Data 101/Info 258 Spring 2024 Discussion 14 – April 25th, 2024

MapReduce, Data Cubes, and OLAP Review

Professor Aditya Parameswaran *Covers Lecture(s): 22*

I MapReduce

A MapReduce-style system, such as Apache Hadoop and Apache Spark, requires each stage of computation to be composed of a *map* function and a *reduce* function. The functions have the following type signatures:

$$Map(K, V) \to \langle K', V' \rangle^*$$
$$Reduce(K', \langle V' \rangle^*) \to \langle K'', V'' \rangle^*$$

1. *Word count.* Write a MapReduce program to count the appearance of each word in a set of sentences. Specifically, implement the following two functions:

   - The map function should take a sentence as input, split it into words, then output a dictionary of words and their counts in the document.
   - The reduce function should take a word, and a list of counts from the map tasks. It should output the total count of this word in all documents.

```
def map_fn(document: str) -> dict[str, int]:
    word_count = defaultdict(int)
    for word in document.split():
        word_count[word] += 1
    return word_count

def reduce_fn(word: str, counts: list[int]) -> dict[str, int]:
    return {word: sum(counts)}
```

2. Let us reason about how this program will be executed in a distributed MapReduce environment. Suppose the input strings are:

   - hello world
   - hello world mapreduce
   - mapreduce example
   - hello hello hello

   (a) There will be 4 mappers, each processing one sentence. What are their outputs?

      i. {hello: 1, world: 1}
      ii. {hello: 1, world: 1, mapreduce: 1}
      iii. {mapreduce: 1, example: 1}
      iv. {hello: 3}

   (b) There will be 4 reducers. The *shuffle* stage, handled by the execution system, will dispatch all the mapper outputs to their corresponding reducers. What will be the inputs to each reducer? For each input value, which mapper will it come from?

i. example: {example: 1} from mapper 3
ii. hello: {hello: 1} from mapper 1, {hello: 1} from mapper 2, {hello: 3} from mapper 4
iii. mapreduce: {mapreduce: 1} from mapper 2, {mapreduce: 1} from mapper 3
iv. world: {world: 1} from mapper 1, {world: 1} from mapper 2

(c) What will be the outputs of the reducers?

- {example: 1}
- {hello: 5}
- {mapreduce: 2}
- {world: 2}

3. Consider the following SQL query to compute the total number of page views of each Wikipedia article. The `daily_page_views` table stores the total views of an article by day, so we want to aggregate them all. How can we implement this in MapReduce style? First write in English what the map and reduce functions should do, then implement these functions.

```
SELECT title, SUM(num_views) FROM daily_page_views GROUP BY title ORDER BY title;
```

```
def map_fn(key: str, value: int) -> dict[str, int]:
    return {key: value}

def reduce_fn(key: str, values: list[int]) -> dict[str, int]:
    return key, sum(values)
```

The map function takes as input each record (row) of the table, which includes fields such as *title* and *num_views*. For each record, it will output a pair (title, num_views). Then, in the shuffle stage, the system will send the pairs to their corresponding reducers (one reducer for each title).

In the output stage, the input contains a *title*, and a list of values (i.e. a list of *num_views* associated with the title). It will sum up the values and output a final *num_views* for each title.

This seems simple, but the burden is on the execution system, which must execute this at scale. Think 1 million articles, aggregated over 10 years of data. For example, it must figure out:

- How to run the mappers and reducers over batches of data?
- How to efficiently manage the *shuffle* operation, which must send the mapper outputs to the correct reducers?
- How to run this on hundreds of machines, with possible network and software failures?

II OLAP Concepts

4. (a) What do MOLAP and ROLAP stand for, respectively?

MOLAP is Multidimensional Online Analytical Processing;
ROLAP is Relational Online Analytical Processing.

(b) MOLAP or ROLAP?

- MOLAP. Suitable for data fit into the data cube model
- ROLAP. Supports data that does not fit into the data cube model
- MOLAP. Fast response to queries thanks to precomputation
- MOLAP. Expensive ETL time due to precomputation
- MOLAP. Provides OLAP-specific optimizations in addition to traditional SQL query optimization
- ROLAP. Supports generalized SQL tools
- ROLAP. Needs SQL knowledge to specify new materialized views

5. True or false:

   (a) A fact table typically contains keys that link to dimension tables and numerical measurements that are the results of business transactions or events.

