Brought to you by:

IBM

Spark™ for dummies

A Wiley Brand

Discover Big Data today

Deploy Spark in your enterprise

Get ten tips for your Spark experience

Robert D. Schneider
Jeff Karmiol

2nd IBM Limited Edition

Spark

2nd IBM Limited Edition

by Robert D. Schneider
and Jeff Karmiol

for dummies®
A Wiley Brand

Spark For Dummies®, 2nd IBM Limited Edition

Published by
John Wiley & Sons, Inc.
111 River St.
Hoboken, NJ 07030-5774
www.wiley.com

LIMIT OF LIABILITY/DISCLAIMER OF WARRANTY: THE PUBLISHER AND THE AUTHOR MAKE NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE ACCURACY OR COMPLETENESS OF THE CONTENTS OF THIS WORK AND SPECIFICALLY DISCLAIM ALL WARRANTIES, INCLUDING WITHOUT LIMITATION WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE. NO WARRANTY MAY BE CREATED OR EXTENDED BY SALES OR PROMOTIONAL MATERIALS. THE ADVICE AND STRATEGIES CONTAINED HEREIN MAY NOT BE SUITABLE FOR EVERY SITUATION. THIS WORK IS SOLD WITH THE UNDERSTANDING THAT THE PUBLISHER IS NOT ENGAGED IN RENDERING LEGAL, ACCOUNTING, OR OTHER PROFESSIONAL SERVICES. IF PROFESSIONAL ASSISTANCE IS REQUIRED, THE SERVICES OF A COMPETENT PROFESSIONAL PERSON SHOULD BE SOUGHT. NEITHER THE PUBLISHER NOR THE AUTHOR SHALL BE LIABLE FOR DAMAGES ARISING HEREFROM. THE FACT THAT AN ORGANIZATION OR WEBSITE IS REFERRED TO IN THIS WORK AS A CITATION AND/OR A POTENTIAL SOURCE OF FURTHER INFORMATION DOES NOT MEAN THAT THE AUTHOR OR THE PUBLISHER ENDORSES THE INFORMATION THE ORGANIZATION OR WEBSITE MAY PROVIDE OR RECOMMENDATIONS IT MAY MAKE. FURTHER, READERS SHOULD BE AWARE THAT INTERNET WEBSITES LISTED IN THIS WORK MAY HAVE CHANGED OR DISAPPEARED BETWEEN WHEN THIS WORK WAS WRITTEN AND WHEN IT IS READ.

For general information on our other products and services, or how to create a custom *For Dummies* book for your business or organization, please contact our Business Development Department in the U.S. at 877-409-4177, contact info@dummies.biz, or visit www.wiley.com/go/custompub. For information about licensing the *For Dummies* brand for products or services, contact BrandedRights&Licenses@Wiley.com.

ISBN: 978-1-119-57697-6 (pbk); 978-1-119-57696-9 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

Publisher’s Acknowledgments

Some of the people who helped bring this book to market include the following:

Project Editor: Carrie A. Burchfield

Editorial Manager: Rev Mengle

Acquisitions Editor: Steve Hayes

Business Development Representative: Sue Blessing

IBM contributors: Rick Janowski, Neal Ekker

Table of Contents

Introduction

Welcome to *Spark For Dummies,* 2nd IBM Limited Edition! You've come to the right place if you want to get educated about how this exciting open-source initiative — and the technology behemoths that have gotten behind it — is transforming the already dynamic world of Big Data. Apache Spark represents a revolutionary new approach that shatters the previously daunting barriers to designing, developing, and distributing solutions capable of processing the colossal volumes of Big Data that enterprises are accumulating each day.

Where's all this data coming from? Unless you're stranded on a desert island, your routine daily activities all play a part in producing and consuming Big Data. Here's a brief list of some common sources:

- Making a phone call
- Checking email
- Browsing the Web
- Updating Facebook, LinkedIn, or Twitter
- Shopping — whether online or in a brick-and-mortar store, and whether you buy or not!
- Wearing a medical device
- Adjusting a thermostat
- Driving a car

With so many potential data-creating events, just one person's data footprint can get rather hefty just in a single day. But when you multiply that by the billions of smartphones, sensors, medical devices, and cameras that are churning away out there, data gets really big. In fact, according to Forbes, by the year 2020, about 1.7 megabytes of new information will be created every second for every human being on the planet.

Enticed by the possibilities, businesses, government agencies, and not-for-profit organizations are all increasingly getting in on the Big Data action. They're using it to fuel research, reduce risk, improve their products, stay ahead of the competition, and increase profitability.

However, simply identifying and deploying the right technology — such as Spark — to bring your Big Data vision to life can be downright confusing. That's where this book comes in: Our mission is to help you understand the state of Big Data today, along with how Spark can streamline and simplify the ways you interact with and extract value from it.

About This Book

Spark represents the next generation in Big Data infrastructure, and it's already supplying an unprecedented blend of power and ease of use to those organizations that have eagerly adopted it. However, to thoroughly comprehend Spark and its full potential, it's beneficial to view it in the context of larger information processing trends. To help you get the full picture, here's what we've set out to cover in this book:

- How — and why — Big Data has become so pervasive, including what's driving its growth

  This includes attributes such as its quantity (volume), diversity (variety), and the speed at which it's created (velocity).
- The rise of specialized, first-generation information processing strategies and technologies such as MapReduce and Apache Hadoop
- The vital role of open-source initiatives in advancing the state of the art
- The need for, and advent of, newer technologies such as Spark, along with how they protect existing customer investments in earlier generations of Big Data solutions
- A detailed exploration of Spark, including how it works and what it means for your organization
- The continued strengthening and adoption of integrated vendor-backed Big Data solutions such as IBM Spectrum Conductor
- A collection of industry-specific use cases that showcase Spark, both individually as well as in concert with other popular Big Data technologies
- Fitting all the pieces (for example, MapReduce, Hadoop, and Spark) together into a cohesive whole to benefit your organization

Foolish Assumptions

In writing this book, we've assumed that you're an executive, manager, or technology professional. We've also anticipated that you have some familiarity with enterprise software, database systems, or other mainline IT infrastructure. However, if that presumption doesn't describe you — don't worry. Even if this is the first time you've heard of Spark, MapReduce, Hadoop, or even Big Data for that matter, you can still profit from this book's introduction to the technology and its supporting use cases.

Icons Used in This Book

Every *For Dummies* book has small illustrations called icons in the margins; here are the ones we've used:

TIP

The Tip icon guides you to right-on-target information to help you get the most out of your Big Data efforts.

REMEMBER

This icon highlights concepts worth bearing in mind.

TECHNICAL STUFF

Be on the lookout for this icon if you want to explore the next level of detail.

WARNING

Pay close attention when you see this icon; it alerts you to potentially risky situations.

ON THE WEB

Seek out this icon if you want to find out even more about Spark, Big Data, MapReduce, or Hadoop.

Beyond the Book

This short book can't possibly cover all things Spark, so we provide links to several resources for you to peruse beyond this book.

- **`http://spark.apache.org`:** This reference for Spark is its official website. It provides everything you need to download, install, and configure Spark. Because its robust collection of APIs is such a key ingredient in Spark's success, the website also includes links to its API documents.
- **`http://spark-packages.org`:** This site has links to community-contributed packages that can be used seamlessly as Spark libraries. The `spark-xml` package, for example, makes treating XML documents as data sources for SparkSQL a breeze.
- **`http://codait.org`:** The Center for Open-Source Data & AI Technologies (formerly the Spark Technology Center) — spearheaded by IBM — originally focused on Spark, but it has evolved to make open-source and artificial intelligence (AI) technologies accessible to the practitioners that understand real-world problems so that they can develop open-source and AI solutions that solve these problems.

IN THIS CHAPTER

- Understanding what Big Data is
- Looking at MapReduce
- Getting to know Hadoop
- Recognizing the challenges with Hadoop and MapReduce
- Seeing the big picture of Spark
- Seeing how Big Data, MapReduce, Hadoop, and Spark relate

Chapter 1 Big Data Today

At its core, this book is a story about Apache Spark and how it's revolutionizing the way enterprises interact with the masses of data that they're accumulating. It's also about how IBM is revolutionizing the approach for deploying Spark in the enterprise with IBM Spectrum Conductor. But before we get to those fascinating details, it's worthwhile to spend some time exploring the full Big Data picture, including its origins, major technologies, widely employed use cases, and future trajectory, including the emerging world of artificial intelligence (AI). What you discover in this chapter is that Spark isn't a detour, but in fact, it's the next logical step in Big Data's evolution.

What Is Big Data?

Although exact definitions vary, most observers would agree that Big Data refers to information collections that are so large or complex that traditional technologies wither when attempting to store and process them. Additionally, Big Data signifies the new breed of applications and analytics that are being built around this information. However, the Big Data story is much more than a simple tale of enterprises amassing large data sets and then finding new ways to exploit them. In fact, what we label Big Data is the inevitable outcome of three universal trends:

- **Organizations are capturing, managing, and generating huge quantities of information:** Numerous independent market and research studies have found that data volumes continue to expand relentlessly. On top of all this extra new information, many enterprises are also stockpiling three or more years of historic data for regulatory or operational reasons.
- **Much of this data is unstructured:** Studies also indicate that 80 percent of today's data collections aren't arriving in the traditional, highly structured formats — the kind of data that fits in rows and columns and has been stored in relational databases for decades. Instead, this new data is composed of images, audio files, tweets, text messages, and so on. Until recently, most organizations have found it difficult — if not impossible — to take full advantage of all this unstructured information.
- **New breeds of applications are now being developed to exploit these masses of information and new data types:** Many of the tools and technologies that were designed to work with relatively large information volumes haven't evolved much in the past 20 years. This rigidity means that they simply can't keep up with Big Data. In response, enterprises are building — or buying — fresh classes of analytic applications. These new solutions are transforming the way enterprises are conducting business, and they're largely constructed on next-generation Big Data platforms — which we describe in this chapter's sections "What Is MapReduce?" and "What Is Hadoop?"

What can you do with Big Data?

Big Data has the potential to revolutionize the way you do business. It can provide new insights into everything about your enterprise, including the following:

- How your customers locate and interact with you
- The way you deliver products and services to the marketplace

- The position of your organization versus its competitors
- Strategies you can implement to increase profitability

Want some tangible examples? A comprehensive survey would fill a bookshelf, so this list gives you some quick, industry-specific use cases — all made possible by capturing, probing, and taking action on Big Data:

- Financial services
  - Gain deeper knowledge about your customers
  - Be on the lookout for fraudulent activities
  - Offer new, innovative products and services
  - Make better — and faster — trading decisions
- Telecommunications
  - Deliver the highest quality of service
  - Quickly identify and correct network anomalies
  - Make informed decisions about capital investments
  - Offer highly tailored packages to retain more customers
- Retail
  - Offer smarter up-sell and cross-sell recommendations
  - Get a better picture of overall purchasing trends
  - Set optimal pricing and discounts
  - Monitor social media to spot satisfied — or disgruntled — customers
- Energy
  - Detect meaningful patterns from torrents of raw sensor data
  - Identify promising new energy sources more quickly and accurately
  - Adjust inventory levels more accurately
  - Use predictive analytics to spot potential failures before they can cause harm
- Healthcare and Life sciences
  - Streamline and accelerate the process for new discoveries
  - Maintain compliance with regulations such as HIPAA

- Monitor vast numbers of medical devices for incipient crises
- Spot potential medication interactions before they can hurt the patient

What’s so provocative about these scenarios is that they have the potential to take place in real time — that is, however, if your infrastructure is designed properly with performance, scalability, and security in mind. If you want to learn about how enterprise-grade Spark makes this possible and about IBM Spectrum Conductor, a solution designed to streamline the deployment and management of enterprise Spark, take a look at Chapter 3. You find more detail about Big Data scenarios in Chapter 4, and if you’re seeking best practices for Spark implementations, Chapter 5 is the place to go.

