International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor:6.887
Volume 5 Issue IX, September 2017- Available at www.ijraset.com

Bandwidth Optimized Continuous Querying In Public Cloud Storage

P. Dileep Kumar Reddy[1], K. Krishnaveni[2]
[1, 2] *lecturer, dept of computer science and engineering, jntua,anantapur.*

Abstract:Bandwidth efficient execution of online large information analytics in telecommunication network demands for tailored solutions. Present Streaming Analytics systems are designed to operate in large data centers, assuming unlimited information measure between data center nodes. It overlooks the fact that available information measure is insufficient and high cost resource making the telecommunication network valuable to end-users. The streaming analytics platform Continuous Hive(CHive) is suitable for distributed tele communication clouds. The fundamental contribution of CHive is that it reduces question plans to minimize their overall information measure reduction when implemented in a distributed tele communication cloud. Early experiments on data from a large mobile operators, indicate that CHive can yield information measure reduction to ninty nine percent.

Key words: Big Data, Cloud, Data processing, Distributed computing, Event Stream Processing, Optimization, Query processing, Telecommunications

I. INTRODUCTION

The web based computing which provides shared information resources to computers and other devices is called as cloud computing. The important service of cloud computing is cloud storage which provides services for data owners to store information in cloud via internet. Large data companies like Facebook, Google, Twitter or LinkedIn continuously collect large amounts of data from users and their

devices. According to the business model of these companies, data has become a very valuable asset to extract knowledge about user interests for instance to offers. Traditional tele communication companies, in contrast, adopt a different business model. These companies generate revenues by selling their communication services, including network badwidth. However, similar to the web scale companies listed above, tele communication companies also have access to huge amount of information, in particular data related to how their networks are being used. This includes both network traffic information and statistics about the operational status of the deployed tele communication equipment. By efficiently combining the traditional multi-authority scheme with TMACS, construct the hybrid scheme which is more suitable for the real scenario, in which attributes come from different authority-sets and multiple authorities in an authority-set combinly maintain a subset of the whole attribute set. This method addresses not only attributes coming from different authorities but also security and robustness. How to reasonably select the values of (t, n) in theory and design optimized interaction protocols will be addressed in further work

II. RELATED WORK

Tele communication networks are designed to enable communication between terminals.

A. TMACS

In the previous approaches we propose a multi authority CP-ABE access control method which deals with the single-point bottleneck on both security and performance. In this scheme, authorities combinely manage the attribute set but neither of them has total control on any specific attribute. CP-ABE schemes, having a secret key (SK) which is used to generate attribute private keys. The secret key can be shared among n authorities by using (t , n) secret sharing method. the master key cannot be obtained by any authority alone. It is just because of using the (t , n) threshold secret sharing method. To the best of our knowledge, it is the first try to address the single point bottleneck problem on both security and performance in CPABE access control methods in public cloud storage.

B. ESP

Tele communication networks are designed to enable communication between terminals. These networks are built up in a hierarical manner including access layer networks connecting to end user homes. In Event Stream Processing(ESP) systems they store event processing workflows rather than data. ESP systems facilitate composing and readying distributed event stream processing flows.

The current state of the art in ESP does not offer an SQL like declarative API that enables non programmers to define streaming queries with out writing a single line of code. The ESP platforms does not support question plan optimizations for information measure reductions.

C. CEP

Complex Event Processing solutions gives an analyzing continuous event streams. It is an open source Complex Event Processing tool developed by EsperTech. It can operate as a standalone application. Esper gives a high level SQL like query language.

III. PROPOSED WORK

The streaming analytics platform Continuous Hive(CHive) is suitable for distributed tele communication clouds.

