Improved Real-Time Data Elasticity on Stream Cloud

Mr. Chetan A. Joshi[#1], Mr. Rohit N. Devikar [*2]

[#]M,E.I.T Second Year. Information Technology Dept.

[#2]Assistant Professor, AVCOE, Sangamner, Information Technology Department
Dist Ahmednagar. Maharashtra, India

Abstract — *Most of the applications in cloud domains such as online data processing, Fraud detection, large scale sensor network etc. where large amount of data should processed in real time. Earlier, for data stream processing, the centralized system environment was using with store and then process paradigms. After that some advancement has been introduced with distributed environment for data stream processing. Data Stream processing using novel computing paradigm which take query as input and splits that query into multiple sub queries and process the data on multiple sub clusters in such a way that reduces the distribution overheads. This kind of application generates very high input data which needs to process with the available clusters So High availability and elasticity are two key characteristics on the cloud computing services. High availability ensures that the cloud applications are sensible to failure. Elasticity is a key feature of cloud computing where availability of resources are related with the runtime demand. So in this paper we present a comprehensive framework for obtaining elasticity and scheduling technique for highly availability.*

Keywords — *Scalability, Elasticity, High availability, Load balancing, Reliability.*

I. Introduction

Number of real time applications in which large amounts of data should process continuously. But there are some limitation comes with the traditional store the process paradigm [1]. So for overcoming this issue, some advancement has been presented in the stream process engines. Stream process engines are computing systems which are designed to process continuous stream of input data with the minimum time delay. Instead of store then process, in this system data streams are process on the fly using continuous queries. This is due to the amount of input data which discourage persistent storage and the prompt result requirement. Here the query is continually standing in streaming tuple and produces continues output

Here in this system there is substantial development in the stream processing engine. Earlier it was running on the centralized stream processing engine [2]. Centralized engine using store then process paradigm which causes unnecessary value storage and other limitations. But now it's also running on the distributed environment. With distributed environment stream process engine distributes different queries among a cluster of nodes which we is called it as interquery parallelism or distributing different operators of the query across different nodes which is called as interoperator parallelism [3]. Most of the applications for scalable stream processing engine which need to aggregate the computing power of hundreds which need to process the millions of tuples per second. Here for obtaining higher scalability and avoiding the single node bottleneck problem stream process engine need to lies in distributed stream process engine with intra operator parallelism [4].

While doing the query parallelization, this requires to addressing additional number of challenges. Query parallelization should be semantically and syntactically transparent. Semantic transparent means query should produce exact the same output like non parallel queries. Syntactically transparency means the query should get automatically parallelized. It should be oblivious to the user. While doing parallelization, usage of resources should also be the cost effective. The parallel stream process engine should be elastic and it should manage the amount of its resources to the workload. The elasticity also combined with the dynamic load balancing technique. It should able to manage load across available nodes or the clusters.

In this paper we are presenting as inproved real time data elasticity on stream cloud [6] and elastic stream process engine which provides a transparent query parallelization. That is stream processing engine will accept the input query which is automatically paralyzed. This query will splits in multiple sub query and process individual sub query on clusters of nodes [6]. Stream process engine handles the stream of tuples. A stream is potentially infinite sequence of tuples which is sharing a given schema. All tuples having a time stamp attribute which sets at the data source [7]. The data source has clocks which are synchronized with the other system nodes. When clock synchronization is not feasible tuple can be time stamped at the entry point of the data streaming system. In SPE, query is defined as a cyclic graph where as node is an operator and edges defines the data flow. Here focus has given on stateless and steteful operators [8]. Stateless operator does not keep any state across tuples and perform computing operation only based on input tuple. (eg. Map,union and filter. Stateful operators perform computation operations on sliding windows of tuple defined over a fixed time period (eg. Aggregate, cartision product and join).

So in the proposed system stream cloud gives high scalability, reliability for stream processing engines. The input queries which are executing are automatically and all tuples provide transparent parallelization. System gives high scalability by giving interoperator parallelism.