What is driving the growth of Big Data?

Just as no single, universal definition of Big Data exists, there’s also no specific cause for what’s behind its rapid rate of adoption. Instead, multiple distinct trends have contributed to Big Data’s momentum. Five of the most notable ingredients have been new data sources, larger information quantities, broader data categories, commoditized hardware and software, and the rapid adoption and growth of artificial intelligence.

Before we tell you about these contributors to Big Data’s expansion, it’s worth pointing out that none of these advances would’ve been possible without new techniques to capture, work with, and analyze enormous amounts of unstructured raw data. Firms such as Google, Facebook, and Yahoo!, along with the not-for-profit Apache Software Foundation, all played a major role in moving things forward — in many cases, not just for the good of the industry but also to improve their own internal daily operations.

New data sources

Today, more information generators exist than ever before. These include devices such as mobile phones, tablet computers, sensors, medical equipment, and other technologies that gather vast quantities of information. Many of these newer sources are commonly labeled as belonging to the Internet of Things (IoT), and their impact is dwarfing everything that came before them.

TECHNICAL STUFF

For example, the number of connected things is already in the tens of billions and still growing. Some sources even state that there will be over 20 billion connected things in use by the year 2020.

Meanwhile, conventional enterprise applications are changing, too: E-commerce, financial risk management, and increasingly powerful scientific solutions (such as pharmaceutical, meteorological, and simulation, to name a few) are all contributing to the overall growth of Big Data.

Larger information quantities

As you might surmise from its name, Big Data also means that dramatically larger data volumes are now being captured, managed, and analyzed. To demonstrate just how much bigger Big Data can be, consider this: Over a history that spans more than 40 years, SQL database servers have traditionally held gigabytes of information, and reaching that milestone took a long time. In the past 30 years, data warehouses and enterprise analytics expanded these volumes to terabytes. But in the last ten years, the distributed file systems that store Big Data now routinely house petabytes of information, and the International Data Corporation (IDC) now forecasts that we will generate 175 zettabytes (175 billion terabytes) by 2025. This isn't a surprise when certain modern hybrid cars can generate around 25 gigabytes of data per hour.

Broader data categories

How does your enterprise's data suddenly balloon from gigabytes to hundreds of terabytes and then on to petabytes? One way is to start working with entirely new classes of information. While some of this new information is relational in nature, most is not.

In the past, most relational databases only held records of complete, finalized transactions. In the world of Big Data, *sub-transactional data* — which we define as information that's gathered while the transaction hasn't yet been completed — is also part of the picture. Here are just a few examples of sub-transactional data:

- Click trails through a website
- Shopping cart manipulation
- Tweets
- Text messages

Relational databases and associated analytic tools are designed to interact with structured information. But much of the information that makes up today's Big Data is unstructured or semi-structured, such as

- Photos
- Video
- Audio
- XML and JSON documents

TECHNICAL STUFF

Today's enterprise applications commonly transmit data by using documents encoded in Extensible Markup Language (XML) or JavaScript Object Notation (JSON). Even though they're at the heart of modern software, they've proven demanding for earlier generations of analytic tools to digest. This is thanks to their habitually massive size and their semi-structured nature.

Commoditized hardware and software

The final piece of the Big Data puzzle is the low-cost hardware and software environments that have transformed technology, particularly in the past ten years. Capturing and exploiting Big Data would be much more difficult and costly without the contributions of these cost-effective advances.

Artificial Intelligence

Artificial intelligence (AI) applications, which includes machine learning (ML) and deep learning (DL), need huge quantities of data to train the AI models to be accurate and effective. This depends heavily on high-quality, diverse, and dynamic data inputs. You can learn more about AI in Chapter 2.

Comparing Big Data with traditional enterprise data

REMEMBER

Thinking of Big Data as "just a lot more enterprise data" is tempting, but it's a serious mistake. First, Big Data is notably larger — often by several orders of magnitude. Secondly, Big Data is commonly generated outside of traditional enterprise applications. And finally, Big Data is often composed of unstructured or semi-structured information types that continually arrive in enormous amounts.

Because transactions are increasingly mobile and digital in nature, firms need to leverage Big Data to gain deeper insights, improve decision making, and provide better services. To do so, business operations need to integrate Big Data and Big Data sources into their traditional enterprise workflows, applications, and data sets. For example, a retailer might want to associate its website visitor behavior logs (a classic Big Data application) with purchase information (traditional enterprise data commonly found in relational databases). In another case, a mobile phone provider might want to offer a wider range of smartphones to customers (inventory maintained in a relational database) based on text and image message volume trends (unstructured Big Data).

The original challenges of Big Data

As is the case with any exciting new movement, first generation Big Data analytics came with its own unique set of barriers:

- **Information growth:** Unstructured data makes up the vast majority of what's being captured today. These massive volumes threatened to swamp all but the most well prepared IT organizations.
- **Processing power:** The customary approach of using a single, expensive, powerful computer to crunch information just didn't scale when confronted with the gargantuan amounts that typify Big Data. Instead, the smarter approach was to create an interconnected network of servers built on cheap, commoditized hardware and then distribute small work tasks to each server.
- **Physical storage:** Capturing and managing all this information consumed enormous resources, outstripping all budgetary expectations.
- **Data issues:** Lack of data mobility, proprietary formats, and interoperability obstacles all made working with Big Data complicated.
- **Costs:** Extract, transform, and load (ETL) processes for Big Data were expensive and time-consuming, particularly in the absence of specialized, well-designed software.

In the early days of Big Data, these complications proved to be an insurmountable hurdle for many organizations. Building applications using the same tools and technologies as had been used for decades was difficult, expensive, and delivered results too slowly. Driven by the deficiencies of using legacy approaches to work with Big Data, dedicated new computation and storage techniques began gaining traction.

Big Data programming methodologies

New programming practices made it possible to employ distributed computing on massive data sets: In fact, entire products and ecosystems have grown up around these advances. We tell you about MapReduce in the later section "What Is MapReduce?"; you also get a quick overview of Spark in the section "Spark from 30,000 Feet," with much more information on that subject available in Chapter 2.

Big Data storage architectures

Software development methodologies had a side effect of shaking up the previously boring world of data storage. New technologies quickly arose to support new distributed processing architectures, including large file systems running on inexpensive commodity hardware. One example of a new data storage technology is Hadoop Distributed File System (HDFS), which serves as a repository for enormous amounts of both structured and unstructured data.

The virtualization of applications has also caused substantial shifts in the world of data storage and its various architectures. Traditional architectures employed Network File Systems (NFS) and a few other protocols on local disks, while applications were each hosted on individual, dedicated servers. The advent of virtual machines changed all that. Now, multiple applications each supported by its own "virtual machine" often run on one physical server, and storage architectures have moved into their own systems called Storage Area Networks (SANs). These shared storage architectures are becoming prevalent, with shared storage systems offering in-place analytics that enable accessing the same data using many different file protocols, including NFS, SMB, POSIX, Object, and HDFS. "Containers," sometimes considered a lightweight version of virtual machines, though not mainstream yet, are the next step in this on-going evolution.

IBM Spectrum Scale represents a high-performance solution for storing and processing the colossal information quantities that are so prevalent in Big Data environments. IBM Spectrum Scale can utilize all the different protocols and does an excellent job of managing SANs and shared storage in Big Data environments. We tell you about both HDFS and IBM Spectrum Scale in this chapter's sections "What Is MapReduce?" and "What Is Hadoop?"

What Is MapReduce?

Old techniques for working with information simply don't scale to Big Data: They're too costly, time-consuming, and complicated. Thus, a new way of interacting with all this data became necessary, which is where MapReduce came in.

In a nutshell, MapReduce is built on the proven concept of divide and conquer by using distributed computing and parallel processing: It's much faster to break massive tasks into smaller chunks, allocate them to multiple servers, and process them in parallel.

Dividing and conquering

While this concept may appear new, in fact there's a long history of this style of computing, going all the way back to LISP and parallel computing in the 1960s.

REMEMBER

Faced with its own set of unique challenges, in 2004 Google decided to bring the power of parallel, distributed computing to help digest the enormous amounts of data produced during daily operations. The result was a group of technologies and architectural design philosophies that came to be known as MapReduce.

ON THE WEB

Check out `http://research.google.com/archive/mapreduce.html` to see the MapReduce design documents.

TECHNICAL STUFF

In MapReduce, task-based programming logic is placed as close to the data as possible. This technique works nicely with both structured and unstructured data. At its core, MapReduce is composed of two major processing steps: Map and Reduce. Put them together, and you've got MapReduce.

In the Map phase of MapReduce, records from a large data source are divided up and processed across as many servers as possible — in parallel — to produce intermediate values. After all the Map processing is done, the intermediate results are collected and combined (reduced) into final values.

This is a much simpler approach for large-scale computations and is meant to abstract away much of the complexity of parallel processing. Yet despite its simplicity, MapReduce lets you crunch massive amounts of information far more quickly than ever before.

It's no surprise that Google chose to follow a divide-and-conquer approach, given its organizational philosophy of using a lot of commoditized computers for data processing and storage instead of focusing on fewer, more powerful (and expensive!) servers. Along with the MapReduce architecture, Google also authored the Google File System (GFS). This innovative technology is a powerful, distributed file system meant to hold enormous amounts of data. Google optimized this file system to meet its voracious information processing needs. However, this was just the starting point: Google's MapReduce served as the foundation for subsequent technologies such as Hadoop, while GFS was the basis for HDFS.

The influence of MapReduce

If Google was the only organization deploying MapReduce, the story would end here. But as we pointed out earlier in this chapter in "What Is Big Data?", the explosive growth of Big Data placed IT organizations in every industry under great stress. The old procedures for handling all this information no longer scaled, and organizations needed a new approach. *Distributed processing* had proven to be an excellent way of coping with massive amounts of input data. Commodity hardware and software made it cost-effective to employ hundreds or thousands of servers — working in parallel — to answer a question.

REMEMBER

MapReduce was just the beginning: It provided a well-validated technology architecture that helped solve the challenges of Big Data, rather than a commercial product per se. Instead, MapReduce laid the groundwork for the next step in Big Data's evolution: Hadoop.

What Is Hadoop?

MapReduce was a great start, but it required you to expend a significant amount of developer and technology resources to make it work in your organization. This wasn't feasible for most enterprises, and the relative complexity led to the advent of Hadoop.

The first question you're probably asking is, "What's a Hadoop?" Believe it or not, the original Hadoop was a toy elephant, specifically, the toy elephant belonging to Doug Cutting's son. Doug — who created the Hadoop implementation of MapReduce — gave this name to the new initiative.

Hadoop is a well-adopted, standards-based, open-source software framework built on the foundation of Google's MapReduce and GFS papers. It's meant to leverage the power of massive parallel processing to take advantage of Big Data, generally by using a lot of inexpensive commodity servers.

REMEMBER

Hadoop was designed to abstract away much of the complexity of distributed processing. This let developers focus on the task at hand, instead of getting lost in the technical details of deploying such a functionally rich environment. But as we portray in this chapter's "Looking at Hadoop and MapReduce Challenges" section, the MapReduce processing strategy running on Hadoop environments has proven to have its own set of challenges — which in turn have been addressed by Spark.

