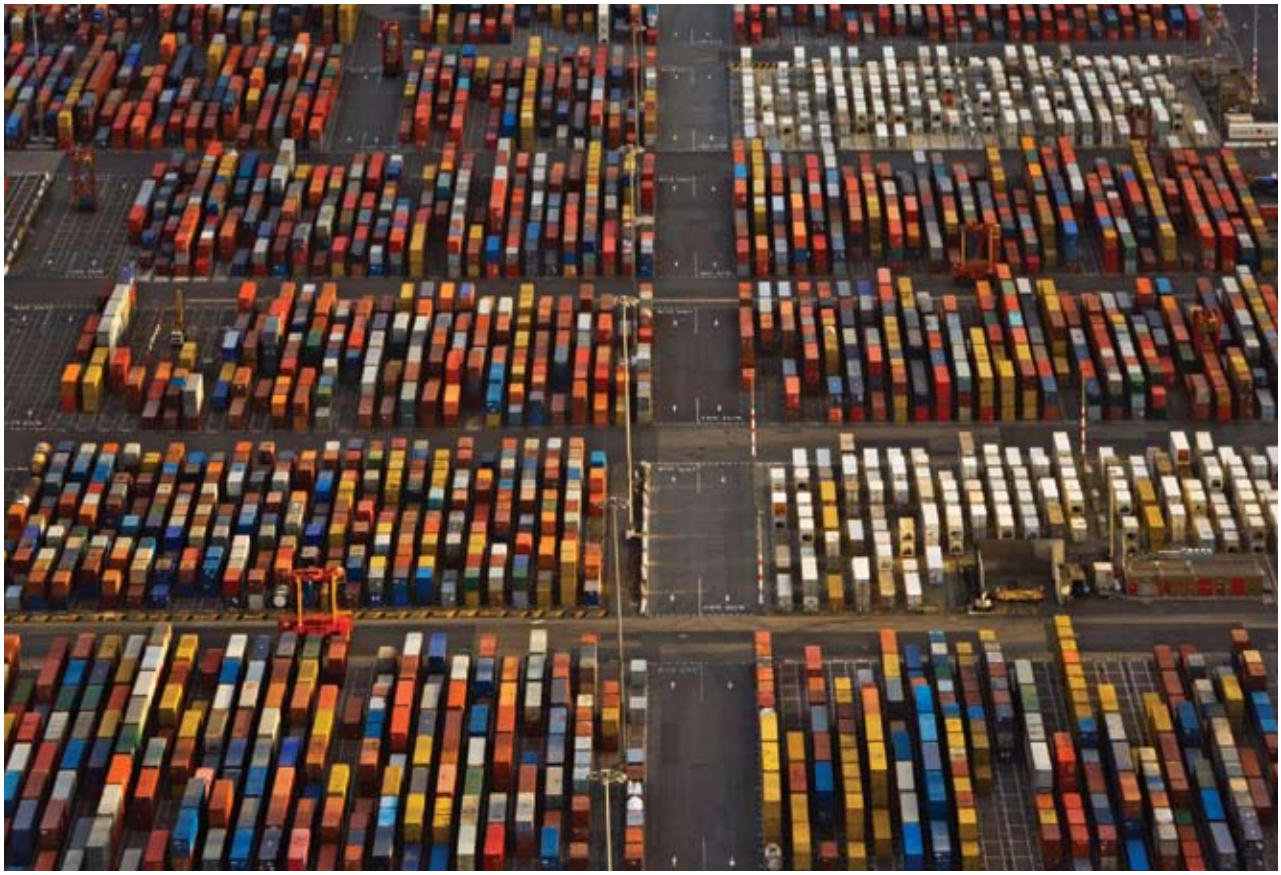
O'REILLY® radar

Release 2.0 11

Issue 2.0.11, February 2009
http://r2.oreilly.com



"Ultimately, big data is more about attitude than tools; data-driven organizations look at big data as a solution, not a problem."

Roger Magoulas and Ben Lorica, from *Big Data: Technologies and Techniques for Large Scale Data*, page 32

Release 2.0
Issue 11, February 2009
ISSN 1935-9446

Published six times a year by
O'Reilly Media, Inc.,
1005 Gravenstein Highway North,
Sebastopol, CA 95472
http://r2.oreilly.com

This newsletter covers the world of information technology and the Internet—and the business and societal issues they raise.

subscription information
Release 2.0
PO Box 17046
North Hollywood, CA 91615-9588
http://dispatchservice.oreilly.com

customer service
1.800.889.8969
1.707.827.7019
r2@oreilly.com

Contents

Roger Magoulas is the Director of Research at O'Reilly. **Ben Lorica** is a Senior Analyst in O'Reilly's Research group.

Big Data:
Technologies and Techniques for Large-Scale Data

Preface: Stories from the Field

You take a leave of absence from an organization known for handling big data to work on the data analysis systems for the Obama campaign. You're faced with one big server and five terabytes of messy voter registration data from multiple sources in multiple formats. You're tasked with optimizing "get out the vote" efforts by finding out who has already voted and removing those names from the call-bank lists used by canvassers—real-time, on election day. With only a few weeks to build the system, you assemble a small team of people comfortable with several aspects of big data management, i.e., the size and state of the data, analytics, and serving the data to many users and many devices.

In the end, while there are problems on election day, you are able to clean 1.6 million voters from the call lists the campaign distributes to canvassers that afternoon, making those lists 25% shorter, on average. Your knowledge and experience with big data management makes a complex task manageable in a tight timeframe with a small team. And, you spare 1.6 million supporters an unnecessary phone call.

Alternatively, you work for a large social networking website and you're tasked with creating an analysis infrastructure that serves a unique group of users. The data is already large and growing fast. There's no real business plan and the data is interesting beyond just its business insights—there's fodder for real sociology research in your social graph. There's no time and no guidance, but everyone thinks the data is important.

You determine that the Hadoop implementation of MapReduce provides the scaling, performance, and flexibility you need. There's no requirement to predefine a schema, so you can just throw data into the Hadoop platform. Your developers and analysts can build and use the various access points, with the help of a few tutorials. Over time, the most effective and repeatable analysis

patterns become clear and you start to develop access tools that support different classes of users, e.g., programming language APIs for developers and SQL-like access for analysts. Your now data-driven organization has the infrastructure to capture all the data generated by your website, run experiments and one-off analyses, and identify opportunities to build more formal, repeatable data analysis interfaces when needed.

We heard these stories from folks doing leading-edge big data implementations, and their experiences aren't isolated or unusual. More and more organizations are facing the challenges and opportunities of working with big data, and they're transforming themselves in the process. We've seen that the organizations that best handle their big data challenges can gain competitive advantage and improve their product and service offerings.

For those organizations facing new challenges and new opportunities regarding big data, we present a roadmap of choices and trade-offs for large-scale data management. There's a lot to make sense of and many competing perspectives. The high-level descriptions and guidance regarding what to consider can inform a deeper dive into making decisions about your big data environment.

Introduction to Big Data

Big Data: when the size and performance requirements for data management become significant design and decision factors for implementing a data management and analysis system. For some organizations, facing hundreds of gigabytes of data for the first time may trigger a need to reconsider data management options. For others, it may take tens or hundreds of terabytes before data size becomes a significant consideration.

We're at the front edge of a data deluge, brought on by new, pervasive data sources. A few years ago, a retailer analyzing thousands of T-shirt sales to discern customer behavior thought it was dealing with big data. Today, social networking companies examine hundreds of millions of personal interactions to identify social trends and relationships, and energy utilities plow through petabytes of sensor data to understand use trends and demand projections. Given the scale of today's data sets, traditional approaches to data acquisition, management, and analysis don't always measure up.

Over the past year, we've noticed a number of faint signals, from the people we talk to and the data we research, that making sense of large-scale data

stores is increasingly interesting and important. What we find most notable is the broad array of organizations, from large enterprises and government agencies down to startups, that are tackling big data—the size of the organization no longer directly correlates with the size of the data challenges. We're also seeing the role of data becoming more central to business strategy. Companies like Google (where analytics is at the heart of how they manage ad revenue), Facebook, (which is attempting to harness the power of data on its social graph of users to develop its business plan), or Twitter (focusing on analyzing its micro-messaging data as the basis for a business model) are examples of companies organized around data insight. Vendors and various open source communities are responding with a new set of tools and techniques to handle this emerging focus on data. We see new big data challenges, growing interest in the topic, and an increasingly diverse set of tools available to address these challenges.

Big data is a big topic. To help make sense of all that big data entails, we divide the topic into three broad activities:

- Data acquisition
- Data management
- Analysis and insight

Data acquisition—whether from data-collecting sensors, increasingly computerized systems, web content, telematics, social networks, or ubiquitous computing—leads to the need to store and manage more data, data that can become valuable with access and iterative, repeatable analysis. This puts data management at the center of big data—scaling to acquire more data and providing fast, convenient access and sophisticated analysis to all that data. In this report we focus on data management as a critical link in the big data story. We'll investigate different approaches to handling large-scale data, describe the technology, identify key trade-offs, and address resource requirements.

Why Big Data Matters

We believe that organizations need to embrace and understand data to make better sense of the world. (We believe it so much that O'Reilly co-sponsored an "unconference" covering Collective Intelligence topics, e.g., data mining and analytics around human behavior.) Big data matters because:

- The world is increasingly awash in sensors that create more data—both explicit sensors like point-of-sales scanners and RFID tags, and implicit sensors like cell phones with GPSs and search activity.

Key Takeaway

Building and making sense of massive databases is the core competency of the information age. Being better at data is why Google beat Yahoo! and Microsoft in search and one reason why Barack Obama beat John McCain. A bad economy accelerates the importance of big data—companies without big data competencies will be left behind.

