Tradeoffs in Massively Parallel Analytical Systems
MIT IAP Talk

1/10/2013

Andrew Lamb (aalamb@alum.mit.edu)
MIT VI-2 2002, MEng 2003

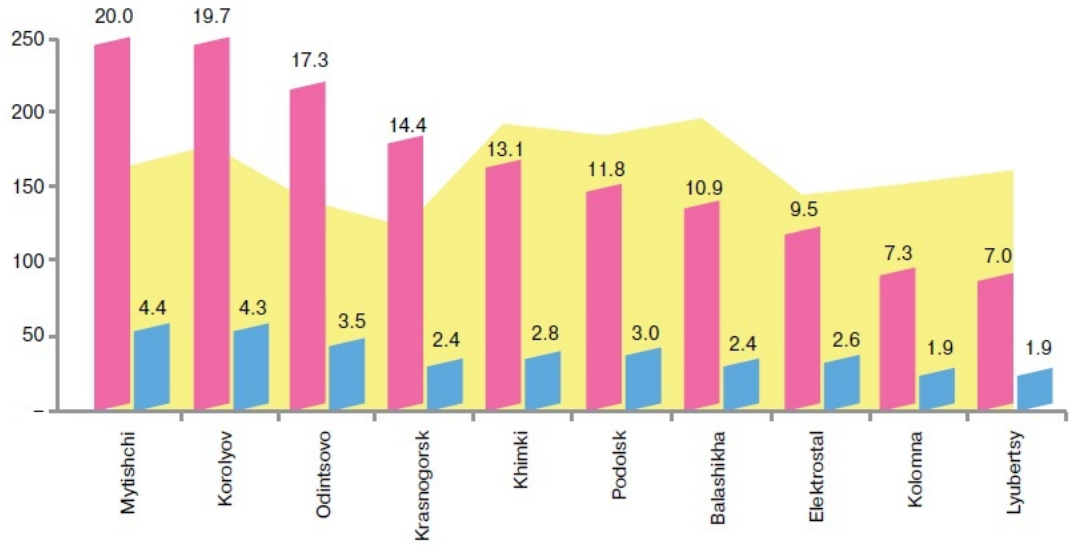
VERTICA

Assumptions

- **Transactional Workloads (Oracle, SQL Server)**
  - Large # transactions per second (thousands)
  - Each transaction involves a few rows on average
  - Example: Credit Card transaction processing

Assumptions

- **Analytic Workloads**
  - Fewer transactions per second (10s)
  - Each transaction touches large number of rows
  - Example: Aggregate sales by region



Analytic Systems: Criteria for Inclusion

Massively Parallel Processing (MPP)

Often suggested as solutions for Analytics or Big Data

Distributed Systems – run on a bunch of 64-bit Linux commodity servers, don't rely on a shared disk or filesystem

Freely Available (at least to try)

Systems Compared – Parallel Databases

- Vertica/HP, Greenplum/EMC, etc.
- Native distributed databases (not grafted on to a shared disk system)
- SQL as native query language
- JDBC/ODCB/etc. Programming APIs

Systems Compared – Hadoop

- "Software library for distributed processing of large datasets across clusters of computes using simple programming model (Map/Reduce)"
- Includes a distributed file system, HDFS
- Open Source clone of Google's GFS/MapReduce administered by Apache Software Foundation

Systems Compared – Hive

- “Dataware house system for Hadoop”
- SQL-like language (HiveQL) on top of Hadoop
- A way to bind structure to data in HDFS
- Compiles queries to Hadoop Map/Reduce jobs

Systems Compared – Pig

- “High level language for data analysis programs”
- Dataflow language, Pig Latin, on top of Hadoop
- Compiles down to Hadoop Map/Reduce jobs

Systems Compared – HBase

- Key Value store which can host “very large tables -- billions of rows X millions of columns -- atop clusters of commodity hardware.”
- Open Source clone of Google's Big Table
- Not really design for Analytical workloads, but included out of interest and common confusion

Now is a good time to point out...

- I work for Vertica/HP
- I wrote a lot of its SQL optimizer
- I am not a NoSQL fanboy
  - Know SQL before you NoSQL

Now is a good time to point out...

- My opinions:
  - Relational model (aka SQL) works fine for Big Data
  - Legacy RDBMS implementations were designed and matured for different workloads and hardware

- My evidence: 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

Moving on ….

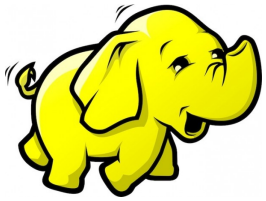
Rest of the talk focused on the tradeoffs

Handling Single Row Operations

- CRUD Operations on a single record at a time: Create Read Update Delete
- Sometimes referred to as 'Point Queries' as they lookup ~1 record based on key
- Don't confuse storing / retrieving a huge number of keys (aka one key per web page) with analysis (e.g. trend identification for human consumption)

Handling Single Row Operations

- Analytic Systems are typically bad for single row operations:
- Vertica + other parallel databases, Hadoop (MapReduce), Pig, Hive
- Optimized for large number of rows per operation
  - Startup time to begin execution going often dominates for small workloads
  - Incremental cost per row is very small