After it started becoming popular, the not-for-profit Apache Software Foundation took over maintenance of Hadoop, with Yahoo! making significant contributions. Hadoop has gained tremendous adoption in a wide variety of organizations, including

- Social media (Facebook, Twitter, LinkedIn)
- Life sciences
- Financial services
- Retail
- Government

Hadoop architecture

Envision a Hadoop environment as consisting of three basic layers. These represent a logical hierarchy with a full separation of concerns — in other words, each layer has its own specializations that it carries out independently of the other layers. Together, these levels deliver a full MapReduce implementation. The Hadoop architecture is shown in Figure 1-1.

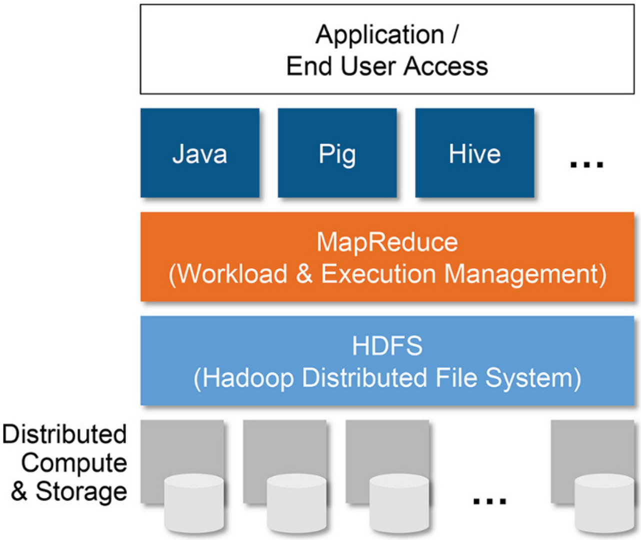

Application / End User Access
Java
Pig
Hive
...
MapReduce (Workload & Execution Management)
HDFS (Hadoop Distributed File System)
Distributed Compute & Storage
...

FIGURE 1-1: The layers of the Hadoop architecture.

Application layer/end-user access layer

This stratum provides a programming framework for software developers to apply distributed computing techniques to extract meaning from large data sets. Thus, it serves as the primary point of contact for applications to interact with Hadoop. These applications may be internally written solutions, or third-party tools such as business intelligence, packaged enterprise software, and so on.

TECHNICAL STUFF

To build one of these applications, you normally employ a popular programming interface such as Java, Pig (a specialized, higher-level MapReduce language), or Hive (a specialized, SQL-based MapReduce language). Meanwhile, to schedule Hadoop jobs, you'll likely incorporate the software developed from the Apache Oozie project.

ON THE WEB

Learn more about Apache Pig at `pig.apache.org`, Apache Hive at `hive.apache.org`, and Apache Oozie at `oozie.apache.org`.

MapReduce workload management layer

Commonly known as *JobTracker*, this Hadoop component supplies an open-source runtime job execution engine. This engine coordinates all aspects of your Hadoop environment, such as scheduling and launching jobs, balancing the workload among different resources, and dealing with failures and other issues — all essential responsibilities in a highly distributed data landscape.

Distributed parallel file systems/data layer

This layer is responsible for the actual storage of information. For the sake of efficiency, it commonly uses a specialized distributed file system. In most cases, this file system is HDFS.

TECHNICAL STUFF

Along with HDFS, this layer may also consist of commercial and other third-party storage implementations. These include IBM Spectrum Scale, the MapR filesystem from MapR Technologies, Kosmix's CloudStore, and Amazon's Simple Storage Service (S3).

The inventors of HDFS made a series of important design decisions:

- **Files are stored in larger blocks:** Many file systems store their information in blocks of 4 kilobytes (KB); some use even smaller values. In contrast, HDFS blocks are hundreds of times bigger, with a default of either 64 or 128 megabytes (MB). Storing Hadoop's information in these large blocks helps reduce the overall number of blocks that must be maintained and also minimizes overhead.
- **Reliability is achieved through replication:** Each block is replicated across two or more servers that are assigned the job of storing information. Formally known as *DataNodes,* most Hadoop environments replicate to a default number of three.
- **A single master *NameNode* coordinates access and metadata:** This simplifies and centralizes management.
- **No data caching:** It's not worth it given the large data sets and sequential scans.
- **There's a familiar interface with a customizable application programming interface (API):** This lets you simplify the problem and focus on distributed applications, rather than performing low-level data manipulation.

While HDFS was a reasonable choice for a file system, over the past few years numerous other information storage options have become popular. This diversity not only led to some challenges for Hadoop and MapReduce but also was a catalyst for fresh techniques such as Spark.

Hadoop's ecosystem

Hadoop's modular structure has spawned a thriving ecosystem of highly specialized technologies and vendors. Major elements in the Hadoop ecosystem include

- **Distributed data storage:** These technologies are used to maintain large data sets on commodity storage clusters.
- **Distributed MapReduce runtime:** This software is assigned the important responsibility of scheduling and distributing jobs that consume information kept in distributed data storage.
- **Software development tools and applications:** An expansive range of solutions lets programmers and non-programmers alike derive value and meaning from the information stored in Hadoop.
- **Hadoop distributions:** These commercial packages provide a single, integrated group of components that are pre-tested and certified to work together.
- **Business intelligence and related tools:** These popular technologies have been in use for years working with traditional relational data, and they've now been extended to work with data that's accessible via Hadoop.

Looking at Hadoop and MapReduce Challenges

While Hadoop and MapReduce were a great start — and they continue to be heavily utilized — they're not the end of the story for optimal Big Data processing. Techniques and tools have continued to evolve, in part driven by some notable solution development and operational headaches such as

- **Batch processing:** From the beginning, Hadoop running MapReduce has been most commonly deployed in batch-processing environments. But once users get a taste of what's possible with Big Data, they only want more — and they want answers right away.

- **Performance:** Hadoop and MapReduce are heavily reliant on time-consuming read/write interactions with disk drives. As both data volumes and user expectations grow, these slowdowns become less and less acceptable.
- **Solution development complexity:** While Hadoop and MapReduce application construction methodologies are vastly simpler than earlier hand-cobbled application development techniques, they remain well out of reach for the vast majority of software developers: The learning curve is still far too steep.
- **Silo proliferation:** With users enticed by the possibilities of rapid access to reams of valuable data — and in the absence of proper oversight — matters can speedily devolve into an expensive collection of isolated information silos. Naturally, each data island will have its own dedicated, underutilized hardware and will be inaccessible from the rest of the organization. Along with being an administrative and security nightmare, this proliferation results in higher expenses and diminished return on investment (ROI).
- **Supporting multiple versions:** Open-source software — especially critical infrastructure such as Hadoop — is characterized by endless updates. Many enterprises find it necessary to run multiple versions of their open-software technologies at the same time, with the upshot being an ever-expanding collection of isolated data silos.
- **Supporting multiple tenants:** When an organization makes the not-insubstantial expenditures for Big Data hardware, software, and expertise, it can expect to derive maximum return from its investment. This means making data and related solutions available to as many users as possible. The act of sharing, however, opens up all sorts of potential security risks, especially when these users get a taste of what's possible and encourage even more people to participate. And once again, it's common for this requirement to result in multiple Hadoop implementations and ensuing information silos.

- **Infrastructure rightsizing:** As an enterprise rolls out the underlying infrastructure meant to support its Big Data initiatives, it's easy to fall into one of two capacity planning traps:
  - If there's not enough server capacity to meet user needs, the resulting performance bottlenecks can make things unpleasant for the user community.
  - On the other hand, if the organization acquires excess capacity, then it's wasted time, money, and personnel that could be better deployed elsewhere.
- **Storage restrictions:** Partially by design, and partially by common usage, Hadoop and MapReduce applications have largely been focused on a relatively small number of information repositories, with HDFS being the most popular. However, nearly a dozen widely adopted repository standards and copious other information stockpiles are in use. Making the most of all this data means that a Big Data environment must be capable of working with the complete range of these sources.

Aside from Hadoop's and MapReduce's inherent drawbacks, new technologies, methodologies, and market trends are affecting Big Data and changing the way these initiatives are carried out. Some of the most significant movements include

- **Fast-evolving software ecosystems:** The pace of change in software ecosystems is speeding up. For example, Spark — which we introduce in the later section "Spark from 30,000 Feet" — appeared on the scene quite rapidly after Hadoop and MapReduce began gaining commercial adoption.
- **Born-in-the-cloud technologies:** We're reaching a point where the vast majority of enterprise software is designed, developed, and deployed in the cloud. This has major implications for your Big Data strategy.
- **New data storage and processing models:** Technology visionaries continue to push the envelope, leveraging all of the processing power, storage capacity, and network bandwidth that they can. The overall direction is to democratize access to information — particularly Big Data — and create innovative solutions in the process.

- **Cloud maturation:** In the early days of Hadoop and MapReduce, enterprises were just beginning to fully grasp the potential of cloud computing. At this point, most sophisticated organizations have a cloud computing strategy that incorporates a blend of public, private, and hybrid instances.

The history and challenges set the stage for the next generation of techniques for working with real-time Big Data, such as Spark.

Spark from 30,000 Feet

While the remainder of this book is dedicated to Spark, here's a brief introduction to give you some initial insights.

Spark is a memory-optimized data processing engine that can execute operations against Hadoop data (in HDFS) with better performance than MapReduce. It can also process data on other clustering environments, such as Cassandra, or even on its own clusters and files. By minimizing disk reads and writes, Spark delivers very high-performance access to dynamic, complex data collections and provides interactive functionality such as ad hoc query support in a scalable way.

Spark extends the batch-processing paradigm that's so popular in MapReduce toward the real-time applications that users crave. Developers appreciate the productivity gains that Spark supplies via its broad base of programming languages, including Java, Scala, Python, and R. Supporting so many different development approaches also helps to diminish the number of technical barriers to building Spark solutions.

For structured data access on Spark, several SQL-based options are available, including Spark SQL. Spark can work with a wide variety of data sources, including SQL databases.

Relating Big Data, MapReduce, Hadoop, and Spark

The earlier parts of this chapter describe Big Data, MapReduce, Hadoop, and Spark. To provide some additional clarity, here's a quick rundown of how they relate:

- **Big Data:** Today most enterprises are facing a lot of new data, which arrives in many different forms. Big Data analytics has the potential to provide insights that can transform every business. And Big Data use has spawned a whole new industry of supporting architectures such as MapReduce.
- **MapReduce:** A new programming framework — created and successfully deployed by Google — that uses the divide-and-conquer method (and a lot of commodity servers) to break down complex Big Data problems into small units of work, and then processes them in parallel. These problems can now be solved faster than ever before, but deploying MapReduce alone is far too complex for most enterprises. This reality led to Hadoop.
- **Hadoop:** A complete technology stack that implements the concepts of MapReduce to exploit Big Data. Hadoop has also spawned a robust marketplace served by open-source and value-add commercial vendors. In recent years, data scientists and application developers have been drawn toward pairing Spark with Hadoop.
- **Spark:** A new approach — composed of software infrastructure along with simple yet powerful application development techniques — to developing and delivering Big Data applications. Instead of being viewed as a replacement for MapReduce, Spark can best be appreciated as the next step in Big Data's evolution. In fact, most Hadoop distributions also include a Spark distribution to be installed in the same environment.

