Addison Wesley Data & Analytics Series

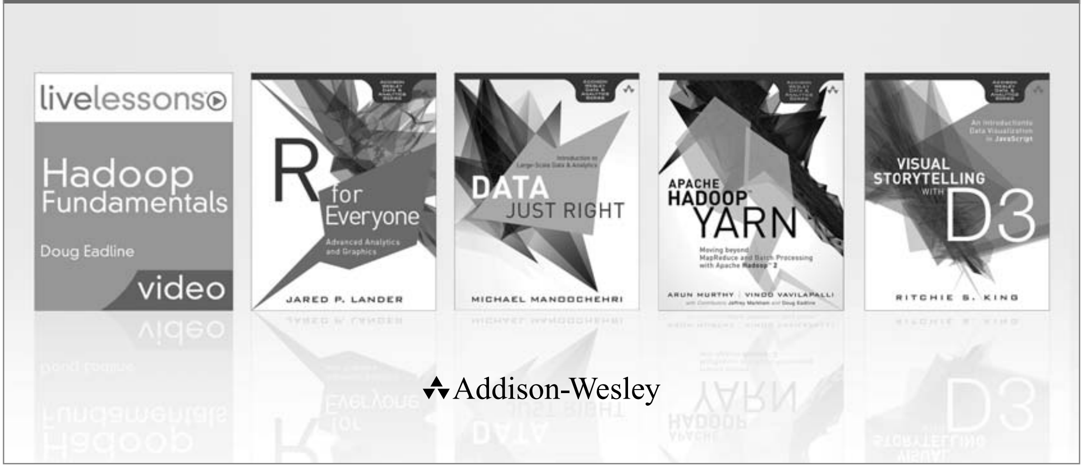
Introduction to Large-Scale Data & Analytics

DATA JUST RIGHT

MICHAEL MANOOCHEHRI

Data Just Right

The Addison-Wesley Data and Analytics Series



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

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.

Make sure to connect with us!
informit.com/socialconnect

informIT.com the trusted technology learning source | Addison-Wesley | Safari Books Online

ALWAYS LEARNING PEARSON

Data Just Right

Introduction to Large-Scale Data & Analytics

Michael Manoochehri

Addison-Wesley

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

For information about buying this title in bulk quantities, or for special sales opportunities (which may include electronic versions; custom cover designs; and content particular to your business, training goals, marketing focus, or branding interests), please contact our corporate sales department at corpsales@pearsoned.com or (800) 382-3419.

For government sales inquiries, please contact governmentsales@pearsoned.com.

For questions about sales outside the United States, please contact international@pearsoned.com.

Visit us on the Web: informit.com/aw

Library of Congress Cataloging-in-Publication Data

Manoochehri, Michael.
Data just right : introduction to large-scale data & analytics / Michael Manoochehri.
pages cm
Includes bibliographical references and index.
ISBN 978-0-321-89865-4 (pbk. : alk. paper)—ISBN 0-321-89865-6 (pbk. : alk. paper)
1. Database design. 2. Big data. I. Title.
QA76.9.D26M376 2014
005.74’3—dc23
2013041476

ISBN-13: 978-0-321-89865-4
ISBN-10: 0-321-89865-6
Text printed in the United States on recycled paper at RR Donnelley in Crawfordsville, Indiana.
First printing, December 2013

❖

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.

❖

This page intentionally left blank

Contents

Foreword

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

This page intentionally left blank

Preface

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 as 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.

Not all data challenges are purely technical. Part VII, "Looking Ahead," covers practical strategies for dealing with organizational uncertainty in the face of data-analytics innovations. Chapter 13, "When to Build, When to Buy, When to Outsource," covers strategies for making purchasing decisions in the face of the highly innovative field of data analytics. The chapter also takes a look at the pros and cons of building data solutions with open-source technologies.

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.

This page intentionally left blank

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.

This page intentionally left blank

About the Author

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.

This page intentionally left blank

1

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.

1. Brynjolfsson, Erik, Lorin Hitt, and Heekyung Kim. "Strength in Numbers: How Does Data-Driven Decisionmaking Affect Firm Performance?" (2011).

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

2. http://wordpress.org/news/2011/08/state-of-the-word/

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.

Index

A

B

H

I

N

O

P

S

T

X

Y

Z