System also contains load balancing, task assignment and scheduling for the execution nodes which executes its operation based on available execution information. Additionally system calculates the execution power capacity of each node. So execution task assignment can be performed efficiently System also uses heart bit technology which gives node alive status with transferring the signals in between each others for informing the live status.

II. LITRATURE SURVEY

A literature survey includes related work in the data stream processing in the stream line cloud for real time data processing which shows the system reliability and scalability. Some of legacy applications and most of the real time applications data processing should be continuous. So for such applications we need to use stream process engines. There has been advancement from centralized to distributed environment. There is a substantial change from store then process to tuple-on-the fly. It is also called as continues queries in which queries are continuously standing with a streaming data for real time processing. while doing the parallel stream processing, attention must be given at stateful operators (Aggregate, joins and Cartesian products) and stateless operators (map unions and filters) .Also here the basically two factors has for number of hops performed by each tuple and communication fan out 0f each node has considered. Here there are different strategies for parallelization such as,

A. Operator cloud strategy

In the operator cloud strategy, the query parallelization unit is a single operator. So each of input data deployed on different subset of node. We can also called it as a subclustor. If we will consider that there are 15 nodes presented and 5 operators are presented. Communication happens from every subclustor to all its peers in the next presented subclustors. So total number of hopes is 5 and fan out for every node is 15.

B. Operator set cloud strategy

The above operator cloud strategy has been exhibits the trade-off in-between the distribution cost and number of hopes. The operator set cloud strategy introduced for minimizing both things at the same time. Here for guarantee semantic transparency the communication is required to be done with stateful operators. Here each input query is splits in between the multiple sub queries as stateful operators plus an additional one. Sub query consists of stateful operators followed by stateless operators which are connected to its output [6].

Here these both strategies minimize the number of hopes and fan out.

For the effective tuple distribution and parallel query processing, we are using some special operators which is called as LB (ie Load balancers) and IM (ie Input mergers).

Load balancers are basically using for distribution the input tuple from one local sub query to all its downstream peers to guarantee that tuple should be joined together are indeed received by the same instance. For performing the load balancing it is using join operators, CP ie. General join operators and aggregate operators [9].

Input mergers is simply forwards tuple which comes from its upstream load balance might lead to incorrect result. This is basically using for physically merging streams which are processed by different instances.

III. IMPLEMENTATION OF PROPOSED SYSTEM

In the proposed system, we have focused on the Load balancing, HA and elasticity with the help of operator cloud strategies and operator set cloud strategies with some modified algorithms and concepts. These steps are as follows,

A. Proposed System Algorithm

```
Input: Query data Output: Result files.
1.  Data € Query data.
2.  Split data using split criteria
3.  BuildTasks() -> task
4.  GetClientStatus ()
5.  Check busy Bit of the individual client.
6.  Check client Attribute -> RAM, Processor,
    Memory
7.  While(checkBusyBit())
8.  Start
9.  If (allTaskCompleted)
10. Break;
11. Client feasibility
    checking for associated task.
```

```
        12.     Schedule task to client
        13.     TaskTssign() -> Client
        14.     End
        15.     Contribute Results ()
        16.     Display Results ()
End.
```

B. Result Analysis

In the presented Real-time data streaming system, we have checked the scenario and it has been observed that some changes in original and presented system readings in operation execution time. We have calculated with multiple data size and multiple datasets. We have tested different cases while doing the analysis.

Case I: Data size Vs. Time required

Data set file size	Original system time	Contribution system time
Data 1 (100 Files)	3154 MS	3115 MS
Data 2 (200 Files)	6162 MS	6141 MS
Data 3 (300 Files)	9176 MS	6141 MS

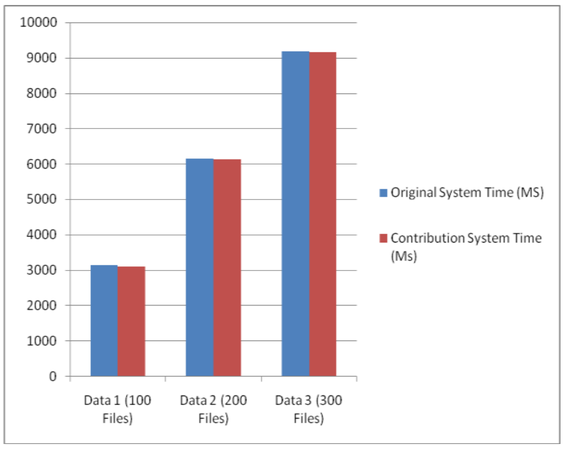

Fig. 1 Data size Vs. Time required graph.

