xing, S.B. Zdonik, and J.-H. Hwang, "Dynamic Load Distribution in the Borealis Stream Processor," ProcInt'lConfData Eng. (ICDE), pp. 791-802,20.

BIOGRAPHY

J.Rethna Virgil Jeny received her B.E and M.E degrees in Computer Science and Engineering from Bharathidasan University, Trichy in 1997 and Annamalai University in 2005 respectively. She is currently doing Ph.D in wireless sensor Networks at MS University. She has received Lady Engineer Award by IEI. She is a member of IEEE, IEEE computer Society,ACM, ISTE, IEI , IAENG and a senior member of IACSIT. Her research interests includes Cross layer routing in Wireless Sensor Networks.

Mr, Chetan Joshi his B.E. in information Technology from Dr. BAMU Aurangabad University, in 2009. He is currently doing his M.E. in Information Technology from University of Pune.