- Harnessing both explicit and implicit human contribution leads to far more profound and powerful insights than traditional data analysis alone, e.g.:
  - Google can detect regional flu outbreaks seven to ten days faster than the Centers for Disease Control and Prevention by monitoring increased search term activity for phrases associated with flu systems, [M. Helft, "Google Uses Searches to Track Flu's Spread," *New York Times,* 11/11/2008]
  - MIT researchers were able to predict location and social interactions by analyzing patterns in geo/spatial/proximity data collected from students using GPS-enabled cell phones for a semester, [N. Eagle and A. Pentland, "Reality mining: sensing complex social systems," *Personal and Ubiquitous Computing,* Vol 10, #4, 255-268]
  - IMMI captures media rating data by giving participants special cell phones that monitor ambient noise and identify where and what media (e.g., TV, radio, music, video games) a person is watching, listening to, or playing, [J. Pontin, "Are Those Commercials Working? Just Listen." *New York Times,* 9/9/2007]
- Competitive advantage comes from capturing data more quickly, and building systems to respond automatically to that data.
- The practice of sensing, processing, and responding (based on pre-built models of what matters, "the database of expectations," so to speak) is arguably the hallmark of living things. We're now starting to build computers that work the same way. And we're building enterprises around this new kind of sense-and-respond computing infrastructure.
- As our aggregate behavior is measured and monitored, it becomes feedback that improves the overall intelligence of the system, a phenomenon Tim O'Reilly refers to as harnessing collective intelligence.
- With more data becoming publicly available, from the Web, from public data sharing sites like Infochimps, Swivel, and IBM's Many Eyes, from increasingly transparent government sources, from science organizations, from data analysis contests (e.g., Netflix), and so on, there are more opportunities for mashing data together and open sourcing analysis. Bringing disparate data sources together can provide context and deeper insights than what's available from the data in any one organization.
- Experimentation and models drive the analysis culture.
  - At Google, the search quality team has the authority and mandate to fine-tune search rankings and results. To boost search quality and relevancy, they focus on tweaking the algorithms, not analyzing the data.

- Models improve as more data becomes available, e.g., Google's automatic language translation tools keep getting better over time as they absorb more data*.
- Models and algorithms become the focus, not data management.

Big data repositories provide the opportunity, via analysis, for insights that can help you understand and guide your organization's activities and behaviors. You can improve results by combining more data from more sources with more sophisticated analysis and models.

The power of big data needs to be tempered with the responsibility of protecting privacy and civil liberties, preventing sensitive data from getting hacked or inappropriately shared, and treating people generating the data fairly. The insights gained from big data can be used to improve products and customer service, but they can also be used in ways that creep out customers and make them feel uncomfortable or watched. The industry doesn't have all the answers, e.g., academic research shows it's difficult to create truly anonymous data. There are techniques, such as aggregating data beyond the level of an individual, that can protect privacy while still allowing insightful analysis. Although it's beyond the scope of this report to address privacy issues in detail, you'll need to consider them as you work with big data.

Big Data Technologies

Surveying the technology around big data, there are three fundamental strategies for storing and providing fast access to large data sets:

- Improved hardware performance and capacity
  - Faster CPUs
  - More CPU cores
    - Requires parallel/threaded operations to take advantage of multi-core CPUs
  - Increased disk capacity and data transfer throughput
  - Increased network throughput
- Reducing the size of data accessed
  - Data compression

*** FOOTNOTE:**

"How Google translates without understanding; Most of the right words, in mostly the right order," by Bill Softky, *The Register,* May 15, 2007. http://www.theregister.co.uk/2007/05/15/google_translation/

Key Takeaway

Big data is driving new approaches: MPP, MapReduce, column-oriented data are all becoming essential parts of the database toolkit.

The relational model is no longer the only database that matters. For many problems, MapReduce-style processing is superior. What's more, it's easier for many programmers to understand and implement than SQL.

Parallelism is the answer to big data challenges: it lets you "divide and conquer," and it's built to scale.

- Data structures that, by design, limit the amount of data required for queries (e.g., bitmaps, column-oriented databases)
- Distributing data and parallel processing
  - Putting data on more disks to parallelize disk I/O
  - Various schemes to put slices of data on separate compute nodes that can work on these smaller slices in parallel; includes custom hardware and commodity server architectures
  - Massively distributed architectures lead to an emphasis on fault tolerance and performance monitoring
    - As the number of nodes in a cluster increases, failures or slowdowns are inevitable and the systems needs resiliency to recover from faults in a reliable manner
  - Higher-throughput networks to improve data transfer between nodes

These three technology strategies undergird the discussion that follows. Because they are not discrete, mutually exclusive approaches, we don't offer simple "apples-to-apples" comparisons; each separably and in combination can provide effective platforms for handling big data challenges.

Massively Parallel Processing (MPP)

The MPP relational/SQL database architecture spreads data over a number of independent servers, or nodes, in a manner transparent to those using the database. We focus on analytic MPP systems usually called "shared-nothing" databases, as the nodes that make up the cluster operate independently, communicating via a network but not sharing disk or memory resources (see sidebar). With modern multi-core CPUs, MPP databases can be configured to treat each core as a node and run tasks in parallel on a single server.

By distributing data across nodes and running database operations across those nodes in parallel, MPP databases are able to provide fast performance even when handling very large data stores. The massively parallel, or "shared-nothing," architecture allows MPP databases to scale performance in a near-linear fashion as nodes are added, i.e., a cluster with eight nodes will run twice as fast as a cluster with four nodes for the same data.

The collection of servers that make up an MPP system is known as a cluster. Within an MPP cluster there are two topologies:

- Master node as a single point for all cluster connections, for aggregating results and orchestrating activities on the rest of the nodes in the cluster.

Key Takeaway

Massively Parallel Processing (MPP) can dramatically improve query and load performance for all data types. It works with existing relational tools and infrastructure, and you can throw hardware at a cluster to improve performance.

- Peer architecture where all nodes in the cluster can be used to connect to data, and can aggregate results and coordinate activity across the cluster.

The topologies represent trade-offs involving the number of connections, ease of adding and removing compute or data nodes, and availability. Peer architectures offer more flexibility with more coordination overhead than master node architectures.

A key to MPP performance is distributing the data evenly across all the nodes in the cluster. This requires identifying a key whose value is random enough that, even over time, the data does not concentrate in one or a subset of nodes. The MPP databases have algorithms that help keep the data distributed—in practice, using fields like dates or states for distribution may lead to a skewed data distribution and poor performance.

MPP databases are available in three flavors: tightly coupled with proprietary hardware as a data appliance, loosely coupled with a specially configured server platform as a data appliance, and independent as software. Data appliances help reduce installation and configuration complexity for clients and help vendors optimize performance; they are a popular option for MPP databases (see Hardware Options section for more detail).

The focus of modern shared-nothing MPP system vendors is on easy implementation and operations coupled with SQL compliance. MPP systems can create more work for system administrators and designers:

- The need to administer all server nodes in a cluster and a dedicated network—more parts to break and take care of
- More complex database performance monitoring and problem resolution
- Identifying keys that evenly distribute the data across the nodes in the clusters, especially over time and to support joins
- Rebalancing/redistributing data if the number of nodes in the cluster is changed (some MPP systems offer options to help redistribute data to new nodes)

MPP database advantages:

- Fast query and load performance; scalable by throwing more hardware at the cluster
- Standard SQL
  - Easy integration with ETL (extract/transform/load), visualization, and display tools
  - No new skills required for SQL or SQL abstraction layer-savvy developers

MPP Architectures

There are shared-everything architectures for MPP from Oracle (RAC) and IBM. Shared-everything is optimized to support OLTP (transactional) operations and requires synchronization overhead between nodes to ensure data integrity (i.e., transactions are atomic and complete with no collisions). Shared-nothing MPP are optimized for read-intensive tasks associated with analysis.

Shared-nothing MPP databases are not new. There were versions available from a number of vendors in the mid-'80s. In the '90s, NCR's Teradata became the most prominent MPP database vendor, selling an MPP appliance to mostly enterprise customers. Using custom hardware, Teradata's appliance became a popular choice for handling big data needs that extended beyond what one machine could handle. In recent years, a number of vendors have emerged with shared-nothing MPP architectures, including Netezza, Greenplum, Kognitio WX2, and Aster nCluster. These new entrants compete by offering better value, either via running on commodity servers or clever hardware configurations or by removing restrictions on data warehouse design. Shared-nothing MPP databases are not designed for OLTP workloads and they don't have the connections, robust transaction support, and other features associated with transactional systems.

- Generally fast to install, configure, and run
- Parallelization available via standard SQL; no special coding required

Other considerations:

- Performance is affected by the choice of distribution keys; skewed data distributions slow queries
- Hardware costs and energy costs—even with low-cost commodity servers
- Limits to the number of connections (hundreds of users at the upper limit)
  - Managing simultaneous operations can be difficult, as orchestrating parallel operations is complex and analysis environments tend to run many table scans.
- Queries with complex, multi-table joins can run slowly when too much data needs transferring between nodes
  - Attention to distribution keys can improve join performance by co-locating commonly joined data
  - MPP vendors concentrate their engineering effort to reduce and speed up interconnect traffic, and, to co-locate related data on the same nodes
- Data needs to be redistributed and rebalanced when new nodes are added
- MPP databases benefit from multi-core CPUs (parallelism is available for each core), large amounts of RAM, and direct-attached disks
- Clusters of heterogenous nodes are limited by the performance of the slowest node with evenly distributed data
- Check with vendors on recommended network options; some MPP database vendors suggest high-speed networks for optimal performance.

MPP databases are generally sold in two flavors:

- As an appliance with hardware and software bundled together.
- As software that runs on commodity hardware. Software MPP databases can be packaged with commodity hardware and sold as an appliance.

The feature set for MPP databases is evolving to include MapReduce and column-oriented storage, putting the MPP database at the center of a big data architecture that supports multiple operating modes. Here's a summary of recently introduced or planned features for MPP databases:

- Integrating MapReduce with the database
- Adding column-oriented storage options
- Increased monitoring and adaptive operations

- Support for increasing the number of simultaneous connections, via improved queueing algorithms
- Fast data loading via direct writes to files
- Options for rebalancing data
  - Making new compute nodes available before the data is rebalanced
  - Background rebalancing (to avoid a complete dump and reload of data; not a simple task when dealing with terabytes or more of data)
- Options to keep data "in-memory" to increase performance
- More compression options, geared towards reducing the amount of storage needed while minimizing the impact on load and query performance—a tricky balancing act that depends on data and system load

MPP databases are a relatively easy transition for an organization already steeped in relational database technologies and resources.

Column-Oriented Databases

Relational Database Management Systems (RDBMSs) typically store table data as rows, i.e., all the columns associated with a row are stored and retrieved together—regardless of the number of columns in the row used. In a column-oriented database, the data is stored by columns, and, when possible, turned into bitmaps or compressed in other ways to reduce the amount of data stored (see sidebar). Compressing columns reduces how much data needs storing; the combination of compressed data and retrieving only the columns requested speeds query performance by reducing the amount of I/O required and increasing the amount of query data that can be stored in fast memory.

