ISSN:0975-9646

Dhanshri S. Lad et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (3) , 2014, 4057-4061

Different Cube Computation Approaches: Survey Paper

Dhanshri S. Lad #, Rasika P. Saste*

M.Tech. Student , * M.Tech. Student
Department of CSE, Rajarambapu Institute of Technology, Islampur(Sangli), MS, India

***Abstract*— Efficient computation of aggregations plays important role in Data Warehouse systems. Multidimensional data analysis applications aggregate data across many dimensions looking for anomalies or unusual patterns. The SQL aggregate functions and the GROUP BY operator are used for aggregation. But Data analysis applications need the N-dimensional generalization of these operators. Data Cube is introduced which is a way of structuring data in N-dimensions so as to perform analysis over some measure of interest. Data cube computation is essential task in data warehouse implementation. The precomputation of all or part of a data cube can greatly reduce the response time and enhance the performance of on-line analytical processing. There are several methods for cube computation, several strategies to cube materialization and some specific computation algorithms, namely Multiway array aggregation, BUC, Star Cubing, the computation of shell fragments and parallel algorithms. But these techniques have limitation so new MapReduce based approach is used.**

***Keywords*—** Data Cubes, Cube Computation Techniques, Star Cubing, BUC, MapReduce

I. INTRODUCTION

A data warehouse is a subject oriented, united, non volatile and time-variant collection of data organized in support of management decision-making. Several factors distinguish data warehouses from operational databases. Because the two systems provides quite different functionalities and require different kinds of data, it is necessary to maintain data warehouses separately from operational database. In Data Warehouse systems efficient computation of aggregations plays a key role.

The aggregations are mentioned to as GROUP-BY'S. Online Analytical Processing (OLAP)or multidimensional data analysis applications typically aggregate data across many dimensions looking for anomalies or unusual patterns. Computing multidimensional aggregates is a performance bottleneck for these applications. The SQL aggregate functions and the GROUP BY operator are used for aggregation and produce zero-dimensional or one dimensional aggregates respectively. Data analysis applications need the N dimensional generalization of these operators. The Data cube is the N-dimensional generalization of simple aggregate functions, which is introduced by Grey[1]. In OLAP systems, a data cube is a way of organizing data in N-dimensions so as to perform analysis over some measure of interest. Measure is a term used to refer numerical facts that can be algebraic (SUM,COUNT etc.) or non-algebraic (DISTINCT, TOP-K etc.).The data cube is used for conveniently supporting multiple aggregates in OLAP databases. It requires computing group-bys on all possible combinations of a list of attributes, and is equivalent to the union of a number of standard group-by operations.

The cube operator is widely significant to the histogram, roll-up, drill-down, cross-tabulation, and sub-total constructs. The basic cube problem is to compute all of the aggregates as efficiently as possible. Concurrently computing the aggregates offers the opportunity to share partitioning and aggregation costs between various group-bys. The chief difficulty is that the cube problem is exponential in the number of dimensions. In addition, the size of each group-by depends upon the cardinality of its dimensions. As many techniques are proposed for efficient cube computation.

Our paper is organized as follows: Firstly explanation of various cubing algorithms. Next is Limitations of these techniques. Then MapReduce based Approach used for data cube materialization and mining over massive data sets using important subset of holistic measure. And last the conclusion of our study.

II. DIFFERENT TECHNIQUES FOR CUBE COMPUTATION

1) General Cube Computation with Optimizing Techniques : Multi- Dimensional aggregate computation [2]

Authors extended basic sort based and hash based methods to compute multiple group-bys by incorporating optimizations techniques like smallest-parent, cache-results, amortize-scans, share-sorts and share-partitions.

Smallest-parent: This optimization aims at computing a group by from the smallest previously computed group-by. In this, each group-by can be computed from a number of other group bys.

Cache-results: This optimization aims at caching (in memory) the results of a group-by from which other group-bys are computed to reduce disk I/O.