IN THIS CHAPTER

- » Looking at Spark's background and history
- » Understanding common use cases
- » Processing information with Spark
- » Seeing how Spark benefits the entire organization
- » Detailing the core technology components
- » Comparing Hadoop/MapReduce and Spark
- » Looking at the challenges of open-source Apache Spark

Chapter 2
Introducing Spark

In Chapter 1, we describe the continual evolution of Big Data and the technologies that support it. To call it a dynamic, ever-changing environment is an understatement. In fact, it's in constant flux, with new and exciting open-source solutions regularly popping up and then rapidly maturing.

The good news here is that the designers of these technologies have been doing a great job regarding interoperability, plotting logical roadmaps, and so on. This progress has helped protect the often-substantial investments that Big Data adopters have made to develop and roll out production systems.

In this chapter, we tell you all about Apache Spark — the next generation of Big Data application development technology. To help put things in context, we tell you about its history, why it was developed, what it's useful for, and where it's heading. We also set the stage for Chapter 3, where we discuss enterprise Spark deployments and introduce IBM Spectrum Conductor, a software platform designed to simplify the deployment and management of enterprise Spark solutions.

Spark's Background and History

Thanks to its early successes and advantages versus prior methods, for years MapReduce has been the de facto software development methodology for distributed Big Data batch processing (more on MapReduce in Chapter 1). However, batch processing, while useful, isn't the answer for every computational situation. Driven by continuing, widespread adoption of Big Data, and user hunger for additional use cases (which was partially driven by the first generation of innovative solutions), researchers began looking at ways to improve on MapReduce's capabilities. Two interesting outcomes of this research were Apache Drill and Spark.

Drill is a high-performance Structured Query Language (SQL) engine meant to provide self-service data exploration features to users, letting them navigate through all this new information without having to submit requests to their colleagues in the IT department. On the other hand, Spark was designed to be a general-purpose computational engine for interactive, batch, and streaming tasks that were capable of leveraging the same types of distributed processing resources that had heretofore powered MapReduce initiatives.

TECHNICAL STUFF

When viewed against the backdrop of Big Data's open-source history, the continual search for better solutions isn't surprising. For example, Apache Pig (a specialized, higher-level MapReduce language) was the outcome of Yahoo!'s quest for more user-friendly access to data stored in Hadoop; Apache Hive (which brought the power of SQL to MapReduce) was the upshot of Facebook's probe for more interactive, SQL-driven data interactions. The trend continues with other Apache-backed initiatives such as Impala (meant for business analytic-style queries for Hadoop) and Presto (a high speed, distributed processing engine for running SQL queries against massive databases).

The designers of Spark created it with the expectation that it would work with petabytes of data that were distributed across a cluster of thousands of servers — both physical and virtual.

From the beginning, Spark was meant to exploit the potential — particularly speed and scalability — offered by in-memory information processing. These time savings really add up when compared with how MapReduce jobs tend to continually write intermediate data back to disk throughout an often lengthy processing cycle. A rule of thumb metric is that Spark delivers results on the same data set up to 100 times faster than MapReduce.

In comparison with earlier approaches that were based on MapReduce, Spark offers four primary advantages for developing Big Data solutions:

- Performance
- Simplicity
- Ease of administration
- Faster application development

Thanks to these augmentations, Spark gained adoption quickly and was promoted to a top-level Apache Software Foundation project in 2014. While more than 300 Apache Software Foundation projects are in the works, a minute number attain this status. These select few are considered to be highly significant and are showered with much more attention, resources, and contributors. We explain how Spark demonstrates this superiority in the later section "How Spark Benefits the Entire Organization."

Common Use Cases for Spark

It's impossible to even scratch the surface of how many different valuable applications there are for Spark-based solutions. Its speed, simplicity, and developer productivity make enormous numbers of previously lengthy and expensive projects much more practical than with earlier Big Data tooling.

With that said, Spark is particularly compelling in a few common applications:

- Streaming data
- Artificial intelligence, machine learning, and deep learning
- Business intelligence
- Data integration

REMEMBER

To make these examples more realistic, in this section we use situations likely to be encountered during the daily operations of a financial services firm. These types of use cases are applicable across every industry. In Chapter 4, we provide many more examples of specialized solutions for

- Financial services
- Telecommunications
- Retail
- Energy
- Life sciences

Streaming data

Although the specific sources and usages vary by industry, many enterprises are now guzzling a constant torrent of streaming data, fed by

- Sensors
- Financial trades
- Mobile devices
- Medical monitors
- Website access logs
- Social media updates from Facebook, Twitter, and LinkedIn

Working directly on streaming data is different from the recent past, where enterprise software-driven transactions arrived at a more leisurely rate — and in smaller volumes.

Using the financial services example, a Spark application can analyze raw incoming credit card authorizations and completed transactions in essentially real time, rather than wading through this information after the fact. This time-saver is a game-changer for solutions like security, fraud prevention (instead of just detection), and so on.

For example, the firm could carry out a fact-based business decision to reject a bogus credit card transaction at the point of sale in real time, rather than permitting the fraudster to acquire merchandise and thus trigger a loss for the bank and retailer.

Artificial intelligence, machine learning, and deep learning

Enterprises are increasingly using artificial intelligence (AI), machine learning (ML), and deep learning (DL) algorithms to develop powerful new analytic capabilities across many use cases, from computer vision and object detection, through improved human computer interaction via natural language processing, to sophisticated anomaly detection capabilities. At the heart of any use case associated with AI, ML, and DL are sophisticated pattern recognition and classification capabilities that serve as the birthplace for revolutionary applications and insights of the future.

AI projects begin as Big Data problems, and the accuracy and quality of an AI model is directly affected by the quality and quantity of data used. Experts estimate that emerging AI solutions will need eight to ten times the data volume used for current Big Data solutions. This data comes from many sources such as organizational data from ERP systems, databases, data lakes, sensors, collaborators and partners, public data, mobile apps, social media, and legacy data. It's structured and unstructured and in many different formats.

In-memory processing and highly efficient data pipelines make Spark an ideal analytics engine for data preparation, transformation, and manipulation for AI projects. Many steps are involved:

- Identifying and connecting to data sources
- Extracting the data
- Tagging the data
- Using tools and scripts to manipulate the data (for example, removing extraneous elements, breaking large images down to "tile" size so they will fit in GPU memory, and so on)

Business intelligence

To be more effective at their jobs, business analysts and end-users alike continually seek more up-to-date, accurate visibility into the full spectrum of an organization's data. Instead of being served in a canned report, ideally these queries should be carried out right away and also give the users the flexibility to change their inquiries, drill down on the data, and so on.

A BRIEF INTRODUCTION TO AI, ML, AND DL

AI is a term first coined in 1956 by John McCarthy and is defined as the capability of a machine to imitate intelligent human behavior. A machine can perform tasks that are characteristic of human intelligence. While this is a rather general description, it includes things like planning, understanding language, recognizing objects and sounds, learning, and problem solving.

ML is a form of AI that enables a system to *learn* or be *trained* from data rather than through explicit programming. Data scientists design ML models and train them by having them ingest large amounts of data that identifies the desired behaviors. The more data used to train the model, the more accurate the model. After training, when you provide a model with an input, you receive an output. For example, a predictive algorithm creates a predictive model. Then, when you provide the predictive model with data, you receive a prediction based on the data that trained the model. Machine learning is now essential for creating analytics models.

DL is an advanced method of ML, using more sophisticated models called neural networks. DL models are designed to emulate how the human brain works so computers can be trained to deal with abstractions and problems that are poorly defined. The average five-year-old child can easily recognize the difference between his teacher's face and the face of the crossing guard. In contrast, the computer has to do a lot of work to figure out who is who. Neural networks and DL are often used in image recognition, speech, and computer vision applications.

As the name suggests, the deep learning part of the model includes more decisions in many successive or iterative layers of decision making. DL models are much larger than ML models — usually an order of magnitude larger — and require more data than ML.

REMEMBER

Spark provides the requisite performance necessary to allow on-the-fly analysis — not only by algorithms but also by people. In a financial services investment scenario, this combination of computerized and human analysis of risk factors would result in faster and more accurate trading decisions.

Data integration

Most enterprises deploy an array of business systems. Some are on-premises, while others are cloud-based. Some are internally developed, while others are provided by vendors. Tying all this data together is complex and computationally intensive. This process is known as *extract, transform, and load* (ETL), and countless hours — and IT budget dollars — have been spent getting it to work and continually fine-tuning it.

Bringing Spark into the ETL loop means that you can quickly and easily combine data from multiple silos to get a better picture of the situation. Spark has impressive integration with numerous data storage technology suppliers and platforms and excels at processing these types of workloads.

Building on the financial services streaming and ML examples from earlier in this section, real-time data reported by point-of-sale devices can be combined with historical transaction information from relational databases to further refine the fraud detection process.

Understanding How Spark Processes Information

Spark's designers were quite cognizant of the basic fact that in-memory processing is always faster than interacting with hard disks (a ten times performance gain is a commonly accepted figure). This realization informed all their architectural and product design decisions, which yielded another tenfold improvement. Taken together, this made Spark solutions up to 100 times faster than earlier disk-based MapReduce applications. We begin the discussion in this section by talking about the approach Spark uses to gather, store, and manage information.

Resilient Distributed Datasets (RDDs) are in-memory structures meant to hold the information you want to load into and then process within Spark. They're labeled *resilient* because they're just that: They keep track of their history (for example, edits and deletions) so they can reconstruct themselves if there are any failures along the way. Meanwhile, they're called *distributed* because that's just what they are: dispersed across multiple nodes in the cluster to help safeguard data while also increasing efficiency via parallel processing.

When presented with a request from an application, a Spark solution uses the platform's internal features to create and then load data into an RDD. This information can come from just about any data source you can think of, including the Hadoop File System, Apache Cassandra, Amazon S3, Apache HBase, relational databases — you name it.

After the data in question has been placed into its RDD, Spark permits two primary types of operation: transformations and actions.

Transformations

Transformations include filtering, mapping, or otherwise manipulating the data, which automatically triggers the creation of a new RDD. The original RDD remains unchanged, no matter what happens to subsequent RDDs. This immutability lets Spark keep track of the alterations that were carried out during the transformation, and thereby makes it possible to restore matters back to normal should a failure or other problem crop up somewhere along the line — hence why "resilient" is part of its name.

Transformations aren't executed until the moment they're needed, such as by an action (see the next section). This behavior is known as *lazy evaluation* and helps improve performance by delaying computations until the instant they're required, rather than potentially conducting computations that eventually turned out to be unnecessary.

Actions

Actions are behaviors that interact with the data but don't change it, such as

- Counting
- Retrieving a specific element
- Aggregating

After an application requests an action, Spark evaluates the initial RDD and then creates subsequent RDD instances based on what it's being asked to do. When spawning these next-generation RDDs, Spark examines the cluster to find the most efficient place to put the new RDD.

How Spark Benefits the Entire Organization

Spark offers tangible and noteworthy business benefits for everyone in the enterprise, including, ultimately, your customers. In this section, we break down some of these advantages for three primary constituencies: end-users, data scientists/developers, and operations staff.

End-users

We're starting with end-users because after all, they're the ultimate audience for all Big Data initiatives — particularly Spark. (They also happen to sign the checks to pay for the projects.) Fortunately, they receive a good return on investment for their Spark expenditures.