The techniques for reducing the data footprint of column data work best on integer columns with few distinct values. More complex data types and more complex relationships between columns reduces compression opportunities, increasing data sizes and slowing query performance.

Column-oriented databases are relational, using SQL as the language for accessing and manipulating data, and the same set of theory concepts that undergird conventional RDBMSs, i.e., tables can be joined, filtered, grouped, and ordered. The column orientation exists under the covers for most users of the database. Holding the columns together into table entities requires join indices, an extra layer of storage and abstraction compared to traditional row-oriented relational databases. This difference is felt most by designers and administrators, as they need to map query patterns and requirements to column index and compression strategies.

Key Takeaway

Column stores are impressively fast when used with the right type of data, e.g., time series and trial data. They've gained adherents in the past two years, and are overcoming their undeserved reputation for requiring extra engineering effort.

Because the different column index strategies have different performance characteristics, designers create multiple indexes on the same column and let the optimizer pick the best choice. Using multiple indexes increases how much data needs to be stored, which limits the overall reduction in data size from compressing the columns.

Column-oriented databases provide fast query performance for analysis environments with mostly integer data, e.g., time series and bioinformatics, and when most queries focus on a single or small subset of columns. The advantages of column-oriented DBMSs diminish for queries that require many columns and complex table joins due to the extra overhead of bringing all the columns together. The column-oriented database folks we interviewed designed around complex table joins to avoid the impact on performance.

Column-oriented databases speed performance by limiting the amount of data needed to process queries. With less data to move around, disk throughput and latency becomes less critical and network storage devices* becomes an option. Network storage devices provides the following scaling options: 1) increase storage capacity by adding disks to the storage unit; 2) improve disk I/O by adding more disk spindles to increase parallel reads; 3) increase data throughput between the server and storage unit with a high-speed, dedicated network (e.g., fiber channel); 4) add compute capacity with reader servers attached to the storage unit (additional servers connect to a single storage device).

Column-oriented databases have been around for more than a decade, first popularized by Sybase IQ and joined by a number of commercial and open source offerings in the last few years.

Column-oriented databases are designed to be easy to install. Resource impacts include:

- DBAs and designers need to determine the index strategy for each column; vendors provide index analysis tools that recommend index strategies based on column data
- Developers and analysts should be aware that query performance tuning includes limiting the number of requested columns
- Using network storage devices requires more attention to network and storage unit administration

*** FOOTNOTE:**

We are using *network storage devices* as a generic term for two types of shared, networked-attached storage devices: Storage Area Networks (SAN) and Networ- Attached Storage (NAS).

Column-oriented database advantages:

- Fast read query performance:
  - Data size, memory requirements, and I/O are reduced by data compression and by only accessing requested columns
- Standard SQL integrates with relational database tools and interfaces for database design, data access, query, and analysis
- Compression and fast query performance can allow a smaller hardware platform
- Works best with integer data and queries that access one or a few columns,
  - E.g., time series, finance data, sensor data, or bioinformatics
- Compression reduces the amount of disk storage required
  - Best compression with low cardinality (data with few distinct values), integer data
  - Compression gains can be partially offset by the need for extra indexes to support different query requirements and high cardinality data
- Architectures are available for handling many connections

Other considerations:

- Compressing data and index building can slow write performance
  - Vendors all have fast data-loading options and focus engineering resources on improving write performance
- Performance is impacted by large unstructured text, complex join logic, and high cardinality data (i.e., columns with many distinct values)
- Scaling for performance and data size can be complex, requires planning, depends on hardware topology and DBMS options, and can be expensive
  - Column-oriented databases perform best on systems with lots of RAM and multi-core CPUs
  - For systems with a network storage device, adding disk spindles, RAM caching, and high-throughput, dedicated connections between the storage unit and the server can help improve performance
  - Scaling MPP column-oriented databases is similar to scaling row-oriented MPP databases; for MPP architecture, direct-attached disks are recommended

Future features of column-oriented DBMSs

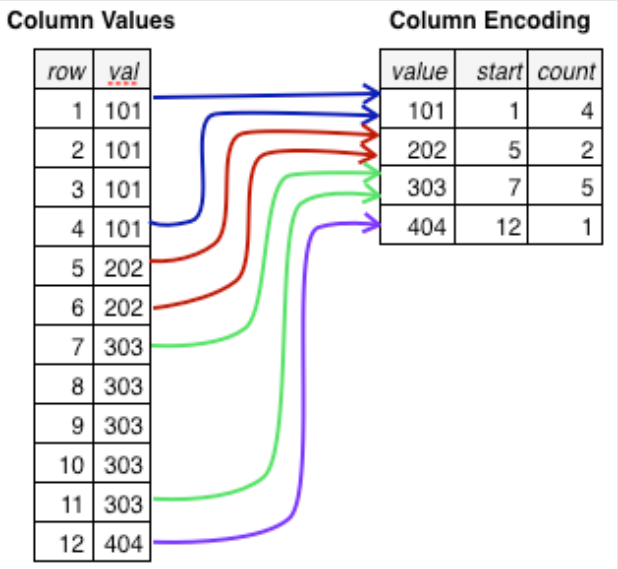
- MPP configurations
- Faster data-loading algorithms
- Increased support for text and unstructured data

Column-oriented DBMSs create performance advantages on a given hardware platform by reducing the amount of data that needs to be processed. With the right data environment, column-oriented DBMSs can be an option to meet performance requirements while maintaining a fit with existing relational database tools and infrastructure—especially for organizations trying to minimize the number of servers required to handle a given data volume.

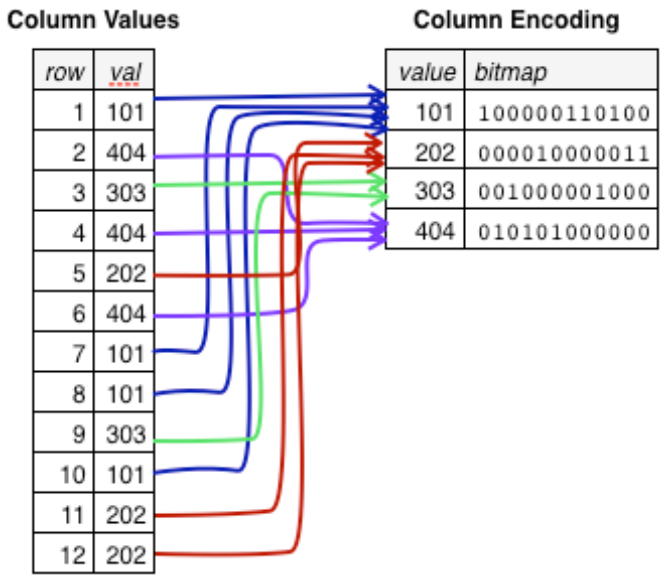
How Column Stores Work

Understanding how columns can be compressed or turned into bitmaps helps explain the technology underlying column-oriented databases or adding column-oriented support to other database schemes. These examples show how column databases manage column data for compression and fast reads (see C-Store: A column-oriented DBMS, Proceedings of the 31st VLDB Conference," 2005, by Stonebraker, Abadi, et al.). Commercial column-oriented databases also have more complex encoding and compression schemes than those outlined below.

For data that can be ordered and has few distinct values, the column can be encoded by storing the value, when the value first appears, and how many times the value appears—reducing the number of rows required to represent a column of many rows to one row for each value. The figure below shows how the original values map to the new (v, s, n) encoding (v -: value, s -: start, n:, number of times v appears). The value 303 is stored as (303, 7, 5), i.e., the first instance of the value 303 is row seven and the value appears five times.



When the column sort depends on a foreign column, the column can be encoded with the value and a bitmap of the relative position of where the value is stored. The bitmaps are typically sparse and can be more efficiently stored and indexed. The following diagram shows the original, unsorted column data and the map to the value bitmap. In this example, the value 303 bitmaps looks like 00100000100, i.e., the value 303 occurs in the 3rd and the 9th position of the bitmap.



These compression techniques lend themselves best to integer data and to columns with low cardinality (i.e., columns with few distinct values). Internal dictionaries and other lookup strategies can be used that process text fields more like integers. Column store strategies don't work as well on columns with many distinct values or on unstructured data.

MapReduce

In the last few years, we've been hearing about—and have been intrigued by—the buzz surrounding MapReduce from startups and large technology companies. Many of the data-intense startups we meet with are using or plan to use Hadoop for some or all of their data management. We see the enthusiasm about MapReduce coming from:

- The massive scale of data Google is processing with their MapReduce infrastructure as proof the technology works

Key Takeaway

MapReduce is the next, new thing in big data, scaling to meet the biggest data processing environments, e.g., petabytes at Google, Yahoo!, Facebook.

- The success of Hadoop at Facebook, Yahoo!, *The New York Times* and others
- The availability of Amazon Web Services (AWS) as a convenient and cheap platform for trying MapReduce
- Small organizations looking for an affordable, scalable way to manage and analyze big data
- The expense of commercial big data products

MapReduce refers both to a style of programming and to a parallel data processing engine for managing large-scale, distributed data.

As a style of programming, the concept is based on combining functions (*map* and *reduce*) common to many programming languages. The map function performs filtering or transformations before creating output records as a key/value pair. A split function, most typically a hash (but any deterministic function that guarantees the data always lands on the same server will work), distributes these records for storage to the servers that make up the system. The reduce function performs some type of aggregate function on all the records in the bucket associated with a key. The map functions can be distributed to run on different nodes in a cluster, with each map given a portion of the input data to process. By analogy to SQL, the map is like "group by" and the reduce is like an aggregate function (e.g., sum or count) for an aggregate query.

MapReduce programming can be applied in other contexts—for example, against a distributed relational database (both Aster Data and Greenplum have released versions of their MPP databases with MapReduce functionality) or key/value pair data stores like CouchDB.

As a parallel data processing engine, MapReduce is most closely associated with Google's implementation and with Hadoop, the open source clone of MapReduce supported by Yahoo!, Facebook, Cloudera, and others.

