CLOUD COMPUTING APPLICATIONS

Analytics in the Cloud: Columnar Storage

Prof. Reza Farivar

History

- **Column-stores.** In recent years, there has been renewed interest in so-called *column-oriented systems*, sometimes also called *column-stores, a.k.a. Columnar Storage*
- MonetDB
- VectorWise → Ingres VectorWise → Actian Verctor
- C-Store → Vertica
- SybaseIQ

Columnar Storage

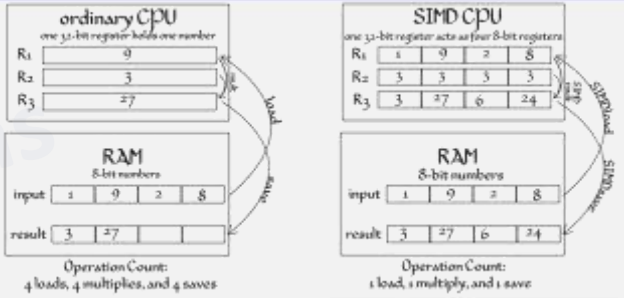
- Traditionally, databases stored records in rows, similar to how a spreadsheet appears
  - e.g. this could include all information about a customer or a retail transaction.
- Retrieving data the traditional way required the system to read the entire row to get one element
- With columnar storage, each data element of a record is stored in a column
- With this approach, a user can query just one data element, such as gym members who have paid their dues, without having to read everything else in that entire record, which may include each member's ID number, name, age, address, city, state, payment information, and so on.
  - Reading the same number of column field values for the same number of rows requires a fraction of the I/O operations and uses a fraction of the memory that would be required for processing row-wise blocks
  - For example, suppose a table contains 100 columns. A query that uses five columns will only need to read about five percent of the data contained in the table.

Columnar Storage

- Column-oriented stores are a good fit for analytical workloads that compute aggregates, such as finding trends, computing average values, etc.
  - Read Optimized
- Processing complex aggregates for when records have multiple fields, but some of them have different importance and are often consumed together
- Note: Column-oriented databases should not be mixed up with wide column stores, such as BigTable or Hbase
  - Data represented as a multidimensional map
  - Columns are grouped into column families (usually storing data of the same type)
  - Inside each column family, data is stored row-wise
  - This layout is best for storing data retrieved by a key or a sequence of keys.

Hardware Optimization: Cache and SIMD

- Disk access pattern
  - One SSD page is 4KB~8KB
  - Row-store: When reading a page, a small number of similar column fields from different rows are loaded
  - Column Store: All the read page are relevant column fields
- Reading multiple values for the same column in one run significantly improves cache utilization and computational efficiency
- On modern CPUs, vectorized instructions (SIMD) can be used to process multiple data points with a single CPU instruction



Hardware Optimization: Compression

- Storing values that have the same data type together (e.g., numbers with other numbers, strings with other strings) offers a better compression ratio
  - Lower information entropy resulting in higher compression
  - We can use different compression algorithms depending on the data type and pick the most effective compression method for each case
- Compression can be automatic by the engine
- Columns shrink and grow independently

```
CREATE TABLE loft_deep_dive (
    aid    INT         ENCODE LZO
   ,loc    CHAR(3)     ENCODE BYTEDICT
   ,dt     DATE        ENCODE RUNLENGTH
);
```

Updates in Columnar Stores

- Update performance in a columnar database is poor
  - Go to every column in order to update one ‘row’
- Many modern columnar databases limit the ability to update data after it is stored
  - Example: Google Dremel paper explains the system as append-only structure
  - No longer a limit of Google BigQuery
  - Performance might be slow
- Redshift
  - Each block is 1MB, blocks are immutable
  - Clone blocks on write to not cause fragmentation
  - Samll writes (~1-10 rows) has similar cost to larger writes (~100K rows)

Metadata

- To reconstruct data tuples, which might be useful for joins, filtering, and multirow aggregates, we need to preserve some metadata on the column level to identify which data points from other columns it is associated with
- Example: AWS Redshift storage nodes have 2.5~3X more storage attached than advertised
  - Used internally for metadata

Column Store File Format

- During the last several years, likely due to a rising demand to run complex analytical queries over growing datasets, we've seen new column-oriented file formats
- Apache Parquet, Apache ORC, RCFile, as well as column-oriented stores, such as Apache Kudu, ClickHouse
  - Parquet: an open source file format for Hadoop
    - Hive, Pig, Impala, Spark
    - Parquet stores nested data structures in a flat columnar format
  - ORC, the Optimized Row Columnar format