DATA JUST RIGHT

Introduction to Large-Scale Data & Analytics

MICHAEL MANOOCHERI

FREE SAMPLE CHAPTER
SHARE WITH OTHERS
Data Just Right
The Addison-Wesley Data and Analytics Series provides readers with practical knowledge for solving problems and answering questions with data. Titles in this series primarily focus on three areas:

1. **Infrastructure**: how to store, move, and manage data
2. **Algorithms**: how to mine intelligence or make predictions based on data
3. **Visualizations**: how to represent data and insights in a meaningful and compelling way

The series aims to tie all three of these areas together to help the reader build end-to-end systems for fighting spam; making recommendations; building personalization; detecting trends, patterns, or problems; and gaining insight from the data exhaust of systems and user interactions.

Visit informit.com/awdataseries for a complete list of available publications.

---

Make sure to connect with us!
informit.com/socialconnect
This book is dedicated to my parents,
Andrew and Cecelia Manoochehri,
who put everything they had into making sure
that I received an amazing education.
# Contents

Foreword xv  
Preface xvii  
Acknowledgments xxv  
About the Author xxvii  

## I Directives in the Big Data Era  1  

1 Four Rules for Data Success  3  
   When Data Became a BIG Deal  3  
   Data and the Single Server  4  
   The Big Data Trade-Off  6  
      Build Solutions That Scale (Toward Infinity)  6  
      Build Systems That Can Share Data (On the Internet)  7  
      Build Solutions, Not Infrastructure  8  
      Focus on Unlocking Value from Your Data  8  
Anatomy of a Big Data Pipeline  9  
The Ultimate Database  10  
Summary  10  

## II Collecting and Sharing a Lot of Data  11  

2 Hosting and Sharing Terabytes of Raw Data  13  
   Suffering from Files  14  
      The Challenges of Sharing Lots of Files  14  
   Storage: Infrastructure as a Service  15  
      The Network Is Slow  16  
   Choosing the Right Data Format  16  
      XML: Data, Describe Thyself  18  
      JSON: The Programmer’s Choice  18  
   Character Encoding  19  
      File Transformations  21  
Data in Motion: Data Serialization Formats  21  
      Apache Thrift and Protocol Buffers  22  
Summary  23  

3 Building a NoSQL-Based Web App to Collect Crowd-Sourced Data 25
   Relational Databases: Command and Control 25
      The Relational Database ACID Test 28
   Relational Databases versus the Internet 28
      CAP Theorem and BASE 30
   Nonrelational Database Models 31
      Key–Value Database 32
      Document Store 33
   Leaning toward Write Performance: Redis 35
   Sharding across Many Redis Instances 38
      Automatic Partitioning with Twemproxy 39
   Alternatives to Using Redis 40
   NewSQL: The Return of Codd 41
   Summary 42

4 Strategies for Dealing with Data Silos 43
   A Warehouse Full of Jargon 43
      The Problem in Practice 45
      Planning for Data Compliance and Security 46
   Enter the Data Warehouse 46
      Data Warehousing’s Magic Words: Extract, Transform, and Load 48
   Hadoop: The Elephant in the Warehouse 48
   Data Silos Can Be Good 49
      Concentrate on the Data Challenge, Not the Technology 50
      Empower Employees to Ask Their Own Questions 50
      Invest in Technology That Bridges Data Silos 51
   Convergence: The End of the Data Silo 51
      Will Luhn’s Business Intelligence System Become Reality? 52
   Summary 53
III Asking Questions about Your Data  55

5 Using Hadoop, Hive, and Shark to Ask Questions about Large Datasets  57
   What Is a Data Warehouse?  57
   Apache Hive: Interactive Querying for Hadoop  60
      Use Cases for Hive  60
      Hive in Practice  61
      Using Additional Data Sources with Hive  65
   Shark: Queries at the Speed of RAM  65
   Data Warehousing in the Cloud  66
   Summary  67

6 Building a Data Dashboard with Google BigQuery  69
   Analytical Databases  69
   Dremel: Spreading the Wealth  71
      How Dremel and MapReduce Differ  72
   BigQuery: Data Analytics as a Service  73
      BigQuery's Query Language  74
   Building a Custom Big Data Dashboard  75
      Authorizing Access to the BigQuery API  76
      Running a Query and Retrieving the Result  78
      Caching Query Results  79
      Adding Visualization  81
   The Future of Analytical Query Engines  82
   Summary  83

7 Visualization Strategies for Exploring Large Datasets  85
   Cautionary Tales: Translating Data into Narrative  86
   Human Scale versus Machine Scale  89
      Interactivity  89
   Building Applications for Data Interactivity  90
      Interactive Visualizations with R and ggplot2  90
      matplotlib: 2-D Charts with Python  92
      D3.js: Interactive Visualizations for the Web  92
   Summary  96
### IV Building Data Pipelines  97

#### 8 Putting It Together: MapReduce Data Pipelines  99

- What Is a Data Pipeline?  99
  - The Right Tool for the Job  100
- Data Pipelines with Hadoop Streaming  101
  - MapReduce and Data Transformation  101
  - The Simplest Pipeline: stdin to stdout  102
- A One-Step MapReduce Transformation  105
  - Extracting Relevant Information from Raw NVSS Data: Map Phase  106
  - Counting Births per Month: The Reducer Phase  107
  - Testing the MapReduce Pipeline Locally  108
  - Running Our MapReduce Job on a Hadoop Cluster  109
- Managing Complexity: Python MapReduce Frameworks for Hadoop  110
  - Rewriting Our Hadoop Streaming Example Using mrjob  110
  - Building a Multistep Pipeline  112
  - Running mrjob Scripts on Elastic MapReduce  113
- Alternative Python-Based MapReduce Frameworks  114
- Summary  114

#### 9 Building Data Transformation Workflows with Pig and Cascading  117

- Large-Scale Data Workflows in Practice  118
- It’s Complicated: Multistep MapReduce Transformations  118
  - Apache Pig: “Ixney on the Omplexicityay”  119
  - Running Pig Using the Interactive Grunt Shell  120
  - Filtering and Optimizing Data Workflows  121
  - Running a Pig Script in Batch Mode  122
- Cascading: Building Robust Data-Workflow Applications  122
  - Thinking in Terms of Sources and Sinks  123
Building a Cascading Application 124
Creating a Cascade: A Simple JOIN Example 125
Deploying a Cascading Application on a Hadoop Cluster 127
When to Choose Pig versus Cascading 128
Summary 128

V Machine Learning for Large Datasets 129

10 Building a Data Classification System with Mahout 131
Can Machines Predict the Future? 132
Challenges of Machine Learning 132
   Bayesian Classification 133
   Clustering 134
   Recommendation Engines 135
Apache Mahout: Scalable Machine Learning 136
   Using Mahout to Classify Text 137
MLBase: Distributed Machine Learning Framework 139
Summary 140

VI Statistical Analysis for Massive Datasets 143

11 Using R with Large Datasets 145
Why Statistics Are Sexy 146
   Limitations of R for Large Datasets 147
   R Data Frames and Matrices 148
Strategies for Dealing with Large Datasets 149
   Large Matrix Manipulation: bigmemory and biganalytics 150
   ff: Working with Data Frames Larger than Memory 151
   biglm: Linear Regression for Large Datasets 152
   RHadoop: Accessing Apache Hadoop from R 154
Summary 155
12 Building Analytics Workflows Using Python and Pandas 157
   The Snakes Are Loose in the Data Zoo 157
   Choosing a Language for Statistical Computation 158
   Extending Existing Code 159
   Tools and Testing 160
Python Libraries for Data Processing 160
   NumPy 160
   SciPy: Scientific Computing for Python 162
   The Pandas Data Analysis Library 163
Building More Complex Workflows 167
   Working with Bad or Missing Records 169
iPython: Completing the Scientific Computing Tool Chain 170
   Parallelizing iPython Using a Cluster 171
Summary 174

VII Looking Ahead 177

13 When to Build, When to Buy, When to Outsource 179
   Overlapping Solutions 179
   Understanding Your Data Problem 181
   A Playbook for the Build versus Buy Problem 182
      What Have You Already Invested In? 183
      Starting Small 183
      Planning for Scale 184
My Own Private Data Center 184
   Understand the Costs of Open-Source 186
   Everything as a Service 187
Summary 187

14 The Future: Trends in Data Technology 189
   Hadoop: The Disruptor and the Disrupted 190
   Everything in the Cloud 191
   The Rise and Fall of the Data Scientist 193
This page intentionally left blank
The array of tools for collecting, storing, and gaining insight from data is huge and getting bigger every day. For people entering the field, that means digging through hundreds of Web sites and dozens of books to get the basics of working with data at scale. That’s why this book is a great addition to the Addison-Wesley Data & Analytics series; it provides a broad overview of tools, techniques, and helpful tips for building large data analysis systems.

Michael is the perfect author to provide this introduction to Big Data analytics. He worked on the Cloud Platform Developer Relations team at Google, helping developers with BigQuery, Google’s hosted platform for analyzing terabytes of data quickly. He brings his breadth of experience to this book, providing practical guidance for anyone looking to start working with Big Data or anyone looking for additional tips, tricks, and tools.

The introductory chapters start with guidelines for success with Big Data systems and introductions to NoSQL, distributed computing, and the CAP theorem. An introduction to analytics at scale using Hadoop and Hive is followed by coverage of real-time analytics with BigQuery. More advanced topics include MapReduce pipelines, Pig and Cascading, and machine learning with Mahout. Finally, you’ll see examples of how to blend Python and R into a working Big Data tool chain. Throughout all of this material are examples that help you work with and learn the tools. All of this combines to create a perfect book to read for picking up a broad understanding of Big Data analytics.

—Paul Dix, Series Editor
Did you notice? We’ve recently crossed a threshold beyond which mobile technology and social media are generating datasets larger than humans can comprehend. Large-scale data analysis has suddenly become magic.

The growing fields of distributed and cloud computing are rapidly evolving to analyze and process this data. An incredible rate of technological change has turned commonly accepted ideas about how to approach data challenges upside down, forcing companies interested in keeping pace to evaluate a daunting collection of sometimes contradictory technologies.