Case II: Base/Existing system with multiple clients

Data set File size	Client 1	Client 2
Data 1 (100 Files)	54 MS	60 MS
Data 2 (200 Files)	51 MS	55 MS
Data 3 (300 Files)	60 MS	76 MS

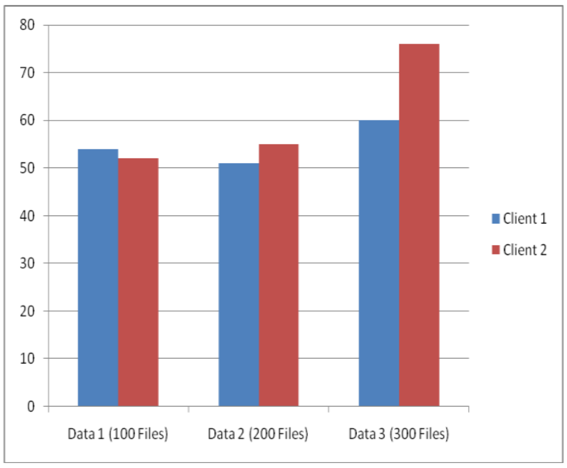

Fig. 2 Base/Existing system with multiple client graph.

Case III: Presented system with multiple clients

Data set File size	Client 1	Client 2
Data 1 (100 Files)	45 MS	32 MS
Data 2 (200 Files)	40 MS	67 MS
Data 3 (300 Files)	52 MS	72 MS

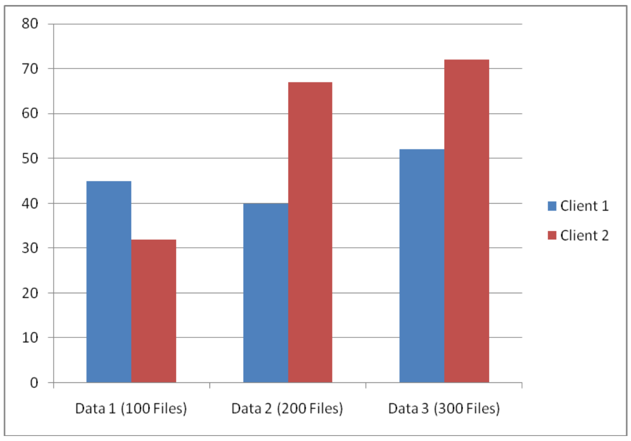

Fig. 3 Presented system with multiple clients

IV. SYSTEM ARCHITECTURE

Architecture diagram shows the improved real-time data system on stream cloud system. Here figure represents complete 3T system. In this system, First Q input query comes and it is divided into multiple sub queries SQ1, SQ2. The spitted sub queries have been assigned to stream cloud instance for execution. While doing this, Stream cloud served uses some operations like parsing, Mapping and Job creation and job assignment. While doing this, it has to take care of the different status of the client nodes like computing power, CPU, RAM etc. So as per this criteria job has been assigning to execution. System also contains the heart bit signal passing mechanism which is using for checking the live status of the available nodes. System passes signals continuously for knowing node active status after fixed time of interval. If in case reply from node will not get within the time then system assumes that node as dead. This technique is useful for improving efficiency of the system. It also continuously checks whether new node is presented in the system which helps for HA. Once the operation done successfully, then it is directly giving the output.