Amortize-scans: This optimization aims at amortizing disk reads by computing as many group-bys as possible, together in memory.

Share-sorts: This optimization is specific to the sort-based algorithms and aims at sharing sorting cost across multiple group bys.

Share-partitions: This optimization is specific to the hash-based algorithms. When the hash table is too large to fit in memory, data is partitioned and aggregation is done for each partition that fits in memory. We can save on partitioning cost by sharing this cost across multiple group bys.

2) Top-Down Approach : Multi-Way array aggregation [3]

The computation starts from the larger group-bys and proceeds towards the smallest group-bys. As show in below Fig 1 :

In this, a partition-based loading algorithm designed and implemented to convert a relational table or

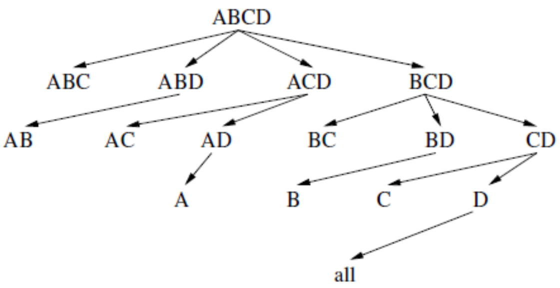

Fig 1: Top-Down Approach

external load file to a (possibly compressed) chunked array. There is no direct tuple comparisons.It perform Simultaneous aggregation on multiple dimensions. In MultiWay array aggregation Intermediate aggregate values are re-used for computing ancestor cuboids .It cannot do Apriori pruning means it cannot perform iceberg cube optimization .

In Multi-Way array aggregation, It partition arrays into chunks (a small sub cube which fits in memory). It uses compressed sparse array addressing: (chunk_id, offset) and compute aggregates in — "multiway" by visiting cube cells in the order which minimizes the # of times to visit each cell, and reduces memory access and storage cost.

What is the best traversing order to do multi-way aggregation?

- Method: the planes should be sorted and computed according to their size in ascending order
- Idea: keep the smallest plane in the main memory, fetch and compute only one chunk at a time for the largest plane.
- Limitation of the method: computing well only for a small number of dimensions .
- If there are a large number of dimensions, top-down computation and iceberg cube computation methods can be explored.

3) Bottom-Up Approach : Bottom-Up Computation (BUC) [4]

BUC is an algorithm for sparse and iceberg cube computation.BUC uses the bottom-up approach that allows to prune unnecessary computation by recurring to A-priori pruning strategy. if a given cell does not satisfy minsup, then no discendant will satisfy minsup either.The Iceberg cube problem is to compute all group-bys that satisfy an iceberg condition.

First, BUC partitions dataset on dimension A, producing partitions a1, a2, a3, a4.Then, it recurses on partition a1,the partition a1 is aggregated and BUC produces <a1,*,*,*>.

Next, it partitions a1 on dimension B. It produces <a1,b1,*,*> and recurses on partition a1,b1. Similarly, it produces <a1,b1,c1,*> and then <a1,b1,c1,d1>. Now, it returns from recursion and produces <a1,b1,c1,d2> etc. After processing partition a1, BUC processes partition a2 and so on as shown in Fig.2 below,

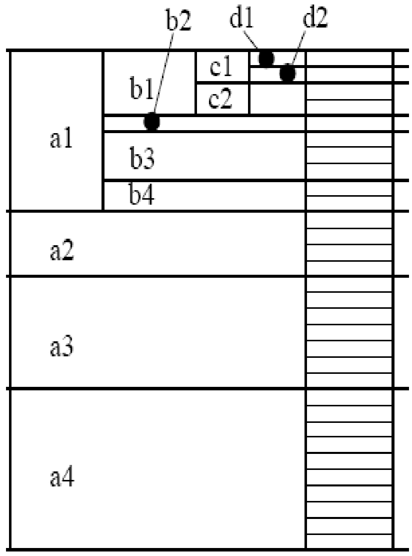

Fig 2 : BUC Partitioning

BUC is sensitive to data skew and to the order of the dimensions processing first most discriminating dimensions improves performance. It shares partitioning costs.BUC does not share computation between parent and child cuboids.