Relational databases, long the drivers of business-intelligence applications, are now being joined by radical NoSQL open-source upstarts, and features from both are appearing in new, hybrid database solutions. The advantages of Web-based computing are driving the progress of massive-scale data storage from bespoke data centers toward scalable infrastructure as a service. Of course, projects based on the open-source Hadoop ecosystem are providing regular developers access to data technology that has previously been only available to cloud-computing giants such as Amazon and Google.

The aggregate result of this technological innovation is often referred to as Big Data. Much has been made about the meaning of this term. Is Big Data a new trend, or is it an application of ideas that have been around a long time? Does Big Data literally mean lots of data, or does it refer to the process of approaching the value of data in a new way? George Dyson, the historian of science, summed up the phenomena well when he said that Big Data exists “when the cost of throwing away data is more than the machine cost.” In other words, we have Big Data when the value of the data itself exceeds that of the computing power needed to collect and process it.

Although the amazing success of some companies and open-source projects associated with the Big Data movement is very real, many have found it challenging to navigate the bewildering amount of new data solutions and service providers. More often than not, I’ve observed that the processes of building solutions to address data challenges can be generalized into the same set of common use cases that appear over and over.

Finding efficient solutions to data challenges means dealing with trade-offs. Some technologies that are optimized for a specific data use case are not the best choice for others. Some database software is built to optimize speed of analysis over flexibility, whereas the philosophy of others favors consistency over performance. This book will help you understand when to use one technology over another through practical use cases and real success stories.
Who This Book Is For

There are few problems that cannot be solved with unlimited money and resources. Organizations with massive resources, for better or for worse, can build their own bespoke systems to collect or analyze any amount of data. This book is not written for those who have unlimited time, an army of dedicated engineers, and an infinite budget.

This book is for everyone else—those who are looking for solutions to data challenges and who are limited by resource constraints. One of the themes of the Big Data trend is that anyone can access tools that only a few years ago were available exclusively to a handful of large corporations. The reality, however, is that many of these tools are innovative, rapidly evolving, and don’t always fit together seamlessly. The goal of this book is to demonstrate how to build systems that put all the parts together in effective ways. We will look at strategies to solve data problems in ways that are affordable, accessible, and by all means practical.

Open-source software has driven the accessibility of technology in countless ways, and this has also been true in the field of Big Data. However, the technologies and solutions presented in this book are not always the open-source choice. Sometimes, accessibility comes from the ability of computation to be accessed as a service.

Nonetheless, many cloud-based services are built upon open-source tools, and in fact, many could not exist without them. Due to the great economies of scale made possible by the increasing availability of utility-computing platforms, users can pay for supercomputing power on demand, much in the same way that people pay for centralized water and power.

We’ll explore the available strategies for making the best choices to keep costs low while retaining scalability.

Why Now?

It is still amazing to me that building a piece of software that can reach everyone on the planet is not technically impossible but is instead limited mostly by economic inequity and language barriers. Web applications such as Facebook, Google Search, Yahoo! Mail, and China’s Qzone can potentially reach hundreds of millions, if not billions, of active users. The scale of the Web (and the tools that come with it) is just one aspect of why the Big Data field is growing so dramatically. Let’s look at some of the other trends that are contributing to interest in this field.

The Maturity of Open-Source Big Data

In 2004, Google released a famous paper detailing a distributed computing framework called MapReduce. The MapReduce framework was a key piece of technology that Google used to break humongous data processing problems into smaller chunks. Not too long after, another Google research paper was released that described BigTable, Google’s internal, distributed database technology.
Since then, a number of open-source technologies have appeared that implement or were inspired by the technologies described in these original Google papers. At the same time, in response to the inherent limits and challenges of using relational-database models with distributed computing systems, new database paradigms had become more and more acceptable. Some of these eschewed the core features of relational databases completely, jettisoning components like standardized schemas, guaranteed consistency, and even SQL itself.

The Rise of Web Applications

Data is being generated faster and faster as more and more people take to the Web. With the growth in Web users comes a growth in Web applications.

Web-based software is often built using application programming interfaces, or APIs, that connect disparate services across a network. For example, many applications incorporate the ability to allow users to identify themselves using information from their Twitter accounts or to display geographic information visually via Google Maps. Each API might provide a specific type of log information that is useful for data-driven decision making.

Another aspect contributing to the current data flood is the ever-increasing amount of user-created content and social-networking usage. The Internet provides a frictionless capability for many users to publish content at almost no cost. Although there is a considerable amount of noise to work through, understanding how to collect and analyze the avalanche of social-networking data available can be useful from a marketing and advertising perspective.

It’s possible to help drive business decisions using the aggregate information collected from these various Web services. For example, imagine merging sales insights with geographic data; does it look like 30% of your unique users who buy a particular product are coming from France and sharing their purchase information on Facebook? Perhaps data like this will help make the business case to dedicate resources to targeting French customers on social-networking sites.

Mobile Devices

Another reason that scalable data technology is hotter than ever is the amazing explosion of mobile-communication devices around the world. Although this trend primarily relates to the individual use of feature phones and smartphones, it’s probably more accurate to think of this trend as centered on a user’s identity and device independence. If you both use a regular computer and have a smartphone, it’s likely that you have the ability to access the same personal data from either device. This data is likely to be stored somewhere in a data center managed by a provider of infrastructure as a service. Similarly, the smart TV that I own allows me to view tweets from the Twitter users I follow as a screen saver when the device is idle. These are examples of ubiquitous computing: the ability to access resources based on your identity from arbitrary devices connected to the network.
Along with the accelerating use of mobile devices, there are many trends in which consumer mobile devices are being used for business purposes. We are currently at an early stage of ubiquitous computing, in which the device a person is using is just a tool for accessing their personal data over the network. Businesses and governments are starting to recognize key advantages for using 100% cloud-based business-productivity software, which can improve employee mobility and increase work efficiencies.

In summary, millions of users every day find new ways to access networked applications via an ever-growing number of devices. There is great value in this data for driving business decisions, as long as it is possible to collect it, process it, and analyze it.

The Internet of ... Everything

In the future, anything powered by electricity might be connected to the Internet, and there will be lots of data passed from users to devices, to servers, and back. This concept is often referred to as the *Internet of Things*. If you thought that the billions of people using the Internet today generate a lot of data, just wait until all of our cars, watches, light bulbs, and toasters are online, as well.

It’s still not clear if the market is ready for Wi-Fi-enabled toasters, but there’s a growing amount of work by both companies and hobbyists in exploring the Internet of Things using low-cost commodity hardware. One can imagine network-connected appliances that users interact with entirely via interfaces on their smartphones or tablets. This type of technology is already appearing in televisions, and perhaps this trend will finally be the end of the unforgivable control panels found on all microwave ovens.

Like the mobile and Web application trends detailed previously, the privacy and policy implications of an Internet of Things will need to be heavily scrutinized; who gets to see how and where you used that new Wi-Fi-enabled electric toothbrush? On the other hand, the aggregate information collected from such devices could also be used to make markets more efficient, detect potential failures in equipment, and alert users to information that could save them time and money.

A Journey toward Ubiquitous Computing

Bringing together all of the sources of information mentioned previously may provide as many opportunities as red herrings, but there’s an important story to recognize here. Just as the distributed-computing technology that runs the Internet has made personal communications more accessible, trends in Big Data technology have made the process of looking for answers to formerly impossible questions more accessible.

More importantly, advances in user experience mean that we are approaching a world in which technology for asking questions about the data we generate—on a once unimaginable scale—is becoming more invisible, economical, and accessible.
How This Book Is Organized

Dealing with massive amounts of data requires using a collection of specialized technologies, each with their own trade-offs and challenges. This book is organized in parts that describe data challenges and successful solutions in the context of common use cases. Part I, “Directives in the Big Data Era,” contains Chapter 1, “Four Rules for Data Success.” This chapter describes why Big Data is such a big deal and why the promise of new technologies can produce as many problems as opportunities. The chapter introduces common themes found throughout the book, such as focusing on building applications that scale, building tools for collaboration instead of silos, worrying about the use case before the technology, and avoiding building infrastructure unless absolutely necessary.

Part II, “Collecting and Sharing a Lot of Data,” describes use cases relevant to collecting and sharing large amounts of data. Chapter 2, “Hosting and Sharing Terabytes of Raw Data,” describes how to deal with the seemingly simple challenge of hosting and sharing large amounts of files. Choosing the correct data format is very important, and this chapter covers some of the considerations necessary to make good decisions about how data is shared. It also covers the types of infrastructure necessary to host a large amount of data economically. The chapter concludes by discussing data serialization formats used for moving data from one place to another.

Chapter 3, “Building a NoSQL-Based Web App to Collect Crowd-Sourced Data,” is an introduction to the field of scalable database technology. This chapter discusses the history of both relational and nonrelational databases and when to choose one type over the other. We will also introduce the popular Redis database and look at strategies for sharding a Redis installation over multiple machines.

Scalable data analytics requires use and knowledge of multiple technologies, and this often results in data being siloed into multiple, incompatible locations. Chapter 4, “Strategies for Dealing with Data Silos,” details the reasons for the existence of data silos and strategies for overcoming the problems associated with them. The chapter also takes a look at why data silos can be beneficial.

Once information is collected, stored, and shared, we want to gain insight about our data. Part III, “Asking Questions about Your Data,” covers use cases and technology involved with asking questions about large datasets. Running queries over massive data can often require a distributed solution. Chapter 5, “Using Hadoop, Hive, and Shark to Ask Questions about Large Datasets,” introduces popular scalable tools for running queries over ever-increasing datasets. The chapter focuses on Apache Hive, a tool that converts SQL-like queries into MapReduce jobs that can be run using Hadoop.

Sometimes querying data requires iteration. Analytical databases are a class of software optimized for asking questions about datasets and retrieving the results very quickly. Chapter 6, “Building a Data Dashboard with Google BigQuery,” describes the use cases for analytical databases and how to use them as a complement for
batch-processing tools such as Hadoop. It introduces Google BigQuery, a fully managed analytical database that uses an SQL-like syntax. The chapter will demonstrate how to use the BigQuery API as the engine behind a Web-based data dashboard.

