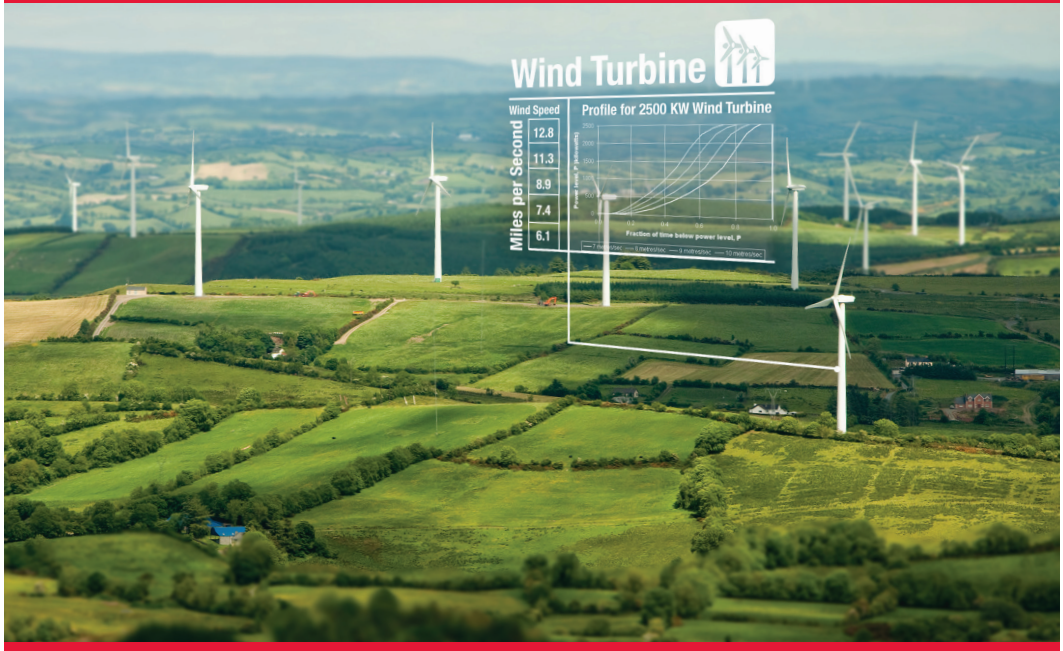
Planning for Big Data

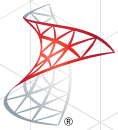
A CIO's Handbook to the Changing Data Landscape

O'Reilly Radar Team



O'REILLY®

O'REILLY®
Strata
Making Data Work



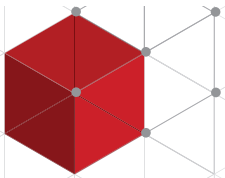
Microsoft®
SQL Server® 2012

FURTHER. FORWARD. FASTER.

SQL SERVER IS THE #1 DATABASE*

used by enterprises to run their mission critical applications.

» learn more at www.microsoft.com/sqlserver



*IDC, "2009 Mission Critical Application Platform Study," December 2009

Microsoft®

Change the world with data. We'll show you how.

strataconf.com

O'REILLY®

Strata RX CONFERENCE

Data Makes a Difference

Sep 25 – 27, 2013
Boston, MA

O'REILLY®

Strata CONFERENCE + HADOOP WORLD

Oct 28 – 30, 2013
New York, NY

O'REILLY®

Strata CONFERENCE

Making Data Work

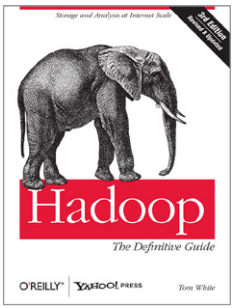
Nov 11 – 13, 2013
London, England

O'REILLY®

Spreading the knowledge of innovators.

O'REILLY®

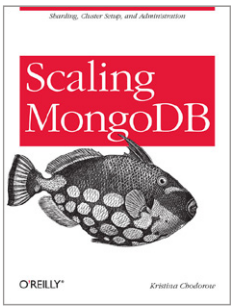
Related eBooks



Hadoop: The Definitive Guide, 3rd edition

By Tom White
Released: May 2012
Ebook: **$39.99**

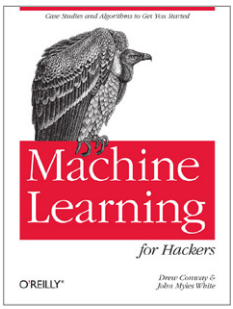
Buy Now



Scaling MongoDB

By Kristina Chodorow
Released: January 2011
Ebook: **$16.99**

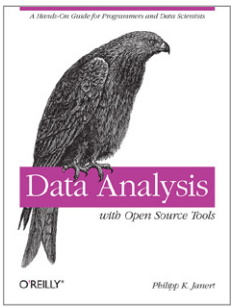
Buy Now



Machine Learning for Hackers

By Drew Conway and John Myles White
Released: February 2012
Ebook: **$31.99**

Buy Now



Data Analysis with Open Source Tools

By Philipp K. Janert
Released: November 2010
Ebook: **$31.99**

Buy Now

Planning for Big Data

O'Reilly Radar Team

O'REILLY®
Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo

Planning for Big Data
by O'Reilly Radar Team

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://my.safaribooksonline.com*). For more information, contact our corporate/institutional sales department: (800) 998-9938 or *corporate@oreilly.com*.

Editor: Edd Dumbill

Cover Designer: Karen Montgomery
Interior Designer: David Futato
Illustrator: Robert Romano

March 2012: First Edition.

Revision History for the First Edition:
2012-03-12 First release
See *http://oreilly.com/catalog/errata.csp?isbn=9781449329679* for release details.

ISBN: 978-1-449-32967-9

[LSI]

1331315384

Table of Contents

Introduction

In February 2011, over 1,300 people came together for the inaugural O'Reilly Strata Conference in Santa Clara, California. Though representing diverse fields, from insurance to media and high-tech to healthcare, attendees buzzed with a new-found common identity: they were data scientists. Entrepreneurial and resourceful, combining programming skills with math, data scientists have emerged as a new profession leading the march towards data-driven business.

This new profession rides on the wave of big data. Our businesses are creating ever more data, and as consumers we are sources of massive streams of information, thanks to social networks and smartphones. In this raw material lies much of value: insight about businesses and markets, and the scope to create new kinds of hyper-personalized products and services.

Five years ago, only big business could afford to profit from big data: Walmart and Google, specialized financial traders. Today, thanks to an open source project called Hadoop, commodity Linux hardware and cloud computing, this power is in reach for everyone. A data revolution is sweeping business, government and science, with consequences as far reaching and long lasting as the web itself.

Every revolution has to start somewhere, and the question for many is "how can data science and big data help my organization?" After years of data processing choices being straightforward, there's now a diverse landscape to negotiate. What's more, to become data-driven, you must grapple with changes that are cultural as well as technological.

The aim of this book is to help you understand what big data is, why it matters, and where to get started. If you're already working with big data, hand this book to your colleagues or executives to help them better appreciate the issues and possibilities.

I am grateful to my fellow O'Reilly Radar authors for contributing articles in addition to myself: Alistair Croll, Julie Steele and Mike Loukides.

Edd Dumbill

Program Chair, O'Reilly Strata Conference

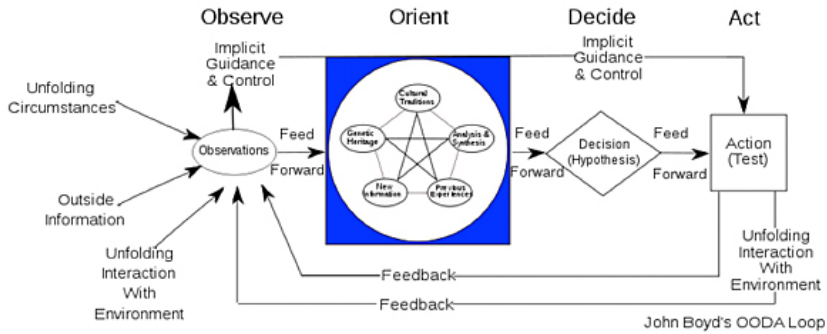
February 2012

CHAPTER 1

The Feedback Economy

By Alistair Croll

Military strategist John Boyd spent a lot of time understanding how to win battles. Building on his experience as a fighter pilot, he broke down the process of observing and reacting into something called an Observe, Orient, Decide, and Act (OODA) loop. Combat, he realized, consisted of observing your circumstances, orienting yourself to your enemy's way of thinking and your environment, deciding on a course of action, and then acting on it.



The Observe, Orient, Decide, and Act (OODA) loop. Larger version available here..

The most important part of this loop isn't included in the OODA acronym, however. **It's the fact that it's a loop**. The results of earlier actions feed back into later, hopefully wiser, ones. Over time, the fighter "gets inside" their opponent's loop, outsmarting and outmaneuvering them. The system learns.

Boyd's genius was to realize that winning requires two things: being able to collect and analyze information better, and being able to act on that informa-

tion faster, incorporating what's learned into the next iteration. Today, what Boyd learned in a cockpit applies to nearly everything we do.

Data-Obese, Digital-Fast

In our always-on lives we're flooded with cheap, abundant information. We need to capture and analyze it well, separating digital wheat from digital chaff, identifying meaningful undercurrents while ignoring meaningless social flotsam. Clay Johnson argues that we need to go on an information diet, and makes a good case for conscious consumption. In an era of information obesity, we need to eat better. There's a reason they call it a feed, after all.

It's not just an overabundance of data that makes Boyd's insights vital. In the last 20 years, much of human interaction has shifted from atoms to bits. When interactions become digital, they become instantaneous, interactive, and easily copied. It's as easy to tell the world as to tell a friend, and a day's shopping is reduced to a few clicks.

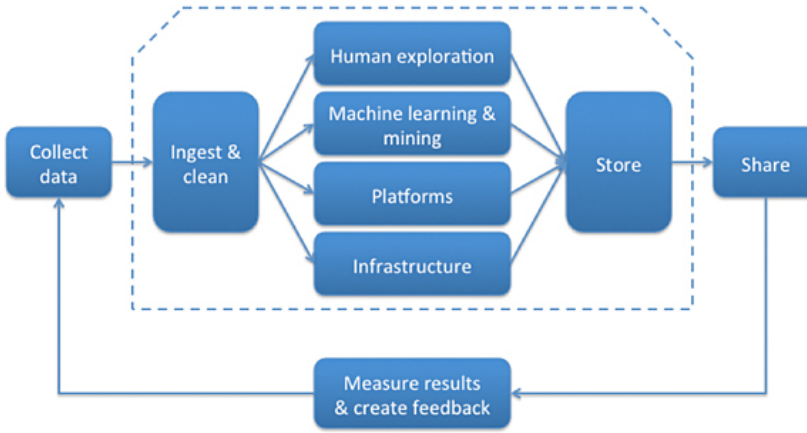
The move from atoms to bits reduces the coefficient of friction of entire industries to zero. Teenagers shun e-mail as too slow, opting for instant messages. The digitization of our world means that trips around the OODA loop happen faster than ever, and continue to accelerate.

We're drowning in data. Bits are faster than atoms. Our jungle-surplus wetware can't keep up. At least, not without Boyd's help. In a society where every person, tethered to their smartphone, is both a sensor and an end node, we need better ways to observe and orient, whether we're at home or at work, solving the world's problems or planning a play date. And we need to be constantly deciding, acting, and experimenting, feeding what we learn back into future behavior.

We're entering a feedback economy.

The Big Data Supply Chain

Consider how a company collects, analyzes, and acts on data.



The big data supply chain. Larger version available here..

Let's look at these components in order.

Data collection

The first step in a data supply chain is to get the data in the first place.

Information comes in from a variety of sources, both public and private. We're a promiscuous society online, and with the advent of low-cost data marketplaces, it's possible to get nearly any nugget of data relatively affordably. From social network sentiment, to weather reports, to economic indicators, public information is grist for the big data mill. Alongside this, we have organization-specific data such as retail traffic, call center volumes, product recalls, or customer loyalty indicators.

The legality of collection is perhaps more restrictive than getting the data in the first place. Some data is heavily regulated — HIPAA governs healthcare, while PCI restricts financial transactions. In other cases, the act of combining data may be illegal because it generates personally identifiable information (PII). For example, courts have ruled differently on whether IP addresses aren't PII, and the California Supreme Court ruled that zip codes are. Navigating these regulations imposes some serious constraints on what can be collected and how it can be combined.

The era of ubiquitous computing means that everyone is a potential source of data, too. A modern smartphone can sense light, sound, motion, location, nearby networks and devices, and more, making it a perfect data collector. As consumers opt into loyalty programs and install applications, they become sensors that can feed the data supply chain.

In big data, the collection is often challenging because of the sheer volume of information, or the speed with which it arrives, both of which demand new approaches and architectures.

Ingesting and cleaning

Once the data is collected, it must be ingested. In traditional business intelligence (BI) parlance, this is known as Extract, Transform, and Load (ETL): the act of putting the right information into the correct tables of a database schema and manipulating certain fields to make them easier to work with.

One of the distinguishing characteristics of big data, however, is that the data is often unstructured. That means we don't know the inherent schema of the information before we start to analyze it. We may still transform the information — replacing an IP address with the name of a city, for example, or anonymizing certain fields with a one-way hash function — but we may hold onto the original data and only define its structure as we analyze it.

Hardware

The information we've ingested needs to be analyzed by people and machines. That means hardware, in the form of computing, storage, and networks. Big data doesn't change this, but it does change how it's used. Virtualization, for example, allows operators to spin up many machines temporarily, then destroy them once the processing is over.

Cloud computing is also a boon to big data. Paying by consumption destroys the barriers to entry that would prohibit many organizations from playing with large datasets, because there's no up-front investment. In many ways, big data gives clouds something to do.

Platforms

Where big data is new is in the platforms and frameworks we create to crunch large amounts of information quickly. One way to speed up data analysis is to break the data into chunks that can be analyzed in parallel. Another is to build a pipeline of processing steps, each optimized for a particular task.