4) Mixed Approach : Star cubing [8]

Star Cubing integrate the top-down and bottom-up methods. It explore shared dimensions .E.g., dimension A is the shared dimension of ACD and AD. ABD/AB means cuboid ABD has shared dimensions AB. Star cubing allows for shared computations .e.g., cuboid AB is computed simultaneously as ABD . Star Cubing aggregate in a top-down manner but with the bottom-up sub-layer underneath which will allow Apriori pruning. It's shared dimensions grow in bottom-up fashion. As shown in Fig 3.

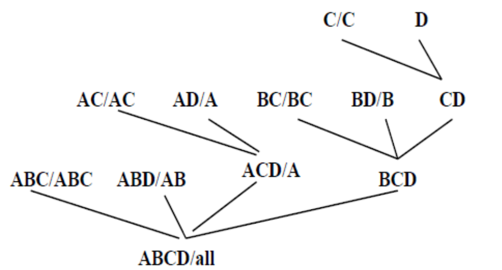

Fig 3 : An Integrating method : Star Cubing

Star-Cubing Algorithm—DFS on Lattice Tree

Properties of Proposed Method

- Partitions the data vertically
- Reduces high-dimensional cube into a set of lower dimensional cubes
- Online re-construction of original high-dimensional space
- Lossless reduction

Offers tradeoffs between the amount of pre-processing and the speed of online computation

Further Implementation Considerations

Incremental Update:

- Append more TIDs to inverted list
- Add <tid: measure> to ID_measure table
- Incremental adding new dimensions
- Form new inverted list and add new fragments
- Bitmap indexing
- May further improve space usage and speed
- Inverted index compression
- Store as d-gaps
- Explore more IR compression methods

5) High-Dimensional OLAP : A Minimal Cubing Approach [9]

In many applications, like bioinformatics, statistics and text processing, datasets are characterized by high dimensionality e.g. over 100 dimensions -> 2100 cuboids in a full cube. As huge cube there is infeasible computation time. Iceberg cube is not an ultimate solution as it cannot be incrementally updated. In this low minsup requires too space and high minsup gives no significant results.

A minimal cubing approach, a new semi-online computational model is based on the computation of shell fragments**.** This method partitions 'vertically' a high dimensional dataset into a set of disjoint low dimensional datasets called fragments. Then, for each fragment, it computes local data cube. In shell fragment efficiency is obtained by using inverted index, i.e. a list of record-ids associated to each dimension value. Given the pre-computed fragment cubes, intersection among fragments is performed online by re-assembling cuboids of the required data cube. It reduces high dimensionality of the data cube to lower dimensionality.Online operations of re-construction of original dimensional space. There is Tradeoffs between the pre-processing phase and the performance of online computation.

***6)* Parallel Approaches [7] :**

Parallel Algorithms are introduced for cube computation over small PC clusters. Algorithm BPP (Breadth-first Writing, Partitioned, Parallel BUC), in which the dataset is not replicated, but is range partitioned on an attribute basis. The output of cuboids is done in a breadth-First fashion, as opposed to the depth-first writing that BUC do. In Depth First writing, cells may belong to different cuboids. For example, the cell a1 belongs to cuboid A, the cell a1b1 to cuboid AB, and the cells a1b1c1 and a1b1c2 belong to ABC. The point is that cuboids is scattered. This clearly incurs a high I/O over-head. It is possible to use buffering to help the scattered writing to the disk. However, this may require a large amount of buffering space, thereby reducing the amount of memory available for the actual computation. Furthermore, many cuboids may need to be maintained in the buffers at the same time, causing extra management overhead. In BPP, this problem is solved by breadth-first writing, implemented by first sorting the input dataset on the "prefix" attributes. Breadth-First I/O is a significant improvement over the scattering I/O used in BUC.