A. The Chive Query Language

CHive having the high-level command language support to distributed event stream process. The CHive command language (CHiveQL) is powerfully inspired by Esper's event process language (EPL). EPL is associate SQL like language with FROM, WHERE, GROUP BY, HAVING, ORDER BY and LIMIT clauses to support streaming analytics, EPL replaces information tables with event streams, generating knowledge tuples in a very continuous fashion. what is more, EPL offers varied SQL extensions supporting the expression of streaming queries, as well as statements to derive and mixture data from one or additional event streams. As associate example, EPL permits to outline varied styles of window-based views on knowledge streams, as well as time-based windows (e.g. to stay all events generated throughout the last fifteen minutes) and fixed-length windows (to keep 10K events in memory). For instance a EPL streaming question, the listing below depicts associate EPL question expression calculating each second the top-10 HTTP hosts generated. This question is employed in the remainder of this text for instance the bestowed work providing a close summary of EPL or CHive Query Language.

```
select http_host, sum(rec_bytes) as download_volume
from src.win:time(15 minutes)
where http_host <> 'unknown'
group by http_host
output snapshot every 1 sec
order by download_volume desc
limit 10
```

B. Query plan & primitives

In order to execute a question, a question string is remodeled into a query plan, representing advancement through question primitives. Every question primitive implements a passionate perform of the question and is characterized by having one or additional input gates (to receive incoming information tuples), like wise together or additional output gates. ChiveQL presently supports the SQL like question primitives

Project: retains only those attributes of the input schema required for executing the query
Filter: retains only those records/events matching the filter criteria
Map: applies a function to query attribute values
Group: aggregates records/events into buckets
Order: sorts the result set
Limit: retains only a specified number of records/ events in the result set
Join: combines two streams using a hash join function preserving only those event combinations that meet one or more join condition(s)
Union: two sets are joined together as a new set and some more primitives will be added by using CHiveQL
Partition: Events are distributed over the multiple instances of the question plan's downstream primitives According to the value of a hash function computing on a particular attribute
Broadcast: Incoming events are redundant to all output gates
Merge: The inverse of broadcast, joints events from each input gate by applying a given aggregate function.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor:6.887
Volume 5 Issue IX, September 2017- Available at www.ijraset.com

C. System Architecture

CHive having high-level command language support to distributed event stream process. It's a layer that sits on top of existing Event Stream process (ESP) platforms, like Storm, Akka, etc. As such, the standard non-functional support is inherited from the CHive to ESP platform. the rest of this section highlights the fore most Components of the CHive design additionally the algorithms for question plan generation and readying. The following figure represents the CHive design, as well as the following main components

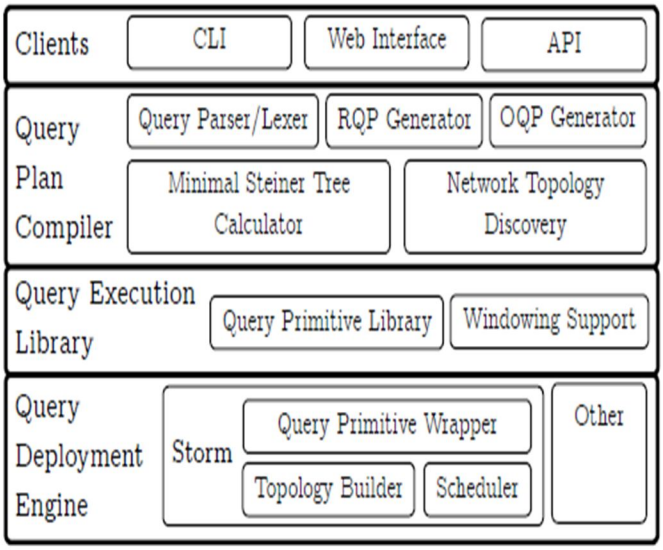

Figure 1

1) *Client APIS:* CHive having various client API's to submit CHive question expressions,event source data (i.e. their address, name and event kind name) and the schemas of event types. These API's presently embody a command line interface (CLI), an online interface (HTTP) and a Java API.
2) *Query Plan Compiler:* This element generates a distributed CHive question plan to be readying onto a given or discovered configuration.
3) *Query Execution Library:* This library gives all needed functionalities to execute CHive question plans, together with a implementation of all supported question primitives.
4) *Query Deploying Engine:* Deploys the generated reduced CHive question plans onto the process nodes within the topology.