Big data is often about fast results, rather than simply crunching a large amount of information. That's important for two reasons:

1. Much of the big data work going on today is related to user interfaces and the web. Suggesting what books someone will enjoy, or delivering search results, or finding the best flight, requires an answer in the time it takes a

page to load. The only way to accomplish this is to spread out the task, which is one of the reasons why Google has nearly a million servers.

2. We analyze unstructured data iteratively. As we first explore a dataset, we don't know which dimensions matter. What if we segment by age? Filter by country? Sort by purchase price? Split the results by gender? This kind of "what if" analysis is exploratory in nature, and analysts are only as productive as their ability to explore freely. Big data may be big. But if it's not fast, it's unintelligible.

Much of the hype around big data companies today is a result of the retooling of enterprise BI. For decades, companies have relied on structured relational databases and data warehouses — many of them can't handle the exploration, lack of structure, speed, and massive sizes of big data applications.

Machine learning

One way to think about big data is that it's "more data than you can go through by hand." For much of the data we want to analyze today, we need a machine's help.

Part of that help happens at ingestion. For example, natural language processing tries to read unstructured text and deduce what it means: Was this Twitter user happy or sad? Is this call center recording good, or was the customer angry?

Machine learning is important elsewhere in the data supply chain. When we analyze information, we're trying to find signal within the noise, to discern patterns. Humans can't find signal well by themselves. Just as astronomers use algorithms to scan the night's sky for signals, then verify any promising anomalies themselves, so too can data analysts use machines to find interesting dimensions, groupings, or patterns within the data. Machines can work at a lower signal-to-noise ratio than people.

Human exploration

While machine learning is an important tool to the data analyst, there's no substitute for human eyes and ears. Displaying the data in human-readable form is hard work, stretching the limits of multi-dimensional visualization. While most analysts work with spreadsheets or simple query languages today, that's changing.

Creve Maples, an early advocate of better computer interaction, designs systems that take dozens of independent, data sources and displays them in navigable 3D environments, complete with sound and other cues. Maples' studies

show that when we feed an analyst data in this way, they can often find answers in minutes instead of months.

This kind of interactivity requires the speed and parallelism explained above, as well as new interfaces and multi-sensory environments that allow an analyst to work alongside the machine, immersed in the data.

Storage

Big data takes a lot of storage. In addition to the actual information in its raw form, there's the transformed information; the virtual machines used to crunch it; the schemas and tables resulting from analysis; and the many formats that legacy tools require so they can work alongside new technology. Often, storage is a combination of cloud and on-premise storage, using traditional flat-file and relational databases alongside more recent, post-SQL storage systems.

During and after analysis, the big data supply chain needs a warehouse. Comparing year-on-year progress or changes over time means we have to keep copies of everything, along with the algorithms and queries with which we analyzed it.

Sharing and acting

All of this analysis isn't much good if we can't act on it. As with collection, this isn't simply a technical matter — it involves legislation, organizational politics, and a willingness to experiment. The data might be shared openly with the world, or closely guarded.

The best companies tie big data results into everything from hiring and firing decisions, to strategic planning, to market positioning. While it's easy to buy into big data technology, it's far harder to shift an organization's culture. In many ways, big data adoption isn't a hardware retirement issue, it's an employee retirement one.

We've seen similar resistance to change each time there's a big change in information technology. Mainframes, client-server computing, packet-based networks, and the web all had their detractors. A NASA study into the failure of Ada, the first object-oriented language, concluded that proponents had over-promised, and there was a lack of a supporting ecosystem to help the new language flourish. Big data, and its close cousin, cloud computing, are likely to encounter similar obstacles.

A big data mindset is one of experimentation, of taking measured risks and assessing their impact quickly. It's similar to the Lean Startup movement, which advocates fast, iterative learning and tight links to customers. But while

a small startup can be lean because it's nascent and close to its market, a big organization needs big data and an OODA loop to react well and iterate fast.

The big data supply chain is the organizational OODA loop. It's the big business answer to the lean startup.

Measuring and collecting feedback

Just as John Boyd's OODA loop is mostly about the loop, so big data is mostly about feedback. Simply analyzing information isn't particularly useful. To work, the organization has to choose a course of action from the results, then observe what happens and use that information to collect new data or analyze things in a different way. It's a process of continuous optimization that affects every facet of a business.

Replacing Everything with Data

Software is eating the world. Verticals like publishing, music, real estate and banking once had strong barriers to entry. Now they've been entirely disrupted by the elimination of middlemen. The last film projector rolled off the line in 2011: movies are now digital from camera to projector. The Post Office stumbles because nobody writes letters, even as Federal Express becomes the planet's supply chain.

Companies that get themselves on a feedback footing will dominate their industries, building better things faster for less money. Those that don't are already the walking dead, and will soon be little more than case studies and colorful anecdotes. Big data, new interfaces, and ubiquitous computing are tectonic shifts in the way we live and work.

A Feedback Economy

Big data, continuous optimization, and replacing everything with data pave the way for something far larger, and far more important, than simple business efficiency. They usher in a new era for humanity, with all its warts and glory. They herald the arrival of the feedback economy.

The efficiencies and optimizations that come from constant, iterative feedback will soon become the norm for businesses and governments. We're moving beyond an information economy. Information on its own isn't an advantage, anyway. Instead, this is the era of the feedback economy, and Boyd is, in many ways, the first feedback economist.

Alistair Croll is the founder of Bitcurrent, a research firm focused on emerging technologies. He's founded a variety of startups, and technology accelerators, including Year One Labs, CloudOps, Rednod, Coradiant (acquired by BMC in 2011) and Networkshop. He's a frequent speaker and writer on subjects such as entrepreneurship, cloud computing, Big Data, Internet performance and web technology, and has helped launch a number of major conferences on these topics.

CHAPTER 2
What Is Big Data?

By Edd Dumbill

Big data is data that exceeds the processing capacity of conventional database systems. The data is too big, moves too fast, or doesn't fit the strictures of your database architectures. To gain value from this data, you must choose an alternative way to process it.

The hot IT buzzword of 2012, big data has become viable as cost-effective approaches have emerged to tame the volume, velocity and variability of massive data. Within this data lie valuable patterns and information, previously hidden because of the amount of work required to extract them. To leading corporations, such as Walmart or Google, this power has been in reach for some time, but at fantastic cost. Today's commodity hardware, cloud architectures and open source software bring big data processing into the reach of the less well-resourced. Big data processing is eminently feasible for even the small garage startups, who can cheaply rent server time in the cloud.

The value of big data to an organization falls into two categories: analytical use, and enabling new products. Big data analytics can reveal insights hidden previously by data too costly to process, such as peer influence among customers, revealed by analyzing shoppers' transactions, social and geographical data. Being able to process every item of data in reasonable time removes the troublesome need for sampling and promotes an investigative approach to data, in contrast to the somewhat static nature of running predetermined reports.

The past decade's successful web startups are prime examples of big data used as an enabler of new products and services. For example, by combining a large number of signals from a user's actions and those of their friends, Facebook has been able to craft a highly personalized user experience and create a new kind of advertising business. It's no coincidence that the lion's share of ideas

and tools underpinning big data have emerged from Google, Yahoo, Amazon and Facebook.

The emergence of big data into the enterprise brings with it a necessary counterpart: agility. Successfully exploiting the value in big data requires experimentation and exploration. Whether creating new products or looking for ways to gain competitive advantage, the job calls for curiosity and an entrepreneurial outlook.

What Does Big Data Look Like?

As a catch-all term, "big data" can be pretty nebulous, in the same way that the term "cloud" covers diverse technologies. Input data to big data systems could be chatter from social networks, web server logs, traffic flow sensors, satellite imagery, broadcast audio streams, banking transactions, MP3s of rock music, the content of web pages, scans of government documents, GPS trails, telemetry from automobiles, financial market data, the list goes on. Are these all really the same thing?

To clarify matters, the three Vs of *volume, velocity and variety* are commonly used to characterize different aspects of big data. They're a helpful lens through which to view and understand the nature of the data and the software platforms available to exploit them. Most probably you will contend with each of the Vs to one degree or another.

Volume

The benefit gained from the ability to process large amounts of information is the main attraction of big data analytics. Having more data beats out having better models: simple bits of math can be unreasonably effective given large amounts of data. If you could run that forecast taking into account 300 factors rather than 6, could you predict demand better?

This volume presents the most immediate challenge to conventional IT structures. It calls for scalable storage, and a distributed approach to querying. Many companies already have large amounts of archived data, perhaps in the form of logs, but not the capacity to process it.

Assuming that the volumes of data are larger than those conventional relational database infrastructures can cope with, processing options break down broadly into a choice between massively parallel processing architectures — data warehouses or databases such as Greenplum — and Apache Hadoop-based solutions. This choice is often informed by the degree to which the one of the other "Vs" — variety — comes into play. Typically, data warehousing approaches involve predetermined schemas, suiting a regular and slowly evolving dataset. Apache Hadoop, on the other hand, places no conditions on the structure of the data it can process.

At its core, Hadoop is a platform for distributing computing problems across a number of servers. First developed and released as open source by Yahoo, it implements the MapReduce approach pioneered by Google in compiling its search indexes. Hadoop's MapReduce involves distributing a dataset among multiple servers and operating on the data: the "map" stage. The partial results are then recombined: the "reduce" stage.

To store data, Hadoop utilizes its own distributed filesystem, HDFS, which makes data available to multiple computing nodes. A typical Hadoop usage pattern involves three stages:

- loading data into HDFS,
- MapReduce operations, and
- retrieving results from HDFS.

This process is by nature a batch operation, suited for analytical or non-interactive computing tasks. Because of this, Hadoop is not itself a database or data warehouse solution, but can act as an analytical adjunct to one.

One of the most well-known Hadoop users is Facebook, whose model follows this pattern. A MySQL database stores the core data. This is then reflected into Hadoop, where computations occur, such as creating recommendations for

you based on your friends' interests. Facebook then transfers the results back into MySQL, for use in pages served to users.

Velocity

The importance of data's velocity — the increasing rate at which data flows into an organization — has followed a similar pattern to that of volume. Problems previously restricted to segments of industry are now presenting themselves in a much broader setting. Specialized companies such as financial traders have long turned systems that cope with fast moving data to their advantage. Now it's our turn.

Why is that so? The Internet and mobile era means that the way we deliver and consume products and services is increasingly instrumented, generating a data flow back to the provider. Online retailers are able to compile large histories of customers' every click and interaction: not just the final sales. Those who are able to quickly utilize that information, by recommending additional purchases, for instance, gain competitive advantage. The smartphone era increases again the rate of data inflow, as consumers carry with them a streaming source of geolocated imagery and audio data.

It's not just the velocity of the incoming data that's the issue: it's possible to stream fast-moving data into bulk storage for later batch processing, for example. The importance lies in the speed of the feedback loop, taking data from input through to decision. A commercial from IBM makes the point that you wouldn't cross the road if all you had was a five-minute old snapshot of traffic location. There are times when you simply won't be able to wait for a report to run or a Hadoop job to complete.

Industry terminology for such fast-moving data tends to be either "streaming data," or "complex event processing." This latter term was more established in product categories before streaming processing data gained more widespread relevance, and seems likely to diminish in favor of streaming.

There are two main reasons to consider streaming processing. The first is when the input data are too fast to store in their entirety: in order to keep storage requirements practical some level of analysis must occur as the data streams in. At the extreme end of the scale, the Large Hadron Collider at CERN generates so much data that scientists must discard the overwhelming majority of it — hoping hard they've not thrown away anything useful. The second reason to consider streaming is where the application mandates immediate response to the data. Thanks to the rise of mobile applications and online gaming this is an increasingly common situation.

Product categories for handling streaming data divide into established proprietary products such as IBM's InfoSphere Streams, and the less-polished and still emergent open source frameworks originating in the web industry: Twitter's Storm, and Yahoo S4.

As mentioned above, it's not just about input data. The velocity of a system's outputs can matter too. The tighter the feedback loop, the greater the competitive advantage. The results might go directly into a product, such as Facebook's recommendations, or into dashboards used to drive decision-making.

It's this need for speed, particularly on the web, that has driven the development of key-value stores and columnar databases, optimized for the fast retrieval of precomputed information. These databases form part of an umbrella category known as NoSQL, used when relational models aren't the right fit.

Variety

Rarely does data present itself in a form perfectly ordered and ready for processing. A common theme in big data systems is that the source data is diverse, and doesn't fall into neat relational structures. It could be text from social networks, image data, a raw feed directly from a sensor source. None of these things come ready for integration into an application.

Even on the web, where computer-to-computer communication ought to bring some guarantees, the reality of data is messy. Different browsers send different data, users withhold information, they may be using differing software versions or vendors to communicate with you. And you can bet that if part of the process involves a human, there will be error and inconsistency.

A common use of big data processing is to take unstructured data and extract ordered meaning, for consumption either by humans or as a structured input to an application. One such example is entity resolution, the process of determining exactly what a name refers to. Is this city London, England, or London, Texas? By the time your business logic gets to it, you don't want to be guessing.

The process of moving from source data to processed application data involves the loss of information. When you tidy up, you end up throwing stuff away. This underlines a principle of big data: *when you can, keep everything*. There may well be useful signals in the bits you throw away. If you lose the source data, there's no going back.

Despite the popularity and well understood nature of relational databases, it is not the case that they should always be the destination for data, even when tidied up. Certain data types suit certain classes of database better. For instance, documents encoded as XML are most versatile when stored in a dedicated XML store such as MarkLogic. Social network relations are graphs by

nature, and graph databases such as Neo4J make operations on them simpler and more efficient.

Even where there's not a radical data type mismatch, a disadvantage of the relational database is the static nature of its schemas. In an agile, exploratory environment, the results of computations will evolve with the detection and extraction of more signals. Semi-structured NoSQL databases meet this need for flexibility: they provide enough structure to organize data, but do not require the exact schema of the data before storing it.