Another Parallel algorithm PT (Partitioned Tree) works with tasks that are created by a recursive binary division of a tree into two sub trees having an equal number of nodes. In PT, there is a parameter that controls when binary division stops.PT tries to exploit a affinity scheduling. During processor assignment, the manager tries to assign to a worker processor a task that can take advantage of prefix affinity based on the root of the subtree.PT is top-down. But interestingly, because each task is a sub tree, the nodes within the sub tree can be traversed/computed in a bottom-up fashion. In fact, PT calls BPP-BUC, which offers breadth-first writing, to complete the processing. Algorithm PT load-balances by using binary partitioning to divide the cube lattice as evenly as possible PT is the algorithm of choice for most situations.

III. Limitations Of Existing Techniques

There are three main limitations in the existing techniques:

1. They are designed for a single machine or clusters with small number of nodes [16]. It is difficult to process data with a single (or a few) machine(s) at many companies where data storage is huge (e.g., terabytes per day)
2. Many of the techniques use the algebraic measure [1] and use this property to avoid processing groups with a large number of tuples. This allows parallelized aggregation of data subsets whose results are then post processed to derive the final result. Many important analyses over logs, involve computing holistic (i.e.,nonalgebraic) measures. Holistic measures pose significant challenges for distribution.
3. Existing techniques failed to detect and avoid extreme data skew.

Extension of cube analysis usage can be avoided by these limitations. There is need of technique to compute cube efficiently in parallel and identification of interesting cube groups on important subset of holistics measures over massive data sets. Hadoop based Mapreduce [8] environment handles large amount of data in clusters with thousands of machines. So MapReduce based technique which supports holistic measures is best option for data analysis. It helps to detect extreme data skew problem.

IV. Mapreduce Based Approach

Mapreduce [19] is rapidly becoming one of most popular parallel execution frameworks. Introduced in 2004 by Google Corporation, it automatically parallelizes task execution, given that users formulate algorithm as map and reduce steps. Data partitioning, fault tolerance, execution scheduling are provided by mapreduce framework itself. Mapreduce was designed to handle large data volumes and huge clusters (thousands of servers). MapReduce is a programming framework that allows to execute user code in a large cluster. Hadoop [18] is an open-source implementation of this framework. All the user has to write two functions: Map and Reduce.

During the Map phase, the input data are distributed across the mapper machines, where each machine then processes a subset of the data in parallel and produces one or more <key; value> pairs for each data record. Next, during the Shuffle phase, those <key, value> pairs are repartitioned (and sorted within each partition) so that values corresponding to the same key are grouped together into values {v1; v2; :::} Finally, during the Reduce phase, each reducer machine processes a subset of the <key,{v1; v2; :::}> pairs in parallel and writes the final results to the distributed file system. The map and reduce tasks are defined by the user while the shuffle is accomplished by the system.

So MR-Cube, a MapReduce-based algorithm was introduced [13] for efficient cube computation and identification of interesting cube groups on holistic measures. Here each node in the lattice represents one possible grouping/aggregation. We use the term cube region to denote a node in the lattice and the term cube group to denote an actual group belonging to the cube region. First we begin by identifying a subset of holistic measures that are easy to compute in parallel than an arbitrary holistic measure. We call them partially algebraic measures. This notion is inspired by common ad hoc practices for computing a single holistic measure from an extremely large number of data tuples.

Then two techniques needed for effectively distribute the data and computation workload. Value Partitioning is used for effectively distribute data for that we are going to run Naïve Algorithm [12].we want to perform value partitioning only on groups that are likely to be reducer-unfriendly and dynamically adjust the partition factor. we adopt a sampling approach where we estimate the reducer-unfriendliness of each cube region based on the number of groups it is estimated to have, and perform partitioning for all groups within the list of cube regions (a small list) that are estimated to be reducer unfriendly.