First, they profit from all the new functionality offered by the Spark-based solution. These advances help them be more productive, serve customers better, and so on. Along with the applications themselves, to further interact with Spark's information, end-users are also able to use familiar tools and techniques, such as

- **Structured Query Language (SQL):** This interactive data query language has been popular for decades and is well understood by users, business analysts, and data scientists. Spark SQL offers native, built-in support for structured data stored within Spark as well as other repositories such as relational databases.
- **Business intelligence software:** Enterprises have invested significant sums in acquiring and rolling out powerful analytics tools from software heavyweights such as Microsoft, Oracle, SAP, and IBM. These are easily connected to Spark-processed information.
- **Spreadsheets:** Even these workhorses are able to interact with the results of your Spark solutions. In fact, this avenue is often the fastest way for end-users to gain additional insight.

Data scientists and application developers

Assigned the job of creating and distributing solutions built on Spark, this community (which had already been working with Hadoop) is the primary beneficiary of its amazing application development productivity enhancements. One knock against MapReduce (covered in more detail in Chapter 1) was that it was relatively difficult to gain proficiency and was thus out of reach for many software designers and developers. On the other hand, Spark uses a single — and simpler — programming model, along with supporting today's most popular Big Data development interfaces and languages, including

- Python
- R
- Scala
- Java

Spark's suite of unified, well-documented application programming interfaces (APIs) is a productivity booster for numerous use cases, such as the following:

- Streaming
- SQL
- AI, ML, and DL
- DataFrames
- Graph processing

If you want to learn more about what these APIs can do and the impact they can have for your business, be sure to check out "Core Spark Technology Components" just ahead in this chapter.

Web-based notebooks are interactive computational environments that make it possible to combine multiple building blocks into a single solution. These applications present a cohesive, visually attractive user interface consisting of

- Rich media
- Mathematics

- Code execution
- Rich text
- Plots

ON THE WEB

Anaconda and Jupyter are two of the more popular notebook development environments that can leverage Spark's data. You can find out more about both of these compelling application development technologies at `www.anaconda.com` and `www.jupyter.org`.

When you combine Spark's architecture, programming model, and development tools with its inherent speed, the outcome is a new breed of high-performance, more intuitive applications.

Another complaint about MapReduce was the somewhat awkward ways applications were developed, which led to long learning curves. MapReduce solutions can be a bit disjointed, requiring software developers to cobble together multiple Hadoop open-source ecosystem components. On the other hand, Spark is inherently able to blend multiple software development tactics when processing data by abstracting away much of the complexity associated with MapReduce.

This flexibility means that Spark can coordinate interactive, streaming, and batch workflows at the same time. It's also able to abstract other computational engines, which lets developers benefit from others' work without having to decipher deep technical details. The resulting data pipelines can be rich, complex, fast, and reusable, too.

TECHNICAL STUFF

Spark's data pipelines are logical objects that prepare data as well as conduct analysis on it. Yet they're much more straightforward to design, develop, and maintain than earlier approaches. In MapReduce, this would've required multiple jobs to be linked into a cohesive whole, with performance-sapping disk persistence taking place all along the way. Much of this complexity has been abstracted away in Spark, and because it's in memory, performance immediately is much faster.

REMEMBER

The productivity and application development speed means that developers and data scientists can use these shorter cycles to quickly build pipelines, get their results, learn from them, make adjustments, and start the cycle anew.

IT operations and administrators

Charged with keeping the organization's mission-critical applications running smoothly, IT operations and administrators benefit from Spark's performance: Its in-memory architecture and information processing techniques reduce reliance on hard drive I/O. The results are up to 100 times faster for in-memory processing, and 10 times faster for disk-based access when compared with MapReduce.

Along with its speed advantages, Spark solutions are highly scalable using distributed servers. Happily, Spark also protects existing investments by working well with existing Hadoop/MapReduce infrastructure, data, and its entire ecosystem.

Spark has a bright future as core IT infrastructure. A thriving open-source community, with rapidly growing numbers of commits, contributors, and Google searches, is bolstering it. In fact, Spark is now the Apache project with the most activity, with hundreds of contributors and companies helping to drive its advances.

REMEMBER

You can deploy Spark standalone; it doesn't require an existing Hadoop/MapReduce implementation.

Core Spark Technology Components

As the next step in the ongoing evolution of Big Data distributed processing technology, Spark is comprised of a collection of highly specialized components. These all work together in concert to help data scientists and developers quickly create impressively powerful applications, with end-users also capable of gaining access to Spark's information.

To give you a better idea of what's under the covers, this section introduces the major elements that constitute your Spark environment.

REMEMBER

We're describing "vanilla" open-source Spark here; we go into more detail about value-add technologies in Chapter 3.

Spark Core and RDD

As you might surmise from its name, you find the Spark Core at the center of the platform. It's responsible for creating and managing RDDs. It's also tasked with keeping things running smoothly across the cluster, such as scheduling tasks and coordinating activities.

A Spark environment is typically composed of many machines working together. Because the data is split across these servers, it's imperative to ensure that the computing resources are being used as effectively as possible. Schedulers act like coordinators for the cluster; they help start the containers or processes for each application's tasks.

Rather than dictating a specific scheduler, Spark's designers left that decision up to individual administrators. This means that you can choose from a collection of scheduler options:

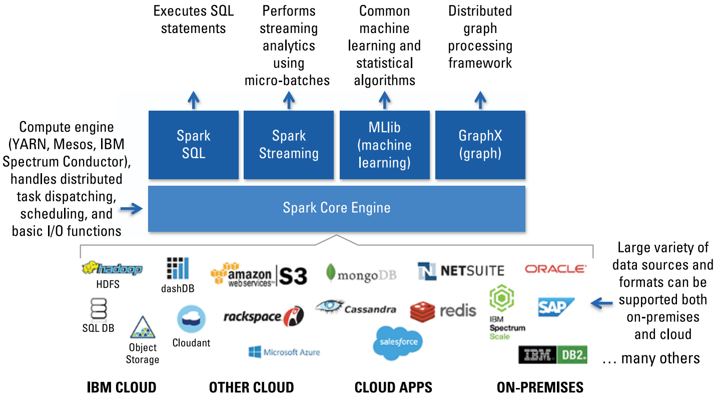
- **YARN:** Yet Another Resource Negotiator (YARN). Delivered in standard distributions, this scheduler is the open-source standard for Hadoop clusters.
- **Apache Mesos:** Another open-source cluster management system that's available as a separate download. Mesos is often used to manage data centers — something that YARN can't do.
- **IBM Spectrum Conductor:** This integrated solution — which we describe throughout Chapter 3 — incorporates Spark bolstered by numerous value-added software components and backed by IBM support and service organizations. One of these assets is a high-performance alternative to YARN that allows Spark to run on a scalable enterprise-grade framework.

Spark also comes with its own scheduler. However, the built-in scheduling system is rarely used in practice unless the cluster is solely dedicated to Spark.

Spark's libraries

Regardless of your chosen scheduling software, your Spark applications will incorporate behaviors from its Core, along with one or more of the libraries we describe in this section. In the world of

Spark, the whole is definitely greater than the sum of the parts: Stacking multiple libraries lets you build incredibly innovative Big Data solutions. Figure 2-1 does a nice job of placing Spark's libraries into its overall architecture.



Executes SQL statements
Performs streaming analytics using micro-batches
Common machine learning and statistical algorithms
Distributed graph processing framework
Spark SQL
Spark Streaming
MLlib (machine learning)
GraphX (graph)
Compute engine (YARN, Mesos, IBM Spectrum Conductor), handles distributed task dispatching, scheduling, and basic I/O functions
Spark Core Engine
HDFS
dashDB
amazon web services S3
mongoDB
NETSUITE
ORACLE
SQL DB
Object Storage
Cloudant
rackspace
Cassandra
redis
IBM Spectrum Scale
SAP
Microsoft Azure
salesforce
IBM DB2
Large variety of data sources and formats can be supported both on-premises and cloud
… many others
IBM CLOUD
OTHER CLOUD
CLOUD APPS
ON-PREMISES

FIGURE 2-1: The broad reach of Spark's libraries.

Spark streaming

With streaming data becoming one of the fastest growing and most significant contributors to Big Data, this library helps abstract away much of the complexity of creating an application based on this exciting class of information.

Along with industry-specific streams (for example, financial, medical monitors, and mobile devices) we describe in this chapter's "How Spark Benefits the Entire Organization" section, Spark streaming is also capable of creating fault-tolerant and scalable interactions with the sources of streaming data that are becoming so prevalent.

For example, Apache Kafka is a distributed streaming platform that's tailored for building all sorts of interesting solutions. Tracking activity on your website, aggregating metrics from diverse applications, and even replacing legacy message brokers are just a few instances of what it can do. On the other hand, Apache Flume is more intended for efficiently collecting, aggregating, and transporting the massive amounts of log data that are being generated today.

MLlib, Spark's scalable ML library

ML was previously feasible only for highly trained data scientists with sizeable hardware budgets. Spark expands these types of applications to a far wider audience. MLlib is a complete ML library with a diverse set of algorithms meant to standardize and streamline the job of developing an application that gains expertise over time. These algorithms cover an enormous amount of ground for data scientists, such as

- Regression
- Recommendation
- Decision trees, random forests, and gradient-boosted trees
- Model evaluation and hyper-parameter tuning
- Frequent items and sequential pattern mining
- Distributed linear algebra
- Statistics

ON THE WEB

You can view the full list of these algorithms at `spark.apache.org/mllib`.

Spark SQL

Spark SQL is a specialized Spark module that builds on the basic Spark RDD API. By supplying details about the structure of the data, along with the computations that are being performed on it, it's ideal for interacting with the kinds of relational data that is still so prevalent in enterprise applications.

Spark furnishes datasets and DataFrames to make things even easier — and more familiar — for software developers used to creating traditional applications. *Datasets* are distributed information collections that deliver all the advantages of RDDs on top of the enhancements provided by Spark's SQL processing engine. *DataFrames* go one step further: They represent the information contained in datasets as familiar relational database tables or the DataFrames found in the R programming language.

Finally, Open Database Connectivity (ODBC) and Java Database Connectivity (JDBC) are longtime, well-established standards for enabling applications and tools to work with these databases; unsurprisingly, Spark also offers ODBC and JDBC connections.

TECHNICAL STUFF

You can even use JDBC connectors as a conduit to the schema-free Drill SQL query engine, which further expands Spark's structured data reach. Furthermore, you can connect Spark with the open-source Hive data warehouse project, including using its SQL-like HiveQL syntax.

GraphX

Every time you interact with social networking platforms such as Facebook and LinkedIn, your experience is largely driven by what's contained in massive graph databases. These information repositories are designed to keep track of the web of relationships among the data that's placed in them. What makes graph databases so compelling — and fast — is that they're architected to store details about the connections *along* with the data itself.

REMEMBER

In this context, a *graph* isn't a visual object (like a chart or diagram). Instead, it's a representation of a network of relationships among people or data.

Graph computations are becoming a very popular way of interacting with this type of information. Spark's GraphX module, which supports many of the most popular graph algorithms, makes it relatively straightforward to include them in your Big Data applications.

SparkR

The R programming language is wildly popular with data scientists and statisticians. This library presents a clean and simple way for this audience to gain access to all of the Spark capabilities.

Storage systems

Hadoop/Spark has two major storage deployment models:

- The "share-nothing" storage model in which compute and storage resources both come from commercial storage-rich servers
- The shared storage model in which a shared storage system, like IBM Elastic Storage Server (ESS), provides storage service for the entire Hadoop/Spark cluster.