In Practice

We have explored the nature of big data, and surveyed the landscape of big data from a high level. As usual, when it comes to deployment there are dimensions to consider over and above tool selection.

Cloud or in-house?

The majority of big data solutions are now provided in three forms: software-only, as an appliance or cloud-based. Decisions between which route to take will depend, among other things, on issues of data locality, privacy and regulation, human resources and project requirements. Many organizations opt for a hybrid solution: using on-demand cloud resources to supplement in-house deployments.

Big data is big

It is a fundamental fact that data that is too big to process conventionally is also too big to transport anywhere. IT is undergoing an inversion of priorities: it's the program that needs to move, not the data. If you want to analyze data from the U.S. Census, it's a lot easier to run your code on Amazon's web services platform, which hosts such data locally, and won't cost you time or money to transfer it.

Even if the data isn't too big to move, locality can still be an issue, especially with rapidly updating data. Financial trading systems crowd into data centers to get the fastest connection to source data, because that millisecond difference in processing time equates to competitive advantage.

Big data is messy

It's not all about infrastructure. Big data practitioners consistently report that 80% of the effort involved in dealing with data is cleaning it up in the first

place, as Pete Warden observes in his Big Data Glossary: "I probably spend more time turning messy source data into something usable than I do on the rest of the data analysis process combined."

Because of the high cost of data acquisition and cleaning, it's worth considering what you actually need to source yourself. Data marketplaces are a means of obtaining common data, and you are often able to contribute improvements back. Quality can of course be variable, but will increasingly be a benchmark on which data marketplaces compete.

Culture

The phenomenon of big data is closely tied to the emergence of data science, a discipline that combines math, programming and scientific instinct. Benefiting from big data means investing in teams with this skillset, and surrounding them with an organizational willingness to understand and use data for advantage.

In his report, "Building Data Science Teams," D.J. Patil characterizes data scientists as having the following qualities:

- Technical expertise: the best data scientists typically have deep expertise in some scientific discipline.
- Curiosity: a desire to go beneath the surface and discover and distill a problem down into a very clear set of hypotheses that can be tested.
- Storytelling: the ability to use data to tell a story and to be able to communicate it effectively.
- Cleverness: the ability to look at a problem in different, creative ways.

The far-reaching nature of big data analytics projects can have uncomfortable aspects: data must be broken out of silos in order to be mined, and the organization must learn how to communicate and interpet the results of analysis.

Those skills of storytelling and cleverness are the gateway factors that ultimately dictate whether the benefits of analytical labors are absorbed by an organization. The art and practice of visualizing data is becoming ever more important in bridging the human-computer gap to mediate analytical insight in a meaningful way.

Know where you want to go

Finally, remember that big data is no panacea. You can find patterns and clues in your data, but then what? Christer Johnson, IBM's leader for advanced

analytics in North America, gives this advice to businesses starting out with big data: first, decide what problem you want to solve.

If you pick a real business problem, such as how you can change your advertising strategy to increase spend per customer, it will guide your implementation. While big data work benefits from an enterprising spirit, it also benefits strongly from a concrete goal.

Edd Dumbill is a technologist, writer and programmer based in California. He is the program chair for the O'Reilly Strata and Open Source Convention Conferences.

CHAPTER 3

Apache Hadoop

By Edd Dumbill

Apache Hadoop has been the driving force behind the growth of the big data industry. You'll hear it mentioned often, along with associated technologies such as Hive and Pig. But what does it do, and why do you need all its strangely-named friends such as Oozie, Zookeeper and Flume?

Hadoop brings the ability to cheaply process large amounts of data, regardless of its structure. By large, we mean from 10-100 gigabytes and above. How is this different from what went before?

Existing enterprise data warehouses and relational databases excel at processing structured data, and can store massive amounts of data, though at cost. However, this requirement for structure restricts the kinds of data that can be processed, and it imposes an inertia that makes data warehouses unsuited for agile exploration of massive heterogenous data. The amount of effort required to warehouse data often means that valuable data sources in organizations are never mined. This is where Hadoop can make a big difference.

This article examines the components of the Hadoop ecosystem and explains the functions of each.

The Core of Hadoop: MapReduce

Created at Google in response to the problem of creating web search indexes, the MapReduce framework is the powerhouse behind most of today's big data processing. In addition to Hadoop, you'll find MapReduce inside MPP and NoSQL databases such as Vertica or MongoDB.

The important innovation of MapReduce is the ability to take a query over a dataset, divide it, and run it in parallel over multiple nodes. Distributing the computation solves the issue of data too large to fit onto a single machine. Combine this technique with commodity Linux servers and you have a cost-effective alternative to massive computing arrays.

At its core, Hadoop is an open source MapReduce implementation. Funded by Yahoo, it emerged in 2006 and, according to its creator Doug Cutting, reached "web scale" capability in early 2008.

As the Hadoop project matured, it acquired further components to enhance its usability and functionality. The name "Hadoop" has come to represent this entire ecosystem. There are parallels with the emergence of Linux: the name refers strictly to the Linux kernel, but it has gained acceptance as referring to a complete operating system.

Hadoop's Lower Levels: HDFS and MapReduce

We discussed above the ability of MapReduce to distribute computation over multiple servers. For that computation to take place, each server must have access to the data. This is the role of HDFS, the Hadoop Distributed File System.

HDFS and MapReduce are robust. Servers in a Hadoop cluster can fail, and not abort the computation process. HDFS ensures data is replicated with redundancy across the cluster. On completion of a calculation, a node will write its results back into HDFS.

There are no restrictions on the data that HDFS stores. Data may be unstructured and schemaless. By contrast, relational databases require that data be structured and schemas defined before storing the data. With HDFS, making sense of the data is the responsibility of the developer's code.

Programming Hadoop at the MapReduce level is a case of working with the Java APIs, and manually loading data files into HDFS.

Improving Programmability: Pig and Hive

Working directly with Java APIs can be tedious and error prone. It also restricts usage of Hadoop to Java programmers. Hadoop offers two solutions for making Hadoop programming easier.

- Pig is a programming language that simplifies the common tasks of working with Hadoop: loading data, expressing transformations on the data, and storing the final results. Pig's built-in operations can make sense of semi-structured data, such as log files, and the language is extensible using Java to add support for custom data types and transformations.
- Hive enables Hadoop to operate as a data warehouse. It superimposes structure on data in HDFS, and then permits queries over the data using a familiar SQL-like syntax. As with Pig, Hive's core capabilities are extensible.

Choosing between Hive and Pig can be confusing. Hive is more suitable for data warehousing tasks, with predominantly static structure and the need for frequent analysis. Hive's closeness to SQL makes it an ideal point of integration between Hadoop and other business intelligence tools.

Pig gives the developer more agility for the exploration of large datasets, allowing the development of succinct scripts for transforming data flows for incorporation into larger applications. Pig is a thinner layer over Hadoop than Hive, and its main advantage is to drastically cut the amount of code needed compared to direct use of Hadoop's Java APIs. As such, Pig's intended audience remains primarily the software developer.

Improving Data Access: HBase, Sqoop, and Flume

At its heart, Hadoop is a batch-oriented system. Data are loaded into HDFS, processed, and then retrieved. This is somewhat of a computing throwback, and often interactive and random access to data is required.

Enter HBase, a column-oriented database that runs on top of HDFS. Modeled after Google's BigTable, the project's goal is to host billions of rows of data for rapid access. MapReduce can use HBase as both a source and a destination for its computations, and Hive and Pig can be used in combination with HBase.

In order to grant random access to the data, HBase does impose a few restrictions: performance with Hive is 4-5 times slower than plain HDFS, and the maximum amount of data you can store is approximately a petabyte, versus HDFS' limit of over 30PB.

HBase is ill-suited to ad-hoc analytics, and more appropriate for integrating big data as part of a larger application. Use cases include logging, counting and storing time-series data.

The Hadoop Bestiary

Ambari	Deployment, configuration and monitoring
Flume	Collection and import of log and event data
HBase	Column-oriented database scaling to billions of rows
HCatalog	Schema and data type sharing over Pig, Hive and MapReduce
HDFS	Distributed redundant filesystem for Hadoop
Hive	Data warehouse with SQL-like access
Mahout	Library of machine learning and data mining algorithms
MapReduce	Parallel computation on server clusters
Pig	High-level programming language for Hadoop computations
Oozie	Orchestration and workflow management
Sqoop	Imports data from relational databases
Whirr	Cloud-agnostic deployment of clusters
Zookeeper	Configuration management and coordination

Getting data in and out

Improved interoperability with the rest of the data world is provided by Sqoop and Flume. Sqoop is a tool designed to import data from relational databases into Hadoop: either directly into HDFS, or into Hive. Flume is designed to import streaming flows of log data directly into HDFS.

Hive's SQL friendliness means that it can be used as a point of integration with the vast universe of database tools capable of making connections via JBDC or ODBC database drivers.

Coordination and Workflow: Zookeeper and Oozie

With a growing family of services running as part of a Hadoop cluster, there's a need for coordination and naming services. As computing nodes can come and go, members of the cluster need to synchronize with each other, know where to access services, and how they should be configured. This is the purpose of Zookeeper.

Production systems utilizing Hadoop can often contain complex pipelines of transformations, each with dependencies on each other. For example, the arrival of a new batch of data will trigger an import, which must then trigger recalculates in dependent datasets. The Oozie component provides features to manage the workflow and dependencies, removing the need for developers to code custom solutions.

Management and Deployment: Ambari and Whirr

One of the commonly added features incorporated into Hadoop by distributors such as IBM and Microsoft is monitoring and administration. Though in an early stage, Ambari aims to add these features to the core Hadoop project. Ambari is intended to help system administrators deploy and configure Hadoop, upgrade clusters, and monitor services. Through an API it may be integrated with other system management tools.

Though not strictly part of Hadoop, Whirr is a highly complementary component. It offers a way of running services, including Hadoop, on cloud platforms. Whirr is cloud-neutral, and currently supports the Amazon EC2 and Rackspace services.

Machine Learning: Mahout

Every organization's data are diverse and particular to their needs. However, there is much less diversity in the kinds of analyses performed on that data. The Mahout project is a library of Hadoop implementations of common analytical computations. Use cases include user collaborative filtering, user recommendations, clustering and classification.

Using Hadoop

Normally, you will use Hadoop in the form of a distribution. Much as with Linux before it, vendors integrate and test the components of the Apache Hadoop ecosystem, and add in tools and administrative features of their own.

Though not *per se* a distribution, a managed cloud installation of Hadoop's MapReduce is also available through Amazon's Elastic MapReduce service.

CHAPTER 4
Big Data Market Survey

By Edd Dumbill

The big data ecosystem can be confusing. The popularity of "big data" as industry buzzword has created a broad category. As Hadoop steamrolls through the industry, solutions from the business intelligence and data warehousing fields are also attracting the big data label. To confuse matters, Hadoop-based solutions such as Hive are at the same time evolving toward being a competitive data warehousing solution.

Understanding the nature of your big data problem is a helpful first step in evaluating potential solutions. Let's remind ourselves of the definition of big data:

> "Big data is data that exceeds the processing capacity of conventional database systems. The data is too big, moves too fast, or doesn't fit the strictures of your database architectures. To gain value from this data, you must choose an alternative way to process it."

Big data problems vary in how heavily they weigh in on the axes of volume, velocity and variability. Predominantly structured yet large data, for example, may be most suited to an analytical database approach.

This survey makes the assumption that a data warehousing solution alone is not the answer to your problems, and concentrates on analyzing the commercial Hadoop ecosystem. We'll focus on the solutions that incorporate storage and data processing, excluding those products which only sit above those layers, such as the visualization or analytical workbench software.

Getting started with Hadoop doesn't require a large investment as the software is open source, and is also available instantly through the Amazon Web Services cloud. But for production environments, support, professional services and training are often required.

Just Hadoop?

Apache Hadoop is unquestionably the center of the latest iteration of big data solutions. At its heart, Hadoop is a system for distributing computation among commodity servers. It is often used with the Hadoop Hive project, which layers data warehouse technology on top of Hadoop, enabling ad-hoc analytical queries.

Big data platforms divide along the lines of their approach to Hadoop. The big data offerings from familiar enterprise vendors incorporate a Hadoop distribution, while other platforms offer Hadoop connectors to their existing analytical database systems. This latter category tends to comprise massively parallel processing (MPP) databases that made their name in big data before Hadoop matured: Vertica and Aster Data. Hadoop's strength in these cases is in processing unstructured data in tandem with the analytical capabilities of the existing database on structured or structured data.

Practical big data implementations don't in general fall neatly into either structured or unstructured data categories. You will invariably find Hadoop working as part of a system with a relational or MPP database.

Much as with Linux before it, no Hadoop solution incorporates the raw Apache Hadoop code. Instead, it's packaged into distributions. At a minimum, these distributions have been through a testing process, and often include additional components such as management and monitoring tools. The most well-used distributions now come from Cloudera, Hortonworks and MapR. Not every distribution will be commercial, however: the BigTop project aims to create a Hadoop distribution under the Apache umbrella.

Integrated Hadoop Systems

The leading Hadoop enterprise software vendors have aligned their Hadoop products with the rest of their database and analytical offerings. These vendors don't require you to source Hadoop from another party, and offer it as a core part of their big data solutions. Their offerings integrate Hadoop into a broader enterprise setting, augmented by analytical and workflow tools.

EMC Greenplum

EMC Greenplum

Database

Greenplum Database

Deployment options

Appliance (Modular Data Computing Appliance), Software (Enterprise Linux)

Hadoop

Bundled distribution (Greenplum HD); Hive, Pig, Zookeeper, HBase

NoSQL component

HBase

Links

Home page, case study

Acquired by EMC, and rapidly taken to the heart of the company's strategy, Greenplum is a relative newcomer to the enterprise, compared to other companies in this section. They have turned that to their advantage in creating an analytic platform, positioned as taking analytics "beyond BI" with agile data science teams.

