Advanced Data Engineering: Final Assn.

NGUYEN T. Hoang - SID: 15M54097
(ホアン)

Fall 2015, W831 Tue. Period 5-6
Due date: 2016/02/15

Problems

- Discuss at least three approaches and their combinations to improve the performance for preparing a Data Cube in MOLAP under the parallel processing environment.
- Discuss the effect of communication costs in parallel and distributed database operations (including data partitioning, join, aggregation, skew handling, and optimization).
- Discuss the role of RDBMS in the environment of Cloud (with consideration regarding RDF and XML databases).
- Choose a method or a technique in which is most interesting to you during this course, and describe the reason why you select it.
- Write your comments on this course.

Question 1: Data Cube in MOLAP under parallel processing environment

Discuss at least three approaches and their combinations to improve the performance for preparing a Data Cube in MOLAP under the parallel processing environment.

Answer: The process of preparing a Data Cube in MOLAP can be a time consuming process. Locally, there are many methods to optimize the calculation process: Smallest parent, effective use of cache, optimization of disk scan, etc. [8] [3]. In the parallel processing environment, the overall speed of Data Cube preparation is increased but there are many problems related to parallel environment arise such as:

- Inter-processor communication. [3]
- Load balancing (partition skew and cuboid skew). [4]
- Selective Data Cube computation.

Inter-processor communication can be minimized by using sort-based method or hash-based data cube loading [3]. The problem of allocating local and non-local computation is modeled as a graph problem and by using minimum cost matching in a bipartite graph, the minimum cost of inter-processor communication is achieved.

Load balancing. The problem of skew arise when the attributes are not uniformly distributed among the processing elements [4]. This phenomenon results in imbalance of workload in each PE, causing the time for parallel processing equals the time of the slowest PE. There are two type of skew in Data Cube preparation proposed by [4]: Partition Skew and Cuboid Skew. By redistributing the tuple from PE that has larger amount of tuples to a lesser one until a flat distribution is achieved, we can achieve a balanced system. However, this scheme's communication cost is large.

Selective Data Cube computation. Instead of computing all possible "group-by" computation, we can calculate the most relevant combination. This approach may miss out some data, but in practice, the uncommon "group-by" is not worth materializing.

Combination of methods. Each approach mentioned above can be combined in order to achieve a better system. By applying the load balancing scheme and the minimize the inter-processor communication, we can distribute the workload evenly so that the computation take the most out of our hardware facilities.

Question 2: Communication cost in distributed environment

Discuss the effect of communication costs in parallel and distributed database operations (including data partitioning, join, aggregation, skew handling, and optimization). s

Answer: Under different scope, we have different definition of communication cost. In this answer, I will assume the *communication cost* in parallel and distributed database is the cost of moving data from one processing element memory to another without concerning the cost of writing or reading that data into disk. To make the answer compact, I use only the Shared-Nothing model from the Disk Oriented Classification. This model is illustrated in Figure 1 [6]. On another matter, the communication cost depends on the overhead needed to initialize the connection. This number depends on the number of nodes in the network and also the technology. However, the overhead cost can be amortized or overcast by other data intensive operation in the database.

2.1 Data partitioning

There are many scheme for data partitioning: Vertical, horizontal (including Round-Robin, Hash, Value Range), duplicate. This data operation often requires the most of the network, since it sends out almost all the database. I will take an example to illustrate the cost of data partitioning.

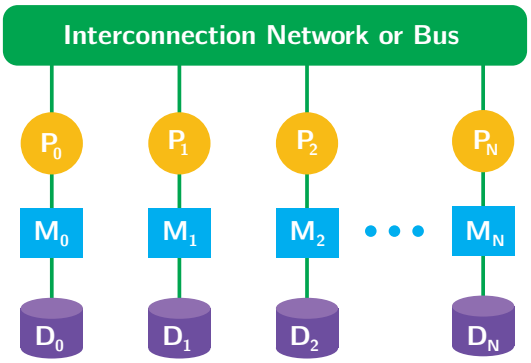

Figure 1: *Shared nothing parallel system.*

Hash partitioning Assuming the relational database is collected in parallel. Which means our database is distributed among all nodes. In this case, denote r_c is the average "correct" rate of the data tuple, which means r_c portion of the database is in its assigned node. The total communication cost for this scheme is:

$$C_{\text{com}} = R \times (1 - r_c)$$

where R is the total size of the database. As we can see, under this scheme of hash partitioning, the number of PEs (N) plays a crucial role in communication cost. If the network has enough bandwidth, the communication cost is divided by N times because of parallel messages sending. The communication cost also depends on random chances. On another perspective, the cost for I/O in this case is also $R \times (1 - r_c)$. However, in practice, the data rate of disk I/O is much larger than the data rate of the network (5Gbps vs. 1Gbps). Therefore, if we have a small number of PEs, then the communication cost is

the major factor in this case. On the other hand, if we have a large number of PEs and a well-connected network, the disk I/O cost might become a bottle neck factor.

2.2 Join operation

Join operation is one of the most intensively used operation in practice. In this analysis, I will consider distributed Sort-Merge Join and GRACE Hash Join.

Sort-Merge Join According to the lecture note [5], the communication cost for parallel processing (ignoring pipeline effect) is:

$$C_{\text{com}} = (2 - \frac{1}{N}) \times \{R\}$$

$$C_{\text{I/O}} = 6 \times (1 - \frac{1}{N}) \times |R|$$

In this case, the communication cost becomes dominant as there are more processing elements. In the case of increasing N, the sort operation will be speed up, but the cost of sending data to the merge node is much higher.

GRACE Hash Join In this scheme, the all-to-all communication cost is denoted as α. For the sake of simplicity, I assume that each PE has enough space to hold $|R|/N$ and $|S|/N$. Because of this assumption, we can reduce the number of communications to the number of processing element N. For this case, network setup time is negligible. There are N phrases for the communication. The communication cost in this case is:

$$\alpha = N \times \frac{|R| + |S|}{N} \times \text{ transfer rate}$$

As we can see here, for GRACE Hash Join, the communication cost for the operation itself does not depend on the network. Therefore, in this case, the communication cost depends on the size of the relation itself and also it is largely affected by the setup time in case each PE does not have enough space to hold the relation.

2.3 Aggregation

The most common aggregation is the "group-by" operator. This operation is commonly used in Data Warehouse and MapReduce. The communication cost of this operation depends linearly on the number of PEs, since there is only $N - 1$ data paths. Each PE performs aggregation locally and then sends the result to the result node. This communication situation is similar to "select" operation or data partitioning operation.

2.4 Skew handling

Skew handling is important for load-balancing in distributed hash join operations. As in lecture note about skew handling [7], here I will focus on the Fine Bucket Method. The total I/O cost for the Fine Bucket Method is:

$$C_{\text{I/O}} = 5 \times \frac{|R| + |S|}{N}$$

This estimation can also be considered as communication cost for the fine bucket method. As we can see here, the cost depends on the number of PEs. In this scheme, communication cost also depends linearly on the data size.

2.5 Optimization

The optimization problem for communication cost depends on each operation itself. Typically, communication cost depends on the amount of data sent. Therefore, if we can minimize the data sent, we can optimize the communication cost. One vivid example of this is the distributed hash join operations. By using hash function, we know exactly the destination of the data tuple, without broadcasting the data to all PEs.

Question 3: RDBMS in the Cloud environment

Discuss the role of RDBMS in the environment of Cloud (with consideration regarding RDF and XML databases).

Answer: Relational databases are facilitated through Relational Database Management Systems (RDBMS), some examples are Oracle, SQL Server, MySQL, Sybase, etc. The data stored in RDBMS is structured, robust, and often the best combination of performance and scalability. According to [1], the characteristic of RDBMS is defined as follow:

- Database contains tables, tables contains columns and rows, and rows are made up of column values. Rows within a table all have the same schema.
- The data model is well defined in advance. A schema is strongly typed and it has constraints and relationships that enforce data integrity.
- The data model is based on a "natural" representation of the data it contains, not on an application's functionality.
- The data model is normalized to remove data duplication. Normalization establishes table relationships. Relationships associate data between tables.