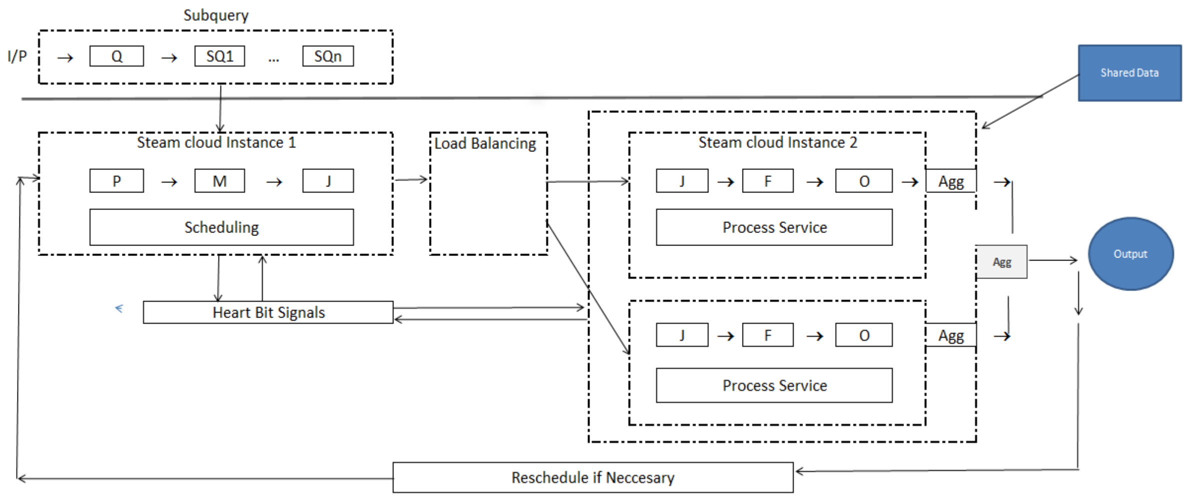

Fig 1: If necessary, the images can be extended both columns

Stream cloud compliments elastic resource management with dynamic load balancing for guaranty that the new instances are only be provisioned when a subclustors of node is not able to cope the incoming load. Here system also checks processing power of the computer system like CPU and RAM. So as per the collected information task can be assigned for the execution which helps for balancing the load across subclustors. System uses different strategies for obtaining elasticity such as

A. Elastic reconfiguration protocols

Subclustor reconfiguration required the transferring the owner ship from one instance of subclustor to another ie. from the old instance to new one in the same subclustor. This triggers reconfiguration by one or more reconfiguration actions.

B. Reconfiguration start

The process is initiated by the elastic manager that decides to perform a reconfiguration either for provisioning, decommissioning, or load balancing purposes.

C. Windows reconfiguration protocol

The Window Recreation protocol aims at avoiding communication between the instances being reconfigured.

V. CONCLUSIONS

In this paper, we have presented improved Real-time data elasticity on stream cloud is presented. System also presents transparent query parallelization that keeps the syntax and semantics of the centralized system. HA, Elasticity and scalability are attained by means of novel parallelization strategy which minimizes the distribution overheads and also improves the performance of the system. Stream cloud elasticity and dynamic load balancing gives efficiency with minimizing number of resources. This evolution demonstrates the scalability, elasticity and high availability 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] M.T. O ¨ zsu and P. Valduriez, Principles of Distributed Database Systems, third ed. Springer, 2011

[5] V. Gulisano, R. Jime´nez-Peris, M. Patin˜ o-Martı´nez, and P. Valduriez, "Streamcloud: A Large Scale Data Streaming System," Proc. Int'l Conf. Distributed Computing Systems (ICDCS '10), pp. 126-137, 2010.

[6] StreamCloud: An Elastic and Scalable Data Streaming System ,Vincenzo Gulisano,Ricardo Jime´nez-Peris,Marta Patin˜ o-Martı´nez,Claudio Soriente,Patrick Valduriez VOL. 23, NO. 12, DECEMBER 2012.

[7] 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.

[8] 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.

[9] Y. Xing, S.B. Zdonik, and J.-H. Hwang, "Dynamic Load Distribution in the Borealis Stream Processor," Proc. Int'l Conf. Data Eng. (ICDE), pp. 791-802, 2005.