We mention shared storage in Chapter 1. Shared storage is becoming popular primarily because it separates storage from compute, enabling compute and storage to grow independently. This significantly helps to control server cluster sprawl and data center footprint. Adoption of containerized workloads is another reason why shared storage deployments are being considered. Most of the commercial shared storage offerings, such as ESS, allow different applications to access the same data using a range of access protocols like NFS, S3, HDFS, and so on. This enables you to build a single common data lake for all your Hadoop, Spark, and traditional applications.

Spark's designers did a great job of opening it up to work with numerous file systems — commercial and open source alike. This is in complete concordance with Spark's mission of interacting with the full range of enterprise data. First, Spark supports any Hadoop-compliant storage layer. *Hadoop-compliant* means that these other technologies support the high-level Hadoop API. Hadoop-compliant storage systems include

- The Hadoop File System (HDFS)
- IBM Spectrum Scale (upon which IBM ESS solutions are built)
- Redhat's GlusterFS
- Amazon's S3
- Microsoft's Azure Blob Storage (WASB)
- Alluxio (formerly Tachyon)

Spark doesn't stop there, however. Its architects designed it to connect to any data source while providing the same end-user experience via consistent APIs. Spark can use various distributed file systems as data sources, and also supports these popular repositories:

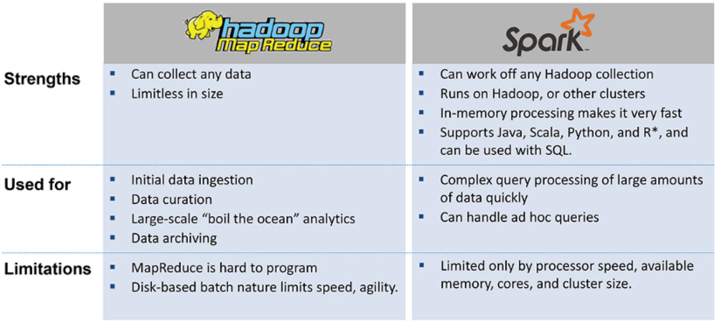
- HBase
- Cassandra
- Solr
- Redshift
- MongoDB
- Elastic
- MapR
- Couchbase

- Hazelcast
- Any JDBC source (Oracle, Sybase, MySQL, Postgres, and so on)

More support is on the way. If this list isn't comprehensive enough, you're free to roll your own data source and plug it into Spark.

Comparing Hadoop/MapReduce and Spark

Spark isn't a replacement for Hadoop or MapReduce. Instead, each technology has a role to play in an intelligently run IT organization. To help you figure out what that role should be, have a look at Figure 2-2.



	hadoop MapReduce	Spark
Strengths	▪ Can collect any data ▪ Limitless in size	▪ Can work off any Hadoop collection ▪ Runs on Hadoop, or other clusters ▪ In-memory processing makes it very fast ▪ Supports Java, Scala, Python, and R*, and can be used with SQL.
Used for	▪ Initial data ingestion ▪ Data curation ▪ Large-scale "boil the ocean" analytics ▪ Data archiving	▪ Complex query processing of large amounts of data quickly ▪ Can handle ad hoc queries
Limitations	▪ MapReduce is hard to program ▪ Disk-based batch nature limits speed, agility.	▪ Limited only by processor speed, available memory, cores, and cluster size.

FIGURE 2-2: Comparing Hadoop/MapReduce and Spark.

REMEMBER

If you want to see examples of Spark, Hadoop, and MapReduce all coexisting beautifully, be sure to read Chapter 4.

Spark's Open-Source Challenges

As is the case with most open-source-based technology advances, Spark has some obstacles that must be overcome. None of them should stop you from proceeding on your Spark journey; instead, they may highlight the value of considering a commercial Spark distribution.

REMEMBER

With time, money, and trained personnel always tight, these challenges are a big reason why many enterprises look to a trusted partner to provide their Big Data infrastructure, including the latest Spark implementations.

Dynamically managing resources

It's easy for a Spark environment to devolve into a collection of siloed servers, data, and so on. If not precisely managed, enterprise assets can be poorly utilized, which wastes time and money, and places undue burdens on administrators and operational staff.

Living up to Service Level Agreements (SLAs)

As Big Data — and Spark by extension — becomes more mainstream, SLAs are now a fact of life in most organizations. Open-source Spark can have some difficulty meeting contractual obligations related to

- Performance
- Scalability
- Security
- Resource utilization

Incorporate many other data sources and frameworks

Open-source Spark excels at interacting with data hosted in the Hadoop environment, but easily getting to other sources can be problematic. However, to get the most from the investment in Spark, it must also work with all of the other data generated and stored by the enterprise.

Coping with the speed of open-source updates

An open-source Big Data environment has enormous numbers of moving parts — even without Spark in the mix. Things are even more complex once Spark is deployed, particularly given the aforementioned dynamic nature of the Spark open-source community. Spark has numerous components that require administration, including

- SQL
- DataFrames
- Spark Streaming
- MLlib
- GraphX

Each of these components is continually being revised, and it's possible that application code changes will be necessary when moving between Spark versions. Keeping pace with the torrent of updates — while not violating SLAs or introducing security risks — can exceed the capabilities of the IT team.

Performance demands

Hardware that hosts Spark environments typically serves multiple tenants, each of which has different workloads that are subject to different SLAs. But because Spark applications are quite varied, it can be difficult for IT to prioritize appropriately.

Production support

Open-source Big Data platforms, including Spark, mandate that administrators and developers use community resources such as blogs, Q&A forums, and meetups to get answers to their questions and issues. It's hard to achieve the SLAs required for mission-critical applications without dedicated and accountable support resources.

IN THIS CHAPTER

- **Looking into performance considerations**
- **Managing resources**
- **Providing maximum operational flexibility**
- **Keeping things running smoothly**

Chapter 3
Deploying Spark in the Enterprise

Because they now underpin numerous production applications, Big Data technologies, including Apache Spark, warrant being treated as mission-critical information and resources by the enterprise. This task requires significant time and expertise, because most Big Data infrastructure offerings are composed of numerous open-source components, and downloading various elements and then assembling a stable, scalable, manageable environment isn't straightforward.

An integrated solution from a vendor provides a single point of contact to help get the Big Data infrastructure up and running — and to keep it running if you have problems. As such, it's become a popular way for enterprises to reap the benefits of Spark, while minimizing operational overhead.

IBM has made — and continues to make — enormous contributions and investments in open-source Spark. To complement these efforts, IBM also created an all-inclusive, turnkey commercial distribution that delivers all of Spark's advantages, while making it easier for enterprises that are basing new applications around it.

IBM Spectrum Conductor is a complete enterprise Spark solution that includes

- A Spark distribution from the Center for Open-Source Data and AI Technologies (CODAIT), formerly known as the Spark Technology Center
- A resource scheduler that's been proven in some of the world's most demanding customer environments
- Workload management
- Monitoring
- Alerting
- Reporting
- Diagnostics
- IBM services and support

All are managed from a single graphical user interface.

In this chapter, we introduce you to IBM Spectrum Conductor and explain how it addresses many of the challenges faced by open-source Spark deployments (covered in Chapter 2).

IBM Spectrum Conductor

IBM Spectrum Conductor is a secure and scalable platform for orchestrating multi-tenant enterprise Big Data and data science applications, including artificial intelligence (AI). It concurrently supports multiple application frameworks such as Spark and Anaconda, and distributed databases such as MongoDB and Cassandra, on a shared computing grid both on-premises and in the cloud.

With IBM Spectrum Conductor, organizations can

- Securely deploy, manage, and scale multiple application frameworks, including Spark
- Gain valuable insights with better access to sophisticated analytics applications
- Improve resource utilization and use local and cloud resources more efficiently

- Reduce administrative costs with automated application and resource management
- Accelerate time to results through efficient resource scheduling and sharing

In many organizations, multiple separate server clusters are deployed for each application framework and for other technical or business reasons. This process can lead to underutilization and the inability to share infrastructure across applications. IBM Spectrum Conductor removes these barriers, allowing all applications to see a single resource pool and providing flexible resource sharing to reduce cost and improve utilization.

IBM Spectrum Conductor is production proven in some of the world's largest supercomputers and enterprise grids and includes an end-to-end secure architecture with runtime isolation. Because of IBM Spectrum Conductor's advanced multi-tenant capabilities, multiple versions of Spark can be managed at the same time on the same cluster. This feature is important in production environments where applications may be tied to particular versions of Spark or Spark libraries.

IBM Spectrum Conductor also delivers dramatic performance gains for Spark workloads. In fact, in an audited benchmark IBM Spectrum Conductor performed between 30 percent and 224 percent faster than competing resource orchestrators on the same hardware environment.

TIP

For Spark and other workloads, IBM Spectrum Conductor provides a better infrastructure for scalable enterprise analytics because it

- Supports diverse applications and frameworks
- Allows users to share resources dynamically while maintaining a secure environment (runtime isolation while sharing resources dynamically)
- Delivers superior performance and scalability
- Allows compute resources to be scaled independent of storage
- Simplifies environments, helping reduce administrative costs

Performance Considerations

Getting answers to critical business questions is one of the primary reasons why enterprises select Spark. And faster answers can give businesses an important competitive advantage. Consequently, you should place special emphasis on making sure that your Spark implementation delivers results as quickly as possible.

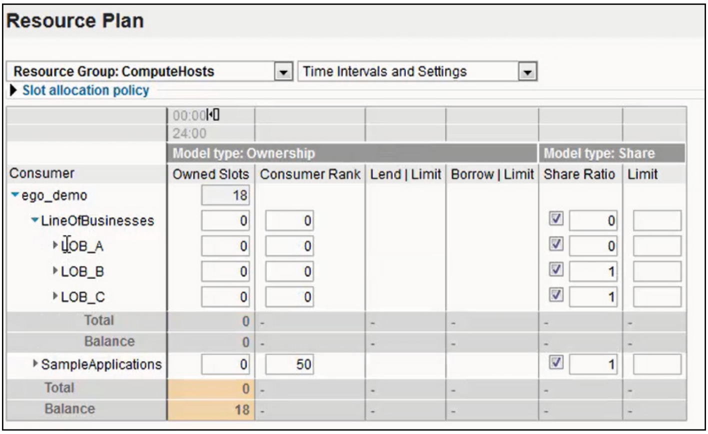
Spark's resource manager coordinates activities and scheduling tasks throughout the cluster. For "assemble-it-yourself" environments, this will commonly be handled by either the open-source Apache Hadoop YARN ("Yet Another Resource Negotiator") or the open-source Apache Mesos distributed systems kernel that offers abstraction capabilities for storage, memory, and computational capabilities across the network. Regardless of your selected resource manager, its performance and functionality will have a major impact on your Spark environment.

TIP

IBM Spectrum Conductor offers many advantages over YARN and Mesos, including sophisticated scheduling policies. It also eliminates many of the challenges of keeping your open-source environment up to date and configured properly. IBM Spectrum Conductor features include

- Time-based scheduling policies
- A centralized GUI for managing resource plans (see Figure 3-1)
- Dynamic updates of resource plans
- Pre-emption with sophisticated reclaim logic
- Proven multi-tenancy with service level agreement (SLA) management providing Quality of Service guarantees

An independently audited benchmark has demonstrated the speed advantages of IBM Spectrum Conductor: It was able to deliver 41 percent greater throughput than Hadoop YARN, and 58 percent greater throughput than Mesos. In addition, the IBM solution provided significantly superior consistency in how long any individual task took to complete, a particularly important consideration for interactive applications where a user is waiting for the result. The resource scheduler at the core of IBM Spectrum Conductor has had impressive adoption and throughput metrics, with scalability proven in some of the world's most demanding customer environments:



Resource Plan

Resource Group: ComputeHosts ▾ | Time Intervals and Settings ▾

▸ Slot allocation policy

	00:00 24:00					
	Model type: Ownership				Model type: Share	
Consumer	Owned Slots	Consumer Rank	Lend | Limit	Borrow | Limit	Share Ratio	Limit
▾ego_demo	18					
▾LineOfBusinesses	0	0			☑ 0	
▸LOB_A	0	0			☑ 0	
▸LOB_B	0	0			☑ 1	
▸LOB_C	0	0			☑ 1	
Total	0	-	-	-	-	-
Balance	0	-	-	-	-	-
▸SampleApplications	0	50			☑ 1	
Total	0	-	-	-	-	-
Balance	18	-	-	-	-	-

FIGURE 3-1: The resource manager GUI.