Handling Single Row Operations

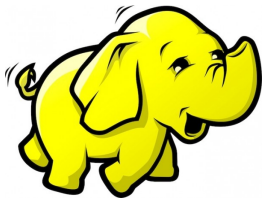
- Key/Value and Document Stores excel at single row operations
- Design point is to handle large numbers of individual CRUD operations and they do it very well
- HBase

Declarative vs Procedural Analytics

- Declarative Analytics: Specify **what** you want, system figures out **how** to compute it.
- Pros:
  - Abstraction frees you from lots of nasty details
  - You don't have to know how to program, only write queries → much larger number of potential users
- Cons:
  - Hope you can express what you want in SQL or equivalent/derivative.
- Vertica + other parallel databases, Hive, ***Pig***

Declarative vs Procedural Analytics

- Procedural Analytics: Explicitly specify computation in your language using APIs provided by system.
- Pros:
  - Can compute 'anything' with SMOP*
  - Commonly preferred (at least at first) by programmers
- Cons:
  - Must be a programmer to use
  - Significant amounts of code for simple questions
- Hadoop, ***Pig***, HBase



* *Simple Matter Of Programming*

Query Performance vs Query Flexibility

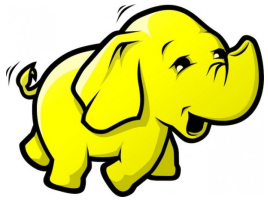
- "Processing Unstructured Data" is a fallacy
- The tradeoff is really **when** you bind structure to the data for processing.

Query Performance vs Query Flexibility

- If you bind structure to the data at **load** time:
- Pros:
  - System can optimized physical structures based on the data structure resulting in faster query processing
- Cons
  - Need to spend time declare your schema before you can even load it, less flexibility
- Vertica + other parallel databases

Query Performance vs Query Flexibility

- If you bind structure to the data at **query** time
- Pros:
  - Lower startup cost: can just load data without defining / determining its structure
- Cons
  - Slower query processing due to general physical data structures. Limited optimization potential (data is opaque until runtime)
- Hadoop (MapReduce), Pig, Hive

Query Performance vs Query Flexibility

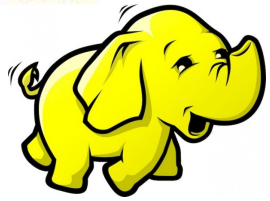
- If you **never** bind structure to the data
- Pros:
  - System is very simple, handles keys and opaque data items
- Cons
  - Optimization potential (other than CRUD) is non-existent.
- HBase and other key-value stores

Latency between load and querability

- How long between when you tell the system to load data and when you can access it via queries?

Latency between load and querability

- High ~ minutes
  - Significant per-job startup overhead
  - Hadoop, Pig, Hive

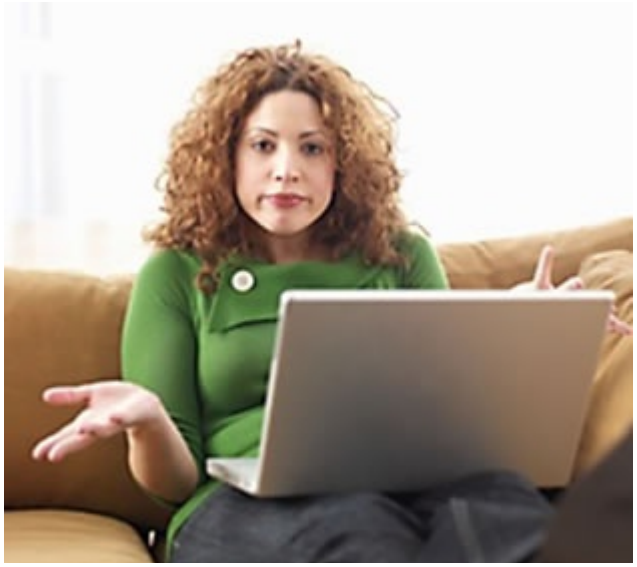


- Medium ~ 100s of milliseconds
  - Parse / Validate / Optimize incoming data
  - Vertica + other parallel databases

- Low ~ milliseconds
  - Working set is all in memory
  - HBase

Consistency

- What happens if two people put data in at once?
- What happens if two people take data out at the same time (do they see the same thing)?
- What happens if someone looks at two different tables at the same time?

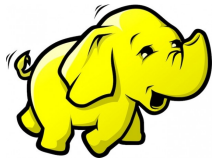


Consistency

- Strong
  - ACID consistency via transactions
  - Vertica + other parallel databases

- Limited
  - Strong consistency for a particular key, no cross key consistency
  - HBase

- None
  - Consistency, if needed, guaranteed by application layer
  - Hadoop (MapReduce), Pig, Hive

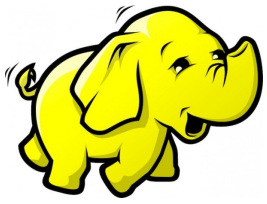


Cost Structures

- Greater Upfront Investment
  - 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)

Cost Structures

- 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


Thank you

- Questions?