Dremel - Interactive Analysis of Web-Scale Dataset

Paper Review

Arinto Murdopo

November 8, 2012

1 Motivation

The main motivation of the paper is the inability of existing big data infrastructure (MapReduce-BigTable and Hadoop) to perform fast ad-hoc explorations/queries into web-scale data-sets. In this context, ad-hoc explorations/queries mean on-the-fly queries, which are issued by users when they need it. The execution time of ad-hoc queries is expected to be fast so that users can interactively explore the datasets.

Secondary motivation of the paper is the needs of more user-friendly query mechanism to make data analytic in big data infrastructure easier and faster. Pig and Hive try to solve this challenge by providing SQL-like query language in Hadoop, but they only solve the usability issue and not the performance issue.

Solving these issues allows faster, more efficient and more effective data analytic in big data infrastructure which implies higher productivity for data scientists and engineers who are working in big data analytic. Hence, these issues are interesting and important to be solved.

2 Contributions

The main contribution of Dremel is high-performance technique in processing web-scale data-set for ad-hoc query usage. The mechanism solves these following challenges:

1. Inefficiency in storing data for ad-hoc query.

   Ad-hoc queries most of the time do not need all the available field/column in a table. Therefore the authors propose columnar data model that improve data retrieval performance. It was novel solution because well-known big data processing platforms (such as MapReduce on Hadoop) work at record-structure data model at that time.

2. Performance and scalability challenge in processing the data.

Multi-level serving tree model allows high parallelism in processing the data. It also allows optimization of query in each of the tree level. Query scheduling allows prioritization of the execution. Both techniques solve performance challenge in data processing.

Scalability challenge is also solved since we can easily add more nodes to process more data in multi-level serving tree model. Separation of query scheduler from root server is good because it decouples query scheduling responsibility with job tracking (note that: job tracking is performed by root server). This scheduler model implies higher scalability when the number of leaf servers is huge.

The minor contribution of this paper is the use of actual Google data set in the experiment. This usage gives other researchers insight on the practical magnitude of web-scale data-sets

3 Solutions

3.1 Columnar Data Model

As mentioned before, ad-hoc queries only need small subset of fields/columns in tables. Record-structure data model introduces significant inefficiencies. The reason is in order to retrieve the needed fields/columns, we need to read the whole record data, including unnecessary fields/columns. To reduce these inefficiencies, columnar data model is introduced. Columnar data model allows us to read only the needed fields/columns for ad hoc queries. This model will reduce the data retrieval inefficiencies and increase the speed. The authors also explain how to convert from record-structure data model to columnar model and vice versa.

3.2 Multi-level Serving Tree

The authors use multi-level serving tree in columnar data processing. Typical multi-level serving tree consists of a root server, several intermediate servers and many leaf servers. There are two reasons behind multi-level serving tree:

1. Characteristics of ad hoc queries where the result set size is small or medium. The overhead of processing these kinds of data in parallel is small.

2. High degree of parallelism to process small or medium-size data.

3.3 Query Dispatcher

The authors propose a mechanism to regulate resource allocation for each leaf servers and the mechanism is handled by module called *query dispatcher*. Query dispatcher works in *slot*(number of processing unit available for execution) unit. It allocates appropriate number of tablets into their respective slot. It deals with stragglers by moving the tablets from slow straggler slots into new slots.

3.4 SQL-like query

People with SQL background can easily use Dremel to perform data analytics in web-scale datasets. Unlike Pig and Hive, Dremel does not convert SQL-like queries into MapReduce jobs, therefore Dremel should have faster execution time compared to Pig and Hive.

4 Strong Points

1. Identification of the characteristics of ad-hoc queries data set.

   The authors correctly identify the main characteristic of data set returned from ad-hoc queries, which is: only small number of fields are used by ad hoc-queries. This finding allows the authors to develop columnar data model and use multi-level serving tree to process the columnar data model.

2. Fast and lossless conversion between nested record structure model and columnar data model.

   Although columnar data model has been used in other related works, the fast and lossless conversion algorithm that the authors propose is novel and one of the key contributions of Dremel.

3. Magnitude and variety of datasets for experiments.

   The magnitude of datasets is huge and practical. These magnitude and variety of data-sets increase the confirming power of Dremel solution and proof its high performance.

5 Weak Points

1. Record-oriented data model can still outperform columnar data model. This is the main shortcoming of Dremel, however credit must be given on the authors since they do not hide this shortcoming and they provide some insight on this shortcoming.

2. Performance analysis on the data model conversion is not discussed. They claim that the conversion is fast, but they do not support this argument using experiment data.

3. "Cold" setting usage in *Local Disk* experiment. In *Local Disk* experiment, the authors mention that "all reported times are cold". Using cold setting in database or storage benchmarking is not recommended because the data is highly biased with disk access performance. When the database is "cold", query execution performance in the start of the experiment will highly depend on the disk speed. In the start of the experiment, most of the operations involve moving data from disk to OS cache, and the execution performance will be dominated by disk access.