- 100,000 cores in production
- Thousands of nodes deployed
- Hundreds of applications on a single cluster
- More than one billion tasks carried out per day
- Multiple sessions running over one million tasks per session

Resource Management

It's quite common for Spark environments to store petabytes of data distributed across thousands of nodes. Acquiring, operating, and maintaining all of these resources can get quite costly. However, by sharing computing infrastructure across applications and business groups, resources that would otherwise be idle due to lack of local demand can be made available to meet other groups' current workload requirements. This puts idle resources to work running Spark jobs, speeding time to results and optimizing resource utilization. This approach improves service levels and reduces costs because the resulting greater utilization allows the same amount of work to be accomplished using fewer resources.

IBM Spectrum Conductor lets administrators use and manage thousands of computers as one. You can aggregate heterogeneous resources from multiple sites — on-premises as well as elsewhere. Furthermore, these resources may be physical or virtual machines.

IBM Spectrum Conductor allows users to run large numbers of Spark instances and applications, including different versions of Spark, on resources across the cluster. The product offers a central point of control that supports dynamic scheduling policies that can be changed in real time to reflect shifting business priorities.

Providing Maximum Operational Flexibility

One advantage of rolling out a pre-configured, turnkey Spark implementation such as IBM Spectrum Conductor is that it's been architected with a goal of making daily operations as streamlined as possible. This strategy is evident in the product's flexibility regarding other data sources and file systems, as well as how well it addresses security considerations.

Creating AI, ML, and DL environments

Building AI models involves a series of complex and time-consuming tasks that includes data ingestion and preparation, model training and tuning, scaling applications, and moving them into production. IBM Watson Machine Learning Accelerator, built on IBM Spectrum Conductor, enables you to build AI, machine learning (ML), and deep learning (DL) environments that provide an end-to-end workflow so data scientists can focus on training, tuning, and deploying models into production. This IBM Watson offering takes advantage of the IBM Spectrum Conductor distributed Spark environment to accelerate time to insights.

Integration with other data sources and frameworks

Although IBM Spectrum Conductor is fully compatible with the Hadoop Distributed File System (HDFS), it also provides its own storage option: IBM Spectrum Scale. This is a high-performance, shared-disk parallel file management solution that can easily plug into your Spark environment. IBM Spectrum Scale offers several advantages when compared with HDFS:

- It was designed to be Portable Operating System Interface (POSIX) compliant, which provides cross operating system compatibility.
- It's not burdened with a single point of failure, in contrast with how HDFS is dependent on availability of the NameNode process.
- It consumes a smaller footprint via a block size that's more appropriate for the transactional data likely to be found in a Spark environment. This avoids wasting space on your storage resources because large block sizes are a poor choice for the small files that are so common in this type of environment. This strategy minimizes unused storage and helps save money by more optimally matching block sizes with the data that will reside in them.

REMEMBER

As well as being true software-defined storage (SDS) that can be deployed on any appropriate commercial servers, IBM Spectrum Scale can also be deployed in a pre-assembled, comprehensive, high-performance, highly scalable shared storage solution called IBM Elastic Storage Server (ESS) that provides much more than just file system storage. The ESS building block architecture helps you start small and grow your storage capacity and performance as needed. ESS is becoming more and more popular to support Hadoop and Spark workloads and has been deployed in some of the largest AI environments, including the Summit supercomputer at Oak Ridge National Laboratory and Sierra at Lawrence Livermore, two of the most powerful supercomputers in the world.

Security considerations

Most organizations assign a specialized team of users to manage diverse system administrative duties. The permission levels within this group may vary, based on application, resource, security clearances, or other factors. IBM Spectrum Conductor provides role-based access control (RBAC) to help protect Spark environments while conforming to your enterprise's administrative requirements.

REMEMBER

RBAC enables organizations to assign permissions for performing specific tasks according to the role of each user. This minimizes the opportunity for any individual to cause accidental or malicious damage to the system. RBAC is also fully compatible with the high-availability recovery capabilities of IBM Spectrum Conductor.

Keeping Things Running Smoothly

Deploying your Spark infrastructure is only the beginning of your Big Data journey. It's vital to select a vendor that's committed to open-source Spark's continual evolution and can provide the training and consulting services to accelerate your team's productivity.

Spark — like all open-source Big Data software — is continually benefiting from product improvements, new capabilities, and integration with additional Apache projects. IBM Spectrum Conductor can support multiple tenant environments, which means that administrators can simultaneously run multiple instances and versions of Spark. This fine-grained control lets administrators match — by application — the exact Spark versions they wish to support. There's no need to run separate clusters for each instance, and they can migrate workloads from older to newer versions at their own pace.

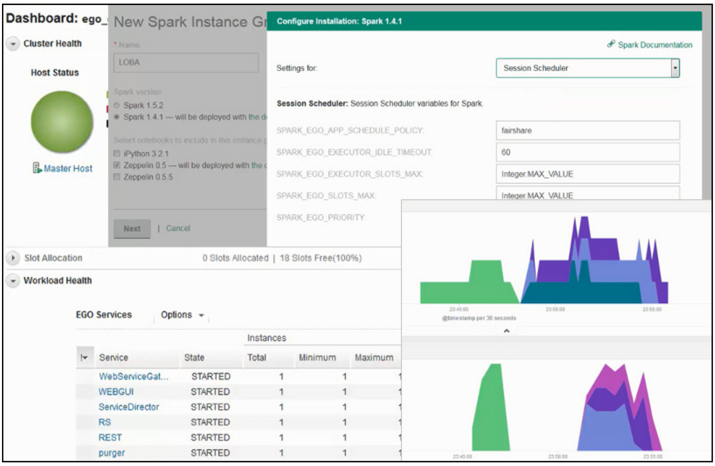
IBM has made an impressive investment in both Big Data and Spark, via

- Establishing the Spark Technology Center, now renamed CODAIT, the Center for Open-Source Data & AI Technologies
- Contributing an internally designed ML library to open source
- Dedicating more than 3,000 engineers and researchers to Spark projects
- Creating a robust professional services organization to help get customers up to speed on Big Data, Hadoop, MapReduce, and Spark
- Undertaking to educate more than one million data scientists and engineers on Spark

ON THE WEB

Visit `codait.org` to see how the Center for Open-Source Data & AI Technologies (founded as the Spark Technology Center and spearheaded by IBM) makes open-source and AI technologies accessible to the practitioners that understand real-world problems, so that they themselves can develop AI solutions that solve these problems.

IBM has made Spark the cornerstone of its Big Data analytics and AI strategies, incorporating Spark into its numerous software solutions, both on-premises and in the cloud. IBM Spectrum Conductor offers an extensive collection of integrated administrative tools, backed by IBM services and support, meant to keep the total cost of ownership low. These tools are shown in Figure 3-2.



* Name: LOBA

Spark version:

○ Spark 1.5.2

● Spark 1.4.1 — will be deployed with the d

Select notebooks to include in this instance g

□ iPython 3.2.1

☑ Zeppelin 0.5 — will be deployed with the c

□ Zeppelin 0.5.5

Settings for: Session Scheduler

Session Scheduler: Session Scheduler variables for Spark.

SPARK_EGO_APP_SCHEDULE_POLICY: fairshare

SPARK_EGO_EXECUTOR_IDLE_TIMEOUT: 60

SPARK_EGO_EXECUTOR_SLOTS_MAX: Integer.MAX_VALUE

SPARK_EGO_SLOTS_MAX: Integer.MAX_VALUE

SPARK_EGO_PRIORITY:

0 Slots Allocated | 18 Slots Free(100%)

		Instances		
Service	State	Total	Minimum	Maximum
WebServiceGat...	STARTED	1	1	1
WEBGUI	STARTED	1	1	1
ServiceDirector	STARTED	1	1	1
RS	STARTED	1	1	1
REST	STARTED	1	1	1
purger	STARTED	1	1	1

FIGURE 3-2: The collection of administrative tools in IBM Spectrum Conductor.

IN THIS CHAPTER

- » Selecting the right tool for the job
- » Seeing Big Data in real-world examples

Chapter 4
How Spark, Hadoop, and MapReduce Work Together

Whenever a new, hot technology (and Apache Spark's blazing adoption certainly puts it in that category) hits the market, it's easy to mistakenly assume that everything that came before it has suddenly been rendered obsolete. This is also the case with Spark — some people believe that it's a replacement for MapReduce and that all existing MapReduce-based applications must immediately be thrown out and rewritten in Spark.

This idea has been — in part — driven by the demonstrable performance, application development, and flexibility advantages of Spark in comparison with MapReduce. Even with these improvements in mind, don't lose sight that Spark isn't a replacement for Hadoop, and MapReduce remains a valid software development architecture. In actuality, many organizations are discovering that Spark is quite capable of extending the reach of existing Hadoop implementations and delivering additional value from these investments.

After all, Spark can happily run on top of a Hadoop instance, utilizing the YARN cluster manager as well as Hadoop's storage layer, such as Hadoop Distributed File System (HDFS) or HBase. On the other hand, Spark has no intrinsic need for a Hadoop instance: It can use other cluster managers (Mesos and IBM Spectrum Conductor) and different storage layers, such as Cassandra, Amazon S3, and IBM Spectrum Scale.

This independence means that a comprehensive Big Data strategy will commonly include a blend of Hadoop, MapReduce, and Spark. To show you what we mean, this chapter offers numerous examples of Big Data scenarios that feature these technologies either alone or working together cooperatively.

Choosing the Optimal Big Data Solution

The entire Big Data universe of products and design standards is continually improving. This holds true for Hadoop as well: the YARN resource manager was a major enhancement in second generation Hadoop. MapReduce continues to be tested in production and is delivering tangible value every day. The result is that enterprises will be working with Hadoop, MapReduce, and Spark for a long time to come.

When evaluating the specific approach to addressing your Big Data needs, it's not an either/or question, but instead it's a matter of choosing the right tool for the job at hand. In fact, in many circumstances you'll want to blend a MapReduce solution with one developed in Spark.

TIP

A well-rounded toolbox includes diverse implements like a hammer, screwdriver, and wrench. A well-rounded Big Data toolbox includes technologies like Hadoop, MapReduce, and Spark.

While every situation is different, this section gives you some general guidelines that you can use to identify the optimal approach for your Big Data application.

When to use Hadoop/MapReduce

For the past several years, the Hadoop/MapReduce tandem has been a popular, effective strategy for developing Big Data applications. It remains a good choice in any of the following conditions:

- **Batch processing:** Many Big Data solutions are perfectly fine delivering results using this approach.
- **Homogeneous data storage:** If you're creating an application that will exclusively employ HDFS, Hadoop and MapReduce may be the only technology you need.
- **Developer experience:** If your team has attained expertise in Hadoop and MapReduce, there's no reason to throw that away as you also incorporate Spark into your environment.
- **Existing applications in production:** There's no need to rip and replace what you've already deployed; Spark capabilities can be blended into new versions of current solutions.
- **Existing data lake:** If you've already built and collected a substantial amount of data in your HDFS data lake, you don't need to move it to another data store. Existing Hadoop and the latest Spark platforms can access this data directly without the need for any migration.