For effectively distribute computation we use partitioning technique called Batch Area. Each batch area represents a collection of regions that share a common ancestor region.

The combined process of identifying and value-partitioning unfriendly regions followed by the partitioning of regions into batches is referred to as ANNOTATE .So lattice formed is annotated lattice.

Extreme Data skew is most important challenge which occurs if a few cube groups are unusually large even when they belong to a cube region at the top of the lattice (i.e., those with fine granularities).This causes value partitioning to be applied to the entire cube and therefore reduces the efficiency of algorithm. We are looking into the use of compressed counting data structures such as CM-Sketch [14][17] and Log-Frequency Sketch [15] as a solution. After Extreme data skew detection and try to avoid same, we are using MR-Cube algorithm [13] for cube materialization and identifying interesting cube groups.

Propose system architecture is given in below Fig 4.

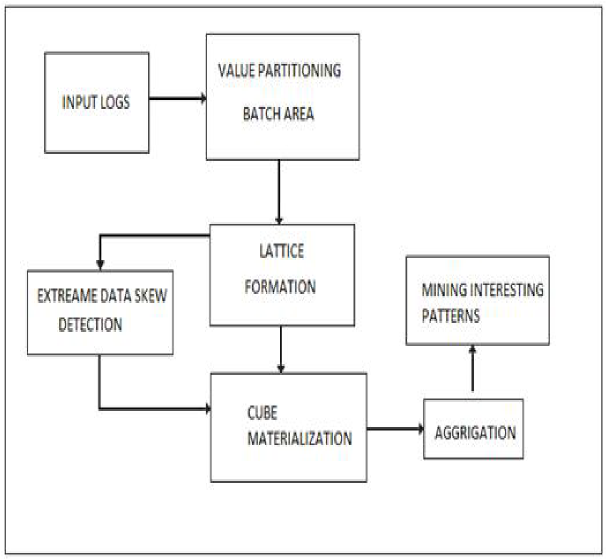

Fig 4 : Proposed System Architecture

In MR-Cube algorithm, the MR-CUBE-MAP emits key:value pairs for each batch area.
In required, keys are appended with a hash based on value partitioning, the shuffle phase then sorts them by key. The BUC Algorithm is run on each reducer, and the cube aggregates are generated. All value partitioned groups need to be aggregated to compute the final measures.

After materializing the cube (i.e., computing measures for all cube groups satisfying the pruning conditions) we can identify interesting cube groups for that cube mining algorithm is used which takes the partially materialized cube. By using the parent group label as the primary key and the group label as the secondary key, measures are clustered based on the parent group level, while ensuring sortedness on the group label. This allows a one-pass discovery of the most interesting group for each parent group-dimension combination. Using above mentioned approach now it is now feasible to perform both large-scale cube materialization and mining in the same distributed framework.

V. CONCLUSION

Efficient Cube computation is important problem in data cube technology. So many techniques are used for computing cube like Multiway array aggregation, BUC, Star Cubing, the computation of shell fragments and parallel algorithms. BUC is sensitive to skew in the data; the performance of BUC degrades as skew increases. However, unlike MultiWay, the result of a parent cuboid does not help compute that of its children in BUC. For the full cube computation, if the dataset is dense, Star Cubing performance is comparable with MultiWay, and is much faster than BUC. If the data set is sparse, Star-Cubing is significantly faster than MultiWay and BUC, in most cases. Parallel algorithm like BPP and PT are designed for small PC clusters and therefore cannot take advantage of the MapReduce infrastructure. Proposed approach effectively distributes data and computation workload .Using important subset of holistic measures we are doing cube materialization and identifying interesting cube groups. MR-Cube algorithm efficiently distributes the computation workload across the machines and is able to complete cubing tasks at a scale where previous algorithms fails.

