International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2014): 5.611

Survey Paper on Cube Computation Techniques

Amar Sawant[1], Madhav Ingle[2]

[1]PG Student of Computer Engineering, Jayawantrao Sawant College of Engineering, Hadapsar, Pune, India

Abstract: ***Efficient aggregations computation is key part in data mining. But for this the N-dimensional generalization of these operators is required. Data Cube is a way of structuring data in N-dimensions. A data cube is defined as a lattice of cuboids. The computation of data cube can reduce the response time and enhance the performance of analytical processing. There are several methods and several strategies for cube computation. But these techniques have limitation so Map Reduce based approach is used***

Keywords: Data Cubes, Cubing, Cube Computation Techniques, BUC, Map Reduce

1. Introduction

A cube can be considered a multi-dimensional generalization of a two- or three-dimensional spreadsheet. Cube is a shortcut for multidimensional dataset, given that data can have an arbitrary number of dimensions Conceiving data as a cube with hierarchical dimensions leads to conceptually straightforward operations to facilitate analysis. To align the data content with a familiar visualization enhances analyst learning and productivity. In data mining systems efficient computation of aggregations plays a key role.

The aggregations are mentioned to as GROUP-BY's. 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 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 basic cube problem 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.

2. Techniques for Cube Computation

2.1 Multi- Dimensional aggregate computation

Here Authors explained the technique 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 is at computing a group by from the smallest previously computed group-by.
- **Cache-results:** This optimization is at caching (in memory) the results of a group-by from which other groupbys are computed to reduce disk I/O.
- **Amortize-scans:** This optimization is 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 used at sharing sorting cost across multiple group bys.
- **Share-partitions:** This optimization is specific to the hash based algorithms.[2]

2.2 Multi-Way array aggregation

This is top down approach where computation starts from the larger group-bys and proceeds towards the smallest group-bys. In this, a partition-based loading algorithm designed and implemented to convert a relational table or external load file to a (possibly compressed) chunked array. There is no direct tuple comparison. It performs simultaneous aggregation on multiple dimension.[3]

2.3 Bottom-Up computation (BUC)

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.[4]

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.

2.4 Star Cubing

Star Cubing integrate the top-down and bottom-up methods. It explores 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 A-priori pruning. Its shared dimensions grow in bottom-up fashion.[5]

2.5 Minimal Cubing Approach

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.[6]

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.

2.6 Parallel approaches

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.

3. 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 [10]. 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.

4. Map Reduce Based Approach

Map Reduce 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 Map Reduce framework itself. Map Reduce was designed to handle large data volumes and huge clusters (thousands of servers). Map Reduce is a programming framework that allows executing user code in a large cluster. Hadoop 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.

5. 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 Map Reduce 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] 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.

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

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

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.

.