A METHOD FOR DISTRIBUTED REAL TIME AGGREGATIONS

[1]Shrikrashna S. Kadam, [2]Manik K. Chavan
Depatrment Of Computer Science and Engineering, Walchand College of Engineering
Sangli, Maharashtra, India
Email:[1]Shrikrishna.kadam4@gmail.com, [2]chavan_manik@yahoo.com

Abstrac—This paper focuses on various approaches and proposes a method for aggregations in distributed environment and its importance for analysing the data. Aggregation operations are the most used operations in the analysis of big data. Traditionally data collected over a period of time was analysed by storing it in the data-warehouse or database with the help of aggregation operations. This paper mainly focuses on the various aggregation operations and their use in business analytics and also the significance and role of distributed computing to perform these aggregations in real time to analyse continuously streaming data to get analysis results in the real time and improve the decision making.

Index Terms— Real-Time Aggregations, Distributed Computing, Analytics, AKKA Actors, KPI's(key performance Indicators)

I. INTRODUCTION

OLAP (Online Analytical Processing) systems play a vital role of analytics in many industries today. By aggregating the individual records of a data set, they provide an non-rational multi-dimensional view on the large volumes of data stored and are used for the purposes of analysis. Recently, the increasing availability of machines with large amounts of main memory and improving processor speeds have led to a surge in the popularity of in-memory OLAP systems like **S**calable **i**n-**me**mory **a**ggregatio**n**(**SIMEAN**) by R. J. Kopaczyk et al.[3], which can process multi-dimensional queries faster than their on-disk counterparts. However, while hardware capacities improve, the amount of data to analyse continues to increase. We can imagine that technological inventions in the area of hardware resources may not be able to keep up with this growth and eventually could reach a halt. Rather than buying the newest cutting-edge machines every time, a better answer to the problem of data volume growth can be to enable a solution to scale out. This means that computation can happen by using a cluster of relatively cheap commodity hardware. Additional hardware machines can then be connected to handle even larger amounts of data.

The Problem: Continuously Growing Data Volume:

While storage capacity and access speeds for various medias (not only main memory) are growing dramatically, and processing speeds of processors have continued to improve, the amount of data collected, stored and analysed over a period of time has not remained at a constant level, either. Will the advantages gained by recent improvements of hardware performance soon be dissipated by the growth in data volume?

The Solution: Distributed Computing?

A naive solution to the problem of continuous data volume growth would be to just buy a better and more expensive machine if storage space runs out and/or processing time becomes inadequate. After all, one may argue, computing hardware gets better every year. However, there are good reasons why this solution may not prove to be viable in the long term:

1. Who guarantees that technological innovation will keep up with the data growth rate? We can, after all, imagine that Moore's Law will not continue to

hold forever. Also, this does not help if the rate of data volume growth exceeds the growth in matching hardware capabilities.

2. Buying highly specialized cutting-edge hardware machine is bound to become expensive, as the newest products in the market are usually sold at higher prices.

There is a cheaper alternative to this. We can buy several low-cost commodity machines, connect them together and make them work to achieve both higher storage capacity and better processing speed at a low cost i.e. distributed system. This also solves the first problem: although there is certain coordination overhead between nodes in a distributed system, we can continue to scale our systems independently of the speed of innovation. J. Bernardino, et al [4] proposed distributed system to improve query performance in OLAP which works on parallelizing queries. Golfarelli M. [5] proposed technology such as BPM (business process management) over the data-warehouse. C. Burnay et al. [6] proposed a framework for designing a system for business analysis by using the OLAP concept. It represents KPI's as facts and aggregation results as measures and schema for hierarchically organizing KPI's.

II. OLAP CONCEPTS

OLAP systems helps in planning, problem solving and decision making. OLAP deals with large amount of historical data. OLAP queries need to access large amount of data and need to perform huge number of aggregation. Following are some concepts used in OLAP systems.

