Tradeoffs in Massively Parallel Analytical Systems

XLDB Sponsor Talk

9/12/2012

Andrew Lamb (alamb@vertica.com)

VERTICA

Vertica Pitch

- **Know** SQL before you NoSQL
  - Relational data model (SQL) works great for modern hardware and huge datasets
  - Legacy RDBMS implementations were designed and matured for different workloads and hardware
  - Don't take my word for it: try the Vertica Community Edition
- Not all bluster – Vertica has
  - 600+ Customers
  - At least 3 customers with more than 1PB in single instance **production** databases (continually load and query)
  - At least one customer which has loaded (and queries) 10,000,000,000,000 (10T) rows in a single table

Actual Content

- Criteria:
  - MPP (distributed systems, no shared disk)
- Parallel Databases (Vertica/HP, Greenplum/EMC, etc.)
- Hadoop (MapReduce)
- Hive (SQL on top of Hadoop)
- Pig (Dataflow language on top of Hadoop)
- HBase (Key Value store), included out of interest

Cost Structures

- Greater Capital Expense (**CapEx**):
  - Commercial software requires license fees before any production deployment, ongoing tech support pre-paid
  - Vertica and other parallel databases + “Enterprise” distributions of H* systems (e.g. Cloudera, Hortonworks)
- Greater Operational Expense (**OpEx**):
  - Open source + community support means initial CapEx is close to $0; ongoing OpEx is higher
  - Less efficient hardware usage
  - Less mature (but maturing) documentation, integrations with existing applications, user base, etc.
  - Hadoop (MapReduce), Hive, Pig, HBase

Declarative vs Procedural Analytics

- Declarative
  - Specify **what** you want, system figures out **how** to compute it. Better hope you can express what you want in SQL
  - Commonly preferred by non-programmers
  - Vertica + other parallel databases, Hive, *Pig*
- Procedural
  - Explicitly specify computation in your language. Better know how to program.
  - Commonly preferred (at least at first) by programmers
  - Hadoop, *Pig*, HBase

Query Performance vs Query Flexibility

- More Performance, Less Flexibility
  - Bind structure to the data during load time, optimized physical structures, query processing
  - Vertica + other parallel databases
- More Flexibility, Less Performance
  - Bind structure to the data at query time, generic physical structures
  - Hadoop (MapReduce), Pig, Hive
  - HBase and other key-value stores designed for high volume insert/update performance rather than analytics

Latency between load and querability

- High ~ minutes
  - Significant per-job startup overhead
  - Hadoop, Pig, Hive
- Medium ~ 100s of milliseconds
  - Parse / Validate / Optimize incoming data
  - Vertica + other parallel databases
- Low ~ milliseconds
  - Working set is all in memory
  - HBase

Consistency

- Strong
  - ACID consistency via transactions
  - Vertica + other parallel databases
- Limited
  - Strong consistency for a particular key, no cross key consistency
  - HBase
- None
  - Consistency guaranteed by application layer
  - Hadoop (MapReduce), Pig, Hive

Single Row Operations

- Two common types of single row operations
  - Single row inserts / updates
  - 'Point queries': lookup ~1 record based on key
- Batch Oriented:
  - Optimized for large number of rows per operation
  - Vertica + other parallel databases, Hadoop (MapReduce), Pig, Hive
- Key/Value stores:
  - Excel at this workload (it was their design point)
  - Hbase