Google implemented MapReduce as a stack that was then used as the inspiration for Hadoop. Since much of the Hadoop documentation references the Google MapReduce stack, we describe the stack in more detail:

- Google File System (GFS)
  - GFS triplicates all data across a cluster. If a node in the cluster becomes unavailable, the data is then automatically replicated.
- BigTable: a distributed data storage system with a multidimensional data structure
  - BigTable uses the terms *rows* and *columns* differently than they are used in relational databases; the rows and columns are really elements of a map (hash in Perl or Ruby, dictionary in Python, associative array in PHP,

object in JavaScript), and BigTable is described as a multidimensional sparse map. A spreadsheet analogy may help—values are accessed by using row and column as an index. All columns are timestamped to allow data versioning, with automatic retrieval of the most recent items. Users often serialize fields into a single column, creating a simpler key/value pair data structure that they can deserialize on retrieval.

  - A complete description of the BigTable data structure is beyond the scope of this report. For more in-depth information, see "Bigtable: A Distributed System for Structured Data" by Chang, Dean, et al. *(http://labs.google.com/papers/bigtable.html)* for the official explanation, or, for a simpler explanation see Jim Wilson's "Understanding HBase and BigTable," at *http://jimbojw.com/wiki/index.php?title=Understanding_Hbase_and_BigTable*.

- MapReduce—a client for performing parallel MapReduce on data stored in BigTable tables on GFS
- Sawzall—a higher-level query and analysis language that runs on top of MapReduce, simplifying filtering, aggregation, and statistics analysis; analogous to SQL
- Workqueue—schedules tasks and restarts jobs that fail
- Chubby—system for coordinating distributed applications, including configuration and synchronization

The MapReduce platform includes node monitoring, fault detection, and queuing processes that help manage MapReduce jobs. From the user perspective, MapReduce provides a platform for operating on many data items in parallel while isolating the user from the details of running a distributed program , i.e., data distribution, replication, fault tolerance, and scheduling. Google also uses the proprietary compression algorithms BMDiff and Zippy to shrink the size of data stored.

Hadoop, an open source Java framework for running applications in parallel across large clusters of commodity hardware, was created by Doug Cutting, the developer of Lucene (search tool) and Nutch (distributed web crawler). Cutting was influenced by what he learned about GFS and MapReduce in 2004 – 2006, and the Hadoop project grew out of his work on Nutch. In 2006 Doug was hired by Yahoo!, got a team of engineers, and started the open source Apache Foundation Hadoop project to give Yahoo! the same type of distributed

processing that Google was enjoying with their MapReduce platform. By early 2008 Hadoop was able to hit web-scale distribution (http://research.yahoo.com/files/cutting.pdf).

Hadoop has become the primary MapReduce platform used outside of Google. A Hadoop Summit in March, 2008, drew more than 400 people. Over half were running Hadoop, with at least 15% running Hadoop on a minimum of 100 nodes. Hadoop has one or more corollaries to the Google MapReduce stack , as shown in the following table.:

Google MapReduce	Hadoop	Notes
GFS	Hadoop Distributed File System (HDFS)	Keeps triplicates of all data distributed across cluster nodes
BigTable	HBase	Alternatives include HyperTable
MapReduce	Hadoop MapReduce	Hadoop Core bundles MapReduce and HDFS
	Hadoop Streaming	Provides Hadoop access to any stdin/stdout binary, includeing interpreted languages like Shell Script, Python, Rails, and Perl
Sawzall	Pig (Pig Latin)	Data flow and execution language
	Hive	Facebook open source query and analysis framework built on top of Hadoop
Workqueue	JobTracker	Schedules and manages jobs
Chubby	ZooKeeper	Coordination system for distributed applications, including configuration and synchronization
Compression: BMDiff, Zippy	zlib/gzip, LZO, bzip2	Google compression optimized for processing speed, not compression

Hadoop has a vibrant and engaged user and developer community. We expect Hadoop and related offerings to continue to improve, add functionality, and generate new businesses that support the Hadoop community.

Implementing MapReduce as a methodology and as a data processing engine is best served by considering your staff's collective skills and experience:

- Hadoop requires learning new system administration skills for installing, configuring, monitoring, tuning, and maintenance.
- Administering multiple servers; adding new servers to scale
- Simpler, more flexible data structures
  - MapReduce table structures should be familiar to developers who work with programming language data structures

    - Developers and analysts most familiar with relational databases will need to learn the different MapReduce data structures
- Getting buy-in from technical resources and/or finding experienced MapReduce resources can help with adoption
- Training and/or pilot projects can help staff learn MapReduce and new approaches to data management
- Programming resources may be required to build high-level user interfaces to replace RDBMS-oriented tools

MapReduce advantages:

- Fast performance enabled by parallel processing on distributed data
- Transparent, fault-tolerant execution of parallel data-handling processes
- Built-in resiliency/fault tolerance coherent with scaling to large clusters
- Scaling to large-scale data volumes
    - Potential for thousands of nodes
    - Largest Hadoop clusters have 2,000 nodes; Google's MapReduce is rumored to use more than 10K servers (perhaps many more) for MapReduce jobs
- Scaling and performance on commodity hardware
    - Can increase and decrease size of cluster via reconfiguration
    - Proven cloud computing options
- Simpler, more flexible data structure
    - No requirement to predefine data structure design
- Parallelism and resiliency come free to developers and analysts, i.e., no explicit coding for parallelism is required

Other considerations:

- Hardware and energy costs rise as the number of servers increases
- Administration, design, and analysis infrastructure tools are available, but still maturing
    - RDBMS-oriented DBA, design, and query tools don't work with MapReduce
- Less compute efficiency compared to more heavily indexed alternatives in some circumstances
    - Depends on data structure design and query complexity
    - Can lead to needing more servers or servers that consume more energy per process

Future features of MapReduce:

- More SQL-like and easier-to-use query and analysis tools (see Hive)
- Increased scaling
  - Hadoop has a design target of 10K node clusters in 2009
- Tools to simplify administration, configuration, installation and monitoring processes

The organizations we spoke with who use MapReduce had a consistently high opinion of their experience. They liked the scaling and flexible data structures. Their developers and analysts were quickly trained and up-to-speed and stayed engaged with the data—and, remain enthusiastic about using MapReduce. MapReduce complements an experimental approach towards data, i.e., loading raw data into simple data structures and running ad hoc and one-off analysis until query patterns emerge that can be turned into more formal and easy-to-analyze data structures. This experimental, discovery-oriented approach helps make MapReduce a good fit for organizations trying to make data central to business strategy and decision making. Drawbacks noted include "trial and error" fiddling to get configuration optimized and maturity issues around documentation and features—all considered relatively insignificant and not a hindrance to adoption.

Key Technology Dimensions

There's a lot to keep in mind when investigating big data technology. Along with careful attention to data, staff skills, and usage, we recommend considering the following technology dimensions to make the best decision for your organization.

Key Takeaway

For most databases, a single server—properly configured—is enough. It's a big leap to move from one to two servers, but once you do, growing a cluster is relatively easy.

Commodity hardware works, rides the mass market innovation curve, and avoids vendor lock-in.

Single Server and Distributed Data/ Parallel Processing Clusters

Single Server

A single box with one or more single or multi-core CPUs and direct-attached or network disk storage.

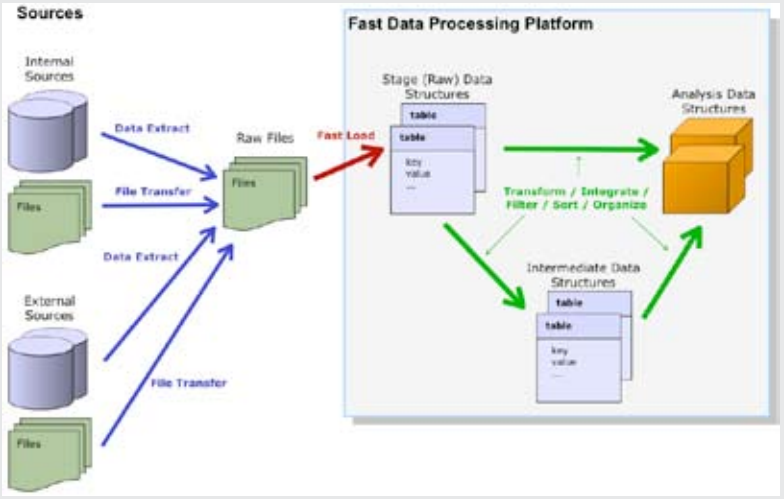
Assumptions:

- Server performance and data capacity continues to improve, i.e., CPUs gain cores and capacity (per Moore's law), hard disk density increases, memory density increases, and other factors make high-end servers powerful enough to handle ever-increasing big data loads

A Data Architecture for Fast Platforms

The data management tools and techniques covered in this report are designed to enable fast processing of large-scale data stores. To take advantage of this speed and reduce data load time, as much of the data processing as possible should be moved to the fast platform (see diagram):

- For data coming from external systems, use the fastest extraction process available (typically reducing load on source system):
  - Files
    - Compress large files before transferring, reducing bandwidth required to send
  - Tables
    - Pull data in the most raw form
      - Don't pull BLOBs (Binary Large OBjects) unless required for analysis
    - When possible, use bulk unload or dump processes (e.g., Oracle SQL*Loader, SQL Server: bcp, MySQL: mysqldump) to create the most compact data set to transfer
      - Consider configuring source to make bulk unload possible, e.g., use partitioning to create a source table containing only the data to be extracted
    - When bulk unload processes are not available or practical, use single table queries and minimal filters
- Configure a fast data processing platform for big data
  - Depending on the tool, generally configure to process data as big chunks
- Use fast load processes to load raw data from files into simple data structures that can then be organized for analysis
  - For distributed systems like MPP databases and MapReduce, use tools and processes that take advantage of parallel data loading
- Run processes that integrate, clean, transform, and organize into the target data structures for analysis on the fast platform
- Tailor target data structures for easy and fast analysis by organizing and aggregating data for common queries, and, where appropriate, sorting and indexing
  - Spend more time transforming and organizing data to help speed queries
- Organize data for different classes of users by providing:
  - More complex data structures for analysts
  - Simpler data for reports and casual users
- Keep raw and intermediate data available to query so that you'll have the flexibility to meet unanticipated analysis needs
  - Analysis against raw and intermediate data structures can become requirements and prototype designs for new, persistent, formal analysis data structures



- Data size and growth rates are predictable within the operating parameters of a single server
- High performance query options exist

Advantages:

- No additional system administration burdens
- Commonly deployed and well-known system monitoring tools can be used
- Fewer machines to break
- No special network topologies or extra network components to run

Downsides:

- Few expansion options if a single server can't handle unanticipated data volumes or performance requirements
- No opportunity to learn how to run and manage parallel data environments—an organizational skill that may become critical if new data sources or styles of data analysis become key to gaining insight or competitive advantage
- Performance degrades as the amount of data increases

Other considerations:

- You'll get the best performance by installing as much RAM as the system can take (and you can afford)
- You can use a network-attached disk (SAN, NAS) to grow disk storage beyond what can be stored in a single box, but at the cost of increased network latency and more system administration work

For many organizations, a single server is the appropriate choice, especially if data size and growth are well understood and within the performance envelope of a single server, and access to technical resources is limited. Column-oriented data management may be an appropriate choice for single server architectures, both for the query performance boost and for the data compression (via bitmaps) they provide.