Data visualization is a rich field with a very deep history. Chapter 7, “Visualization Strategies for Exploring Large Datasets,” introduces the benefits and potential pitfalls of using visualization tools with large datasets. The chapter covers strategies for visualization challenges when data sizes grow especially large and practical tools for creating visualizations using popular data analysis technology.

A common theme when working with scalable data technologies is that different types of software tools are optimized for different use cases. In light of this, a common use case is to transform large amounts of data from one format, or shape, to another. Part IV, “Building Data Pipelines,” covers ways to implement pipelines and workflows for facilitating data transformation. Chapter 8, “Putting It Together: MapReduce Data Pipelines,” introduces the concept of using the Hadoop MapReduce framework for processing large amounts of data. The chapter describes creating practical and accessible MapReduce applications using the Hadoop Streaming API and scripting languages such as Python.

When data processing tasks become very complicated, we need to use workflow tools to further automate transformation tasks. Chapter 9, “Building Data Transformation Workflows with Pig and Cascading,” introduces two technologies for expressing very complex MapReduce tasks. Apache Pig is a workflow-description language that makes it easy to define complex, multistep MapReduce jobs. The chapter also introduces Cascading, an elegant Java library useful for building complex data-workflow applications with Hadoop.

When data sizes grow very large, we depend on computers to provide information that is useful to humans. It’s very useful to be able to use machines to classify, recommend, and predict incoming information based on existing data models. Part V, “Machine Learning for Large Datasets,” contains Chapter 10, “Building a Data Classification System with Mahout,” which introduces the field of machine learning. The chapter will also demonstrate the common machine-learning task of text classification using software from the popular Apache Mahout machine-learning library.

Interpreting the quality and meaning of data is one of the goals of statistics. Part VI, “Statistical Analysis for Massive Datasets,” introduces common tools and use cases for statistical analysis of large-scale data. The programming language R is the most popular open-source language for expressing statistical analysis tasks. Chapter 11, “Using R with Large Datasets,” covers an increasingly common use case: effectively working with large data sets with R. The chapter covers R libraries that are useful when data sizes grow larger than available system memory. The chapter also covers the use of R as an interface to existing Hadoop installations.

Although R is very popular, there are advantages to using general-purpose languages for solving data analysis challenges. Chapter 12, “Building Analytics Workflows Using Python and Pandas,” introduces the increasingly popular Python analytics stack. The chapter covers the use of the Pandas library for working with time-series
data and the iPython notebook, an enhanced scripting environment with sharing and collaborative features.


Finally, Chapter 14, “The Future: Trends in Data Technology,” takes a look at current trends in scalable data technologies, including some of the motivating factors driving innovation. The chapter will also take a deep look at the evolving role of the so-called Data Scientist and the convergence of various data technologies.
Acknowledgments

This book would not have been possible without the amazing technical and editorial support of Robert P. J. Day, Kevin Lo, Melinda Rankin, and Chris Zahn. I’d especially like to thank Debra Williams Cauley for her mentorship and guidance.

I’d also like to thank my colleagues Wesley Chun, Craig Citro, Felipe Hoffa, Ju-kay Kwek, and Iein Valdez as well as the faculty, staff, and students at the UC Berkeley School of Information for help in developing the concepts featured in this book.
Michael Manoochehri is an entrepreneur, writer, and optimist. With the help of his many years of experience working with enterprise, research, and nonprofit organizations, his goal is to help make scalable data analytics more affordable and accessible. Michael has been a member of Google’s Cloud Platform Developer Relations team, focusing on cloud computing and data developer products such as Google BigQuery. In addition, Michael has written for the tech blog ProgrammableWeb.com, has spent time in rural Uganda researching mobile phone use, and holds an M.A. in information management and systems from UC Berkeley’s School of Information.
Four Rules for Data Success

The first rule of any technology used in a business is that automation applied to an efficient operation will magnify the efficiency. The second is that automation applied to an inefficient operation will magnify the inefficiency.

—Bill Gates

The software that you use creates and processes data, and this data can provide value in a variety of ways. Insights gleaned from this data can be used to streamline decision making. Statistical analysis may help to drive research or inform policy. Real-time analysis can be used to identify inefficiencies in product development. In some cases, analytics created from the data, or even the data itself, can be offered as a product.

Studies have shown that organizations that use rigorous data analysis (when they do so effectively) to drive decision making can be more productive than those that do not.¹ What separates the successful organizations from the ones that don’t have a data-driven plan?

Database technology is a fast-moving field filled with innovations. This chapter will describe the current state of the field, and provide the basic guidelines that inform the use cases featured throughout the rest of this book.

When Data Became a BIG Deal

Computers fundamentally provide the ability to define logical operations that act upon stored data, and digital data management has always been a cornerstone of digital computing. However, the volume of digital data available has never been greater than at the very moment you finish this sentence. And in the time it takes you to read this sentence, terabytes of data (and possibly quite a lot more) have just been generated by computer systems around the world. If data has always been a central part of computing, what makes Big Data such a big deal now? The answer: accessibility.

The story of data accessibility could start with the IT version of the Cambrian explosion: in other words, the incredible rise of the personal computer. With the launch of products like the Apple II and, later, the Windows platform, millions of users gained the ability to process and analyze data (not a lot of data, by today's standards) quickly and affordably. In the world of business, spreadsheet tools such as VisiCalc for the Apple II and Lotus 1-2-3 for Windows PCs were the so-called killer apps that helped drive sales of personal computers as tools to address business and research data needs. Hard drive costs dropped, processor speeds increased, and there was no end to the amount of applications available for data processing, including software such as Mathematica, SPSS, Microsoft Access and Excel, and thousands more.

However, there's an inherent limitation to the amount of data that can be processed using a personal computer; these systems are limited by their amount of storage and memory and by the ability of their processors to process the data. Nevertheless, the personal computer made it possible to collect, analyze, and process as much data as could fit in whatever storage the humble hardware could support. Large data systems, such as those used in airline reservation systems or those used to process government census data, were left to the worlds of the mainframe and the supercomputer.

Enterprise vendors who dealt with enormous amounts of data developed relational database management systems (RDBMSs), such as those provided by Microsoft SQL Server or Oracle. With the rise of the Internet came a need for affordable and accessible database backends for Web applications. This need resulted in another wave of data accessibility and the popularity of powerful open-source relational databases, such as PostgreSQL and MySQL. WordPress, the most popular software for Web site content management, is written in PHP and uses a MySQL database by default. In 2011, WordPress claimed that 22% of all new Web sites are built using WordPress.²

RDBMSs are based on a tried-and-true design in which each record of data is ideally stored only once in a single place. This system works amazingly well as long as data always looks the same and stays within a dictated size limit.

## Data and the Single Server

Thanks to the constantly dropping price of commodity hardware, it's possible to build larger and beefier computers to analyze data and provide the database backend for Web applications. However, as we've just seen, there is a limit to the amount of processing power that can be built into a single machine before reaching thresholds of considerable cost. More importantly, a single-machine paradigm provides other limitations that start to appear when data volume increases, such as cases in which there is a need for high availability and performance under heavy load or in which timely analysis is required.

By the late 1990s, Internet startups were starting to build some of the amazing, unprecedented Web applications that are easily taken for granted today: software that

---

provides the ability to search the entire Internet, purchase any product from any seller anywhere in the world, or provide social networking services for anyone on the planet with access to the Internet. The massive scale of the World Wide Web, as well as the constantly accelerating growth of the number of total Internet users, presented an almost impossible task for software engineers: finding solutions that potentially could be scaled to the needs of every human being to collect, store, and process the world’s data.

Traditional data analysis software, such as spreadsheets and relational databases, as reliable and widespread as it had been, was generally designed to be used on a single machine. In order to build these systems to be able to scale to unprecedented size, computer scientists needed to build systems that could run on clusters of machines.

The Big Data Trade-Off

Because of the incredible task of dealing with the data needs of the World Wide Web and its users, Internet companies and research organizations realized that a new approach to collecting and analyzing data was necessary. Since off-the-shelf, commodity computer hardware was getting cheaper every day, it made sense to think about distributing database software across many readily available servers built from commodity parts. Data processing and information retrieval could be farmed out to a collection of smaller computers linked together over a network. This type of computing model is generally referred to as distributed computing. In many cases, deploying a large number of small, cheap servers in a distributed computing system can be more economically feasible than buying a custom built, single machine with the same computation capabilities.

While the hardware model for tackling massive scale data problems was being developed, database software started to evolve as well. The relational database model, for all of its benefits, runs into limitations that make it challenging to deploy in a distributed computing network. First of all, sharding a relational database across multiple machines can often be a nontrivial exercise. Because of the need to coordinate between various machines in a cluster, maintaining a state of data consistency at any given moment can become tricky. Furthermore, most relational databases are designed to guarantee data consistency; in a distributed network, this type of design can create a problem.

Software designers began to make trade-offs to accommodate the advantages of using distributed networks to address the scale of the data coming from the Internet. Perhaps the overall rock-solid consistency of the relational database model was less important than making sure there was always a machine in the cluster available to process a small bit of data. The system could always provide coordination eventually. Does the data actually have to be indexed? Why use a fixed schema at all? Maybe databases could simply store individual records, each with a different schema, and possibly with redundant data.
This rethinking of the database for an era of cheap commodity hardware and the rise of Internet-connected applications has resulted in an explosion of design philosophies for data processing software.

If you are working on providing solutions to your organization’s data challenges, the current era is the Era of the Big Data Trade-Off. Developers building new data-driven applications are faced with all manner of design choices. Which database backend should be used: relational, key-value, or something else? Should my organization build it, or should we buy it? How much is this software solution worth to me? Once I collect all of this data, how will I analyze, share, and visualize it?