Greenplum's Unified Analytics Platform (UAP) comprises three elements: the Greenplum MPP database, for structured data; a Hadoop distribution, Greenplum HD; and Chorus, a productivity and groupware layer for data science teams.

The HD Hadoop layer builds on MapR's Hadoop compatible distribution, which replaces the file system with a faster implementation and provides other features for robustness. Interoperability between HD and Greenplum Database means that a single query can access both database and Hadoop data.

Chorus is a unique feature, and is indicative of Greenplum's commitment to the idea of data science and the importance of the agile team element to effectively exploiting big data. It supports organizational roles from analysts, data scientists and DBAs through to executive business stakeholders.

As befits EMC's role in the data center market, Greenplum's UAP is available in a modular appliance configuration.

IBM

IBM InfoSphere

Database

DB2

Deployment options

Software (Enterprise Linux), Cloud

Hadoop

Bundled distribution (InfoSphere BigInsights); Hive, Oozie, Pig, Zookeeper, Avro, Flume, HBase, Lucene

NoSQL component

HBase

Links

Home page, case study

IBM's InfoSphere BigInsights is their Hadoop distribution, and part of a suite of products offered under the "InfoSphere" information management brand. Everything big data at IBM is helpfully labeled Big, appropriately enough for a company affectionately known as "Big Blue."

BigInsights augments Hadoop with a variety of features, including management and administration tools. It also offers textual analysis tools that aid with entity resolution — identifying people, addresses, phone numbers and so on.

IBM's Jaql query language provides a point of integration between Hadoop and other IBM products, such as relational databases or Netezza data warehouses.

InfoSphere BigInsights is interoperable with IBM's other database and warehouse products, including DB2, Netezza and its InfoSphere warehouse and analytics lines. To aid analytical exploration, BigInsights ships with BigSheets, a spreadsheet interface onto big data.

IBM addresses streaming big data separately through its InfoSphere Streams product. BigInsights is not currently offered in an appliance form, but can be used in the cloud via Rightscale, Amazon, Rackspace, and IBM Smart Enterprise Cloud.

Microsoft

Microsoft

Database

SQL Server

Deployment options

Software (Windows Server), Cloud (Windows Azure Cloud)

Hadoop

Bundled distribution (Big Data Solution); Hive, Pig

Links

Home page, case study

Microsoft have adopted Hadoop as the center of their big data offering, and are pursuing an integrated approach aimed at making big data available through their analytical tool suite, including to the familiar tools of Excel and PowerPivot.

Microsoft's Big Data Solution brings Hadoop to the Windows Server platform, and in elastic form to their cloud platform Windows Azure. Microsoft have packaged their own distribution of Hadoop, integrated with Windows Systems Center and Active Directory. They intend to contribute back changes to Apache Hadoop to ensure that an open source version of Hadoop will run on Windows.

On the server side, Microsoft offer integrations to their SQL Server database and their data warehouse product. Using their warehouse solutions aren't mandated, however. The Hadoop Hive data warehouse is part of the Big Data Solution, including connectors from Hive to ODBC and Excel.

Microsoft's focus on the developer is evident in their creation of a JavaScript API for Hadoop. Using JavaScript, developers can create Hadoop jobs for MapReduce, Pig or Hive, even from a browser-based environment. Visual Studio and .NET integration with Hadoop is also provided.

Deployment is possible either on the server or in the cloud, or as a hybrid combination. Jobs written against the Apache Hadoop distribution should migrate with miniminal changes to Microsoft's environment.

Oracle

Oracle Big Data

Deployment options

Appliance (Oracle Big Data Appliance)

Hadoop

Bundled distribution (Cloudera's Distribution including Apache Hadoop); Hive, Oozie, Pig, Zookeeper, Avro, Flume, HBase, Sqoop, Mahout, Whirr

NoSQL component

Oracle NoSQL Database

Links

Home page

Announcing their entry into the big data market at the end of 2011, Oracle is taking an appliance-based approach. Their Big Data Appliance integrates Hadoop, R for analytics, a new Oracle NoSQL database, and connectors to Oracle's database and Exadata data warehousing product line.

Oracle's approach caters to the high-end enterprise market, and particularly leans to the rapid-deployment, high-performance end of the spectrum. It is the only vendor to include the popular R analytical language integrated with Hadoop, and to ship a NoSQL database of their own design as opposed to Hadoop HBase.

Rather than developing their own Hadoop distribution, Oracle have partnered with Cloudera for Hadoop support, which brings them a mature and established Hadoop solution. Database connectors again promote the integration of structured Oracle data with the unstructured data stored in Hadoop HDFS.

Oracle's NoSQL Database is a scalable key-value database, built on the Berkeley DB technology. In that, Oracle owes double gratitude to Cloudera CEO Mike Olson, as he was previously the CEO of Sleepycat, the creators of Berkeley DB. Oracle are positioning their NoSQL database as a means of acquiring big data prior to analysis.

The Oracle R Enterprise product offers direct integration into the Oracle database, as well as Hadoop, enabling R scripts to run on data without having to round-trip it out of the data stores.

Availability

While IBM and Greenplum's offerings are available at the time of writing, the Microsoft and Oracle solutions are expected to be fully available early in 2012.

Analytical Databases with Hadoop Connectivity

MPP (massively parallel processing) databases are specialized for processing structured big data, as distinct from the unstructured data that is Hadoop's specialty. Along with Greenplum, Aster Data and Vertica are early pioneers of big data products before the mainstream emergence of Hadoop.

These MPP solutions are databases specialized for analyical workloads and data integration, and provide connectors to Hadoop and data warehouses. A recent spate of acquisitions have seen these products become the analytical play by data warehouse and storage vendors: Teradata acquired Aster Data, EMC acquired Greenplum, and HP acquired Vertica.

Quick facts

Aster Data	ParAccel	Vertica
Database	Database	Database
MPP analytical database	MPP analytical database	MPP analytical database
Deployment options	**Deployment options**	**Deployment options**
Appliance (Aster MapReduce Appliance), Software (Enterprise Linux), Cloud (Amazon EC2, Terremark and Dell Clouds)	Software (Enterprise Linux), Cloud (Cloud Edition)	Appliance (HP Vertica Appliance), Software (Enterprise Linux), Cloud (Cloud and Virtualized)
Hadoop	**Hadoop**	**Hadoop**
Hadoop connector available	Hadoop integration available	Hadoop and Pig connectors available
Links	**Links**	**Links**
Home page	Home page, case study	Home page, case study

Hadoop-Centered Companies

Directly employing Hadoop is another route to creating a big data solution, especially where your infrastructure doesn't fall neatly into the product line of major vendors. Practically every database now features Hadoop connectivity, and there are multiple Hadoop distributions to choose from.

Reflecting the developer-driven ethos of the big data world, Hadoop distributions are frequently offered in a community edition. Such editions lack enterprise management features, but contain all the functionality needed for evaluation and development.

The first iterations of Hadoop distributions, from Cloudera and IBM, focused on usability and adminstration. We are now seeing the addition of performance-oriented improvements to Hadoop, such as those from MapR and Platform Computing. While maintaining API compatibility, these vendors replace slow or fragile parts of the Apache distribution with better performing or more robust components.

Cloudera

The longest-established provider of Hadoop distributions, Cloudera provides an enterprise Hadoop solution, alongside services, training and support options. Along with Yahoo, Cloudera have made deep open source contributions to Hadoop, and through hosting industry conferences have done much to establish Hadoop in its current position.

Hortonworks

Though a recent entrant to the market, Hortonworks have a long history with Hadoop. Spun off from Yahoo, where Hadoop originated, Hortonworks aims to stick close to and promote the core Apache Hadoop technology. Hortonworks also have a partnership with Microsoft to assist and accelerate their Hadoop integration.

Hortonworks Data Platform is currently in a limited preview phase, with a public preview expected in early 2012. The company also provides support and training.

An overview of Hadoop distributions (part 1)

	Cloudera	EMC Greenplum	Hortonworks	IBM
Product Name	Cloudera's Distribution including Apache Hadoop	Greenplum HD	Hortonworks Data Platform	InfoSphere BigInsights
Free Edition	CDH	Community Edition		Basic Edition
	Integrated, tested distribution of Apache Hadoop	100% open source certified and supported version of the Apache Hadoop stack		An integrated Hadoop distribution.

	Cloudera	EMC Greenplum	Hortonworks	IBM
Enterprise Edition	Cloudera Enterprise Adds management software layer over CDH	Enterprise Edition Integrates MapR's M5 Hadoop-compatible distribution, replaces HDFS with MapR's C++-based file system. Includes MapR management tools		Enterprise Edition Hadoop distribution, plus BigSheets spreadsheet interface, scheduler, text analytics, indexer, JDBC connector, security support.
Hadoop Components	Hive, Oozie, Pig, Zookeeper, Avro, Flume, HBase, Sqoop, Mahout, Whirr	Hive, Pig, Zookeeper, HBase	Hive, Pig, Zookeeper, HBase, None, Ambari	Hive, Oozie, Pig, Zookeeper, Avro, Flume, HBase, Lucene
Security	Cloudera Manager Kerberos, role-based administration and audit trails			Security features LDAP authentication, role-based authorization, reverse proxy
Admin Interface	Cloudera Manager Centralized management and alerting	Administrative interfaces MapR Heatmap cluster administrative tools	Apache Ambari Monitoring, administration and lifecycle management for Hadoop clusters	Administrative interfaces Administrative features including Hadoop HDFS and MapReduce administration, cluster and server management, view HDFS file content
Job Management	Cloudera Manager Job analytics, monitoring and log search	High-availability job management JobTracker HA and Distributed NameNode HA prevent lost jobs, restarts and failover incidents	Apache Ambari Monitoring, administration and lifecycle management for Hadoop clusters	Job management features Job creation, submission, cancellation, status, logging.
Database connectors		Greenplum Database		DB2, Netezza, InfoSphere Warehouse
Interop features				
HDFS Access	Fuse-DFS	NFS	WebHDFS	

	Cloudera	EMC Greenplum	Hortonworks	IBM
	Mount HDFS as a traditional file-system	Access HDFS as a conventional network file system	REST API to HDFS	
Installation	Cloudera Manager Wizard-based deployment			Quick installation GUI-driven installation tool
Additional APIs				Jaql Jaql is a functional, declarative query language designed to process large data sets.
Volume Management				

An overview of Hadoop distributions (part 2)

	MapR	Microsoft	Platform Computing
Product Name	MapR	Big Data Solution	Platform MapReduce
Free Edition	MapR M3 Edition Free community edition incorporating MapR's performance increases		Platform MapReduce Developer Edition Evaluation edition, excludes resource management features of regualt edition
Enterprise Edition	MapR M5 Edition Augments M3 Edition with high availability and data protection features	Big Data Solution Windows Hadoop distribution, integrated with Microsoft's database and analytical products	Platform MapReduce Enhanced runtime for Hadoop MapReduce, API-compatible with Apache Hadoop
Hadoop Components	Hive, Pig, Flume, HBase, Sqoop, Mahout, None, Oozie	Hive, Pig	
Security		Active Directory integration	
Admin Interface	Administrative interfaces MapR Heatmap cluster administrative tools	System Center integration	Administrative interfaces

	MapR	Microsoft	Platform Computing
			Platform MapReduce Workload Manager
Job Management	High-availability job management JobTracker HA and Distributed NameNode HA prevent lost jobs, restarts and failover incidents		
Database connectors		SQL Server, SQL Server Parallel Data Warehouse	
Interop features		Hive ODBC Driver, Excel Hive Add-in	
HDFS Access	NFS Access HDFS as a conventional network file system		
Installation			
Additional APIs	REST API	JavaScript API JavaScript Map/Reduce jobs, Pig-Latin, and Hive queries	Includes R, C/C++, C#, Java, Python
Volume Management	Mirroring, snapshots		

Notes

- **Pure cloud solutions**: Both Amazon Web Services and Google offer cloud-based big data solutions. These will be reviewed separately.
- **HPCC**: Though dominant, Hadoop is not the only big data solution. LexisNexis' HPCC offers an alternative approach.
- **Hadapt**: not yet featured in this survey. Taking a different approach from both Hadoop-centered and MPP solutions, Hadapt integrates unstructured and structured data into one product: wrapping rather than exposing Hadoop. It is currently in "early access" stage.
- **NoSQL**: Solutions built on databases such as Cassandra, MongoDB and Couchbase are not in the scope of this survey, though these databases do offer Hadoop integration.
- **Errors and omissions**: given the fast-evolving nature of the market and variable quality of public information, any feedback about errors and

omissions from this survey is most welcome. Please send it to *edd+bigdata@oreilly.com*.

CHAPTER 5

Microsoft's Plan for Big Data

By Edd Dumbill

Microsoft has placed Apache Hadoop at the core of its big data strategy. It's a move that might seem surprising to the casual observer, being a somewhat enthusiastic adoption of a significant open source product.

The reason for this move is that Hadoop, by its sheer popularity, has become the de facto standard for distributed data crunching. By embracing Hadoop, Microsoft allows its customers to access the rapidly-growing Hadoop ecosystem and take advantage of a growing talent pool of Hadoop-savvy developers.

Microsoft's goals go beyond integrating Hadoop into Windows. It intends to contribute the adaptions it makes back to the Apache Hadoop project, so that anybody can run a purely open source Hadoop on Windows.

Microsoft's Hadoop Distribution

The Microsoft distribution of Hadoop is currently in "Customer Technology Preview" phase. This means it is undergoing evaluation in the field by groups of customers. The expected release time is toward the middle of 2012, but will be influenced by the results of the technology preview program.