One method to increase the capacity of a single server is to take advantage of the parallel processing available in Graphic Processing Units (GPU). These graphic cards can contain more than 100 cores (and the number keeps growing) that can be used to handle math-oriented tasks in parallel, e.g., sorting.

Distributed Data/Parallel Processing

A cluster of commodity servers or nodes (including virtual machine cloud-based nodes) with a direct-attached disk.

Assumptions:

- Clusters are made of commodity servers
- Cluster nodes can be a server or a CPU core in a multi-core CPU
- Query performance is a near-linear function of degrees of parallelism in the cluster (degrees of parallelism = number of cores/nodes in the server)

Advantages:

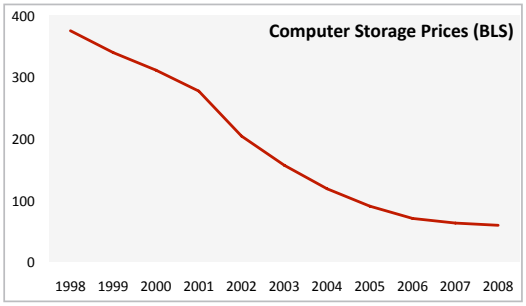
- High performance query and load by distributing data across multiple nodes in a cluster
- Near-linear scaling by adding nodes to a cluster
- Near-unlimited scaling potential by adding nodes to a cluster
- Once you've made the jump from one box to two boxes, making the jump to n boxes is easy.

Downsides:

- System administration for multiple nodes
- More difficult to monitor the cluster
- More hardware to break
- Generally higher cost than the single-server option
- Performance relies on evenly distributed keys
- Connection count limits, limits that can be increased with peer architectures, and improving job queueing algorithms
- Performance can degrade when complex operations introduce too much network traffic

Other considerations:

- You'll get the best performance with a dedicated network
- Direct-attached disk is recommended



Hardware Trends

Those planning big data implementations also need to consider the following trends in hardware performance and cost:

- Moore's law continues to drive increased transistor density in CPUs—however, CPUs are not getting faster, there are just more of them. Multi-core

CPUs have become the rule in big data hardware, and we expect the number of cores to continue to grow.

- Disk prices, while continuing to decline, have not declined much in the last three years, especially when compared to the trend before 2006 (see the Bureau of Labor Statistics (BLS) price chart above).
- Energy consumption and the cost of runnning servers is an increasingly important factor for those making decisions about large systems. Energy calculations for complex and large clusters of servers are beyond the scope of this research; however, energy costs are likely to influence big data hardware platform decisions, especially systems that consist of clusters of many servers.

When we began our research for this report, we assumed that we would focus on distributing data and on parallelism as the primary vehicle for handling big data. During our interviews, we were reminded that, while more and more companies have enormous data stores and the amount of data continues to grow quickly, the chart of data-storage needs by organization looks like a power law relationship—with a "long tail" of organizations having relatively modest data size requirements. For many organizations, a one-server solution may be appropriate. With Moore's law improvements in CPUs and increased disk storage and software advances, single-server platforms can continue to meet expanding big data management needs.

At O'Reilly we run both single-server and MPP platforms for our analysis data stores. The data store that drove us to consider an MPP architecture is a large collection of messy job-post data, now totaling over one billion rows and 5 Tbs of data. The rows include both structured and unstructured data; we focus our analysis on the unstructured job description data using regular expressions and rule bases to investigate technology trends. When we started to outgrow our single-server platform, we had some standard queries that ran for 10 hours. When we tested the MPP database we eventual moved to, the same query ran in six minutes—with no attention to tuning. We've run on the MPP platform for over three years now and spend little time on tuning—which lets us focus our attention on query logic and interpreting results.

Direct-Attached Storage and Network Storage Devices

We touched on this topic earlier, when we compared single-server to distributed systems. Direct-attached storage, most typically SATA (Serial ATA), provides better data throughput, but with a cost—the amount of storage per server is

limited by physical space in the server box for the disk. Direct-attached disk expansion units can help increase the amount of data storage available to a node, up to a limit.

Network storage devices allows more flexible data growth than direct-attached disks do. These storage devices can be added to when more storage is needed, including adding new cabinets. The cost is slower performance (due to network latency) and more work for the system administrators managing the storage devices. Network latency issues can be somewhat mitigated by faster high-throughput networks.

Direct-attached disk is recommended for MPP and MapReduce clusters. Network storage is an option for single-server systems if it's coupled with column-oriented DBMSs (storing appropriate data types and data structures for column-oriented DBMS) and servers with plenty of RAM, all of which reduce network traffic to the storage unit.

Data Partitioning

The big data management tools described in this report that distribute or reorganize data do so below the level of the database designer or the developer—the systems take care of the dirty work, leaving a clean interface to those accessing the data. Independent of row or column orientation, other options allow data structure designers to partition data, either by creating tables made of subsets of columns, called vertical partitioning, or by creating tables made of subsets of rows, called horizontal partitioning or sharding.

Partitioning strategy is best developed with a good understanding of actual or expected query activity, as the goal of partitioning is to reduce the amount of data accessed for the most commonly run queries.

When tables are partitioned, some type of navigation logic may be required to guide queries to the best tables. There are databases with optimizers that understand horizontal partitions and will only pull from the partitions needed to satisfy the query.

Partitioning options need thorough testing and vetting to ensure that do, in fact, improve performance.*

*** FOOTNOTE:** See "Column Stores vs. RowStores: How Different Are They Really?", by Daniel J. Abide, Samuel R. Madden, and Nabil Hachem, SIGMOD 2008, *(http://cs-www.cs.yale.edu/homes/dna/papers/abadi-sigmod08.pdf);* for more details on the performance considerations for partitioning in row and column databases.

MapReduce and SQL

Although SQL has a long history and a large installed base, other approaches such as MapReduce (see upcoming sidebar) are gaining traction among developers. As prominent open source developer and MySQL architect Brian Aker told us: "MapReduce is new and you get to work with a large cluster, plus you don't need to use SQL." Many developers who are comfortable with procedural and object-oriented programming languages are not proficient in nor comfortable with SQL. Those who appreciate the set-theoretic underpinnings of SQL queries eventually embrace it, but they tend to be in the minority. Most programmers prefer to use (object-relational) abstraction layers such as Hibernate or SQLAlchemy. Besides the difficulty of grasping SQL, there are other reasons why developers are exploring new options for querying large data sets. As an example, UC Berkeley professor and leading database researcher Joe Hellerstein observed that the growing popularity of machine learning and data mining algorithms have some developers yearning for alternatives to SQL that support the types of computations (e.g., matrix, vector) common in those domains.

As a non-procedural language, SQL is appropriate for querying data for analysis. However, there are problems that a non-procedural language like SQL cannot easily address. Database vendors have long recognized that there are situations when developers need to use procedural logic; thus most leading databases ship with tools (e.g., pl/sql, tsql) for writing stored procedues.

Some developers we talked to thought that the big data headline is how MapReduce separates data storage, access, and access language. In a relational database, storage and access are completely integrated: SQL is required to read/write bits of data. SQL databases do come with popular connectivity tools such as JDBC/ODBC, but SQL remains the only language for interacting with bits of data (unless you're interested in writing code to parse the proprietary binary data formats often used to store RDBMS data). We should note that databases like MySQL are starting to support more transparent storage engines such as CSV files. In the MapReduce world, the Hadoop Streaming API provides a mechanism for developers to interact with data using their favorite programming language. With the flexibility of building API layers in Hadoop, access to data can be geared to user types.

Nevertheless, SQL has the advantages of incumbency. First, compared to MapReduce, there are many more developers familiar with SQL. Secondly, there are multiple SQL-compliant tools available for all aspects of data management and analysis. There are many Business Intelligence and Reporting Tools (BIRT)

Key Takeaway

MapReduce offers new flexibility for developers and users, because it separates data storage, access, and access language.

MapReduce takes advantage of data structures that are already familiar to many developers.

that interact with databases via SQL. In the data warehousing context, there are popular tools for extracting, loading, and transforming (ETL) data that rely on SQL.

Furthermore, database pioneers David Dewitt and Michael Stonebraker point to an important benefit of using a declarative language (SQL) to access records: with a high-level language like SQL, developers state or describe the subset of data they wish to retrieve. As a consequence, it is relatively easy to modify and understand a SQL statement. In contrast, since MapReduce programmers use low-level record manipulation to retrieve desired subsets of data, subsequent modifications and maintenance may prove more challenging.

Relational and Key/Value Pairs

Some developers we interviewed felt that predefined schemas made it difficult to rapidly experiment with and deploy analytic applications. Transactional and business intelligence databases usually entail carefully designed schemas. In practice, this requires surveying users in advance, to ensure that the database schema adequately captures current and future requirements. Future design changes and modifications are possible, but these can only be done by developers familar with the details.

Key/value structures are especially appealing to developers because they are easy to understand, flexible, and easily processed in a parallel (multi-server) fashion. The appeal is obvious: with minimal upfront design, developers can easily load data into key/value data structures. With the popularity of data interchange formats like JSON, web developers have become comfortable loading, reading, and writing key/value data structures. A senior developer at a large social networking site told us that after his team loaded a petabyte of data into Hadoop, developers and analysts immediately started accessing it using a variety of tools. Not only did they save time by not designing a data mart, but there were enough data access tools to satisfy their wide range of technical and business users. By monitoring the queries against the key/value data structures, certain query patterns emerge as popular enough to create dedicated "views" that make subsequent analysis simpler. The simple key/value pair data structure lends itself to easy experimentation and, when warranted, can lead to developing more permanent, easy-to-use data analysis frameworks.