In practice, a successful data pipeline makes use of a number of different technologies optimized for particular use cases. For example, the relational database model is excellent for data that monitors transactions and focuses on data consistency. This is not to say that it is impossible for a relational database to be used in a distributed environment, but once that threshold has been reached, it may be more efficient to use a database that is designed from the beginning to be used in distributed environments.

The use cases in this book will help illustrate common examples in order to help the reader identify and choose the technologies that best fit a particular use case. The revolution in data accessibility is just beginning. Although this book doesn’t aim to cover every available piece of data technology, it does aim to capture the broad use cases and help guide users toward good data strategies.

More importantly, this book attempts to create a framework for making good decisions when faced with data challenges. At the heart of this are several key principles to keep in mind. Let’s explore these Four Rules for Data Success.

Build Solutions That Scale (Toward Infinity)

I’ve lost count of the number of people I’ve met that have told me about how they’ve started looking at new technology for data processing because their relational database has reached the limits of scale. A common pattern for Web application developers is to start developing a project using a single machine installation of a relational database for collecting, serving, and querying data. This is often the quickest way to develop an application, but it can cause trouble when the application becomes very popular or becomes overwhelmed with data and traffic to the point at which it is no longer acceptably performant.

There is nothing inherently wrong with attempting to scale up a relational database using a well-thought-out sharding strategy. Sometimes, choosing a particular technology is a matter of cost or personnel; if your engineers are experts at sharding a MySQL database across a huge number of machines, then it may be cheaper overall to stick with MySQL than to rebuild using a database designed for distributed networks. The point is to be aware of the limitations of your current solution, understand when a scaling limit has been reached, and have a plan to grow in case of bottlenecks.

This lesson also applies to organizations that are faced with the challenge of having data managed by different types of software that can’t easily communicate or share
with one another. These **data silos** can also hamper the ability of data solutions to scale. For example, it is practical for accountants to work with spreadsheets, the Web site development team to build their applications using relational databases, and financial to use a variety of statistics packages and visualization tools. In these situations, it can become difficult to ask questions about the data across the variety of software used throughout the company. For example, answering a question such as “how many of our online customers have found our product through our social media networks, and how much do we expect this number to increase if we improved our online advertising?” would require information from each of these silos.

Indeed, whenever you move from one database paradigm to another, there is an inherent, and often unknown, cost. A simple example might be the process of moving from a relational database to a key–value database. Already managed data must be migrated, software must be installed, and new engineering skills must be developed. Making smart choices at the beginning of the design process may mitigate these problems. In Chapter 3, “Building a NoSQL-Based Web App to Collect Crowd-Sourced Data,” we will discuss the process of using a NoSQL database to build an application that expects a high level of volume from users.

A common theme that you will find throughout this book is use cases that involve using a collection of technologies that deal with issues of scale. One technology may be useful for collecting, another for archiving, and yet another for high-speed analysis.

**Build Systems That Can Share Data (On the Internet)**

For public data to be useful, it must be accessible. The technological choices made during the design of systems to deliver this data depends completely on the intended audience. Consider the task of a government making public data more accessible to citizens. In order to make data as accessible as possible, data files should be hosted on a scalable system that can handle many users at once. Data formats should be chosen that are easily accessible by researchers and from which it is easy to generate reports. Perhaps an API should be created to enable developers to query data programmatically. And, of course, it is most advantageous to build a Web-based dashboard to enable asking questions about data without having to do any processing. In other words, making data truly accessible to a public audience takes more effort than simply uploading a collection of XML files to a privately run server. Unfortunately, this type of “solution” still happens more often than it should. Systems should be designed to share data with the intended audience.

This concept extends to the private sphere as well. In order for organizations to take advantage of the data they have, employees must be able to ask questions themselves. In the past, many organizations chose a data warehouse solution in an attempt to merge everything into a single, manageable space. Now, the concept of becoming a data-driven organization might include simply keeping data in whatever silo is the best fit for the use case and building tools that can glue different systems together. In this case, the focus is more on keeping data where it works best and finding ways to share and process it when the need arises.
Build Solutions, Not Infrastructure

With apologies to true ethnographers everywhere, my observations of the natural world of the wild software developer have uncovered an amazing finding: Software developers usually hope to build cool software and don’t want to spend as much time installing hard drives or operating systems or worrying about that malfunctioning power supply in the server rack. Affordable technology for infrastructure as a service (inevitably named using every available spin on the concept of “clouds”) has enabled developers to worry less about hardware and instead focus on building Web-based applications on platforms that can scale to a large number of users on demand.

As soon as your business requirements involve purchasing, installing, and administering physical hardware, I would recommend using this as a sign that you have hit a roadblock. Whatever business or project you are working on, my guess is that if you are interested in solving data challenges, your core competency is not necessarily in building hardware. There are a growing number of companies that specialize in providing infrastructure as a service—some by providing fully featured virtual servers run on hardware managed in huge data centers and accessed over the Internet.

Despite new paradigms in the industry of infrastructure as a service, the mainframe business, such as that embodied by IBM, is still alive and well. Some companies provide sales or leases of in-house equipment and provide both administration via the Internet and physical maintenance when necessary.

This is not to say that there are no caveats to using cloud-based services. Just like everything featured in this book, there are trade-offs to building on virtualized infrastructure, as well as critical privacy and compliance implications for users. However, it’s becoming clear that buying and building applications hosted “in the cloud” should be considered the rule, not the exception.

Focus on Unlocking Value from Your Data

When working with developers implementing a massive-scale data solution, I have noticed a common mistake: The solution architects will start with the technology first, then work their way backwards to the problem they are trying to solve. There is nothing wrong with exploring various types of technology, but in terms of making investments in a particular strategy, always keep in mind the business question that your data solution is meant to answer.

This compulsion to focus on technology first is the driving motivation for people to completely disregard RDBMSs because of NoSQL database hype or to start worrying about collecting massive amounts of data even though the answer to a question can be found by statistical analysis of 10,000 data points.

Time and time again, I’ve observed that the key to unlocking value from data is to clearly articulate the business questions that you are trying to answer. Sometimes, the answer to a perplexing data question can be found with a sample of a small amount of data, using common desktop business productivity tools. Other times, the problem
is more political than technical; overcoming the inability of admins across different departments to break down data silos can be the true challenge.

Collecting massive amounts of data in itself doesn’t provide any magic value to your organization. The real value in data comes from understanding pain points in your business, asking practical questions, and using the answers and insights gleaned to support decision making.

**Anatomy of a Big Data Pipeline**

In practice, a data pipeline requires the coordination of a collection of different technologies for different parts of a data lifecycle.

Let’s explore a real-world example, a common use case tackling the challenge of collecting and analyzing data from a Web-based application that aggregates data from many users. In order for this type of application to handle data input from thousands or even millions of users at a time, it must be highly available. Whatever database is used, the primary design goal of the data collection layer is that it can handle input without becoming too slow or unresponsive. In this case, a key–value data store, examples of which include MongoDB, Redis, Amazon’s DynamoDB, and Google’s Google Cloud Datastore, might be the best solution.

Although this data is constantly streaming in and always being updated, it’s useful to have a cache, or a source of truth. This cache may be less performant, and perhaps only needs to be updated at intervals, but it should provide consistent data when required. This layer could also be used to provide data snapshots in formats that provide interoperability with other data software or visualization systems. This caching layer might be flat files in a scalable, cloud-based storage solution, or it could be a relational database backend. In some cases, developers have built the collection layer and the cache from the same software. In other cases, this layer can be made with a hybrid of relational and nonrelational database management systems.

Finally, in an application like this, it’s important to provide a mechanism to ask aggregate questions about the data. Software that provides quick, near-real-time analysis of huge amounts of data is often designed very differently from databases that are designed to collect data from thousands of users over a network.

In between these different stages in the data pipeline is the possibility that data needs to be transformed. For example, data collected from a Web frontend may need to be converted into XML files in order to be interoperable with another piece of software. Or this data may need to be transformed into JSON or a data serialization format, such as Thrift, to make moving the data as efficient as possible. In large-scale data systems, transformations are often too slow to take place on a single machine. As in the case of scalable database software, transformations are often best implemented using distributed computing frameworks, such as Hadoop.

In the Era of Big Data Trade-Offs, building a system data lifecycle that can scale to massive amounts of data requires specialized software for different parts of the pipeline.
The Ultimate Database

In an ideal world, we would never have to spend so much time unpacking and solving data challenges. An ideal data store would have all the features we need to build our applications. It would have the availability of a key–value or document-oriented database, but would provide a relational model of storing data for the best possible consistency. The database would be hosted as a service in the cloud so that no infrastructure would have to be purchased or managed. This system would be infinitely scalable and would work the same way if the amount of data under management consisted of one megabyte or 100 terabytes. In essence, this database solution would be the magical, infinitely scalable, always available database in the sky.

As of this publication, there is currently no such magic database in the sky—although there are many efforts to commercialize cutting-edge database technology that combine many of the different data software paradigms we mentioned earlier in the chapter.

Some companies have attempted to create a similar product by providing each of the various steps in the data pipeline—from highly available data collection to transformation to storage caching and analysis—behind a unified interface that hides some of these complexities.

Summary

Solving large-scale data challenges ultimately boils down to building a scalable strategy for tackling well-defined, practical use cases. The best solutions combine technologies designed to tackle specific needs for each step in a data processing pipeline. Providing high availability along with the caching of large amounts of data as well as high-performance analysis tools may require coordination of several sets of technologies. Along with this, more complex pipelines may require data-transformation techniques and the use of specific formats designed for efficient sharing and interoperability.

The key to making the best data-strategy decisions is to keep our core data principles in mind. Always understand your business needs and use cases before evaluating technology. When necessary, make sure that you have a plan to scale your data solution—either by deciding on a database that can handle massive growth of data or by having a plan for interoperability when the need for new software comes along. Make sure that you can retrieve and export data. Think about strategies for sharing data, whether internally or externally. Avoid the need to buy and manage new hardware. And above all else, always keep the questions you are trying to answer in mind before embarking on a software development project.