A. Cube

For analysing data we use cube in online analytical processing (OLAP) as shown in Fig.1 and Fig.2. OLAP cubes are more famous in analytics for its feature of aligning data in multiple dimensions. These cubes are used to arrange the collected data for easy analysis purpose. This cube can be of type ROLAP, MOLAP or HOLAP. These cube categories are formed on the basis of storage of aggregation results. MOLAP stores source data and

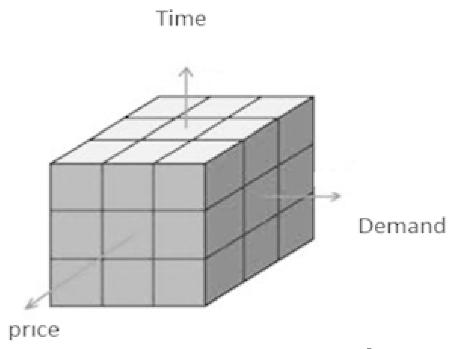

Fig.1 OLAP cube

aggregation results into multidimensional structure, ROLAP stores aggregation results into relational data store, whereas HOLAP is hybrid of ROLAP and MOLAP. We used ROLAP cubes to analyse the data. Each ROLAP cube has set of dimensions, measures, attributes and hierarchies as shown in Fig. 3. Dimensions represent the various fields of data to be analysed. Each dimension has its attributes. These dimensions can be of high cardinality. Dimensions can also be time dimensions whose attribute value is a time value.

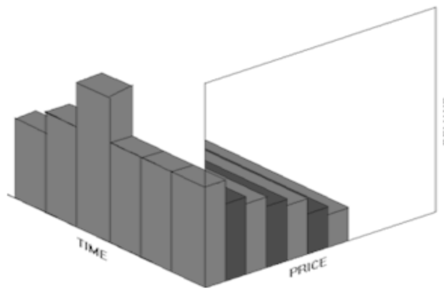

Fig. 2 OLAP cube with multiple dimensions

B. Hierarchy

In ROLAP cube, hierarchy is subset of dimensions arranged in the hierarchical manner for which aggregation values are to be computed. Each hierarchy represents the analysis perspective of a cube. To analyse cube for a hierarchy of dimensions it need to compute aggregation operation at every dimension level called measure or metric. A single hierarchy may have one or more measures. Attributes are values of dimension. Cube can contain multiple hierarchies. Level represents the level of dimension in the hierarchy. These cubes can be represented in the XML format. XML file can contain more than one cube representations. Each hierarchy has its own set of dimensions as well as measurements.

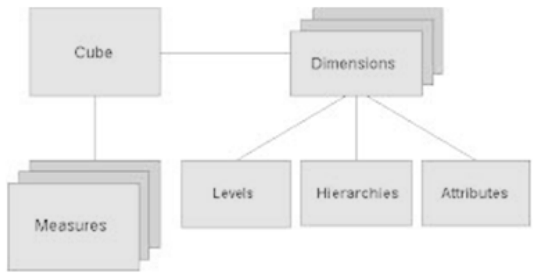

Fig.3 Architecture of a cube with showing its facets

```
<Cube>
        <Hierarchy>
                <Dimensions/>
                <Measurements/>
          </Hierarchy>
        <Hierarchy>
                <Dimensions/>
                <Time Dimensions/>
                <Measurements/>
        </Hierarchy>
</Cube>
```

C. Metrics

Metrics are the quantifiable measures that are used to track and assess the status of a specific business process. Every area of business has specific metrics. These metrics are nothing but the aggregation computations performed on the KPI's at every dimension level in a hierarchy. Metric can be any mathematical measurement such as Sum, Average, MAX-MIN, Standard Deviation etc. These metric functions depend on the analysis to be done.

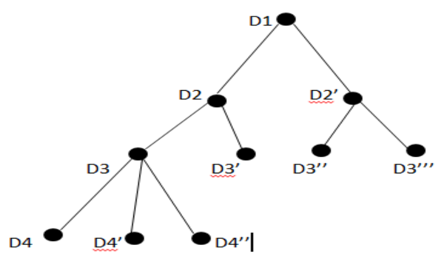

Fig 4 Tree representation of hierarchy with metric node

D. Metric Node

As shown in Fig. 4, it represents the tree structure for a hierarchy. D1, D2, D3, D4 are the levels of dimension in the hierarchy. These dimensions are arranged in hierarchical manner. D2', D3', D3'', D3''', D4' and D4'' are different values of attributes for the respective dimension levels. Thus branching in the given tree depends on the cardinality value of each dimension level. Node in the tree is identified by its hierarchical key formed by its position in the tree. e.g. for D4' node D1_D2_D3D4 will be the key. Each node contains the values computed for the aggregation metrics for its dimension level therefore these nodes are called metric nodes. Higher level node represents the aggregation of metric nodes below that level. Thus in a hierarchy every metric node is a unique entity and it is the lowest granularity of output entity.