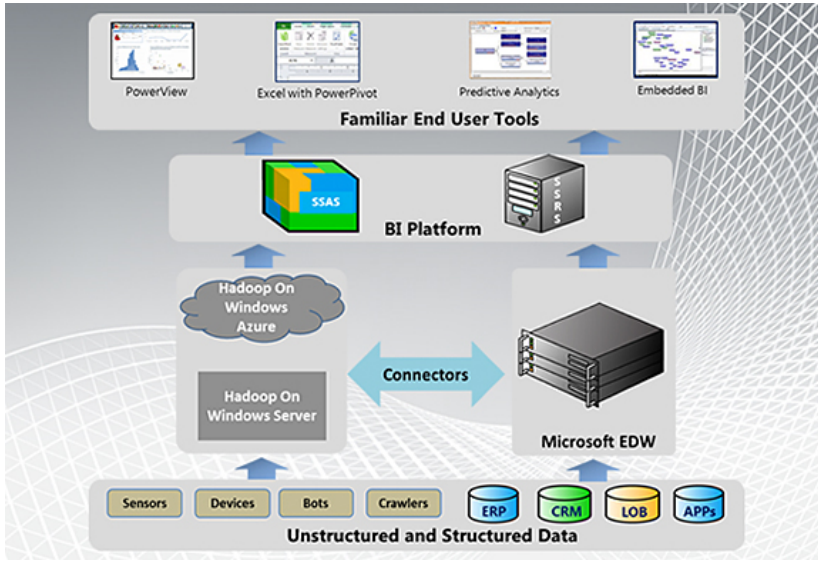
Microsoft's Hadoop distribution is usable either on-premise with Windows Server, or in Microsoft's cloud platform, Windows Azure. The core of the product is in the MapReduce, HDFS, Pig and Hive components of Hadoop. These are certain to ship in the 1.0 release.

As Microsoft's aim is for 100% Hadoop compatibility, it is likely that additional components of the Hadoop ecosystem such as Zookeeper, HBase, HCatalog and Mahout will also be shipped.

Additional components integrate Hadoop with Microsoft's ecosystem of business intelligence and analytical products:

- Connectors for Hadoop, integrating it with SQL Server and SQL Sever Parallel Data Warehouse.
- An ODBC driver for Hive, permitting any Windows application to access and run queries against the Hive data warehouse.
- An Excel Hive Add-in, which enables the movement of data directly from Hive into Excel or PowerPivot.

On the back end, Microsoft offers Hadoop performance improvements, integration with Active Directory to facilitate access control, and with System Center for administration and management.



How Hadoop integrates with the Microsoft ecosystem. (Source: microsoft.com.)

Developers, Developers, Developers

One of the most interesting features of Microsoft's work with Hadoop is the addition of a JavaScript API. Working with Hadoop at a programmatic level can be tedious: this is why higher-level languages such as Pig emerged.

Driven by its focus on the software developer as an important customer, Microsoft chose to add a JavaScript layer to the Hadoop ecosystem. Developers can use it to create MapReduce jobs, and even interact with Pig and Hive from a browser environment.

The real advantage of the JavaScript layer should show itself in integrating Hadoop into a business environment, making it easy for developers to create intranet analytical environments accessible by business users. Combined with Microsoft's focus on bringing server-side JavaScript to Windows and Azure through Node.js, this gives an interesting glimpse into Microsoft's view of where developer enthusiasm and talent will lie.

It's also good news for the broader Hadoop community, as Microsoft intends to contribute its JavaScript API to the Apache Hadoop open source project itself.

The other half of Microsoft's software development environment is of course the .NET platform. With Microsoft's Hadoop distribution, it will be possible to create MapReduce jobs from .NET, though using the Hadoop APIs directly. It is likely that higher-level interfaces will emerge in future releases. The same applies to Visual Studio, which over time will get increasing levels of Hadoop project support.

Streaming Data and NoSQL

Hadoop covers part of the big data problem, but what about streaming data processing or NoSQL databases? The answer comes in two parts, covering existing Microsoft products and future Hadoop-compatible solutions.

Microsoft has some established products: Its streaming data solution called StreamInsight, and for NoSQL, Windows Azure has a product called Azure Tables.

Looking to the future, the commitment of Hadoop compatibility means that streaming data solutions and NoSQL databases designed to be part of the Hadoop ecosystem should work with the Microsoft distribution — HBase itself will ship as a core offering. It seems likely that solutions such as S4 will prove compatible.

Toward an Integrated Environment

Now that Microsoft is on the way to integrating the major components of big data tooling, does it intend to join it all together to provide an integrated data science platform for businesses?

That's certainly the vision, according to Madhu Reddy, senior product planner for Microsoft Big Data: "Hadoop is primarily for developers. We want to enable people to use the tools they like."

The strategy to achieve this involves entry points at multiple levels: for developers, analysts and business users. Instead of choosing one particular analytical platform of choice, Microsoft will focus on interoperability with existing tools. Excel is an obvious priority, but other tools are also important to the company.

According to Reddy, data scientists represent a spectrum of preferences. While Excel is a ubiquitous and popular choice, other customers use Matlab, SAS, or R, for example.

The Data Marketplace

One thing unique to Microsoft as a big data and cloud platform is its data market, Windows Azure Marketplace. Mixing external data, such as geographical or social, with your own, can generate revealing insights. But it's hard to find data, be confident of its quality, and purchase it conveniently. That's where data marketplaces meet a need.

The availability of the Azure marketplace integrated with Microsoft's tools gives analysts a ready source of external data with some guarantees of quality. Marketplaces are in their infancy now, but will play a growing role in the future of data-driven business.

Summary

The Microsoft approach to big data has ensured the continuing relevance of its Windows platform for web-era organizations, and makes its cloud services a competitive choice for data-centered businesses.

Appropriately enough for a company with a large and diverse software ecosystem of its own, the Microsoft approach is one of interoperability. Rather than laying out a golden path for big data, as suggested by the appliance-oriented approach of others, Microsoft is focusing heavily on integration.

The guarantee of this approach lies in Microsoft's choice to embrace and work with the Apache Hadoop community, enabling the migration of new tools and talented developers to its platform.

CHAPTER 6

Big Data in the Cloud

By Edd Dumbill

Big data and cloud technology go hand-in-hand. Big data needs clusters of servers for processing, which clouds can readily provide. So goes the marketing message, but what does that look like in reality? Both "cloud" and "big data" have broad definitions, obscured by considerable hype. This article breaks down the landscape as simply as possible, highlighting what's practical, and what's to come.

IaaS and Private Clouds

What is often called "cloud" amounts to virtualized servers: computing resource that presents itself as a regular server, rentable per consumption. This is generally called infrastructure as a service (IaaS), and is offered by platforms such as Rackspace Cloud or Amazon EC2. You buy time on these services, and install and configure your own software, such as a Hadoop cluster or NoSQL database. Most of the solutions I described in my Big Data Market Survey can be deployed on IaaS services.

Using IaaS clouds doesn't mean you must handle all deployment manually: good news for the clusters of machines big data requires. You can use orchestration frameworks, which handle the management of resources, and automated infrastructure tools, which handle server installation and configuration. RightScale offers a commercial multi-cloud management platform that mitigates some of the problems of managing servers in the cloud.

Frameworks such as OpenStack and Eucalyptus aim to present a uniform interface to both private data centers and the public cloud. Attracting a strong flow of cross industry support, OpenStack currently addresses computing resource (akin to Amazon's EC2) and storage (parallels Amazon S3).

The race is on to make private clouds and IaaS services more usable: over the next two years using clouds should become much more straightforward as vendors adopt the nascent standards. There'll be a uniform interface, whether you're using public or private cloud facilities, or a hybrid of the two.

Particular to big data, several configuration tools already target Hadoop explicitly: among them Dell's Crowbar, which aims to make deploying and configuring clusters simple, and Apache Whirr, which is specialized for running Hadoop services and other clustered data processing systems.

Today, using IaaS gives you a broad choice of cloud supplier, the option of using a private cloud, and complete control: but you'll be responsible for deploying, managing and maintaining your clusters.

Platform solutions

Using IaaS only brings you so far for with big data applications: they handle the creation of computing and storage resources, but don't address anything at a higher level. The set up of Hadoop and Hive or a similar solution is down to you.

Beyond IaaS, several cloud services provide application layer support for big data work. Sometimes referred to as managed solutions, or platform as a service (PaaS), these services remove the need to configure or scale things such as databases or MapReduce, reducing your workload and maintenance burden. Additionally, PaaS providers can realize great efficiencies by hosting at the application level, and pass those savings on to the customer.

The general PaaS market is burgeoning, with major players including VMware (Cloud Foundry) and Salesforce (Heroku, force.com). As big data and machine learning requirements percolate through the industry, these players are likely to add their own big-data-specific services. For the purposes of this article, though, I will be sticking to the vendors who already have implemented big data solutions.

Today's primary providers of such big data platform services are Amazon, Google and Microsoft. You can see their offerings summarized in the table toward the end of this article. Both Amazon Web Services and Microsoft's Azure blur the lines between infrastructure as a service and platform: you can mix and match. By contrast, Google's philosophy is to skip the notion of a server altogether, and focus only on the concept of the application. Among these, only Amazon can lay claim to extensive experience with their product.

Amazon Web Services

Amazon has significant experience in hosting big data processing. Use of Amazon EC2 for Hadoop was a popular and natural move for many early adopters of big data, thanks to Amazon's expandable supply of compute power. Building on this, Amazon launched Elastic Map Reduce in 2009, providing a hosted, scalable Hadoop service.

Applications on Amazon's platform can pick from the best of both the IaaS and PaaS worlds. General purpose EC2 servers host applications that can then access the appropriate special purpose managed solutions provided by Amazon.

As well as Elastic Map Reduce, Amazon offers several other services relevant to big data, such as the Simple Queue Service for coordinating distributed computing, and a hosted relational database service. At the specialist end of big data, Amazon's High Performance Computing solutions are tuned for low-latency cluster computing, of the sort required by scientific and engineering applications.

Elastic Map Reduce

Elastic Map Reduce (EMR) can be programmed in the usual Hadoop ways, through Pig, Hive or other programming language, and uses Amazon's S3 storage service to get data in and out.

Access to Elastic Map Reduce is through Amazon's SDKs and tools, or with GUI analytical and IDE products such as those offered by Karmasphere. In conjunction with these tools, EMR represents a strong option for experimental and analytical work. Amazon's EMR pricing makes it a much more attractive option to use EMR, rather than configure EC2 instances yourself to run Hadoop.

When integrating Hadoop with applications generating structured data, using S3 as the main data source can be unwieldy. This is because, similar to Hadoop's HDFS, S3 works at the level of storing blobs of opaque data. Hadoop's answer to this is HBase, a NoSQL database that integrates with the rest of the Hadoop stack. Unfortunately, Amazon does not currently offer HBase with Elastic Map Reduce.

DynamoDB

Instead of HBase, Amazon provides DynamoDB, its own managed, scalable NoSQL database. As this a managed solution, it represents a better choice than running your own database on top of EC2, in terms of both performance and economy.

DynamoDB data can be exported to and imported from S3, providing interoperability with EMR.

Google

Google's cloud platform stands out as distinct from its competitors. Rather than offering virtualization, it provides an application container with defined APIs and services. Developers do not need to concern themselves with the concept of machines: applications execute in the cloud, getting access to as much processing power as they need, within defined resource usage limits.

To use Google's platform, you must work within the constraints of its APIs. However, if that fits, you can reap the benefits of the security, tuning and performance improvements inherent to the way Google develops all its services.

AppEngine, Google's cloud application hosting service, offers a MapReduce facility for parallel computation over data, but this is more of a feature for use as part of complex applications rather than for analytical purposes. Instead, BigQuery and the Prediction API form the core of Google's big data offering, respectively offering analysis and machine learning facilities. Both these services are available exclusively via REST APIs, consistent with Google's vision for web-based computing.

BigQuery

BigQuery is an analytical database, suitable for interactive analysis over datasets of the order of 1TB. It works best on a small number of tables with a large number of rows. BigQuery offers a familiar SQL interface to its data. In that, it is comparable to Apache Hive, but the typical performance is faster, making BigQuery a good choice for exploratory data analysis.

Getting data into BigQuery is a matter of directly uploading it, or importing it from Google's Cloud Storage system. This is the aspect of BigQuery with the biggest room for improvement. Whereas Amazon's S3 lets you mail in disks for import, Google doesn't currently have this facility. Streaming data into BigQuery isn't viable either, so regular imports are required for constantly updating data. Finally, as BigQuery only accepts data formatted as comma-separated value (CSV) files, you will need to use external methods to clean up the data beforehand.

Rather than provide end-user interfaces itself, Google wants an ecosystem to grow around BigQuery, with vendors incorporating it into their products, in the same way Elastic Map Reduce has acquired tool integration. Currently in

beta test, to which anybody can apply, BigQuery is expected to be publicly available during 2012.

Prediction API

Many uses of machine learning are well defined, such as classification, sentiment analysis, or recommendation generation. To meet these needs, Google offers its Prediction API product.

Applications using the Prediction API work by creating and training a model hosted within Google's system. Once trained, this model can be used to make predictions, such as spam detection. Google is working on allowing these models to be shared, optionally with a fee. This will let you take advantage of previously trained models, which in many cases will save you time and expertise with training.

Though promising, Google's offerings are in their early days. Further integration between its services is required, as well as time for ecosystem development to make their tools more approachable.

Microsoft

I have written in some detail about Microsoft's big data strategy in Microsoft's plan for Hadoop and big data. By offering its data platforms on Windows Azure in addition to Windows Server, Microsoft's aim is to make either on-premise or cloud-based deployments equally viable with its technology. Azure parallels Amazon's web service offerings in many ways, offering a mix of IaaS services with managed applications such as SQL Server.

Hadoop is the central pillar of Microsoft's big data approach, surrounded by the ecosystem of its own database and business intelligence tools. For organizations already invested in the Microsoft platform, Azure will represent the smoothest route for integrating big data into the operation. Azure itself is pragmatic about language choice, supporting technologies such as Java, PHP and Node.js in addition to Microsoft's own.

As with Google's BigQuery, Microsoft's Hadoop solution is currently in closed beta test, and is expected to be generally available sometime in the middle of 2012.

Big data cloud platforms compared

The following table summarizes the data storage and analysis capabilities of Amazon, Google and Microsoft's cloud platforms. Intentionally excluded are IaaS solutions without dedicated big data offerings.