Now that we’ve established some of the ground rules for playing the game in the Era of the Big Data Trade-Off, let’s take a look at some winning game plans.
Abstraction model. See Cascading

Access
- authorization for BigQuery API, 76–78
- running mrjob scripts on Elastic MapReduce, 113–114

Access control lists (ACLs), in BigQuery, 74

Accessibility, data
- with cloud computing, 25
- current revolution in, 6
- with Hadoop for large-scale processing, 191
- machine learning challenges, 132–133
- with MapReduce, 71
- story of, 4

ACID (atomicity, consistency, isolation, and durability) test
- BASE alternative to, 31
- for relational databases, 28
- VoltDB compliance with, 41

ACLs (access control lists), in BigQuery, 74

Administrators, system, 145

Aesthetics
- communicating visual information, 85, 88, 96
- complex data visualizations with ggplot2 library, 91–92
- statistical graphic of Charles Minard, 86–88

Aggregate queries
- analytical databases for, 191
- BigQuery speeding up, 74
- for complex MapReduce jobs, 61, 67
- on relational databases, 58–59
- for specialized analytics systems, 53
- speeding up with Dremel, 72

Alagappan, Muthu, 134–135

Algorithms, machine learning
- k-means clustering, 135
- naïve Bayesian classifier, 134

Amazon
- Redshift. See Redshift
- scalability of, 32
- using mrjobs with Elastic MapReduce, 113–114

AMPLab
- Shark project, 51, 65–66, 82
- Spark project, 51, 65–66, 139

Analysis software
- anatomy of big data pipelines, 9
- decision making with, 44
- developing with Python, 160

Analytics, data
- adding to data processing pipeline, 60
- convergence and end of data silos, 51
- data silos useful for, 49–50
- data warehouses designed for, 47
- empowering employees for, 50–51
- for aggregate queries, 69
- future of query engines, 82–83
- growth of new, 191
- operational data stores for, 58
- understanding, 69–71

Analytics workflows
- building more complex workflows, 167–168
- choosing language for statistical computation, 158–159
- extending existing code, 159–160
- iPython, 170–174
- overview of, 157
- Pandas data analysis library, 163–167
- Python libraries for data processing, 160–163
- summary, 174–175
- tools and testing, 160
- working with bad or missing records, 169–170
Andreessen, Marc, 43–44
Apache
   Avro, 23
   Cassandra, 33, 40–41
   Derby, 60
   Hadoop. See Hadoop
   Hive. See Hive
   Mahout, 136–139
   Nutch Web crawler, Yahoo!, 71
   Pig. See Pig
   Thrift, 22–23
API (application programming interface)
   BigQuery, 74–82
   Dremel, 73
Apollo Guidance Computer, 147
Apple
   data accessibility and Apple II, 4
   iTunes Music Store, 44
Applications, cascading, 122–128
Arrays, NumPy, 161
ASCII (American Standard Code for Information Interchange) characters, 20
Atomic data types, in R, 148, 152
Attributes, relational database, 26
Authorization, for BigQuery API, 76–78
Autocomplete, iPython, 171
Automation
   of ETL processes, data warehousing, 48, 50
   first rule of technology, 3
   of parallelization, Hadoop/MapReduce, 104
   of partitioning, with Twemproxy, 39–40
   of predictive business value, 131
Availability
   of big data pipelines, 9
   CAP theorem of, 30–31
   relational databases vs. Internet, 29–30
   of ultimate database, 195

B
Bar charts, creating with D3.js, 94–96
BASE architecture, CAP theorem and, 30–31
BASH, MapReduce workflows, 58
Batch processing
   HDFS designed for, 60–61
   hosted services for, 187
with MapReduce jobs, 57, 59, 73
running Pig in, 122
Bayesian classification
   machine learning and, 133–134
   of text, with Mahout, 137–139
BI (business intelligence), 43–45
Big Blue, 27. See also IBM
Big Data
   accessibility of, 3–4
   anatomy of pipeline, 9
   build solutions, not infrastructure, 8
   build solutions that scale, 6–7
   build solutions to share data, 7–8
   data and single server, 4–5
   focus on unlocking value from data, 8–9
   overview of, 3
   sharing lots of files. See Formats
   summary, 10
   trade-offs, 5–9
   ultimate database, 10–11
   utility computing and, 190
Biglm, in R, 153–154
Big.matrix objects, in R, 150–151, 153–154
Bigmemory, in R, 150
BigQuery
   authorizing access to, 76–78
   bridging data silos, 51
   building custom big data dashboard, 75–76
   caching query results, 79–80
   data analytics as a service and, 73–74
   Dremel, 71–72
   Dremel vs. MapReduce, 72–73
   future of analytical query engines, 82–83
   managed services for tasks in cloud, 187
   not traditional database, 73
   overview of, 69
   query language, 74–75
   running query/retrieving result, 78–79
   summary, 83–84
   understanding analytical databases, 69–71
   use cases for iterative querying tasks
      solved by, 73
   visualization, 81–82
Billing model, BigQuery, 74
Bleeding-edge technologies, lacking support options, 186
Blogs, as document stores, 33–35
Bostock, Mike, 92
Bq command-line tool, 75
Brewer, Dr. Eric, 30
Broad Street pump, cholera map, 86–87
Broken data, Pandas for, 169–170
Bubble chart visualization, Trendalyzer, 89
Build vs. buy problem
costs of open-source, 186–187
evaluating current technology investments, 183
everything as a service, 187
machines unable to predict, 132
“my own private data center,” 184–186
overview of, 179–181
planning for scale, 184
playbook for, 182–183
starting small, 183–184, 187–188
understanding your problem, 181–182
Business
applying rules in ETL process, 48
automation of predictive value, 131

C
Caching layer
anatomy of big data pipelines, 9
Redis database, 36
Caching query results, BigQuery, 79–80
Callback URL, BigQuery API access, 77
CAP (consistency, availability, and partition tolerance) theorem, 30–31
Cascading
building application, 124–125
choosing Pig vs., 128–129
deploying application on Hadoop cluster, 127–128
JOIN example, 125–127
overview of, 122–123
sources and sinks, 123–124
summary, 128
use cases, 181
writing MapReduce workflows, 58
Character encoding
file transformations, 21
overview of, 19–21
CitrusDB, SQL on Hadoop project, 196
Cleaning raw data, ETL process, 48
Client ID, BigQuery API access, 77–78
Cloud computing
build vs. buy and, 179–180
changing systems administrator roles, 145
commonly used applications available as, 73
data warehousing in, 52, 66–67
defined, 192
distributed-data applications using, 186
future trends, 191–192
mobile computing devices using, 189
“my own private data center” vs., 184–186
present-time/future uses of, 187
rule of building/buying solutions hosted in, 8
ultimate database, 195–196
Cloud Console, 77
Cloudera. See Impala
Cluster
machine learning analyzing, 134–135
parallelizing iPython using, 171–174
running MapReduce job on Hadoop. See Hadoop
Codd, Edgar F.
Internet vs. database design of, 28–30
newSQL and, 41
relational database model of, 26–28
Code generator, Apache Thrift, 22–23
CoGroup process, Cascading, 125–126
Columnar formats
Dremel storing data on disk in, 72
loading data into Hive, 62
Compatibility
CSV files for, 18
problems of, 118
Compliance
data warehousing challenges, 50
planning for security, 46
Components, Pig, 119
Conjars website, Cascading, 124
Consistency
BASE systems for, 31
CAP theorem of, 30–31
of Google’s Spanner database, 41
Consistency (continued)
  limitations in Redis, 36–38
  VoltDB ACID-compliance, 41
Consistency, relational database
  ACID test, 28
  Google’s new F1, 196
  of transactions across systems, 70
  Web availability vs., 30
Consistent hashing, 40
Convergence
  of cultures, 196–197
  data warehousing/distributed computing, 51
  ultimate database and, 195–196
Coordinate system, SVG graphic files for
  Web, 93–94
Coordinated Universal Time (UTC), Pandas, 165–167
Core competencies, 181–182
Corpora Collection, 137
Cost
  build vs. buy. See Build vs. buy problem
  IAAS storage model, 15–16
  of moving data between systems, 22
  of moving from one database to another type, 7
  of open-source, 186–187
CouchDB, 35
CPU-bound problems, 158
CRAN (Comprehensive R Archive Network), 158–159
CSV (comma-separated value) format
  sharing large numbers of files, 16–18
  with text files, 20–21
  time series manipulation of IBM, 165–167
  XML and JSON compared to, 19
Cultures, trend for convergence of, 196–197
Customer-facing applications, 70
Customization, big data dashboard, 75–76

D
Data analytics. See also Statistical analysis
  convergence and end of data silos, 51
  data silos useful for, 49–50
  data warehouses designed for, 47
  empowering employees for, 50–51
  future of query engines, 82–83
  operational data stores for, 58
  as a service, 73–74
Data compliance, 46
Data dashboards
  adding visualization, 81–82
  analytical databases, 69–71
  authorizing access to BigQuery API, 76–78
  BigQuery, query language, 74–75
  BigQuery and data analytics as a service, 73–74
  building custom, 75–76
  caching query results, 79–80
  Dremel, 71–72
  Dremel vs. MapReduce, 72–73
  future of analytical query engines, 82–83
  overview of, 69
  running query/retrieving result, 78–79
  summary, 83–84
Data frames, R, 149, 151–152
Data inputs (sources), Cascading model, 123–124
Data integrity, 48, 70
Data outputs (sinks), Cascading model, 123–124
Data pipelines
  building MapReduce. See MapReduce
data pipelines
  combining tools for, 100–101
  complexity of, 118
  Hadoop streaming for, 101–105
  need for, 99–100
Data processing as a service, 185–186, 192
Data replication, and Hive, 60
Data rules for success. See Big Data
Data scientists
  current state of data technologies, 180
  definitions of, 192–193
  rise and fall of, 192–195
Data serialization formats
  Apache Avro, 23
  Apache Thrift and Protocol Buffers, 22–23
  overview of, 21–22
Data silos
  benefits of, 49–51