The recent trend for a database system is to use the Cloud, which is a form of distributed database system. The most common reason for this trend is the scalability of the Cloud framework. While RDBMS is scalable within a single node, it is not so trivial concerning

a distributed environment. On the other hand, the database system nowadays must be flexible in term of application. In an organization, here are many applications and services need the support from the database. Due to the limitation of RDBMS in the Cloud environment, there are two rising database scheme: RDF (key-value) and XML database.

RDF is defined by [1] as follow:

- Domain can initially be thought of like a table, but unlike a table you don't define any schema for a domain. A domain is bassically a bucket that you put them into. Items within a single domain can have differing schema.
- Items are identified by keys, and a given item can have a dynamic set of attributes attached to it.
- In some implementations, attributes are all of a string type. In other implementations, attributes have simple types that reflect code types, such as ints, string arrays, and lists.
- No relationship are explicitly defined between domains or within a given domain.

By its definition and characteristics, the RDF is more suitable for the clouds compare to RDBMS. Besides, it is natural fit with code, which means the data in the cloud can be used in many multimedia applications. On the other hand, lacking the integrity and structure compare to RDBMS make it difficult to maintain the database and limit analytic operations. In conclusion, RDF is best in the following scenarios: Data is heavily document-oriented, development is heavily object-oriented, need a cheap solution for data storage, and the foremost concern is scalability.

XML database The is one emerging trend that is using XML and RDF together [2]. An XML document is a collection of data. If being considered as a "database" format, XML has some advantages. For example, it is self-describing, it can describe complex data structure (graph, tree). The characteristic of XML makes it a good fit with RDF in Cloud environment, especially for some Natural Language Processing and Machine Learning task.

In conclusion, RDBMS provides a robust, integrity database management system but its lack of distributed scalability makes it not a good fit for cloud computing. On the other hand, RDF and XML database, by nature a semi-structured database scheme, is a perfect fit for cloud computing if the users' foremost concern is scalability.

Question 4: Most interesting technique in the course

Choose a method or a technique which is most interesting to you during this course, and describe the reason why you select it.

Answer: During the course, I think my favorite topics are the GRACE Hash Join method and the relational database cost estimation model ($|R|$ and $\{R\}$).

Cost model provides a high-level view of the relational database operations. Using simple, straight forward analysis, this cost model can give a big overall picture of the local relational operations as well as distributed operations. The result of the analysis can be used to compare between methods, and give prediction on the expected behavior of the database system.

GRACE Hash Join is a fantastic method for join operation in general and distributed join operation in specific. By using the same hash partition function, all PEs can "communicate" instantly by mathematics. All data communication in this scheme is "necessary", which means there is minimized amount of excessive data communication. I believe this idea and technique can be further apply to many other researches in the future.

Question 5: Course comment

Write your comments on this course.

Answer: I think this is a very informative course, and I enjoyed the process of studying database system. However, there are 2 minor points that I want to point out:

- There is no textbook or reference paper. I think it is necessary to have some reference textbooks or papers that are closely related to the course.
- The course can be more interesting if there is a project and demonstration in class (project based learning). Personally, I prefer working on a project to learning only theory.

References

[1] Is the relational database doomed? http://readwrite.com/2009/02/12/is-the-relational-database-doomed. Accessed: 2016-02-10.

[2] Native xml databases and rdf. http://www.dataversity.net/native-xml-databases-and-rdf/. Accessed: 2016-02-10.

[3] Sanjay Goil, A. C. Parallel data cube construction for high performance on-line analytical processing. *Processing of 4th International Conference on High Performance Computing* (1997).

[4] Seigo Muto, M. K. A dynamic load balancing strategy for parallel datacube computation. *ACM 2nd Annual Workshop on Data Warehousing and OLAP* (1999).

[5] Yokota, H. *Lecture 10-11: Parallel Join Operation*. Computer Science Department - Tokyo Institute of Technology, December 2015.

[6] Yokota, H. *Lecture 9: Parallel Processing and Architecture*. Computer Science Department - Tokyo Institute of Technology, December 2015.

[7] Yokota, H. *Lecture 12: Skew handling*. Computer Science Department - Tokyo Institute of Technology, January 2016.

[8] Yokota, H. *Lecture 3: Data Warehousing*. Computer Science Department - Tokyo Institute of Technology, January 2016.