D. Optimized query plan generation algorithm

The query plan optimization algorithm for deployment onto a given network tree is based on the following premises:

1) The reference question plan(RQP) contains the right sequence of primitives for information measure reduction purposes. RQP is a question execution graph reduced for readying on a centralized cluster or datacenter.
2) Question premitives in that are not stream parallelizable, need to be allocated on the trunk, defined as the tail end of the deployment tree that only has a single path towards the sink.
3) Question primitives that have an output to input ratio less than 1, need to be placed as close as possible to the relevant event source.since over all bandwidth consumption on each deployment tree edge, reducing bandwidth early on a path from source to sink, translate into cumulative gains for that path.
4) output to input ratio larger than 1, need to be placed as close as possible to the sink.
5) Since step-3&4 act as opposing forces, the optimization algorithm must operate on sub-sequences of question primitives rather than on individual ones to determine the most beneficial output to input ratio.

IV. PERFORMANCE EVALUATION

The maximum achievable bandwidth reduction heavily depends on the query and characteristics of the event streams and the no.of layers in the deployment hierarchy.

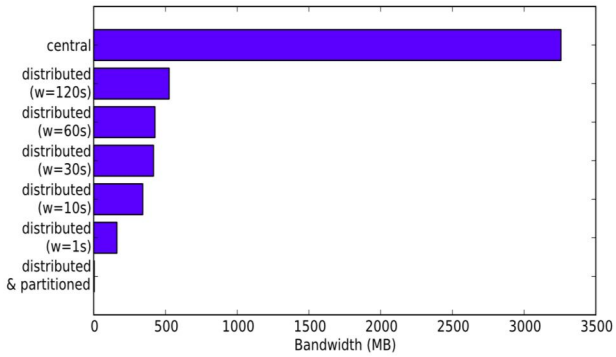

Figure 2

figure shows bandwidth reduction of various configurations of a query. The figure shows the bandwidth reductions up to 99.9% for partitioned streams and between 84% and 95% for unpartitioned streams. For unpartitioned streams, most gains are realized by the projection step which reduces events to only retain those attributes. Partitioned streams gains from the aggregation step followed by the limit. The lower the output frequency of the aggregation step, the reduction will be high. A marginal effect of window size(w) on the reduction factor larger the time window, the more events are aggregated into the window.

V. CONCLUSION

CHive is the brand new technology with in the event stream processing toolbox which enabling information measure efficient distributed stream process in massive information networks and distributed cloud environments. It simplifies and reduces streaming analytics in telecommunication clouds. It offers a streaming query language enriching Esper's Event Processing Language(EPL) to facilitate question execution. CHive's question plan compiler generates a question deployment plan that minimizes the expected over all information measure consumption. Minimizing information measure consumption is gained by reducing event streams. This needs a large amount of correspondence with in the question plans, the CHive question compiler includes a set of substitution rules for changing question primitives with semantic equivalents. A mathematical analysis and experiments using real operator information indicate that CHive can yield information measure reduces up to ninty nine percent.

REFERENCES

[1] Apache hadoop. [Online]. Available: http://hadoop.apache.org/
[2] Spark - lightning-fast cluster computing. [Online]. Available:http://spark.apache.org
[3] Storm - distributed and fault-tolerant period of time computation. [Online]. Available: http://storm.incubator.apache.org
[4] S. Babu and J. Widom, "Continuous queries over knowledge streams," SIGMOD Rec., vol. 30, no. 3, pp.109–120,Sep.2001.
[5] I. Satoh, "Mapreduceprocess on iot clouds," 2013 IEEE fifth International Conference on Cloud Computing Technology and Science, vol. 1, pp. 323–
[6] J. Dean and S. Ghemawat, "Mapreduce: Simplified processing on giant clusters," Commun. ACM, vol. 51, no. 1, pp. 107–113, Jan. 2008.