data compliance and security, 46
data warehouse solution to, 46–48
data warehousing ETL, 48
data warehousing/distributed computing convergence and, 51–52
Hadoop and, 48–49
hampering scalability, 7
jargon, 43–45
problems of, 45–46
summary, 53
Data transformation
MapReduce and, 101–102
one-step MapReduce, 105–109
Data types
Hive, 62
NumPy array, 161
Pandas, 164
R atomic, 148
Data warehouses
choosing over Hive, 60
cloud-based, 52, 66–67
convergence with distributed computing, 51–52
different meanings for, 57–59
distributed. See Hive; Spark project
ETL process, 48
negatives of, 50
overcoming data silos with, 46–48
Database(s)
anatomy of Big Data pipelines, 9
Big Data trade-offs, 6
document store, 33–35
for enormous amounts of data, 4
hierarchical manner of early, 26
key–value, 32–33
Redis. See Redis database
relational. See Relational databases
ultimate, 10, 195–196
Data-driven journalism, 92–93
Data-driven organizations, 50–51
DataFrames, Pandas
dealing with bad or missing records, 169–170
for more complex workflows, 167–168
overview of, 164
time series manipulation, 165–167
Data-modeling, CSV challenges, 17
Datasets
asking questions. See Questions, asking about large datasets
building data dashboard with BigQuery, 74, 76
machine learning for large. See ML (machine learning)
statistical analysis for massive. See R
strategies for large datasets
visualization strategies for large. See Visualization for large datasets
DB-Engines.com, 36
Debugging, Hadoop scripts locally, 154
Decision making
with analysis software, 44
applying computer input to, 131
machines able to report probability, 132
Denormalized data, star schema, 47
Device independence
cloud-based trends, 192
mobile computing, 189
DevOps engineer role, 145
Dimension tables, star schema in data warehouse system, 47–48
Distributed computing
CAP theorem and BASE, 30–31
data warehousing convergence with, 51–52
file transformation process, 21
new software solutions for databases, 41
overview of, 5–6
Distributed data warehousing
Hive. See Hive
Spark project, 51, 65–66, 139
Distributed file system, 109–110
Distributed machine learning systems, 136–139
Distributed-software systems, 145
Document stores, 33–35
Documentation, BigQuery, 77
Dot-distribution map, 86–87
Dremel
vs. MapReduce, 72–73
as new analytical database, 191
overview of, 71–72
Drill, analytical database, 191
Drivers, interacting with Hive, 65
Dumbo, 114
Durability
  BASE alternative to, 31
  relational database ACID test, 28
  VoltDB compliance, 41
DynamoDB, 73

Economies of scale, 15–16
EDW (enterprise data warehouse)
  ETL process, 48
  negatives of, 50
  overcoming data silos with, 46–48
Elastic MapReduce (EMR) service, 113–114
Ellison, Larry, 27–28
Email, spam filtering, 133–134
Embarrassingly parallel problems, 102
Employees, empowering to ask own questions, 50–51
EMR (Elastic MapReduce) service, 113–114
Engines, iPython, 171
Enron Corporation scandal, 46
Era of the Big Data Trade-Off, 6
Errors, Twemproxy handling, 39
ETL (Extract, Transform, and Load) process
  building pipelines, 58
  data warehousing, 48
  Hive addressing challenges of, 60–61
  solving data challenges without, 59
EXPLAIN statement, Hive performance, 64
Extensibility, Hive, 60
External tables, Hive, 62

F
F1 database, Google, 196
Facebook
  Hadoop data process model, 59
  interactive querying with Hive. See Hive Thrift, 22–23
  trends in data technology, 189–190
Fact table, 47
Fault tolerance, with HDFS, 59
Filtering
  in Cascading, 124
  in Pig workflow, 121–122
  spam, with machine learning, 133–134
Financial reporting regulations, SOX, 46
Firebase database, 196
Flat data, CSV format, 17
Flow map example, 86–88
Formats
  character encoding, 19–21
  comparing JSON, CSV, and XML, 19
  CSV, 16–18
  data serialization, 21–23
  file transformations, 21
  incompatible, 118
  JSON, 18–19
  optimizing Hive performance, 64
  shared data, 7
  supported by Hive, 62
  SVG graphic files for Web, 93–96
  XML, 18
Four rules for data success
  build solutions, not infrastructure, 8
  build solutions that scale, 6–7
  build solutions to share data, 7–8
  focus on unlocking value from data, 8–9
Future
  of analytical query engines, 82–83
  trends in data technology. See Technologies, future trends in data

G
Galton, Francis, 152
Gapminder Foundation, 89
Garbage collector, memory usage in R, 147–148
Gartner Hype Cycle curve, 190
General-purpose language, Python as, 159–160, 180–181
GFS, indexing Internet with, 71
ggplot2 library, interactive visualizations, 91–92
Google
  BigQuery. See BigQuery
  Charts API, 81–82
  Dremel, 191
  F1 database, 196
  Spanner database, 41
Governments, maximizing data accessibility, 15
Graphical user interface, of R, 146
Index 205

Grep command, text files, 20–21
Grunt shell, running Pig, 120–121

H

Hadoop
  administrative overhead of running, 171
  building multistep pipeline, 112–113
  Cassandra integration with, 41
  concepts behind, 60
  connecting R with, 154–155
  convergence and end of data silo, 51–52
  data transformation with, 21
  deficiencies of, 190–191
  defined, 71
  empowering users to store/process large data, 191
  for huge amounts of data, 59
  interactive querying for. See Hive
  negatives of, 49–50
  Pig abstracting data workflows for MapReduce tasks, 120
  Python MapReduce frameworks for, 110–111
  running Cascading application on cluster, 127–128
  running MapReduce jobs, 102, 109–110, 157–158
  running Pig script with cluster, 122
  starting small vs. using, 183–184
  streaming utility, 102–105
  summary, 66
  support options, 186
  as synonymous with Big Data, 190
  traditional data warehousing vs., 48–49
  using Apache Avro data serialization, 23
  using Shark with, 66
  Hadoop jar command, 127–128
  Hadoop Sequence files, 62
  Hadoop Streaming API, 154
Hardware
  changing role of systems administrators, 145
  IAAS providers handling failures of, 15–16
  maintaining and building own, 186
  offloading responsibilities to service providers, 73
Hash tables, 32
Hashing
  consistent, 40
  Twemproxy support for, 40
HBase database, connecting R with, 154
HBase tables, 65, 66
HDFS (Hadoop Distributed File System)
  concepts behind, 60
  connecting R with, 154
  creating Hive tables, 62–63
  Hive metastore, 60
  running MapReduce job on Hadoop cluster, 109–110
  running Pig script with, 122
  splitting data tasks across different machines, 58
  summary, 66
Hive
  additional data sources with, 65
  concepts behind, 61
  interactive querying for Hadoop, 60, 191
  loading data into, 62–63
  metastore, 62
  optimizing query performance, 64–65
  querying data with HiveQL, 63–64
  summary, 66–67
  use cases for, 60–61
  using Shark in conjunction with, 66
HiveQL, querying data with, 63–64
Hosting
  batch processing services, 187
  collection of data in JSON format, 19
  HTML5 APIs, storing local data, 79
Human readable files, challenges of CSV, 18

I

IAAS (infrastructure as a service)
  industry of, 8
  storage model for terabytes of raw data, 15–16
IBM
  Big Blue, 27
  developing relational database, 27
  time series manipulation of, 165–167
Image data, using SciPy for, 163
Impala
  future of analytical query engines, 82
Impala (continued)
as new analytical database, 191
overview of, 66
potential benefits of, 82
as visualization tool, 51
Infrastructure
avoiding overhead of managing, 185–186
building solutions rather than, 8
IAAS storage model avoiding, 16
managing scalable software services in cloud, 187
In-memory databases
defined, 36
of next-generation systems, 41
Redis, 36–38
sharding across many Redis instances, 38–41
In-memory environment
avoiding memory limitations of R, 147–148
Spark, 65–66
Inserting data
in document stores, 34–35
using Redis command-line interface, 37–38
Interactive visualizations
2D charts using matplotlib in Python, 92
building applications for, 90
with D3.js for Web, 92–96
with ggplot2 in R, 90–92
of large datasets, 89
Internet
accessibility of open-source relational databases, 4
aspect of BI system envisioned by Luhn, 52
big data trade-off, 5–6
building systems to share data, 7
global access to, 189
interactive visualizations with D3.js, 92–96
network latency issues, 16
relational databases vs., 28–29
single server and, 4–5
Web application development for, 4–5
IO-bound systems, 158
iPython
interactive shell, 92
notebook mode, 171
overview of, 170–171
parallelizing before using cluster, 171–174
Isolation
BASE alternative to, 31
relational database ACID test, 28
VoltDB compliance, 41
Iterative queries
Dremel speeding up, 71–72
Hive speeding up, 67
use cases for BigQuery, 73
iTunes Music Store, 44
J
JAR files, Cascading, 124, 127–128
Jargon
data silos, 43–45
data warehousing, 57–59
Java
Hadoop written in, 102
Mahout libraries, 137
Pig installation with, 120
Java Virtual Machine (JVM)-based API,
Cascading as, 128
JDBC drivers, interacting with Hive, 65
JOIN queries
creating Cascading applications, 125–127
OLAP systems avoiding excessive, 71
Journalism, data-driven, 92–96
JSON (JavaScript Object Notation) format
Avro using, 23
comparing to CSV and XML, 19
data serialization formats, 22
defined, 18
messages sent to BigQuery in, 78
sharing large numbers of files with, 18–19
Julia, numeric computations, 145
K
Ketama algorithm, Twemproxy, 40
Key–value data stores
Amazon.com, 32
anatomy of Big Data pipelines, 9
Cassandra, 40–41
Memcached, 39
Project Voldemort, 40
Redis as most popular. See Redis database using HBase with Hadoop, 65
Key–value pairs, MapReduce transformation, 106–108
K–means clustering algorithm, 135

