A Workload Assignment Strategy for Efficient ROLAP Data Cube Computation in Distributed Systems

Ilhyun Suh, Department of IT Convergence, Korea University, Seoul, South Korea
Yon Dohn Chung, Department of Computer Science and Engineering, Korea University, Seoul, South Korea

ABSTRACT
Data cube plays a key role in the analysis of multidimensional data. Nowadays, the explosive growth of multidimensional data has made distributed solutions important for data cube computation. Among the architectures for distributed processing, the shared-nothing architecture is known to have the best scalability. However, frequent and massive network communication among the processors can be a performance bottleneck in shared-nothing distributed processing. Therefore, suppressing the amount of data transmission among the processors can be an effective strategy for improving overall performance. In addition, dividing the workload and distributing them evenly to the processors is important. In this paper, the authors present a distributed algorithm for data cube computation that can be adopted in shared-nothing systems. The proposed algorithm gains efficiency by adopting the workload assignment strategy that reduces the total network cost and allocates the workload evenly to each processor, simultaneously.

KEYWORDS
Data Cube, Data Warehouse, Distributed Processing, OLAP, ROLAP, Shared-Nothing Architecture

INTRODUCTION
Data cube is an essential part of analytical processing, and it is widely used for analyzing multidimensional data (Gray et al., 1997). Data cube allows users to explore multidimensional data from various perspectives and at different hierarchical summarization levels. By exploring data cube, users can easily gain insights from multidimensional data. For this reason, data cube plays a key role in On-line Analytical Processing (OLAP) systems (Chaudhuri & Dayal, 1997).

In OLAP systems, multidimensional data are usually gathered from sources that generate a large amount of data. Good examples of such sources are sales recording systems, transaction logging systems, and sensors that report their measurements periodically. The generated data are gathered and integrated into an underlying tier called the data warehouse (Han, Kamber, & Pei, 2011). As the size of the data warehouse grows, the complexity of data cube computation becomes a performance bottleneck and makes it difficult for users to perform analysis at a tolerable time. This is even more problematic in these days because the amount of data being generated is growing explosively.

Since data cube computation requires a high computational cost, given its exponential growth of computation with the growth in the number of dimensions, it is unlikely to cope with massive data using a single machine. In this context, some distributed solutions have been proposed (Chen, 2016).
Dehne, Eavis, & Rau-Chaplin, 2004b, 2004a, 2008; Lee, Jo, & Kim, 2015; Lee, Kim, Moon, & Lee, 2012; Nandi, Yu, Bohannon, & Ramakrishnan, 2012; Sergey & Yury, 2009). These studies exploit distributed computing power using a set of processors in order to achieve high performance. In distributed computing, a task is divided into multiple sub-tasks and distributed to the processors. With regard to the sub-task assignments, having the workload evenly distributed among the processors is a key factor for achieving maximal parallelism and scalability.

There are three architectural choices for building a distributed system: shared-disk, shared-memory and shared-nothing (DeWitt & Gray, 1992). The shared-nothing architecture is known to have the best scalability among the three architectures because it can scale the I/O bandwidth (Babu & Herodotou, 2013). Typically, limited I/O bandwidth becomes a performance bottleneck in data intensive jobs such as data cube computation. Despite the scalable I/O, frequent network intercommunication and massive remote data access between processors can be a critical overhead in shared-nothing distributed systems. This is because network communication is the slowest component among the operations of distributed processing. Thus, reducing network cost can be an effective strategy for improving the overall performance of distributed processing.

In this paper, we present an efficient algorithm for computing relational OLAP (ROLAP) data cube in shared-nothing distributed systems. We focus on ROLAP cube computation because it can be easily incorporated into existing DBMSs. In addition, we focus on shared-nothing architecture since it has proven its scalability and is widely used in distributed processing systems. The main contribution of this paper is as follows:

- We introduce a method for dividing the entire cube computation task into independent sub-tasks.
- We present an algorithm for assigning the sub-tasks to the processors for efficient data cube computation. By reducing the inter-processor network transmission and achieving good load balance among the processors, our algorithm gains efficiency.
- We present additional optimization techniques that can be applied to our algorithm.
- We evaluate the performance of the proposed algorithm with extensive experiments.

The rest of the paper is organized as follows: The second section describes some background information and related work for efficient data cube computation. In the third section, we formalize the problem of distributing the workload of data cube computation over the processors in the shared-nothing architecture. The fourth section describes our algorithm in detail along with further optimization techniques. The effectiveness of our algorithm is demonstrated through various experiment results in the fifth section. In the last section, we present our conclusions.

BACKGROUND AND RELATED WORK

Data Cube

The data cube operator computes GROUP BYs for all possible combinations of specified attributes. For example, consider the relation \( R(A, B, C, M) \) that we use for cube computation. Suppose we use attributes \( A \), \( B \), and \( C \) for grouping and \( M \) as the attribute to be aggregated. We refer to the grouping attributes (i.e., \( A, B \) and \( C \)) as dimensions and the attributes to be aggregated (i.e., \( M \)) as measures. In this case, we compute a three-dimensional data cube where the aggregations are performed using dimension combinations such as \( ABC \), \( AB \), \( AC \), \( BC \), \( A \), \( B \), \( C \), and \( ALL \). Each result view generated by these aggregations is called a cuboid. The union of every cuboid is called full cube. A cell of data cube is defined by a unique combination of dimension values and stores the corresponding aggregated measure values.
When Spatial Analysis Meets OLAP: Multidimensional Model and Operators
Sandro Bimonte, Anne Tchounikine, Maryvonne Miquel and François Pinet (2010). 
*International Journal of Data Warehousing and Mining* (pp. 33-60). 
[www.igi-global.com/article/when-spatial-analysis-meets-olap/46942?camid=4v1a](www.igi-global.com/article/when-spatial-analysis-meets-olap/46942?camid=4v1a)

Data Mining as Part of Intelligence-Led Policing in the Finnish Police
[www.igi-global.com/chapter/data-mining-part-intelligence-led/29069?camid=4v1a](www.igi-global.com/chapter/data-mining-part-intelligence-led/29069?camid=4v1a)