Aside from Hadoop, there are emerging databases (e.g., CouchDB, MemcacheDB, SimpleDB, Cassandra) that are based on the key/value structure. Among the many features of CouchDB is its implementation of the MapReduce analytic framework and views. Views in CouchDB are accessible using a REST interface, and more complex queries can be written using MapReduce. Written

Key Takeaway

Early adopters are building and deploying schema-less distributed key/value stores for web-scale applications.

in Erlang, a concurrent language that lends itself to parallelism, CouchDB developers are planning on a multi-server parallel architecture in the future.

DBMS proponents, led by Dewitt and Stonebraker, point to the benefits of schemas. Since fields and data types are recorded and stored, relational databases ensure that records comply to the predefined structure before they are allowed to be loaded. In contrast, programmers who use key/value structures may have to add programming logic to ensure that incorrect records are excluded. Secondly, in modern relational databases, schemas are stored in system catalogs and are separate from application programs. Application developers can easily learn the underlying structure by using SQL to query the system catalogs. In key/value structures, developers may need to examine programming code or documentation to understand the details of data structures.

Reliability and Resilience

MPP databases assume that the hardware is relatively reliable and gear fault tolerance and recovery strategies around extra-operational processors. MapReduce assumes clusters large enough that hardware will fail on a regular basis and applies monitoring tools and agents to automatically identify problems and recover. The trade-offs are around cluster size and efficiency. MapReduce is architected for thousands of cluster nodes via brute-force parallelism and redundancy at the cost of processing efficiency. MPP databases, while moving towards handling larger clusters, are engineered to maximize and depend on the output of each node in the cluster, avoiding the processing costs of automated monitoring and constant replication for redundancy.

Hardware Options

A spectrum of hardware options is available for big data platforms, from specialized, proprietary appliances to run-of-the-mill commodity servers.

Proprietary hardware:

- Optimized for data warehouse storage and performance requirements
  - High-throughput architecture
  - Tightly coupled hardware and software
- More expensive than commodity
- Vendor lock-in
  - Risk of proprietary hardware not keeping up with advances in commodity hardware

Commodity hardware:

- Low cost, good value, many vendors
- Not optimized for big data requirements

Teradata, Netezza, and Kickfire all use proprietary hardware and do not run on commodity servers. All other big data software vendors run on commodity servers or commodity server appliances.

Data Warehouse Appliances

The template for using proprietary hardware to build a data warehouse appliance springs from the success of Teradata's line of hardware/software MPP appliances. Vendors offer both proprietary hardware-based and commodity server-based appliances. In proprietary appliances, the hardware and software are tightly coupled for performance. All appliances come designed and preconfigured to meet big data requirements. Appliances provide the following advantages:

- Optimized for big data storage and performance
  - Balanced CPU-to-disk ratio
  - Dedicated networks
  - Direct-attached, high-throughput disks
  - Large RAM
- Few component decisions
- Appliance is delivered (nearly) ready to run
  - Less work and less training required for system administrators:
    - Software is pre-installed
    - Hardware is set up and configured

Appliances can be a good choice for organizations that lack deep system administration resources or whose system administrators have no experience with cluster-based big data systems.

For those buying separate commodity components, the software providers typically have recommended configurations that can be used as a starting point for building a cluster. Based on our interviews, configuring all the parts of a cluster can be a time-consuming and iterative process.

Big Data Tool Feature Grid

The market for data management tools is vibrant and dynamic, and it includes established companies with mature products, established companies with new products, startups, and open source projects. The grid below shows most of the key players in big data tools and reflects our understanding of their capabilities. Based on our discussions, we expect a lot of changes in these products' functionality and maturity. The presence of a feature on the grid below doesn't mean the feature is fully and reliably implemented. Unless noted as proprietary, the grid assumes commodity servers. [Note that an "x" means the tool already has the feature, and a "y" means the feature is planned or partially implemented.]

Tool	MPP	Column	MapReduce	Other/Notes
Aster nCluster	x		x	Focus on fault tolerant resiliency and commodity hardware; peer-to-peer architecture
CouchDB			x	Open source key/value pair data structure geared towards web content; written in Erlang, distributed architecture planned
Hadoop	x		x	Open source, non-relational data structure; fault tolerant resiliency and commodity hardware
HP Neoview	x			Proprietary appliance
IBM InfoSphere	x			Based on shared-nothing DB2; available as appliance or software
Infobright/MySQL		x		Hybrid column and meta abstraction engine for MySQL
Greenplum	x		x	Available as appliance or as software; master node architecture
Kickfire	y			Proprietary appliance with distributed processing limited to one server box, running MySQL
Kognitio	x	y		Appliance or software or Data-as-a-Service; focus on blade hardware; allows in-memory column projections; peer-to-peer architecture
Microsoft SQL Server	y			Integrating recently purchased DATAllegro MPP technology into SQL Server—unknown release date; DATAllegro uses master node architecture
Netezza	x			Proprietary appliance with field-programmable gate array (FPGA); master node architecture
Oracle	x			Combine Oracle products (e.g., RAC) to create shared-everything MPP platform, appliance or software
ParAccel	x	x		Available as appliance or as software
Sybase IQ		x		Column RDBMS pioneer
Teradata	x			MPP pioneer, proprietary appliance
Vertica	x	x		Available as appliance or as software

Big Data Roadmaps

There are many new options to consider when deciding how to manage large data sets. And there may not be one right choice for your company. We see a future where organizations create a palette of tools for their data analysis needs by integrating what works best to meet different requirements. Another trend we see is towards tiered architectures of massive central data stores that become the source for departmental or focused analysis on subsets of data—with the option of using different data management tools to handle different data scenarios. Data integration to support multiple analysis environments may become a key competency for data-intense organizations. We already see that organizations facing big data challenges are turning to a mix of data management technologies to get the job done. Vendors are responding by converging on a set of technology options, e.g., MPP, column-oriented data structures, MapReduce, and compression. The good news is there are techniques available that do scale to meet the largest data sets—and many people and projects are focusing on making big data tools faster, easier, and more available.

The big data landscape presents a number of parameters to consider when deciding what combination of tools to use:

- Data size and data growth rate
- Type of data
- Resource experience and capacity
- Tolerance towards administering multi-server clusters
- Budget
- Existing analysis tool and skills infrastructure

We've described different technology choices, and their relative strengths, weaknesses, and other considerations. After reviewing the available technologies and considering organizations' differing requirements, we've come up with three example approaches to managing big data.

Work Smarter

For organizations with moderately sized data, a known and reliable data growth rate, and little experience, capacity, or desire to manage complex clusters, single-server solutions are a viable option. Keeping large data sets on a single server requires attention to design and engineering to make the data fit while meeting analysis requirements, and to tune the hardware and database. On the hardware side, multi-core CPUs, large amounts of RAM, and flexible storage

options can provide a platform that has adequate storage and enough query performance when coupled with either a column-oriented or MPP database.

The column-oriented databases provide fast performance by reducing the amount of data processed using compression and by only accessing requested columns. Direct-attached disk extenders or networked-attached storage can provide the option to grow data storage beyond what fits in a single server box. Column-oriented databases work best with integer data, time series, and other data with relatively few distinct values—not an uncommon analysis environment. Column-oriented databases require extra design work to optimize the index and compression strategy for query performance. Careful data design, including aggregating data, can reduce query complexity and also improve query performance.

For unstructured data and the option to move to a distributed data environment, MPP RDBMSs can work on one server. MPP RDBMSs can take advantage of the parallelism available via multi-core CPUs. For example, a server with two four-way CPUs provides eight degrees of parallelism on a single server. MPP RDBMSs work best with direct-attached disks, so network-attached storage is not a good option. Direct-attached storage arrays can extend the amount of storage beyond one server, while still maintaining the single-server architecture. MPP RDBMSs requires extra work to configure the software to work with multi-core CPUs, but they are otherwise easy to install and require little extra work to get running.

Go Parallel

Distributed data and parallel processing are the best current approaches for the large-scale data sets. Environments with rapidly growing data and opportunities for data analysis to provide organizational efficiencies and advantages will be best served by parallel processing's near-linear scaling and capacity, by throwing more hardware at a cluster, to handle enormous data sets and different data types. For organizations with the system administration gravitas to handle cluster systems and an embedded analysis infrastructure geared towards relational databases, MPP and MPP/column databases are a good choice. Getting the storage size and performance scaling of MPP will improve analysis productivity without having to fundamentally change tools and resource skills. Changes include paying attention to data distribution, and increased attention to system administration and increased hardware costs.

Our experience with an MPP database is that little performance tuning is required, once the cluster size is set, and that we can focus on the analysis and not on the database. Once installed, moving to the MPP database is an easy, almost transparent transition.

Think Different

We see MapReduce and Hadoop as proven technology that, when embraced by an organization's technical and analysis staff, can provide the scaling and performance characteristics needed to handle the largest data sets. Yes, it requires a new way of looking at problems, unfamiliar data structures, and new tools—plus people who are excited to learn and work with MapReduce. In return, you get flexibility: flexible scaling via dynamic cluster sizing, flexible data structures with options that don't require predefining schemas, and flexible and tiered data access by designing interfaces geared towards different analysis requirements.

MapReduce is a good solution for an organization that is generating large amounts of data they're not sure how to use. Data can be thrown into a MapReduce platform for developers and analysts to investigate and play with. Once patterns of analysis emerge, more refined data structures, data transformation, and data access processes can be built (including interfaces to relational RDBMSs) that make subsequent inquiries easy to repeat. Sounds like that process requires data-aware developers and analysts? It does. In return you get a staff steeped in knowledge of the data, which is an important characteristic for data-driven organizations.

For new organizations with big data or existing organizations trying to take advantage of new big data opportunities, MapReduce/Hadoop should be considered to augment data workflows—especially when implemented as an Agile-like frontend to a data discovery and insight process. We don't see MapReduce replacing relational data management tools, with their well-developed tools and resource pools; rather, we see MapReduce as a complement that can help organizations more fully integrate data into decision making.

Your Data and Your Team

To make decisions about the appropriate big data platform for your organization, you'll need to:

- Know your data, i.e., data types, data sources, data sizes, data growth rate, and data quality

- Know how data can and may be used for analysis and to gain insight
  - Identify and integrate external data sources
  - Take advantage of opportunities to set up a process of constant discovery and convert repeatable analysis to dedicated data structures and analysis access
- Know your people—what are their technical skills, what are they willing and able to learn, and how do their skills fit with your data management tools and analysis opportunities?