	Amazon	Google	Microsoft
Product(s)	Amazon Web Services	Google Cloud Services	Windows Azure
Big data storage	S3	Cloud Storage	HDFS on Azure
Working storage	Elastic Block Store	AppEngine (Datastore, Blobstore)	Blob, table, queues
NoSQL database	DynamoDB[1 on page 45]	AppEngine Datastore	Table storage
Relational database	Relational Database Service (MySQL or Oracle)	Cloud SQL (MySQL compatible)	SQL Azure
Application hosting	EC2	AppEngine	Azure Compute
Map/Reduce service	Elastic MapReduce (Hadoop)	AppEngine (limited capacity)	Hadoop on Azure[2 on page 45]
Big data analytics	Elastic MapReduce (Hadoop interface[3 on page 45])	BigQuery[2 on page 45] (TB-scale, SQL interface)	Hadoop on Azure (Hadoop interface[3 on page 45])
Machine learning	Via Hadoop + Mahout on EMR or EC2	Prediction API	Mahout with Hadoop
Streaming processing	Nothing prepackaged: use custom solution on EC2	Prospective Search API[4 on page 45]	StreamInsight[2 on page 45] ("Project Austin")
Data import	Network, physically ship drives	Network	Network
Data sources	Public Data Sets	A few sample datasets	Windows Azure Marketplace
Availability	Public production	Some services in private beta	Some services in private beta

Conclusion

Cloud-based big data services offer considerable advantages in removing the overhead of configuring and tuning your own clusters, and in ensuring you pay only for what you use. The biggest issue is always going to be data locality, as it is slow and expensive to ship data. The most effective big data cloud solutions will be the ones where the data is also collected in the cloud. This is an incentive to investigate EC2, Azure or AppEngine as a primary application platform, and an indicator that PaaS competitors such as Cloud Foundry and Heroku will have to address big data as a priority.

It is early days yet for big data in the cloud, with only Amazon offering battle-tested solutions at this point. Cloud services themselves are at an early stage, and we will see both increasing standardization and innovation over the next two years.

However, the twin advantages of not having to worry about infrastructure and economies of scale mean it is well worth investigating cloud services for your big data needs, especially for an experimental or green-field project. Looking to the future, there's no doubt that big data analytical capability will form an essential component of utility computing solutions.

Notes

[1] In public beta.

[2] In controlled beta test.

[3] Hive and Pig compatible.

[4] Experimental status.

CHAPTER 7

Data Marketplaces

By Edd Dumbill

The sale of data is a venerable business, and has existed since the middle of the 19th century, when Paul Reuter began providing telegraphed stock exchange prices between Paris and London, and New York newspapers founded the Associated Press.

The web has facilitated a blossoming of information providers. As the ability to discover and exchange data improves, the need to rely on aggregators such as Bloomberg or Thomson Reuters is declining. This is a good thing: the business models of large aggregators do not readily scale to web startups, or casual use of data in analytics.

Instead, data is increasingly offered through online marketplaces: platforms that host data from publishers and offer it to consumers. This article provides an overview of the most mature data markets, and contrasts their different approaches and facilities.

What Do Marketplaces Do?

Most of the consumers of data from today's marketplaces are developers. By adding another dataset to your own business data, you can create insight. To take an example from web analytics: by mixing an IP address database with the logs from your website, you can understand where your customers are coming from, then if you add demographic data to the mix, you have some idea of their socio-economic bracket and spending ability.

Such insight isn't limited to analytic use only, you can use it to provide value back to a customer. For instance, by recommending restaurants local to the vicinity of a lunchtime appointment in their calendar. While many datasets

are useful, few are as potent as that of location in the way they provide context to activity.

Marketplaces are useful in three major ways. First, they provide a point of discoverability and comparison for data, along with indicators of quality and scope. Second, they handle the cleaning and formatting of the data, so it is ready for use (often 80% of the work in any data integration). Finally, marketplaces provide an economic model for broad access to data that would otherwise prove difficult to either publish or consume.

In general, one of the important barriers to the development of the data marketplace economy is the ability of enterprises to store and make use of the data. A principle of big data is that it's often easier to move your computation to the data, rather than the reverse. Because of this, we're seeing the increasing integration between cloud computing facilities and data markets: Microsoft's data market is tied to its Azure cloud, and Infochimps offers hosted compute facilities. In short-term cases, it's probably easier to export data from your business systems to a cloud platform than to try and expand internal systems to integrate external sources.

While cloud solutions offer a route forward, some marketplaces also make the effort to target end-users. Microsoft's data marketplace can be accessed directly through Excel, and DataMarket provides online visualization and exploration tools.

The four most established data marketplaces are Infochimps, Factual, Microsoft Windows Azure Data Marketplace, and DataMarket. A table comparing these providers is presented at the end of this article, and a brief discussion of each marketplace follows.

Infochimps

According to founder Flip Kromer, Infochimps was created to give data life in the same way that code hosting projects such as SourceForge or GitHub give life to code. You can improve code and share it: Kromer wanted the same for data. The driving goal behind Infochimps is to connect every public and commercially available database in the world to a common platform.

Infochimps realized that there's an important network effect of "data with the data," that the best way to build a data commons and a data marketplace is to put them together in the same place. The proximity of other data makes all the data more valuable, because of the ease with which it can be found and combined.

The biggest challenge in the two years Infochimps has been operating is that of bootstrapping: a data market needs both supply and demand. Infochimps' approach is to go for a broad horizontal range of data, rather than specialize. According to Kromer, this is because they view data's value as being in the context it provides: in giving users more insight about their own data. To join up data points into a context, common identities are required (for example, a web page view can be given a geographical location by joining up the IP address of the page request with that from the IP address in an IP intelligence database). The benefit of common identities and data integration is where hosting data together really shines, as Infochimps only needs to integrate the data once for customers to reap continued benefit: Infochimps sells datasets which are pre-cleaned and integrated mash-ups of those from their providers.

By launching a big data cloud hosting platform alongside its marketplace, Infochimps is seeking to build on the importance of data locality.

Factual

Factual was envisioned by founder and CEO Gil Elbaz as an open data platform, with tools that could be leveraged by community contributors to improve data quality. The vision is very similar to that of Infochimps, but in late 2010 Factual elected to concentrate on one area of the market: geographical and place data. Rather than pursue a broad strategy, the idea is to become a proven and trusted supplier in one vertical, then expand. With customers such as Facebook, Factual's strategy is paying off.

According to Elbaz, Factual will look to expand into verticals other than local information in 2012. It is moving one vertical at a time due to the marketing effort required in building quality community and relationships around the data.

Unlike the other main data markets, Factual does not offer reselling facilities for data publishers. Elbaz hasn't found that the cash on offer is attractive enough for many organizations to want to share their data. Instead, he believes that the best way to get data you want is to trade other data, which could provide business value far beyond the returns of publishing data in exchange for cash. Factual offer incentives to their customers to share data back, improving the quality of the data for everybody.

Windows Azure Data Marketplace

Launched in 2010, Microsoft's Windows Azure Data Marketplace sits alongside the company's Applications marketplace as part of the Azure cloud plat-

form. Microsoft's data market is positioned with a very strong integration story, both at the cloud level and with end-user tooling.

Through use of a standard data protocol, OData, Microsoft offers a well-defined web interface for data access, including queries. As a result, programs such as Excel and PowerPivot can directly access marketplace data: giving Microsoft a strong capability to integrate external data into the existing tooling of the enterprise. In addition, OData support is available for a broad array of programming languages.

Azure Data Marketplace has a strong emphasis on connecting data consumers to publishers, and most closely approximates the popular concept of an "iTunes for Data." Big name data suppliers such as Dun & Bradstreet and ESRI can be found among the publishers. The marketplace contains a good range of data across many commercial use cases, and tends to be limited to one provider per dataset — Microsoft has maintained a strong filter on the reliability and reputation of its suppliers.

DataMarket

Where the other three main data marketplaces put a strong focus on the developer and IT customers, DataMarket caters to the end-user as well. Realizing that interacting with bland tables wasn't engaging users, founder Hjalmar Gislason worked to add interactive visualization to his platform.

The result is a data marketplace that is immediately useful for researchers and analysts. The range of DataMarket's data follows this audience too, with a strong emphasis on country data and economic indicators. Much of the data is available for free, with premium data paid at the point of use.

DataMarket has recently made a significant play for data publishers, with the emphasis on publishing, not just selling data. Through a variety of plans, customers can use DataMarket's platform to publish and sell their data, and embed charts in their own pages. At the enterprise end of their packages, DataMarket offers an interactive branded data portal integrated with the publisher's own web site and user authentication system. Initial customers of this plan include Yankee Group and Lux Research.

Data Markets Compared

	Azure	Datamarket	Factual	Infochimps
Data sources	Broad range	Range, with a focus on country and industry stats	Geo-specialized, some other datasets	Range, with a focus on geo, social and web sources
Free data	Yes	Yes	-	Yes
Free trials of paid data	Yes	-	Yes, limited free use of APIs	-
Delivery	OData API	API, downloads	API, downloads for heavy users	API, downloads
Application hosting	Windows Azure	-	-	Infochimps Platform
Previewing	Service Explorer	Interactive visualization	Interactive search	-
Tool integration	Excel, PowerPivot, Tableau and other OData consumers	-	Developer tool integrations	-
Data publishing	Via database connection or web service	Upload or web/database connection.	Via upload or web service.	Upload
Data reselling	Yes, 20% commission on non-free datasets	Yes. Fees and commissions vary. Ability to create branded data market	-	Yes. 30% commission on non-free datasets.
Launched	2010	2010	2007	2009

Other Data Suppliers

While this article has focused on the more general purpose marketplaces, several other data suppliers are worthy of note.

Social data — Gnip and Datasift specialize in offering social media data streams, in particular Twitter.

Linked data — Kasabi, currently in beta, is a marketplace that is distinctive for hosting all its data as Linked Data, accessible via web standards such as SPARQL and RDF.

Wolfram Alpha — Perhaps the most prolific integrator of diverse databases, Wolfram Alpha recently added a Pro subscription level that permits the end user to download the data resulting from a computation.

Mike Loukides is Vice President of Content Strategy for O'Reilly Media, Inc. He's edited many highly regarded books on technical subjects that don't involve Windows programming. He's particularly interested in programming languages, Unix and what passes for Unix these days, and system and network administration. Mike is the author of "System Performance Tuning", and a coauthor of "Unix Power Tools." Most recently, he's been fooling around with data and data analysis, languages like R, Mathematica, and Octave, and thinking about how to make books social.

CHAPTER 8
The NoSQL Movement

By Mike Loukides

In a conversation last year, Justin Sheehy, CTO of Basho, described NoSQL as a movement, rather than a technology. This description immediately felt right; I've never been comfortable talking about NoSQL, which when taken literally, extends from the minimalist Berkeley DB (commercialized as Sleepycat, now owned by Oracle) to the big iron HBase, with detours into software as fundamentally different as Neo4J (a graph database) and FluidDB (which defies description).

But what does it mean to say that NoSQL is a movement rather than a technology? We certainly don't see picketers outside Oracle's headquarters. Justin said succinctly that NoSQL is a movement for choice in database architecture. There is no single overarching technical theme; a single technology would belie the principles of the movement.

Think of the last 15 years of software development. We've gotten very good at building large, database-backed applications. Many of them are web applications, but even more of them aren't. "Software architect" is a valid job description; it's a position to which many aspire. But what do software architects do? They specify the high level design of applications: the front end, the APIs, the middleware, the business logic--the back end? Well, maybe not.

Since the 80s, the dominant back end of business systems has been a relational database, whether Oracle, SQL Server or DB2. That's not much of an architectural choice. Those are all great products, but they're essentially similar, as are all the other relational databases. And it's remarkable that we've explored many architectural variations in the design of clients, front ends, and middleware, on a multitude of platforms and frameworks, but haven't until recently questioned the architecture of the back end. Relational databases have been a given.

Many things have changed since the advent of relational databases:

- We're dealing with much more data. Although advances in storage capacity and CPU speed have allowed the databases to keep pace, we're in a new era where size itself is an important part of the problem, and any significant database needs to be distributed.
- We require sub-second responses to queries. In the 80s, most database queries could run overnight as batch jobs. That's no longer acceptable. While some analytic functions can still run as overnight batch jobs, we've seen the Web evolve from static files to complex database-backed sites, and that requires sub-second response times for most queries.
- We want applications to be up 24/7. Setting up redundant servers for static HTML files is easy, but a database replication in a complex database-backed application is another.
- We're seeing many applications in which the database has to soak up data as fast (or even much faster) than it processes queries: in a logging application, or a distributed sensor application, writes can be much more frequent than reads. Batch-oriented ETL (extract, transform, and load) hasn't disappeared, and won't, but capturing high speed data flows is increasingly important.
- We're frequently dealing with changing data or with unstructured data. The data we collect, and how we use it, grows over time in unpredictable ways. Unstructured data isn't a particularly new feature of the data landscape, since unstructured data has always existed, but we're increasingly unwilling to force a structure on data a priori.
- We're willing to sacrifice our sacred cows. We know that consistency and isolation and other properties are very valuable, of course. But so are some other things, like latency and availability and not losing data even if our primary server goes down. The challenges of modern applications make us realize that sometimes we might need to weaken one of these constraints in order to achieve another.

These changing requirements lead us to different tradeoffs and compromises when designing software. They require us to rethink what we require of a database, and to come up with answers aside from the relational databases that have served us well over the years. So let's look at these requirements in somewhat more detail.

Size, Response, Availability

It's a given that any modern application is going to be distributed. The size of modern datasets is only one reason for distribution, and not the most impor-

tant. Modern applications (particularly web applications) have many concurrent users who demand reasonably snappy response. In their 2009 Velocity Conference talk, Performance Related Changes and their User Impact, Eric Schurman and Jake Brutlag showed results from independent research projects at Google and Microsoft. Both projects demonstrated imperceptibly small increases in response time cause users to move to another site; if response time is over a second, you're losing a very measurable percentage of your traffic.