L
Latency, global Internet data transfer speed, 16
Leipzig Corpora Collection, 137
Linear regression for large datasets, 153–154
Linear scalability
  of MemSQL, 41
  overview of, 39
  of Project Voldemort, 40
LinkedIn, Project Voldemort, 33
Lists, Python, 160
Loading data
  in ETL process. See ETL (Extract, Transform and Load) process
  into Hive, 62–63
  notebook mode, 92
Log data, CSV format, 17
Luhn, H.P., 44, 52

M
Machine learning. See ML (machine learning)
Mahout, 136–139
Managed tables, Hive, 62
Map phase, MapReduce defined, 61, 101
  one-step transformation, 106–107
  testing pipeline locally, 108–109
MapR, Drill analytical database, 191
MapReduce data pipelines
  alternative Python–based, 114
  building multistep pipeline, 112–113
  data transformation, 101–102
  defining data pipelines, 99–101
  with Hadoop streaming, 101–105
  map phase, 106–107
  one-step transformation, 105–109
  overview of, 99
  Python frameworks for Hadoop, 110–111
  reducer phase, 107–108
running job on Hadoop cluster, 109–110
running mrjob scripts on Elastic
  MapReduce, 113–114
  stdin to stdout, 102–103
  summary, 114–115
  testing locally, 108–109
  using workflow tools for, 118–119
MapReduce framework
  Cascading. See Cascading
  concepts behind, 60, 71–72
  creating job in R with rmr, 154–155
  data transformation, 101–102
  deficiencies of Hadoop, 191
  defined, 101
  defining workflows. See Workflows
  Dremel vs., 72–73
  interactive querying. See Hive
  as interface for Hadoop, 58
  optimizing Hive, 64–65
  Pig abstracting data workflows for, 120
  processing data, 52
  querying data in HiveQL, 63–64
  transforming data, 69
  use cases for batch processing, 73
  using Hadoop for long–running jobs, 66
Mathematical computing. See Numerical data
Matplotlib, 2D charts with Python, 92
Matrices, in R, 148–149
Media, software revolution in, 44
Memory capacity, R
  avoiding limitations of, 147–148
  large matrix manipulation, 150–151
  working with large data frames, 151–152
MemSQL, 41
Metastore, Hive, 60
Metrics, informing decision-making, 43–44
Miasma theory of cholera transmission, 86–87
Microsoft Excel, 51, 88
Minard, Charles Joseph, 86–88
Missing data, Pandas, 169–170
ML (machine learning)
  Apache Mahout, 136–139
  Bayesian classification, 133–134
  challenges of, 132–133
  clustering, 134–135
  defined, 132
ML (machine learning) (continued)
MLBase as distributed framework for, 139–140
overview of, 131–132
prediction of future and, 132
recommendation engines, 135–136
summary, 140
Mobile computing devices, 189
Moneyball: The Art of Winning an Unfair Game (Lewis), 134–135
MongoDB, 35, 186
Moore’s law, 147
Morse code, 19
Movie ratings, recommendation algorithms, 135–136
mrjob scripts
building multistep pipeline, 112–113
Dumbo vs., 114
rewriting Hadoop streaming example, 110–111
running MapReduce tasks on Hadoop clusters, 110
using with Elastic MapReduce, 113–114
Multiple insertion query, HiveQL, 63
Multistep MapReduce pipeline, 112–113
Multistep MapReduce transformations, 118–119
MySQL
alternatives to using. See NoSQL-based Web apps
building open-source databases with, 25
history of, 28
reasons for growth of, 29

N
Naïve Bayesian classifier algorithm, 134
Naming conventions
Hadoop Distributed File System, 109
Hive tables, 62
Natural Language Toolkit (NLTK) library, Python, 167–168
Netflix Prize contest, 135–136
Network latency
global Internet data transfer speeds, 16
moving data between systems, 22
Spanner database limitations, 41
NewSQL, 41
N-grams study, Python, 167–168
NLTK (Natural Language Toolkit) library, Python, 167–168
Nonrelational database models
creation of, 73
document store, 33–35
evolution of, 195–196
key–value databases, 32–33
managing data at scale, 69
overview of, 31–32
Non-Unicode data, 20
Normalization of data
building ETL pipelines, 58
data silo challenges, 45
in relational database model, 26
NoSQL-based Web apps
alternatives to Redis, 40–41
automatic partitioning with Twemproxy, 39–40
CAP theorem and BASE, 30–31
collecting crowd-sourced data, 25
document store, 33–35
evolution of, 195–196
key–value databases, 32–33
NewSQL, 41–42
nonrelational database models, 31–35
relational databases, ACID test, 28
relational databases, command and control, 25–28
relational databases, vs. Internet, 28–31
sharding across many Redis instances, 38–41
summary, 42
write performance with Redis, 35–38
Notebook mode, iPython, 171
Numerical data
computing using Python and Julia, 145, 158
computing using R, 159
large matrix manipulation in R, 150–151
maturity of R for, 180
parallelizing iPython using cluster, 171–174
tools for computing, 158
visualization of. See Visualization for large datasets
NumPy arrays, Python, 160–162, 164
OAuth protocol, 76–77
ODBC drivers
  interacting with Hive, 65
  Shark accessing, 66
OLAP (online analytical processing) systems, 70–71
Old Faithful scatterplot depiction, 90–91
OLTP (online transactional processing) systems, 70
OMQ library, iPython, 171
One-step MapReduce transformation
  map phase, 106–107
  overview of, 105–106
  reducer step, 107–108
  testing locally, 108–109
Online analytical processing (OLAP) systems, 70–71
Online transactional processing (OLTP) systems, 70
Open-source BI projects, 44
Open-source software, costs of, 186–187
Operational data store
  Hive not meant to be used as, 60
  overview of, 58
Operational systems, 46
Organizational culture, 182

Pandas (Python Data Analysis Library)
  data types, 164
  dealing with bad or missing records, 169–170
  overview of, 164
  searching for information on, 164
Parallelization
  of iPython using a cluster, 171–174
  using Hadoop, 104
Parent-child relationships, hierarchical early databases, 26
Partitioning
  automatic, with Twemproxy, 39–40
  optimizing Hive performance, 64

Passwords
  accessing network resources without sharing, 76
  BigQuery API access, 77
PBF (protocol buffer binary format), 23
PCs (personal computers), Big Data directives, 4–6
Performance
  optimizing Hive, 64–65
  optimizing Spark, 65–66
Pig
  Cascading vs., 122–123, 128
  filtering/optimizing workflows, 121–122
  overview of, 119–120
  running script in batch mode, 122–123
  running using interactive Grunt shell, 120–121
  use cases for, 181
  writing MapReduce workflows, 58