As the benefits of becoming a data-driven organization become more obvious, exacerbated by a tough economy, it's more important than ever to make the best choices about big data. Ultimately, big data is more about attitude than tools—data-driven organizations look at big data as a solution, not a problem. ■

How Science Handles Big Data

The scientific research community has been grappling with big data for many years. Outside of Internet search engines, major science projects generate, collect, and manage some of the largest known data sets. As the following table shows, researchers in the fields of High-Energy Physics (HEP), genomics, and astronomy manage databases consisting of several petabytes (1,024 terabytes). In the case of HEP, data generated by particle collisions can range from a few terabytes per second (e.g., the SLAC BaBar detector) to half a petabyte per second (e.g., the Large Hadron Collider, or LHC). Although only a small fraction of the data from billions of collisions is saved, enough data is retained to pose a serious challenge. LHC expects their database to grow at a rate of 15 petabytes per year, with some published reports placing the cost of data management at $150 million.

Field	Source	Type	Size in PB
High-Energy Physics	SLAC BaBar	collisions	4
High-Energy Physics	LHC	collisions	15 per year
Astronomy	NASA Earth Observing System	Images	4
Photon Science	Photon Lasers	Movies	Few PB/year
Genomics	Genomics:GTL	DNA Sequences	Multi PB
Astronomy	LSST	Telescope images	50+

Source: "The Science and Fiction of Petascale Analytics" by Jacek Becla

The big data challenges confronting the scientific community are similar to those faced by users in other fields. In most scientific fields (notably High-Energy Physics), data retention is important. As their models and understanding evolve and improve over time, scientists examine data repeatedly. (For data retention, tape storage is still widely used.) Since some experiments and projects last years and even decades, researchers have a preference for open source and commodity hardware. In the case of long-running (multi-decade) projects, there is a natural inclination to guard against the possibility of commercial companies failing to survive. But there is also an aversion to "black boxes." In a discipline that fosters collaboration and openess, the ability to examine, alter, and recompile source code is a plus. Remote access is also crucial, as science researchers who work together usually reside on different continents. The related areas of energy efficiency and temperature management are of great concern. While writing an article on big data for *Nature,* science fiction writer Cory Doctorow visited several data centers and found IT systems managers to be "universally concerned" with heat and temperature management. Most data center administrators he talked to feared that they would not have enough time to gracefully shut down all their servers if something happened to their energy or temperature management systems. As Doctorow observed, "…the dirtier the shutdown, the longer the subsequent startup, with its rebuilding of databases and replacement of crashed components."

In data analysis, there are important differences between science and industry. Complex queries are common in scientific databases. Business users, on the other hand, tend to rely on reporting tools and summary tables when querying large databases. Because scientists' queries tend to be ad hoc in nature and, in some cases, initial what-if queries are first run after years of data collection, scientists rely on specialized software that are usually hand-coded. In HEP, queries involving hundreds of attributes are run repeatedly as investigators iteratively explore their data. In fields such as astronomy, where researchers are interested in specific objects, or in molecular biology, where researchers might be interested in specific DNA sequences, "needle in the haystack" queries are frequently run. Other popular and more complex big data query types in science include correlations and time series analysis.

So what are the popular data management solutions in science? Although some groups experimented with them, object databases are not widely used. Jacek Becla of SLAC highlights three common approaches. Relational databases are popular, although large multi-server clusters are not common. In some bigger systems, scientists store data in files and metadata in relational databases. Some research groups bypass databases altogether, and store everything in files. Finally, Becla notes that scientists are beginning to explore the MapReduce framework, a technique we discuss in detail in this report.

Looking to the future, scientists want big data to be easier to work with; after developing an algorithm or hypothesis, they want to be able to run tests and queries without the setup costs prevalent today. In some cases, this includes loading the right data slices off of tape storage. The ability to pause, restart, and abort queries would be extremely valuable. Scientists would like tools that allow them to annotate data and expose their annotations to other researchers who query the same data set. Since complex computations are common, scientists would like to have native field/column types to include data structures such as arrays. Along the same lines, scientists want to be able to easily perform spatial and temporal operations. Finally, scientists want analytical tools and techniques that are accessible to a wider base of users and mobile devices, since traditionally, analysis has been limited to elite scientists with access to supercomputers or similar resources.

*** REFERENCE:** *The Science and Fiction of Petascale Analytics* by Jacek Becla

Boldly Going Where No Data Has Gone Before

Emerging/Speculative Technologies

We're just beginning to tap the power of big data. Here's a short survey of emerging and speculative technologies that cover specialized data scenarios, e.g., streams, wavelets, and graph databases, plus other promising technologies (either available or in early development) that have the potential to become useful in the big data space, e.g., solid state drives, compression and the Triadic Continuum.

Solid State Drives/Flash Drives

Solid state drives (SSDs) are based on semiconductor technology to store persistent data. SSDs are also known as flash memory or RAM drives. SSDs can be used to augment or replace electromechanical disk drives for persistent data storage. The advantages of SSDs compared to disk drives are:

- Fast data access and low latency
- No moving parts
- Low energy consumption

Disadvantages include:

- Significantly higher cost per unit of storage
- Limited write cycles
  - Wear-leveling algorithms distribute writes and help extend the life of SSDs

The high cost of SSDs, coupled with the storage requirements for large data sets, limits the adoption of SSDs for big data applications. However, when query performance is the primary requirement, SSDs may be an option. In MPP applications, SSDs can provide a two- or threefold performance improvement.

Compression

Compression is an issue for nearly all the vendors we spoke with. Many offer compression at least as an option. Compression can be used to accomplish either or both of the following goals:

- Reduce the amount of persistent storage space needed
- Increase query performance for I/O bound systems by taking advantage of idle CPU time to compress/decompress data

Because of the special requirements and configurations for big data systems, compression may be deployed to do nothing more than reduce the amount of disk storage. Query performance is considered a higher priority requirement than reducing the size of the data, and the systems are moving towards compression schemes that emphasize compression/decompression speed more than maximizing data compression rates.

Streams

In recent years, finding techniques for querying streams of real-time data has been an active area of research. In domains that generate large and dynamic data sets, there is a real need for operational Business Intelligence (BI). Prime examples include

finance, network monitoring, manufacturing, retail management, and sensor networks. Conventional database systems are not particularly suited to responding to queries that involve continuous and unbounded streams of data. In most traditional data warehouses, data is first loaded, and then indexed, before finally being queried. Fortunately, a different set of tools are being developed and introduced to handle problems involving continuous data streams and queries. A promising academic effort from Stanford University led to a prototype DSMS (Data Stream Management System), built to handle declarative continuous queries over data streams and traditional static data sets.

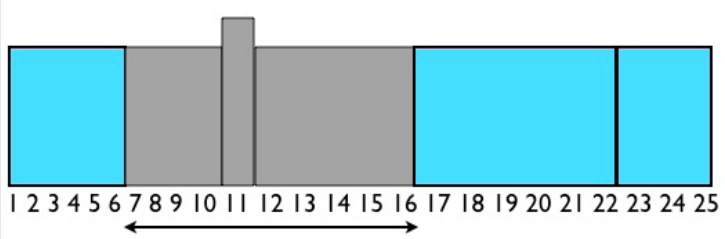
More recently, several startups have attempted to tackle real-time analytics. Silicon Valley-based Truviso has unveiled a scalable platform that extends standard SQL to query live streams of data. Queries run continuously and concurrently, producing instant results. Blog search engine Technorati found that with Truviso's technology, it could easily redraw its live channel tag clouds every few minutes. To help enable supply-chain and logistics optimization, Truviso has also rolled out products designed to help retailers monitor demand in real time. Founded by Michael Stonebraker, Massachusetts company StreamBase has a suite of products built around an extension to standard SQL (StreamSQL), designed for querying and processing continuous data streams. StreamBase technologies have been used in financial services, especially for algorithmic trading, real-time profit/loss calculations, compliance, and market data management. Both Truviso and StreamBase claim that database analysts and developers quickly become comfortable with their SQL extensions.

Approximate Query Processing and Wavelets

When dealing with a large database, an analyst typically executes a SQL query that returns an exact answer. Depending on the size of the database and the complexity of the query, that can take a significant amount of time to complete. For most applications and questions, *approximate* answers are acceptable. Especially in the exploratory phases of an analysis, precision is usually not needed and faster query execution allows analysts to iterate and learn more about the data on hand. With the goal of speeding up execution time, database researchers have developed several *approximate* query processing techniques.

The simplest and most common approach used to answer aggregate queries is to use random samples. (A common aggregate query is the total sales in a given region or time period.) Rather than running queries against the full database, an analyst is given access to a representative random sample. Users just need to be made aware that the results obtained in this fashion are not exact, but rather fall within margins of error of the true answers. Most academic research efforts have been devoted to computationally efficient ways of generating and maintaining random samples. Some commercial databases, including Oracle and IBM Informix, already have (online) sampling operators.

Other researchers have explored efficient methods of summarizing large data sets. Whereas sampling takes a large data set and reduces it to a manageable size, other query-approximation methods are based on carefully constructed synopses. Commonly used in statistics, histograms have also been used in large databases. In the histogram approach, a variable is partitioned into buckets and the frequency of its appearance in a bucket recorded. Popular partitioning schemes include buckets of equal sizes (equi-width), buckets with the same number of records (equi-depth), and compressed partitions (singleton buckets for the most frequently appearing values, and equi-depth buckets for the rest). SQL queries are approximated using the histogram. In the example below, an equi-depth histogram consisting of 6 buckets each with |R| records is used to answer a query:



Query:

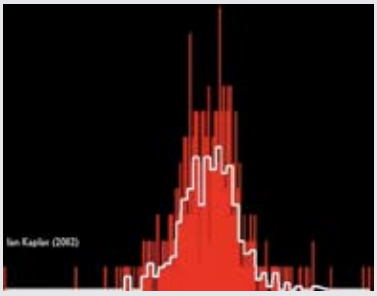
Select count(*)

From table

where field between 7 and 16

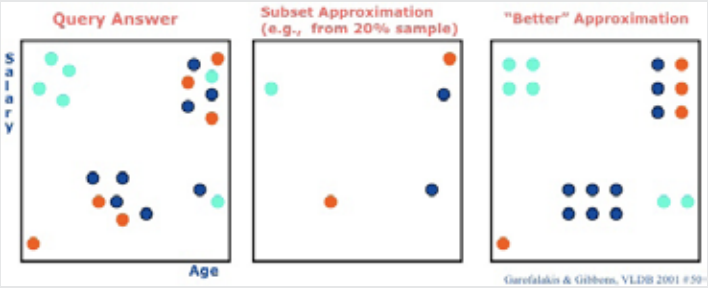
Answer: Approximately 3 * |R|/6

A more recent class of approximate query techniques, available through research prototypes, relies on mathematical techniques from Signal Processing to create synopses of large data sets. Fourier analysis is a mathematical technique that takes general functions and represents them as combinations of periodic functions (sines and cosines). Wavelets are similar mathematical tools used to decompose data or functions into frequency components. Unlike older techniques such as Fourier Analysis, wavelet decomposition uses more complicated functions that allow components to be studied using the proper resolution: when data is examined with a large window, macro features are revealed, while small windows highlight micro properties. In contrast, Fourier analysis relies on non-local functions (sines and cosines) and performs badly when data contains spikes or discontinuties. The ability to optimize resolution levels allows wavelets to better handle such anomalies. Researchers are working on finding efficient algorithms to approximate a wavelet function from a data set. Moreover, because wavelets reduce a problem to a series of coefficients (used in a linear combination of wavelet functions), truncating coefficients below a certain threshold leads to data compression. An early application of wavelets to databases was compressing the large database of fingerprint images used by the FBI. More recently, wavelets have been used for similarity searches in databases of images.