   (b) Dimension tables are usually very large compared to fact tables, as they contain detailed transactional data.

   (c) Fact tables in a star schema data warehouse are often denormalized to optimize query performance.

   (d) Dimension tables in a data warehouse are primarily used for performing aggregations and calculations.

- True. Fact tables contain foreign keys that correspond to primary keys in the dimension tables, along with measurable, quantitative data from business processes.
- False. Dimension tables are generally smaller and contain descriptive data that provide context to the numerical measurements in the fact tables. Fact tables are typically larger as they hold the transactional data.
- True. In a star schema, fact tables are often denormalized to reduce the complexity of data retrieval and improve query performance.
- False. Dimension tables are used to provide contextual information (like time, geography, product details) to the numerical data in the fact tables. Aggregations and calculations are typically performed on the data in fact tables.

6. True or false:

   (a) A data cube allows data to be modeled and viewed in multiple dimensions, which is essential for complex data analysis and decision-making processes.

   (b) A data cube stores data in a flat, two-dimensional structure similar to a traditional spreadsheet.

   (c) In a star schema, the fact table is at the center, surrounded by dimension tables which are directly connected to it.

   (d) In a star schema, dimension tables are typically highly normalized to eliminate data redundancy.

   (e) The snowflake schema is a variation of the star schema where dimension tables are normalized, leading to a structure with more tables and more complex joins.

   (f) Snowflake schemas are typically simpler to query and maintain than star schemas due to their denormalized structure.

- True. A data cube is a multi-dimensional array of values that allows data to be modeled and viewed in several dimensions, making it ideal for complex data analysis.

- False. A data cube uses a multi-dimensional structure, which is more complex than a flat, two-dimensional spreadsheet. It is designed to represent data along multiple dimensions.
- True. The star schema is characterized by a central fact table surrounded by dimension tables, forming a structure that resembles a star.
- False. In a star schema, dimension tables are usually denormalized to simplify the design and improve query performance. Normalization is more characteristic of the snowflake schema.
- True. The snowflake schema involves normalization of dimension tables, which results in a more complex database structure with more tables and joins than in the star schema.
- False. Snowflake schemas, due to their normalized structure, can be more complex to query and maintain compared to the denormalized star schema.

III OLAP in PostgreSQL

Consider the following `sales_data` table:

year	**month**	**sales**
2022	January	100
2022	February	150
2022	March	200
2023	January	180
2023	February	120

7. Write a PostgreSQL query using the `ROLLUP()` function to summarize sales data by year and month from the `sales_data` table. Write the query result.

SELECT year, month, SUM(sales) FROM sales_data GROUP BY ROLLUP(year, month);

year	**month**	**sum**
2022	January	100
2022	February	150
2022	March	200
2022	NULL	450
2023	January	180
2023	February	120
2023	NULL	300
NULL	NULL	750

8. How would you modify the previous query to use the `CUBE()` function instead of `ROLLUP()`? Write the results, and explain the difference between `CUBE` and `ROLLUP`.

SELECT year, month, SUM(sales) FROM sales_data GROUP BY CUBE(year, month);

The result would contain 3 additional rows:

- NULL, January, 280
- NULL, February, 270

- NULL, March, 200

Differences:

- ROLLUP(): It generates a result set that shows aggregates for a hierarchy of values. In this example, ROLLUP provides the total for each month within each year, and totals for each year. It's like adding subtotal rows in a report, moving from the most detailed level up to the summary level. Best used when you need to perform an analysis that follows a clear hierarchical grouping. It's often used in financial reporting or scenarios where a drill-down approach from summary to detail is required.
- CUBE(): It generates a result set that shows all possible combinations of aggregates for the specified group by elements. In this example CUBE provides not only the total for each month within each year and totals for each year, but also totals for each month across all years, and a grand total. Ideal for more complex queries where you need to analyze data across multiple dimensions simultaneously. It's useful in scenarios like sales analysis reports where you might want to see totals by different dimensions (like time, geography, product categories) independently and together.