Building Blocks for Large Analytic Systems
Andrew Lamb (alamb@vertica.com)
Vertica Systems, an HP Company
Oct 19, 2011
Outline

• Emergence of new hardware, software and data volume is driving the wave of new analytic systems in spite of mature existing systems.

• Common architectural choices when building these new systems

• Choices we made when building the Vertica Analytic Database

• Broadly applicable (and widely used)
Architectural Changes

• Driven by changing
  – Hardware: (really) cheap x86_64 servers, high capacity disk, high speed networking
  – Software: Linux, Open Source
  – Requirements: Data Deluge

• Hard for Legacy Software
  – Compatibility is brutal
  – Linux x86_64 today
  – Solaris, HPUX, IRIX, etc. 10 years ago

• Storage Organization and Processing Principles
• Any modern system should run on a cluster of nodes, scale up/down
• Mid range servers are really cheap (to rent or own)
• Aggregate available resources are enormous and scalable:
  – I/O Bandwidth (Disk and Network)
  – Cores, Memory, etc.
[Storage] Keep Your Data Sorted

- For large data sets, extra indexes are expensive to maintain
- Much better to index the data itself by sorting it
- Easy to find what you are looking for, reduces seeks
• “Sharding” -- Distribute data so you can easily find it again (not round robin)
• Segmenting in analytics layer simplifies app layer
• Round Robin computations won't scale: need to swizzle data around the cluster to do most useful thing
• Replication for high availability: not logs
• Rarely do all fields of a data set appear in analytic queries
• Really don't want to waste I/O for data you don't need
• Columns let you be clever about applying predicates individually, further reducing I/O
• Not appropriate for row lookup or transaction processing systems
[Storage] Write it Once, Don't Modify

• Physical use of disk objects should be write once (append only)

• System should present the illusion of mutability

• Immutable storage drastically simplifies coherence in a distributed system
Random vs Sequential Reads

- Spinning disks are very good at large sequential IOs
- You really don't want to whipsaw the read head (another reason why secondary indexes are bad)
- Especially useful with sorted data
[Processing] Trade CPU for I/O Bandwidth

- Use very aggressive compression, even if it seems/is wasteful (keep getting more cores)
- Example: data type specific encoding, then LZO before actually writing to disk
- Hide additional latency with execution pipeline parallelism
• Bring your processing to the data, not data to the processing
• System should push computation down close to data (even if calculation turns out to be redundant)
• Example: multi-phase aggregation, each phase tries to aggregate intermediates before passing up the memory / node hierarchy
• Give users a declarative query language for most tasks
• Writing procedural code for simple queries is wasteful
• Provide procedural extensions for complex analysis
Conclusion

• Emergence of new hardware, software and data volumes implies certain architectural choices in modern big data analytic systems.
• Commonly observed in new systems
• Vertica (unsurprisingly) features all of them
Introducing Community Edition

• Free version of Vertica
  – help steward better analytics and the democratization of data-
closed source, but open access!
• All of the features and advanced analytics of Vertica Enterprise Edition
• Seamless upgrade to Enterprise Edition
• Limited to 1 TB raw data on 3-node hardware cluster
• Revamping Community area of Vertica’s website for knowledge sharing, third party tools, and code sharing
• Launching academic and non-profit research use program as well
• Sign up at: www.vertica.com/community