For database queries, wavelets were used early on for range-query selectivity estimation: given a query, quickly estimate the fraction of database records that satisfy it. Query optimizers maintain histograms for this purpose, and researchers found wavelets particularly suited for creating histograms. (A wavelet-based histogram is computed from the cumulative distribution of a given data set.) In addition, as described earlier, the histograms constucted using wavelets can be used to give approximate answers to some queries.

Researchers have also explored using random samples, histograms, and wavelets to analyze multidimensional data. In particular, researchers have studied whether these approximate query techniques help with joins or range-query selectivity estimates when the data set has multiple dimensions. Random samples can preserve attribute correlations (within a confidence interval) and are viable for aggregate queries (e.g. count, sum, average) on single tables. Table joins are a problem: a table join of random samples need not be a random sample of the joined tables. (The probability of a record/tuple appearing in either set is not identical.) Wavelets are efficient at computing and maintaining histograms involving low-dimensional data. However, both histograms and wavelets are subject to the "curse of dimensionality": researchers have found both to be inefficient with data beyond 5 to 6 dimensions. Nevertheless, research has shown wavelets to be effective for some generic set-valued queries (joins, simple selects, etc.). Even with compression, wavelets can accurately capture regions in low-dimensional data spaces.



Approximate query-processing techniques remain active areas of research, but commercial acceptance has been slow. Sampling can handle some queries involving non-numeric or categorical data, but it's unclear how histograms and wavelets can be used in those situations.*

Graph Databases

Now that websites are incorporating features found on social media properties, such as identifying friends and joining groups, an increasing percentage of all online destinations have a social component. With some social networking websites having databases consisting of hundreds of terabytes of data, databases optimized for network-oriented and semi-structured data have emerged. Social network data is an example of a data set suited for graph databases. Relationships among users of social networks and websites are easily represented in graphical ("tree") structures comprised of nodes ("users") and edges ("relationships").

Unlike relational databases, which are based on tables, rows, and columns, some graph databases have primitive structures consisting of nodes, edges, and their properties. Other graph databases are designed specifically to handle semi-structured data such as this Resource Description Framework (RDF, a data structure advocated by proponents of the semantic Web). Although relational databases can also handle network-oriented and semi-structured data, graph databases are optimized to deliver fast and scalable performance for such data sets.

There are several graph databases available, including the Neo Database, AllegroGraph RDFStore, and Freebase. Developed by the Swedish company Windh Technologies, the Neo Database uses graphical structures as building blocks (or primitives): nodes, edges, and their key/value properties.

Triadic Continuum

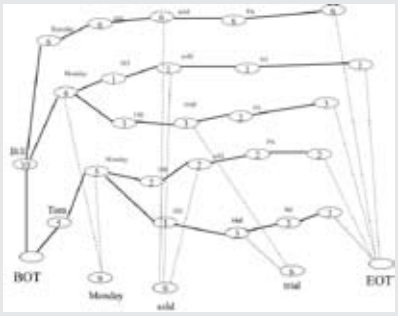
A recent technique with roots in logic (semiotics) and the work of Charles Sanders Peirce simplifies Business Intelligence and data warehousing by introducing novel data structures for storing and querying data. In the 19th century, Peirce studied logic using sign processes, signs, and symbols. Peirce devised a system consisting of three categories ("triadic sign relation"): sign, object, and interpretant. Inspired by previous attempts to build on the work of Peirce, researchers at Unisys extrapolated a self-organizing and self-updating data structure. The data structure is somewhat reminiscent of RDF, a data structure advocated by proponents of the semantic Web.

First developed by researchers at Unisys, the KStore (also referred to as the Triadic Continuum) is a data structure that overcomes some of the perceived limitations of relational databases and data cubes. It takes time to properly design, load, and refresh data cubes, limiting analysts to queries covered by data loaded in the static cubes. Now being refined by Buffalo-based As It Is Inc., the KStore is a dynamic data structure that contains a record of the data and all the relationships between data

*** REFERENCE:** *Approximate Query Processing: Taming the TeraBytes!* by Minos Garofalakis and Phillip B. Gibbons. Proceedings of the 27th International Conference on Very Large Data Bases, 2001

elements, allowing cold (as opposed to cached) queries to be executed efficiently. Data gets loaded into a KStore in either XML or delimited text files, and the resulting data structure is usually smaller (compressed) than the original data files. Once loaded, analysts can immediately execute queries using a graphical user interface or an XML API. This eliminates the need to design special schemas, and extract, transform, and (re)load data cubes or data marts.

As It Is Inc., is currently in the final stages of developing and commercializing the KStore. When we asked Triadic Continuum architect Jane Mazzagatti whether the technology could handle large data sets, she noted that KStore developers have taken real-world (CRM and sales) data warehouses, with hundreds of millions of rows, and loaded them into their data structures in little more than an hour. As of late December 2008, the KStore resides in memory, but Mazzagatti doesn't forsee difficulties in moving it to a disk-based system. Although it's currently only available on Windows, Mazzagatti told us there are plans to make the KStore available on other platforms.



Disclosure

At O'Reilly we run the Greenplum MPP database to analyze a large data store. O'Reilly was Greenplum's first customer, and O'Reilly employees have been used as shills on Greenplum's website and own Greenplum stock.

Acknowledgments

This report would not have been possible without extended conversations with many individuals. Special thanks to the following for their time and insights:

Brian Aker
Chris Anderson and Jan Lehnardt
Jacek Becla
Andy Bruno
Joydeep Das and Dan Lahl
Neil Day
Bernard Golden
Joe Hellerstein
Luke Lonergan
Stephen Gunn, Mikey Dickerson, and Ian Gulliver
Sumit Gupta and Andrew Humber
Jeff Hammerbacher
Gregor Hochmuth
Shawn Kung and Steve Wooledge
Eric Legasse and Steve Miller
Jane Mazzagatti
Bryan O'Sullivan
John Thompson
Peter Zaitsev and Baron Schwartz

Calendar

A selection of significant public events over the next few months.

February 9–11

TOC: Tools of Change for Publishing (New York, NY)
http://en.oreilly.com/toc2009/public/content/home
Reinventing an industry for our networked, customer-driven world.

February 23–24

FOWA: Future of Web Apps (Miami, FL)
http://events.carsonified.com/fowa/2009/Miami
The title says it: future of the browser; web business models; online community; plus a Bar Camp. Across the pond, the Dublin version is March 6.

March 1–2

DEMO (Palm Desert, CA)
http://www.demo.com
Seventy 6–minute presentations from companies hoping to be the Next Big Thing.

March 4–7

DrupalCon (Washington, DC)
http://dc2009.drupalcon.org
The official unconference for developers and users of open source CMS Drupal.

March 9–12

ETech (San Jose, CA)
http://en.oreilly.com/et2009
Because innovation's more important than ever in "interesting" times.

March 13–17

SXSW Interactive (Austin, TX)
http://sxsw.com/interactive
Digital creatives head south.

March 18–20

Mix 09 (Las Vegas, NV)
http://2009.visitmix.com
The March Marathon continues for webheads, with Microsoft's web design and development conference.

March 31–April 3

Web 2.0 Expo (San Francisco, CA)
http://en.oreilly.com/where2009/
Our prescient theme (picked last summer): The Power of Less

April 15–17

The Next Web (Amsterdam, NL)
http://2009.thenextweb.com
Even more web goodness, this time in fabulous Amsterdam.

May 19–21

Where 2.0 (San Jose, CA)
http://www.web2expo.com/webexsf2009
Knowing "where" opens up a world of possibilities, from the mundane (finding pizza) to the world-changing (cultural preservation via indigenous maps).

June 9–11

Found: The Search Acquisition and Architecture Conference
(Burlingame, CA)
http://en.oreilly.com/found
A new O'Reilly conference for web developers, designers, SEO specialists, marketing strategies, and online entrepreneurs.

Release 2.0 Subscription Form

Complete this form and join the other industry executives who rely on *Release 2.0* to stay ahead of the headlines. You can also subscribe online at *http://r2.oreilly.com*.

Your annual *Release 2.0* subscription costs $495 per year, and includes both the print and electronic versions of six every-other-month issues, access to the complete *Release 1.0* and *Release 2.0* archives, a discount on attendance to O'Reilly conferences, and full access to our website, *http://r2.oreilly.com*.

name

title company

address

city state zip country

telephone fax

email (personal email required for electronic access)

url

☐ My colleagues should read *Release 2.0*, too!

Send me information about multiple copy subscriptions and electronic site licenses.

☐ check enclosed ☐ charge my (check one) ☐ american express ☐ mastercard ☐ visa

card number expiration date cvs code

name and billing address ☐ same as above ☐ see below

name

address

city state zip country

signature

Please fax this form to 1.818.487.4501 or mail it to Release 2.0, P.O. Box 17046, North Hollywood CA 91615-9588

Payment must be included with this form. Your satisfaction is guaranteed or your money back.

If you have any questions, please call us at 1.800.889.8969 or 1.707.827.7019 or email us at *r2@oreilly.com*.