If you're not building a web application--say you're doing business analytics, with complex, time-consumming queries--the world has changed, and users now expect business analytics to run in something like real time. Maybe not the sub-second latency required for web users, but queries that run overnight are no longer acceptable. Queries that run while you go out for coffee are marginal. It's not just a matter of convenience; the ability to run dozens or hundreds of queries per day changes the nature of the work you do. You can be more experimental: you can follow through on hunches and hints based on earlier queries. That kind of spontaneity was impossible when research went through the DBA at the data warehouse.

Whether you're building a customer-facing application or doing internal analytics, scalability is a big issue. Vertical scalability (buy a bigger, faster machine) always runs into limits. Now that the laws of physics have stalled Intel-architecture clock speeds in the 3.5GHz range, those limits are more apparent than ever. Horizontal scalability (build a distributed system with more nodes) is the only way to scale indefinitely. You're scaling horizontally even if you're only buying single boxes: it's been a long time since I've seen a server (or even a high-end desktop) that doesn't sport at least four cores. Horizontal scalability is tougher when you're scaling across racks of servers at a colocation facility, but don't be deceived: that's how scalability works in the 21st century, even on your laptop. Even in your cell phone. We need database technologies that aren't just fast on single servers: they must also scale across multiple servers.

Modern applications also need to be highly available. That goes without saying, but think about how the meaning of "availability" has changed over the years. Not much more than a decade ago, a web application would have a single HTTP server that handed out static files. These applications might be data-driven; but "data driven" meant that a batch job rebuilt the web site overnight, and user transactions were queued into a batch processing system, again for processing overnight. Keeping such a system running isn't terribly difficult. High availability doesn't impact the database: if the database is only engaged in batched rebuilds or transaction processing, the database can crash without damage. That's the world for which relational databases were designed: in the 80s, if your mainframe ran out of steam, you got a bigger one. If it crashed, you were down. But when databases became a living, breathing

part of the application, availability became an issue. There is no way to make a single system highly available; as soon as any component fails, you're toast. Highly available systems are, by nature, distributed systems.

If a distributed database is a given, the next question is how much work a distributed system will require. There are fundamentally two options: databases that have to be distributed manually, via sharding; and databases that are inherently distributed. Relational databases are split between multiple hosts by manual sharding, or determining how to partition the datasets based on some properties of the data itself: for example, first names starting with A-K on one server, L-Z on another. A lot of thought goes into designing a sharding and replication strategy that doesn't impair performance, while keeping the data relatively balanced between servers. There's a third option which is essentially a hybrid: databases that are not inherently distributed, but that are designed so they can be partitioned easily. MongoDB is an example of a database that can be sharded easily (or even automatically); HBase, Riak, and Cassandra are all inherently distributed, with options to control how replication and distribution work.

What database choices are viable when you need good interactive response? There are two separate issues: read latency and write latency. For reasonably simple queries on a database with well-designed indexes, almost any modern database can give decent read latency, even at reasonably large scale. Similarly, just about all modern databases claim to be able to keep up with writes at high-speed. Most of these databases, including HBase, Cassandra, Riak, and CouchDB, write data immediately to an append-only file, which is an extremely efficient operation. As a result, writes are often significantly faster than reads.

Whether any particular database can deliver the performance you need depends on the nature of the application, and whether you've designed the application in a way that uses the database efficiently: in particular, the structure of queries, more than the structure of the data itself. Redis is an in-memory database with extremely fast response, for both read and write operations; but there are a number of tradeoffs. By default, data isn't saved to disk, and is lost if the system crashes. You can configure Redis for durability, but at the cost of some performance. Redis is also limited in scalability; there's some replication capability, but support for clusters is still coming. But if you want raw speed, and have a dataset that can fit into memory, Redis is a great choice.

It would be nice if there were some benchmarks to cover database performance in a meaningful sense, but as the saying goes, "there are lies, damned lies, and benchmarks." In particular, no small benchmark can properly duplicate a real test-case for an application that might reasonably involve dozens (or hundreds) of servers.

Changing Data and Cheap Lunches

NoSQL databases are frequently called "schemaless," because they don't have the formal schema associated with relational databases. The lack of a formal schema, which typically has to be designed before any code is written, means that schemaless databases are a better fit for current software development practices, such as agile development. Starting from the simplest thing that could possibly work and iterating quickly in response to customer input doesn't fit well with designing an all-encompassing data schema at the start of the project. It's impossible to predict how data will be used, or what additional data you'll need as the project unfolds. For example, many applications are now annotating their data with geographic information: latitudes and longitudes, addresses. That almost certainly wasn't part of the initial data design.

How will the data we collect change in the future? Will we be collecting biometric information along with tweets and foursquare checkins? Will music sites such as Last.FM and Spotify incorporate factors like blood pressure into their music selection algorithms? If you think these scenarios are futuristic, think about Twitter. When it started out, it just collected bare-bones information with each tweet: the tweet itself, the Twitter handle, a timestamp, and a few other bits. Over its five year history, though, lots of metadata has been added: a tweet may be 140 characters at most, but a couple KB is actually sent to the server, and all of this is saved in the database. Up-front schema design is a poor fit in a world where data requirements are fluid.

In addition, modern applications frequently deal with unstructured data: blog posts, web pages, voice transcripts, and other data objects that are essentially text. O'Reilly maintains a substantial database of job listings for some internal research projects. The job descriptions are chunks of text in natural languages. They're not unstructured because they don't fit into a schema. You can easily create a JOBDESCRIPTION column in a table, and stuff text strings into it. It's that knowing the data type and where it fits in the overall structure doesn't help. What are the questions you're likely to ask? Do you want to know about skills, certifications, the employer's address, the employer's industry? Those are all valid columns for a table, but you don't know what you care about in advance; you won't find equivalent information in each job description; and the only way to get from the text to the data is through various forms of pattern matching and classification. Doing the classification up front, so you could break a job listing down into skills, certifications, etc., is a huge effort that would largely be wasted. The guys who work with this data recently had fits disambiguating "Apple Computer" from "apple orchard"; would you even know this was a problem outside of a concrete research project based on a concrete question? If you're just pre-populating an INDUSTRY column from

raw data, would you notice that lots of computer industry jobs were leaking into fruit farming? A JOBDESCRIPTION column doesn't hurt, but doesn't help much either; and going further, by trying to design a schema around the data that you'll find in the unstructured text, definitely hurts. The kinds of questions you're likely to ask have everything to do with the data itself, and little to do with that data's relations to other data.

However, it's really a mistake to say that NoSQL databases have no schema. In a document database, such as CouchDB or MongoDB, documents are key-value pairs. While you can add documents with differing sets of keys (missing keys or extra keys), or even add keys to documents over time, applications still must know that certain keys are present to query the database; indexes have to be set up to make searches efficient. The same thing applies to column-oriented databases, such as HBase and Cassandra. While any row may have as many columns as needed, some up-front thought has to go into what columns are needed to organize the data. In most applications, a NoSQL database will require less up-front planning, and offer more flexiblity as the application evolves. As we'll see, data design revolves more around the queries you want to ask than the domain objects that the data represents. It's not a free lunch; possibly a cheap lunch, but not free.

What kinds of storage models do the more common NoSQL databases support? Redis is a relatively simple key-value store, but with a twist: values can be data structures (lists and sets), not just strings. It supplies operations for working directly with sets and lists (for example, union and intersection).

CouchDB and MongoDB both store documents in JSON format, where JSON is a format originally designed for representating JavaScript objects, but now available in many languages. So on one hand, you can think of CouchDB and MongoDB as object databases; but you could also think of a JSON document as a list of key-value pairs. Any document can contain any set of keys, and any key can be associated with an arbitrarily complex value that is itself a JSON document. CouchDB queries are views, which are themselves documents in the database that specify searches. Views can be very complex, and can use a built-in mapreduce facility to process and summarize results. Similarly, MongoDB queries are JSON documents, specifying fields and values to match, and query results can be processed by a builtin mapreduce. To use either database effectively, you start by designing your views: what do you want to query, and how. Once you do that, it will become clear what keys are needed in your documents.

Riak can also be viewed as a document database, though with more flexibility about document types: it natively handles JSON, XML, and plain text, and a plug-in architecture allows you to add support for other document types. Searches "know about" the structure of JSON and XML documents. Like

CouchDB, Riak incorporates mapreduce to perform complex queries efficiently.

Cassandra and HBase are usually called column-oriented databases, though a better term is a "sparse row store." In these databases, the equivalent to a relational "table" is a set of rows, identified by a key. Each row consists of an unlimited number of columns; columns are essentially keys that let you look up values in the row. Columns can be added at any time, and columns that are unused in a given row don't occupy any storage. NULLs don't exist. And since columns are stored contiguously, and tend to have similar data, compression can be very efficient, and searches along a column are likewise efficient. HBase describes itself as a database that can store billions of rows with millions of columns.

How do you design a schema for a database like this? As with the document databases, your starting point should be the queries you'll want to make. There are some radically different possibilities. Consider storing logs from a web server. You may want to look up the IP addresses that accessed each URL you serve. The URLs can be the primary key; each IP address can be a column. This approach will quickly generate thousands of unique columns, but that's not a problem--and a single query, with no joins, gets you all the IP addresses that accessed a single URL. If some URLs are visited by many addresses, and some are only visited by a few, that's no problem: remember that NULLs don't exist. This design isn't even conceivable in a relational database: you can't have a table that doesn't have a fixed number of columns.

Now, let's make it more complex: you're writing an ecommerce application, and you'd like to access all the purchases that a given customer has made. The solution is similar: the column family is organized by customer ID (primary key), you have columns for first name, last name, address, and all the normal customer information, plus as many rows as are needed for each purchase. In a relational database, this would probably involve several tables and joins; in the NoSQL databases, it's a single lookup. Schema design doesn't go away, but it changes: you think about the queries you'd like to execute, and how you can perform those efficiently.

This isn't to say that there's no value to normalization, just that data design starts from a different place. With a relational database, you start with the domain objects, and represent them in a way that guarantees that virtually any query can be expressed. But when you need to optimize performance, you look at the queries you actually perform, then merge tables to create longer rows, and do away with joins wherever possible. With the schemaless databases, whether we're talking about data structure servers, document databases, or column stores, you go in the other direction: you start with the query, and use that to define your data objects.

The Sacred Cows

The ACID properties (atomicity, consistency, isolation, durability) have been drilled into our heads. But even these come into play as we start thinking seriously about database architecture. When a database is distributed, for instance, it becomes much more difficult to achieve the same kind of consistency or isolation that you can on a single machine. And the problem isn't just that it's "difficult" but rather that achieving them ends up in direct conflict with some of the reasons to go distributed. It's not that properties like these aren't very important--they certainly are--but today's software architects are discovering that they require the freedom to choose when it might be worth a compromise.

What about transactions, two-phase commit, and other mechanisms inherited from big iron legacy databases? If you've read almost any discussion of concurrent or distributed systems, you've heard that banking systems care a lot about consistency: what if you and your spouse withdraw money from the same account at the same time? Could you overdraw the account? That's what ACID is supposed to prevent. But a few months ago, I was talking to someone who builds banking software, and he said "If you really waited for each transaction to be properly committed on a world-wide network of ATMs, transactions would take so long to complete that customers would walk away in frustration. What happens if you and your spouse withdraw money at the same time and overdraw the account? You both get the money; we fix it up later."

This isn't to say that bankers have discarded transactions, two-phase commit and other database techniques; they're just smarter about it. In particular, they're distinguishing between local consistency and absolutely global consistency. Gregor Hohpe's classic article Starbucks Does Not Use Two-Phase Commit makes a similar point: in an asychronous world, we have many strategies for dealing with transactional errors, including write-offs. None of these strategies are anything like two-phase commit; they don't force the world into inflexible, serialized patterns.

The CAP theorem is more than a sacred cow; it's a law of the Database universe that can be expressed as "Consistency, Availability, Partition Tolerance: choose two." But let's rethink relational databases in light of this theorem. Databases have stressed consistency. The CAP theorem is really about distributed systems, and as we've seen, relational databases were developed when distributed systems were rare and exotic at best. If you needed more power, you bought a bigger mainframe. Availability isn't an issue on a single server: if it's up, it's up, if it's down, it's down. And partition tolerance is meaningless when there's nothing to partition. As we saw at the beginning of this article, distributed systems are a given for modern applications; you won't be able to

scale to the size and performance you need on a single box. So the CAP theorem is historically irrelevant to relational databases: they're good at providing consistency, and they have been adapted to provide high availability with some success, but they are hard to partition without extreme effort or extreme cost.

Since partition tolerance is a fundamental requirement for distributed applications, it becomes a question of what to sacrifice: consistency or availability. There have been two approaches: Riak and Cassandra stress availability, while HBase has stressed consistency. With Cassandra and Riak, the tradeoff between consistency and availability is tuneable. CouchDB and MongoDB are essentially single-headed databases, and from that standpoint, availability is a function of how long you can keep the hardware running. However, both have add-ons that can be used to build clusters. In a cluster, CouchDB and MongoDB are eventually consistent (like Riak and Cassandra); availability depends on what you do with the tools they provide. You need to set up sharding and replication, and use what's essentially a proxy server to present a single interface to cluster's clients. BigCouch is an interesting effort to integrate clustering into CouchDB, making it more like Riak. Now that Cloudant has announced that it is merging BigCouch and CouchDB, we can expect to see clustering become part of the CouchDB core.

We've seen that absolute consistency isn't a hard requirement for banks, nor is it the way we behave in our real-world interactions. Should we expect it of our software? Or do we care more about availability?