REFERENCES

[1] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F.Pellow, and H. Pirahesh, "Data Cube: A Relational Operator Generalizing Group-By, Cross-Tab and Sub-Totals," Proc. 12th Int'l Conf. Data Eng. (ICDE), 1996.

[2] S. Agarwal, R. Agrawal, P. Deshpande, A. Gupta, J. Naughton, R. Ramakrishnan,and S. Sarawagi, "On the Computation of Multidimensional Aggregates," Proc.22nd Int'l Conf. Very Large Data Bases (VLDB), 1996.

[3] Y. Zhao, P. M. Deshpande, and J. F. Naughton."An array-based algorithm for simultaneous multidimensional aggregates". In SIGMOD'97.

[4] K. Ross and D. Srivastava, "Fast Computation of Sparse Datacubes," Proc. 23rd Int'l Conf. Very Large Data Bases (VLDB),1997.

[5] K. Beyer and R. Ramakrishnan, "Bottom-Up Computation of Sparse and Iceberg CUBEs," Proc. ACM SIGMOD Int'l Conf. Management of Data, 1999.

[6] J. Hah, J. Pei, G. Dong, and K. Wang, "Efficient Computation of Iceberg Cubes with Complex Measures," Proc. ACM SIGMOD Int'l Conf. Management of Data,2001.

[7] R.T. Ng, A.S.Wagner, and Y. Yin, "Iceberg-Cube Computation with PC Clusters," Proc. ACM SIGMOD Int'l Conf. Management of Data, 2001.

[8] D. Xin, J. Han, X. Li, and B. W. Wah. Starcubing: Computing iceberg cubes by top-down and bottom-up integration. In VLDB'03.

[9] Xiaolei Li, Jiawei Han,Hector Gonzalez "High-Dimensional OLAP: A Minimal Cubing Approach" University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA.

[10] R. Jin, K. Vaidyanathan, G. Yang, and G. Agrawal, "Communication & Memory Optimal Parallel Datacube Construction," IEEE Trans. Parallel and Distributed Systems, vol. 16, no. 12, pp. 1105-1119, Dec. 2005.

[11] Y. Chen, F.K.H.A. Dehne, T. Eavis, and A. Rau-Chaplin, "PnP:Sequential, External Memory and Parallel Iceberg Cube Computation,"J. Distributed and Parallel Databases, vol. 23, pp. 99-126, 2008.

[12] A. Nandi, C. Yu, P. Bohannon, and R. Ramakrishnan, "Distributed Cube Materialization on Holistic Measures," Proc. IEEE 27th Int'l Conf. Data Eng. (ICDE), 2011.

[13] Arnab Nandi, Cong Yu, Philip Bohannon, and Raghu Ramakrishnan "Data Cube Materialization and Mining over MapReduce " IEEE transaction on Knowledge and Data Engineering, vol. 24, no. 10, Oct 2012.

[14] G. Cormode and S. Muthukrishnan, "The CM Sketch and Its Applications," J. Algorithms, vol. 55, pp. 58-75, 2005.

[15] D. Talbot, "Succinct Approximate Counting of Skewed Data,"Proc. 21st Int'l Joint Conf. Artificial Intelligence (IJCAI), 2009.

[16] K. Sergey and K. Yury, "Applying Map-Reduce Paradigm for Parallel Closed Cube Computation," Proc. First Int'l Conf. Advances in Databases, Knowledge, and Data Applications (DBKDA), 2009.

[17] J. Walker, "Mathematics: Zipf's Law and the AOL Query Database," Fourmilog, 2006.

[18] K.V. Shvachko and A.C. Murthy, "Scaling Hadoop to 4000 Nodes at Yahoo!," Yahoo! Developer Network Blog, 2008.

[19] A. Abouzeid et al., "HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads," Proc. VLDB Endowment, vol. 2, pp. 922-933, 2009.