Pig Latin, 119
Pig Storage module, 120–121
PIL class, Python, 163
Pipe operator (|), Unix, 103
Pipe paradigm, Unix, 123
Pipelines
  anatomy of Big Data, 9
  building ETL, 58
Pipes, Cascading model, 123, 125–127
PostgreSQL
  alternatives to using. See NoSQL-based Web apps
  Amazon's Redshift based on, 66
  building open-source databases, 25
  creator of, 41
  history of, 28
Predictions
  automating business values, 131
  machines making future, 132
Presentation, sharing lots of data files, 14–15
Primary keys, relational database model, 26–27
Primitive data types, Hive, 62
Probability
  Bayesian classifier for spam, 134
  machines able to report, 132
  in statistical analysis, 150
Procedural model, Pig, 120
Professional sports, 134–135
Profile create command, iPython, 173
Programming languages, measuring popularity, 159
Project ID, access to BigQuery API, 77–78
Project Voldemort, 40
Proof-of-concept projects
  build vs. buy problem, 183–184
  using cloud infrastructure, 186
Protocol buffer binary format (PBF), 23
Protocol Buffers, data serialization format, 22–23
Public clouds, 185
Pydoop, 114
PyPI (Python Package Index), 159
Python
  2D charts with matplotlib, 92
  building complex pipeline, 103–105
  building complex workflows, 167–168
  extending existing code, 159–160
  as general-purpose language of choice, 92, 158–159
  iPython, 170–174
  libraries for data processing, 160–164
  lists, 160
  MapReduce frameworks, using Dumbo, 114
  MapReduce frameworks, using mrjob, 110–114
  numeric computations using, 145, 158
  NumPy arrays, 160–162
  Pandas, 163–167
  popularity ratings for, 159
  SciPy, 162–163
  tools and testing, 160
  writing MapReduce workflows, 58
Query engine, BigQuery, 74
Query language, BigQuery, 74–75
Querying results
  data silo challenges, 45–46
  data warehouse reporting capabilities, 47
  with Hive, 60
Questions, asking about large datasets
  data warehousing in cloud, 66–67
  definition of data warehousing, 57–59
  Hadoop project and, 57
  Hive, in practice, 61
  Hive, loading data into, 62–63
  Hive, optimizing query performance, 64–65
  Hive, overview, 60–65
  Hive, use cases for, 60–61
  Hive metastore, 62
  HiveQL, querying data with, 63–64
  overview of, 57
  Shark, 65–66
  summary, 67–68
  using additional data sources with Hive, 65
R
R
  choosing language for data analysis, 158–159
  for interactive visualizations, 90–91
  popularity ratings for, 159
  Python vs., 157
R strategies for large datasets
  biglm, 152–154
  data frames and matrices, 148–149
  definition of, 146
  large matrix manipulation, 150–151
  limitations of, 147–148
  original design for single-threaded machine, 146
  overview of, 145–146
  RHadoop, 154–155
  summary, 155–156
Raw data. See Sharing terabytes of raw data
RCFile (Record Columnar File) format, 62, 64–65
RDBMS (relational database management systems). See Relational databases
RDD (Resilient Distributed Datasets), 65–66
Read performance. See Redis database
Reading Hive files, in RCFiles, 62
Recommendation engines, machine learning, 135–136
Record Columnar File (RCFile) format, 62, 64–65
Redis database
alternatives to, 40–41
automatic partitioning with Twemproxy, 39–40
fast read/write performance of, 35–38
as key–value store, 33
sharding across many instances of, 38–41
Redshift
all-cloud architecture of, 52
managed services for tasks in cloud, 187
overview of, 66–67
Reducer phase, MapReduce
defined, 60, 101
one-step transformation, 107–108
overview of, 107–108
testing pipeline locally, 108–109
Redundancy
built into IAAS, 15–16
in relational database model, 26
Registration, accessing BigQuery API, 77
Regression analysis, 152–154, 165
Regression to the mean, 152
Regulations, financial reporting, 46
Relation, Pig, 120
Relational databases
ACID test, 28
asking questions about structured datasets, 58
best applications for, 31
challenges of large data sizes, 58–59
for customer-facing applications, 70
distributed computing limitations, 5
document stores vs., 33–35
in era of Big Data trade-off, 6
Google's F1, 196
history and characteristics of, 26–28
Hive metastore as, 60
Internet vs., 28–31
non-ideal use-cases for, 32
nonrelational designs in, 196
online analytical processing systems for,
70
operational data stores as, 58
rules, 28
sharding, 5
SQL vs. HiveQL, 63–64
supporting SQL, 58
Relational queries, 26
Reporting, in data warehousing, 47
Research and Innovative Technology
Administration (RITA), 150–151
Resilient Distributed Datasets (RDD), 65–66
REST-based API, BigQuery API, 78–79
RethinkDB, 196
Retrieving data
in document stores, 34–35
using Redis command-line interface, 37–38
RITA (Research and Innovative Technology
Administration), 150–151
Rosling, Hans, 89
Rules of data success, 6–9
S
Sample size, in machine learning, 133
Sarbanes and Oxley Act (SOX) regulations, 46
Scalability
alternatives to Redis, 40–41
of Amazon.com, 32
build vs. buy problem and, 184
building collection of technologies for, 7
CAP theorem and BASE, 30–31
of Google’s Spanner database, 41
of Hive, 60
of IAAS storage model, 16
as key goal of high-throughput databases, 38
linear, 39
of machine learning tasks with Mahout, 136–139
new database systems for. See NoSQL-
based Web apps
relational databases vs. Internet for, 29–30
of ultimate database, 195
Scatterplots
  creating with ggplot2 library, 92
  creating with R graphics, 90–91
Scientific computing
  Fortran for, 158
  iPython for, 170–174
  Julia for, 158
  Python for, 170
  R for, 158, 180
  SciPy for, 162–163
SciPy tools, Python, 162–163
Scope, BigQuery API, 77–78
Security
  data compliance and, 46
  data warehousing and, 50
Sequential access of data, CSV format, 17
Servers
  data and single, 4–5
  distributed computing model, 5–6
Service providers, offloading hardware responsibilities to, 73
Sharding
  automatic partitioning with Twemproxy, 39–40
  availability of software for, 41
  defined, 38
Sharding relational databases
  attempting to scale up, 6
  as data sizes get larger and larger, 41
  distributed computing limitations, 5
  Internet vs., 29–30
Sharing datasets, BigQuery, 74
Sharing terabytes of raw data
  challenges, 14–15
  character encoding, 19–21
  data serialization formats, 21–23
  formats, 16–19
  problems, 13
  storage, infrastructure as a service, 15–16
  summary, 23
Shark project
  fast analytics capability, 51
  future of analytical query engines, 82
  queries at speed of RAM, 65–66
  summary, 67
Shuffle step, MapReduce, 60, 101
Sicular, Svetlana, 190
Simple linear regression, 153–154
Single-machine paradigm, limitations, 4–5
Sinks (data outputs), Cascading model, 123–124
64-bit machine, memory usage in R, 147–148
SLF4J logging library, 124
Sloan Foundation grant, iPython, 171
Smartphones
  archaic computing systems vs., 147
  using differently than PCs, 189
Snapshots
  data warehouse analysis, 47
  operational data stores, 58
Snow, John, 86–87
Snytax, Pig’s workflow, 119
Social networks, 189
Software
  building instead of infrastructure, 8
  different projects addressing similar use cases, 180
  industries transitioning from physical media to, 44
  numerous packages affecting decision making, 180
  revolution in, 44
  sharding, 41
  specialized for big data pipelines, 9
Software Development Laboratories, Oracle, 27
Sources (data inputs), Cascading model, 123–124
SOX (Sarbanes and Oxley Act) regulations, 46
Spam filtering, machine learning for, 133–134
Spark project
  fast analytics capability, 51
  features of, 139
  machine learning tasks of, 139
  overview of, 65–66
Sparklines visualization concept, 89
Split command, text files, 20–21
SQL (Structured Query Language) query
  HiveQL vs., 63–64
  NewSQL, 41
  Pig workflow statements vs., 119
  in relational database model, 26–27, 58
Web apps not based on. See NoSQL-based Web apps
SQL on Hadoop project, CitusDB, 196
SQL-like interface
  Dremel, 72
  Hive, 60, 63–64
SQL-like syntax, BigQuery, 74
Standardization
  ASCII encoding, 20
  challenges of CSV files, 17
  Unicode Standard, 20
Star schema
  in data warehouse system, 47–48
  solving data challenges without, 59
Statistical analysis
  building analytics workflows. See Analytics workflows
determining probability, 150
maturity of R for, 180
using R for. See R strategies for large datasets
Statisticians
  growing need for, 146
  role of, 145
Stdin, complex pipeline in Python, 103–105
Stdin, simple pipeline in Unix, 102–103
Stdout, complex pipeline in Python, 103–105
Stdout, simple pipeline in Unix, 102–103
Storage
  challenges of sharing lots of data files, 14
  IAAS model for, 15–16
Streamgraph, 89
Streaming utility, Hadoop, 102–105
Structured documents, storing with XML, 18
Support, costs of open-source, 186
Survey data, working with, 150
SVG (Scalable Vector Graphics) format, Web, 93
Systems administrators, changing role, 145

T
Tableau, 51
Tables
  BigQuery, 74, 76
  in data warehouse system, 47
  Hive, 62–65
Taps, Cascading model, 123–125
Technologies
  anatomy of big data pipelines, 9
  bridging data silos, 51
  building core, 181–182
  current state/organic growth of data, 180
  evaluating current investment in, 182
  for large-scale data problems, 69, 72
  overcoming challenges of data silos, 50
  unlocking value from data vs. focus on, 8–9
Technologies, future trends in data
  cloud, 191–192
  convergence and ultimate database, 195–196
  convergence of cultures, 196–197
  data scientists, 192–195
  Hadoop, 190–191
  summary, 197–198
  utility computing pattern, 189–190
Testing
  MapReduce pipeline locally, 108–109
  Python scripts locally, 171
  Python tools for, 160
Text
  Bayesian classifier for spam, 134
  classifying with Apache Mahout, 137–139
  working with, 20
Time series data, Pandas, 164–165
TIOBE Programming Community Index, 159
Tools
  evaluating what to build or buy, 75
  Python, 160
TOP results, BigQuery, 74
Trade-offs
  bias variance, in machine learning, 133
  big data, 5–6
  data analysis API, 75
  IAAS storage model, 15–16
Transaction consistency, across systems, 70
Transformations
  big data pipeline anatomy, 9
  ETL process. See ETL (Extract, Transform and Load) process
file, 21
  multistep MapReduce, 118–119
  one-step MapReduce, 105–109
  Pig workflow, 122
Index

Trendalyzer, 89
Tufte, Edward, 86, 89
Tuples
   Pig, 120
   in relational database model, 26
Turing Award, Edgar F. Dodd, 28
Twemproxy, 39–40
Twitter, 189–190
Twitter Streaming API statistics, 167–168
Twitter Tools module, Python, 167–168
2D charts with Python, 92

U
UDFs (user-defined functions), Hive, 60
Ultimate database, 10, 195–196
Unique keys, key–value stores, 32
Unix command line
   building pipelines, 102–103
   pipe paradigm, 123
   text files, 20–21
Unlocking data
   to get value, 8–9
   as misleading metaphor, 118
Use cases
   analytical databases, 69
   batch processing with MapReduce, 73
   big data pipelines, 9
   BigQuery fast aggregate query results, 74
   Cascading vs. Pig, 180
   data warehousing, 58
   different software projects addressing similar, 180
   Hive, 60–62
   machine learning, 131–132
   MapReduce frameworks, 69
   nonrelational database models, 69
   UTC (Coordinated Universal Time), Pandas, 165–167
UTF-8 standard, 20
UTF-16 standard, 20
Utility computing
   adoption of technologies for, 189
   Big Data and, 190
   cloud-based trends, 192
   sharing terabytes of raw data, 15–16
   trend for convergence of cultures, 196–197

V
Value. See also Key–value data stores
   automating predictive business, 131
   focus on unlocking data, 8–9
   MapReduce converting raw text files, 58
Vector data structure, in R, 148
VisiCalc, 4
Visualization for large datasets
   building applications for data interactivity, 90–96
   with D3.js, for Web, 92–96
   with Google Charts API, 81–82
   human scale vs. machine scale, 89
   interactivity, 89
   masterpieces of historical visualizations, 86–88
   with matplotlib, 92
   overview of, 85
   with R and ggplot2, 90–92
   summary, 96
VoltDB, 41

W
Wearable computers, 189
Web services
   accessing, 76
   BigQuery API as, 76
   cloud-based trends, 192
Web-based dashboards, 7
Webmaster roles, in 1990s, 193
WordPress, data accessibility, 4
Workflows
   analytics. See Analytics workflows
   asking questions about data. See R
   Cascading. See Cascading
   Cascading vs. Pig, 128
   large-scale, 118
   multistep MapReduce transformations, 118–119
   overview of, 117
   Pig. See Pig
   summary, 128
   using Python to build more complex, 167–168
   writing MapReduce, 58
Workhorse data types, Python lists, 160
Write performance
Redis database excelling, 35–38
sharding across many Redis instances, 38–41

X
XML (Extensible Markup Language) format
comparing JSON, CSV to, 18–19
data serialization formats, 22
sharing large numbers of files with, 18

Y
Yahoo! distributed systems of commodity
hardware, 71
YAML format, configuring Twemproxy for
Redis, 40
Yelp, creating mrjob, 110

Z
Zero-based, tuples as, 26
ZeroMQ library, iPython, 171