III. PARALALLIZATION APPROACHES

A. Based on input data partitioning

In this approach of parallelization, the input data is divided into non-overlapping parts and partial results are computed for each part. The process of aggregation is, in most cases, simple to parallelize. For performing aggregations in parallel we have to split the input into several non-overlapping parts and process them independently. There may be special design requirements when parallelizing the aggregation and this mainly dependent on the aggregation function used which we want to run in parallel for parallelizing it. Some of the aggregation functions such as sum, minimum or maximum of measures etc. are trivial. All of these functions are binary operator functions and satisfy the associativity property. In other words, the order in which the operations are performed does not matter. There are other aggregation operations which are not easy to parallelize but some of these operations are derived operations which use basic aggregation operations as their basic components. For example, average is the derived aggregation operation of sum and count which again use operator which are associative in nature that's why parallelizing average aggregation operation is not a tedious job. Likewise other operations can be parallelized. This parallelization strategy is solely based on input data partitioning but there are some holistic aggregation operations like TOP-K which are non-trivial and not easy to parallelize by input data partitioning. Due to this limitation this parallelization strategy is not useful.

B. Based on input data partitioning

This approach of parallelization is to parallelize the computation based on the output partitioning. In this approach parallelization is done by computing each non overlapping output entity separately.

IV. PROPOSED WORK

Fig. 5 shows the architecture diagram of proposed system. In the proposed work we used second approach to parallelize analysis of continuously streaming multidimensional time series data according to the dimensionality of OLAP cube [2], to compute measures required for analysis of cubes due to the presence of holistic measures. In this method we parallelized analysis of cube at three places.

At first place we distributed the set of hierarchies to each node in the cluster. Each node will read the configuration of hierarchies to be computed. This distribution is a named distribution i.e. aggregation for any hierarchy can be computed by any computing node if it is enabled for computation on it. In this use case some of hierarchies are highly computation intensive while some are not. So to avoid skew problem we made a unit of two hierarchies for distribution in which one hierarchy is computation intensive and other is not. Hierarchies are distributed to nodes according to these units.

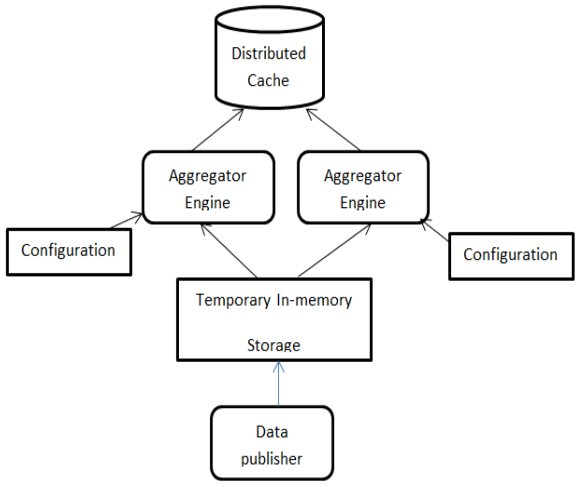

Fig. 5 Architecture of proposed system

In this method, we created a data publisher to send the continuously streaming KPI's called Facts represented in the form of multidimensional time series data. The data received from the publisher is temporarily put into the in-memory storage. These in-memory KPI's are forwarded to each subscribing nodes in the cluster by listening to the put operation preformed in in-memory storage.

In proposed work we have multiple cubes to analyse with one or more hierarchies in each. Streaming multidimensional KPI's for each cube are identified by matching its set of dimension. At second and third place we parallelized analysis of cubes by computing aggregation operation at each dimension in a hierarchy for this purpose we used actors concurrency model.