It depends; the consistency requirements of many social applications are very soft. You don't need to get the correct number of Twitter or Facebook followers every time you log in. If you search, you probably don't care if the results don't contain the comments that were posted a few seconds ago. And if you're willing to accept less-than-perfect consistency, you can make huge improvements in performance. In the world of big-data-backed web applications, with databases spread across hundreds (or potentially thousands) of nodes, the performance penalty of locking down a database while you add or modify a row is huge; if your application has frequent writes, you're effectively serializing all the writes and losing the advantage of the distributed database. In practice, in an "eventually consistent" database, changes typically propagate to the nodes in tenths of a second; we're not talking minutes or hours before the database arrives in a consistent state.

Given that we have all been battered with talk about "five nines" reliability, and given that it is a big problem for any significant site to be down, it seems clear that we should prioritize availability over consistency, right? The architectural decision isn't so easy, though. There are many applications in which inconsistency must eventually be dealt with. If consistency isn't guaranteed by the database, it becomes a problem that the application has to manage. When

you choose availability over consistency, you're potentially making your application more complex. With proper replication and failover strategies, a database designed for consistency (such as HBase) can probably deliver the availability you require; but this is another design tradeoff. Regardless of the database you're using, more stringent reliability requirements will drive you towards exotic engineering. Only you can decide the right balance for your application; the point isn't that any given decision is right or wrong, but that you can (and have to) choose, and that's a good thing.

Other features

I've completed a survey of the major tradeoffs you need to think about in selecting a database for a modern big data application. But the major tradeoffs aren't the only story. There are many database projects with interesting features. Here are a some of the ideas and projects I find most interesting:

- Scripting: Relational databases all come with some variation of the SQL language, which can be seen as a scripting language for data. In the non-relational world, a number of scripting languages are available. CouchDB and Riak support JavaScript, as does MongoDB. The Hadoop project has spawned a several data scripting languages that are usuable with HBase, including Pig and Hive. The Redis project is experimenting with integrating the Lua scripting language.
- RESTful interfaces: CouchDB and Riak are unique in offering RESTful interfaces: interfaces based on HTTP and the architectural style elaborated in Roy Fielding's doctoral dissertation and Restful Web Services. CouchDB goes so far as to serve as a web application framework. Riak also offers a more traditional protocol buffer interface, which is a better fit if you expect a high volume of small requests.
- Graphs: Neo4J is a special purpose database designed for maintaining large graphs: data where the data items are nodes, with edges representing the connections between the nodes. Because graphs are extremely flexible data structures, a graph database can emulate any other kind of database.
- SQL: I've been discussing the NoSQL movement, but SQL is a familiar language, and is always just around the corner. A couple of startups are working on adding SQL to Hadoop-based datastores: DrawnToScale (which focuses on low-latency, high-volume web applications) and Hadapt (which focuses on analytics and bringing data warehousing into the 20-teens). In a few years, will we be looking at hybrid databases that take advantage of both relational and non-relational models? Quite possibly.

- Scientific data: Yet another direction comes from SciDB, a database project aimed at the largest scientific applications (particularly the Large Synoptic Survey Telescope). The storage model is based on multi-dimensional arrays. It is designed to scale to hundreds of petabytes of storage, collecting tens of terabytes per night. It's still in the relatively early stages.
- Hybrid architectures: NoSQL is really about architectural choice. And perhaps the biggest expression of architectural choice is a hybrid architecture: rather than using a single database technology, mixing and matching technologies to play to their strengths. I've seen a number of applications that use traditional relational databases for the portion of the data for which the relational model works well, and a non-relational database for the rest. For example, customer data could go into a relational database, linked to a non-relational database for unstructured data such as product reviews and recommendations. It's all about flexibility. A hybrid architecture may be the best way to integrate "social" features into more traditional ecommerce sites.

These are only a few of the interesting ideas and projects that are floating around out there. Roughly a year ago, I counted a couple dozen non-relational database projects; I'm sure there are several times that number today. Don't hesitate to add notes about your own projects in the comments.

In the End

In a conversation with Eben Hewitt, author of Cassandra: The Definitive Guide, Eben summarized what you need to think about when architecting the back end of a data-driven system. They're the same issues software architects have been dealing with for years: you need to think about the whole ecosystems in which the application works; you need to consider your goals (do you require high availability? fault tolerance?); you need to consider support options; you need to isolate what will changes over the life of the application, and separate that from what remains the same. The big difference is that now there are options; you don't have to choose the relational model. There are other options for building large databases that scale horizontally, are highly available, and can deliver great performance to users. And these options, the databases that make up the NoSQL movement, can often achieve these goals with greater flexibility and lower cost.

It used to be said that nobody got fired for buying IBM; then nobody got fired for buying Microsoft; now, I suppose, nobody gets fired for buying Oracle. But just as the landscape changed for IBM and Microsoft, it's shifting again, and even Oracle has a NoSQL solution. Rather than relational databases being the default, we're moving into a world where developers are considering their

architectural options, and deciding which products fit their application: how the databases fit into their programming model, whether they can scale in ways that make sense for the application, whether they have strong or relatively weak consistency requirements.

For years, the relational default has kept developers from understanding their real back-end requirements. The NoSQL movement has given us the opportunity to explore what we really require from our databases, and to find out what we already knew: there is no one-size-fits-all solution.

CHAPTER 9

Why Visualization Matters

By Julie Steele

A Picture Is Worth 1000 Rows

Let's say you need to understand thousands or even millions of rows of data, and you have a short time to do it in. The data may come from your team, in which case perhaps you're already familiar with what it's measuring and what the results are likely to be. Or it may come from another team, or maybe several teams at once, and be completely unfamiliar. Either way, the reason you're looking at it is that you have a decision to make, and you want to be informed by the data before making it. Something probably hangs in the balance: a customer, a product, or a profit.

How are you going to make sense of all that information efficiently so you can make a good decision? Data visualization is an important answer to that question.

However, not all visualizations are actually that helpful. You may be all too familiar with lifeless bar graphs, or line graphs made with software defaults and couched in a slideshow presentation or lengthy document. They can be at best confusing, and at worst misleading. But the good ones are an absolute revelation.

The best data visualizations are ones that *expose something new* about the underlying patterns and relationships contained within the data. Understanding those relationships—and so being able to observe them—is key to good decision-making. The Periodic Table is a classic testament to the potential of visualization to reveal hidden relationships in even small data sets. One look at the table, and chemists and middle school students alike grasp the way atoms arrange themselves in groups: alkali metals, noble gasses, halogens.

If visualization done right can reveal so much in even a small data set like this, imagine what it can reveal within terabytes or petabytes of information.

Types of Visualization

It's important to point out that not all data visualization is created equal. Just as we have paints and pencils and chalk and film to help us capture the world in different ways, with different emphases and for different purposes, there are multiple ways in which to depict the same data set.

Or, to put it another way, think of visualization as a new set of languages you can use to communicate. Just as French and Russian and Japanese are all ways of encoding ideas so that those ideas can be transported from one person's mind to another, and decoded again—and just as certain languages are more conducive to certain ideas—so the various kinds of data visualization are a kind of *bidirectional encoding* that lets ideas and information be transported from the database into your brain.

Explaining and exploring

An important distinction lies between visualization for *exploring* and visualization for *explaining*. A third category, *visual art*, comprises images that encode data but cannot easily be decoded back to the original meaning by a viewer. This kind of visualization can be beautiful, but is not helpful in making decisions.

Visualization for exploring can be imprecise. It's useful when you're not exactly sure what the data has to tell you, and you're trying to get a sense of the relationships and patterns contained within it for the first time. It may take a while to figure out how to approach or clean the data, and which dimensions to include. Therefore, visualization for exploring is best done in such a way that it can be iterated quickly and experimented upon, so that you can find the signal within the noise. Software and automation are your friends here.

Visualization for explaining is best when it is cleanest. Here, the ability to pare down the information to its simplest form—to strip away the noise entirely—will increase the efficiency with which a decision-maker can understand it. This is the approach to take once you understand what the data is telling you, and you want to communicate that to someone else. This is the kind of visualization you *should* be finding in those presentations and sales reports.

Visualization for explaining also includes infographics and other categories of hand-drawn or custom-made images. Automated tools can be used, but one size does *not* fit all.

Your Customers Make Decisions, Too

While data visualization is a powerful tool for helping you and others within your organization make better decisions, it's important to remember that, in the meantime, your customers are trying to decide between you and your competitors. Many kinds of data visualization, from complex interactive or animated graphs to brightly-colored infographics, can help explain your customers explore and your customer service folks explain.

That's why kinds of companies and organizations, from GE to Trulia to NASA, are beginning to invest significant resources in providing interactive visualizations to their customers and the public. This allows viewers to better understand the company's business, and interact in a self-directed manner with the company's expertise.

As Big Data becomes bigger, and more companies deal with complex data sets with dozens of variables, data visualization will become even more important. So far, the tide of popularity has risen more quickly than the tide of visual literacy, and mediocre efforts abound, in presentations and on the web.

But as visual literacy rises, thanks in no small part to impressive efforts in major media such as The New York Times and The Guardian, data visualization will increasingly become a language your customers and collaborators expect you to speak—and speak well.

Do Yourself a Favor and Hire a Designer

It's well worth investing in a talented in-house designer, or a team of designers. Visualization for explaining works best when someone who understands not only the data itself, but also the principles of design and visual communication, tailors the graph or chart to the message.

To go back to the language analogy: Google Translate is a powerful and useful tool for giving you the general idea of what a foreign text says. But it's not perfect, and it often lacks nuance. For getting the overall gist of things, it's great. But I wouldn't use it to send a letter to a foreign ambassador. For something so sensitive, and where precision counts, it's worth hiring an experienced human translator.

Since data visualization is like a foreign language, in the same way, hire an experienced designer for important jobs where precision matters. If you're making the kinds of decisions in which your customer, product, or profit hangs in the balance, you can't afford to base those decisions on incomplete or misleading representations of the knowledge your company holds.

Your designer is your translator, and one of the most important links you and your customers have to your data.

Julie Steele is an editor at O'Reilly Media interested in connecting people and ideas. She finds beauty in discovering new ways to understand complex systems, and so enjoys topics related to gathering, storing, analyzing, and visualizing data. She holds a Master's degree in Political Science (International Relations) from Rutgers University.

CHAPTER 10

The Future of Big Data

By Edd Dumbill

2011 was the "coming out" year for data science and big data. As the field matures in 2012, what can we expect over the course of the year?

More Powerful and Expressive Tools for Analysis

This year has seen consolidation and engineering around improving the basic storage and data processing engines of NoSQL and Hadoop. That will doubtless continue, as we see the unruly menagerie of the Hadoop universe increasingly packaged into distributions, appliances and on-demand cloud services. Hopefully it won't be long before that's dull, yet necessary, infrastructure.

Looking up the stack, there's already an early cohort of tools directed at programmers and data scientists (Karmasphere, Datameer), as well as Hadoop connectors for established analytical tools such as Tableau and R. But there's a way to go in making big data more powerful: that is, to decrease the cost of creating experiments.

Here are two ways in which big data can be made more powerful.

1. Better programming language support. As we consider data, rather than business logic, as the primary entity in a program, we must create or rediscover idiom that lets us focus on the data, rather than abstractions

leaking up from the underlying Hadoop machinery. In other words: write shorter programs that make it clear what we're doing with the data. These abstractions will in turn lend themselves to the creation of better tools for non-programmers.

2. We require better support for interactivity. If Hadoop has any weakness, it's in the batch-oriented nature of computation it fosters. The agile nature of data science will favor any tool that permits more interactivity.

Streaming Data Processing

Hadoop's batch-oriented processing is sufficient for many use cases, especially where the frequency of data reporting doesn't need to be up-to-the-minute. However, batch processing isn't always adequate, particularly when serving online needs such as mobile and web clients, or markets with real-time changing conditions such as finance and advertising.

Over the next few years we'll see the adoption of scalable frameworks and platforms for handling streaming, or near real-time, analysis and processing. In the same way that Hadoop has been borne out of large-scale web applications, these platforms will be driven by the needs of large-scale location-aware mobile, social and sensor use.

For some applications, there just isn't enough storage in the world to store every piece of data your business might receive: at some point you need to make a decision to throw things away. Having streaming computation abilities enables you to analyze data or make decisions about discarding it without having to go through the store-compute loop of map/reduce.

Emerging contenders in the real-time framework category include Storm, from Twitter, and S4, from Yahoo.

Rise of Data Marketplaces

Your own data can become that much more potent when mixed with other datasets. For instance, add in weather conditions to your customer data, and discover if there are weather related patterns to your customers' purchasing patterns. Acquiring these datasets can be a pain, especially if you want to do it outside of the IT department, and with some exactness. The value of data marketplaces is in providing a directory to this data, as well as streamlined, standardized methods of delivering it. Microsoft's direction of integrating its Azure marketplace right into analytical tools foreshadows the coming convenience of access to data.

Development of Data Science Workflows and Tools

As data science teams become a recognized part of companies, we'll see a more regularized expectation of their roles and processes. One of the driving attributes of a successful data science team is its level of integration into a company's business operations, as opposed to being a sidecar analysis team.

Software developers already have a wealth of infrastructure that is both logistical and social, including wikis and source control, along with tools that expose their process and requirements to business owners. Integrated data science teams will need their own versions of these tools to collaborate effectively. One example of this is EMC Greenplum's Chorus, which provides a social software platform for data science. In turn, use of these tools will support the emergence of data science process within organizations.

Data science teams will start to evolve repeatable processes, hopefully agile ones. They could do worse than to look at the ground-breaking work newspaper data teams are doing at news organizations such as The Guardian and New York Times: given short timescales these teams take data from raw form to a finished product, working hand-in-hand with the journalist.

Increased Understanding of and Demand for Visualization

Visualization fulfills two purposes in a data workflow: explanation and exploration. While business people might think of a visualization as the end result, data scientists also use visualization as a way of looking for questions to ask and discovering new features of a dataset.

If becoming a data-driven organization is about fostering a better feel for data among all employees, visualization plays a vital role in delivering data manipulation abilities to those without direct programming or statistical skills.

Throughout a year dominated by business' constant demand for data scientists, I've repeatedly heard from data scientists about what they want most: people who know how to create visualizations.