1. Concurrent Actors

Today the use of actor based concurrent models is on the peak. Typesafe's AKKA[1] is an open source framework used for real time transaction processing. It is a highly scalable and highly concurrent actor based model. AKKA actors are lightweight entities with

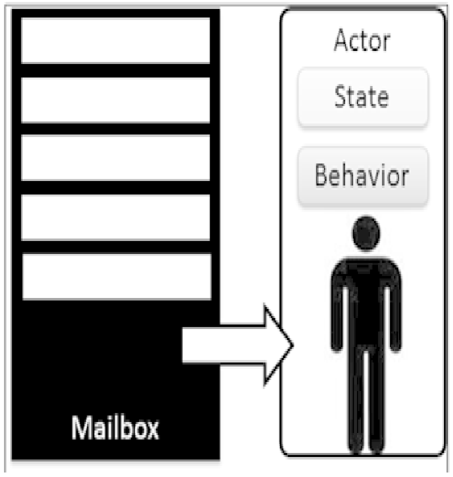

Fig. 6 Architecture of actors in actor based model

asynchronous and event driven processing. Each AKKA actor has its mailbox which can be bounded as well as unbounded. Actors can be created for performing highly concurrent tasks. These actors can be state-full as well as stateless. Actors receive tasks in the order as message in the mailbox. These received messages are identified by using pattern matching. Behaviour of AKKA actors can be defined according to the received message. These actors can be identified in the cluster by using its name as unique id. In-order to achieve maximum concurrency in an application we have to create a thread for its lowest granularity of tasks. Creating large number of threads is not feasible in case of the highly concurrent tasks due to the overhead of maintaining concurrency while using large number of threads. So, in this method at second place of parallelization we created multiple concurrent actors to create batch of tasks and grouped by using the key. This batch of tasks identified by key is distributed to actor identified by same key as name because of this specific type of task is always distributed to specific actor which will guarantee the in order processing of message according to their arrival time.

At third place of parallelization we used AKKA actors to compute and update aggregations for each metric node in the hierarchy tree. AKKA actors are created for lowest granularity i.e. metric node. Actors created executes concurrently to achieve maximum thread utilization. Actor will compute the aggregations results using previous value present in persistence used and KPI values in the multidimensional streaming data. These updated aggregation results of each metric node are stored in the in-memory storage to reduce disk-access delays.

V. EXPERIMENTAL SETUP AND RESULTS

For Experimental setup we used two machines with Intel i7, 2 core and 8GB RAM as two processing nodes. We measured the rate of streaming KPI's processed per second at an average use of 30% of CPU. We used the "thread-pool-dispatcher" as dispatcher service in AKKA actors. Graph in Fig. 7 shows the variation of processing rate with the distribution of number of hierarchies on each node.

VI. CONCLUSION

The applications and profits earned from online analytical processing (OLAP) clearly shows that it is one of the emerging revolutionary technology trend that can be used extensively in managing and analyzing data to get vital information and knowledge. All businesses, big or small, The instant access to information and the apparent

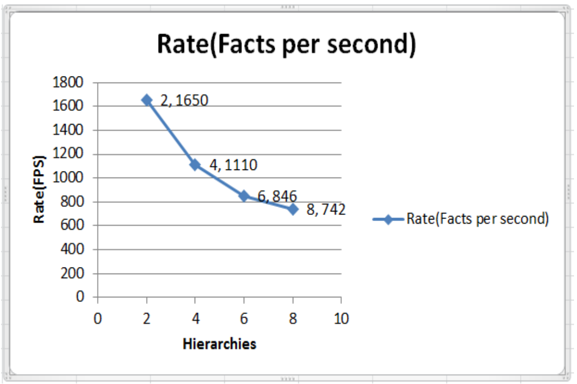

Fig. 7 Results obtained by distributing hierarchies

knowledge gained from the analyzed information is priceless considered against the cost involved in establishing such a setup. Increasing speed of OLAP by distributing its aggregation computation using commodity hardware will increase the cost effectiveness of such applications. Due to use of streaming KPI's it will be online, real-time

REFERENCES

[1] Typesafe, "AKKA actors documentation," Nov. 2014

[2] Z. Jing-hua et al, "OLAP aggregation based on Dimension Oriented storage", IEEE symp. distributed and parallel computing,2012

[3] R. J. Kopaczyk, "Scalable in-memory aggregation," 2011.

[4] J. Bernardino, et al, "Data Warehousing and OLAP: Improving Query Performance Using Distributed Computing",publicpriorart.org/

[5] Golfarelli M., Rizzi S., Cella, "Beyond Data Warehousing: What's Next in Business Intelligence?" Proceedings of the 7th ACM International Workshop on Data Warehousing and OLAP.DOLAP '04, pp. 1–6. ACM, New York (2004)

[6] C. Burnay, I..Jureta et al, "A framework for operationalization of monitoring in business intelligence requirement engineering", J Software System Model, Springer June 2014

[7] http://web.iiit.ac.in/~hemani/files/olap.doc