When to use Spark

There are many good reasons why Spark has become so popular. Consider a Spark-based solution if any of the following characteristics describe your requirements:

- **Developing real-time applications:** By leveraging in-memory processing efficiencies, Spark has the potential to return results much more quickly than MapReduce. This is important in all applications, but especially for solutions that will provide guidance in real time.
- **Working with heterogeneous data:** The most compelling Big Data solutions blend information that's been stored in diverse silos that are built on other architectures. If you're developing a solution that crosses data boundaries, Spark will be a good choice. And remember that it's also adept at working with data stored in HDFS.
- **Less experienced software developers:** The learning curve for creating Spark solutions is less intimidating than for MapReduce. In essence, Spark offers a simpler application development methodology than MapReduce. The upshot of this ease-of-use is faster delivery of more reliable code.

- **Consuming streaming data:** Sensors, mobile devices, and trading systems — to name just a few sources — are generating colossal volumes of streaming data. Spark is particularly adept at coping with these torrents and making it easy for developers to create innovative solutions.
- **Employing artificial intelligence (AI), machine learning (ML), and deep learning (DL):** Previously restricted to only the most educated data scientists working with massive investments, Spark brings ML and DL capabilities within reach of a much larger audience.
- **Seeking a smaller codebase:** Spark's extensive language and API support directly translates to less code to write, which means that there's less code to maintain.

When to deploy both

It's most effective to avoid binary thinking when it comes to your Big Data application development strategy: In many cases, you'll find that the ideal solution to your Big Data challenges will incorporate Hadoop/MapReduce *and* Spark. This is especially true if you've already developed a core system built on MapReduce and want to expand its functionality while still preserving your initial investment.

REMEMBER

Spark adds value to an existing Hadoop/MapReduce application, notably when your goal is to enhance it by incorporating additional features, such as the following:

- **Structured Query Language (SQL) access to data:** Spark offers extensive SQL capabilities, which opens up access to your data to all sorts of tools and supporting applications.
- **Native streaming:** Streaming data is a fact of life within most organizations. Spark provides built-in streaming support, which is a simpler strategy than needing to pair Apache Storm's real-time data stream processing with Hadoop.
- **Built-in graph processing:** Apache Spark offers a more elegant approach to this kind of computation, rather than needing to graft Apache's open-source solutions such as Neo4J or Giraph alongside Hadoop and MapReduce.

Big Data in Action

If you've been reading this chapter up to this point, you can identify the most rational Big Data infrastructure for your initiatives. So in this section, we share a collection of high-level, industry-specific examples of applying modern Big Data technology to help correct problems and encourage innovation. To demonstrate how flexible these solutions are, we describe scenarios that include "greenfield" (for example, net-new) Spark, integrated Spark and MapReduce, and replacements for existing MapReduce applications.

REMEMBER

There are a limitless number of potential usages for these exciting Big Data solutions — far more than space permits us to itemize. Instead, our goal here is to present applications that might pique your interest, whet your appetite, and encourage you to visualize solutions for your own enterprise. With AI, ML, and DL beginning to take hold as standard business tools, Spark is fast becoming the best foundation on which to base these new business processes as well.

Financial services

This is a highly dynamic industry, which sees a continual flow of new products and services designed to create additional revenue streams. These firms also confront a rigorous regulatory environment with non-compliance often leading to expensive civil and even criminal penalties. Additional regulations are on the horizon, so these challenges are sure to continue expanding.

Fraud detection

During the course of a given business day, a bank or brokerage will be inundated with streaming data consisting of quotes, trades, settlements, and so on. It's likely that fraudulent activities will be hidden in this torrent of raw data.

Spark's in-memory processing power and aptitude for working with streaming data make it feasible to inspect activities in real time, compare them with historical models and other proprietary fraud indicators, and then flag suspicious happenings. Based on what's discovered, the firm can reject, delay, or require manual inspection of the transactions in question.

ATM maintenance optimization

ATMs are a principal point of customer contact for retail financial services firms, and every large organization's network is made up of thousands of individual machines. Service outages, errors, or other interruptions reflect very badly on the institution, but it can be difficult to identify which ATMs are about to experience a failure.

The tandem of Hadoop and Spark can deliver a solution that helps prevent downtime before it occurs. Hadoop and MapReduce can be tasked with analyzing reams of historical maintenance data in batch mode, while Spark can be focused on pairing the results of this analysis with the real-time Apache MLlib library to incorporate up-to-the-minute customer complaints and ATM diagnostic messages. By using the results of the linkage between these two previously isolated systems, maintenance personnel can be guided to service machines that are on the verge of a breakdown.

Telecommunications

Telecommunications firms must gracefully balance the heavy infrastructure expenditures necessary to support the ever-multiplying number of mobile devices and sensors while coping with strong competitive and pricing pressures. Maintaining high quality-of-service standards mandates continually expanding and enhancing their networks. Shrewdly applied Big Data solutions can help them do a better job of allocating their investment funds.

Cellphone tower installation

Optimally placing a cellphone tower is a costly, data-intensive endeavor. In one case, an existing Hadoop and MapReduce application had been designed to analyze — in a series of batch calculations — signal strength reports from field sensors to help pinpoint where a new tower should be located. Users were happy with the results, but they also felt that faster delivery would result in better and more accurate installations.

The acquisition of a former competitor — a very common event in this industry — spurred a wholesale reevaluation of the enterprise's Big Data system architecture. Two major factors led to the decision to redevelop the solution as a real-time application using Spark:

- The acquired organization already maintained large amounts of signal strength information on non-HDFS storage technologies, including Amazon S3. This made it difficult for the existing Hadoop/MapReduce solution to gain access to this vital data.
- The combined company faced a scarcity of trained MapReduce developers. This lack of knowledge guaranteed a long learning curve for maintaining the legacy Hadoop/MapReduce solution.

Network outage predictive analytics

Because quality-of-service (QoS) is one of the most consequential factors in maintaining customer satisfaction, QoS is relentlessly under pressure from the addition of new devices to the network. At the same time, the volume of raw and diagnostic data generated by all devices is expanding geometrically.

A fresh Spark solution can be developed to read streaming data from network devices — in real time — and watch for issues that have previously predicted outages and other problems. ML has a particularly important role to play to examine current and historical patterns to identify where performance obstacles, outages, and other QoS degradation can occur; graph data is also useful in this type of introspection. The culmination of these efforts ensures a smarter, self-healing network, happier customers, and better allocation of scarce resources.

Retail

The rise of the Internet — spawning price-comparison-shopping engines and customer-review sites — has made it very difficult for individual retailers to achieve differentiation. This is a brutally competitive business: Customers are price-sensitive, and their loyalty is transient at best. The resultant microscopic profit margins limit budgets for IT infrastructure, software development, and support services. Fortunately, intelligently applied Big Data offers a way for retailers to stand out.

E-commerce up-sell/cross-sell

The checkout process is often the best time to derive additional revenue from an online customer. Making this possible requires an application that rapidly evaluates inventory, margin, and other information to calculate a one-time offer to be presented prior to completing the sale. Simultaneously, it's vital that the promotion keeps the transaction profitable while still delivering the offer in real time. With this many factors to weigh, and given the need for developer efficiency and productivity, Spark would be a logical choice for building this solution.

Recommendation application

It's easy for consumers, especially those buying online, to get overwhelmed by the sheer number of products available for purchase. Reviews and suggestions from friends and family go a long way toward helping potential customers make a selection.

For example, a batch-oriented Big Data solution can utilize MapReduce to analyze historical purchase records for all customers and then combine the results of that research with Spark's graph computational capabilities to create periodic, highly targeted promotional emails based on what a person's friends are buying.

Energy

Energy providers must make hefty upfront capital investments before they can identify a new source and then produce revenue. This means that there are huge risks (and enormous potential rewards) from exploration. At the same time, accidents and other breakdowns can result in devastating environmental and financial consequences.

Wisely exploiting Big Data can give these energy firms the guidance they need when seeking new supplies and can warn them of potential problems.

Pipeline failure predictive analytics

Pipelines can fail — sometimes spectacularly — and cause enormous damage. MapReduce-based solutions excel at wading through the masses of historical maintenance and repair data to detect patterns that indicate an incipient rupture. However, unless this data can be presented to field engineers in near real time, warnings may arrive too late.

TIP

This is a great candidate for a Spark upgrade. The revised solution — which can either leave the existing MapReduce work in place, or revamp it to Spark — will incorporate telemetry from monitoring sensors alongside the existing maintenance details. The results of these predictive analytics can be fed to engineers in the field via mobile devices, in real time. This can go a long way toward eliminating failures before they even occur.

Exploration resource allocation system

Energy surveys are heavily dependent on the right equipment being available at the right time; delays or misallocations can seriously mar profitability. A blended Big Data solution can address this need by intelligently analyzing masses of data to allocate these assets.

MapReduce is assigned the job of evaluating historical data — in batch mode — to help predict where equipment will be needed, while Spark probes real-time inventory details that are maintained in mainline enterprise relational databases.

Life sciences

The life sciences industry features some of the most sophisticated technologies developed by firms willing to evaluate fresh approaches to doing things better. This combination isn't surprising given the life-and-death nature of this business and the resulting huge financial risk/reward ratios that are so prevalent.

For example, pharmaceutical firms engaged in new drug research must make enormous investments in their pipeline: The outcome can make or break the company. They also confront strict regulatory requirements, with violators at risk of being shut down if not compliant.

While new drug development is an ongoing dramatic story, equally interesting events are taking place on the medical devices front. There are a tremendous number of new in-home medical monitors and sensors being deployed. Many of these types of data-intensive devices are even capable of alerting medical personnel to potentially life-threatening crises.

Big Data technologies can play a central role, regardless of the specific life sciences use case.

In-home patient monitoring

With the rise of smartphones, many innovative in-home medical devices are capturing metrics such as blood pressure, blood sugar, epileptic incidents, and numerous other potentially serious events. Although they serve diverse purposes, one thing these devices have in common is that they can each generate hundreds of events per hour. The vast majority of them are unremarkable, but certain ones are potentially deadly.

A MapReduce solution would be a great start toward churning through the vast logs created by these sensors. It could report on potentially serious happenings and alert appropriate personnel. However, MapReduce's batch processing nature would make this system suboptimal.

A better strategy would be to use Spark to monitor the same events, but in real time as they arrive directly from the devices. If a serious situation arises, emergency personnel can be instantaneously alerted.

Drug interaction alert system

Partially driven by human error and insufficient oversight, drug interactions sicken and even kill untold numbers of patients each year. Applications that can monitor a patient's specific medication profile and warn of potential cross-drug issues would go a long way toward reducing these unnecessary events.

Rather than waiting for a dangerous interaction to occur, the best place for this type of solution would be the pharmacy — before the patient even starts treatment. The application would evaluate new prescriptions and associate them with the medications already being taken by the patient. This highly computationally intense solution would then compare the results against a highly detailed database of side effects and interactions, all in real time.

Spark combines the ease-of-development and runtime speed necessary to deliver this kind of solution:

- If it identifies non-serious interactions, the pharmacist could be directed to discuss the situation with the patient.
- If the calculations uncover a potentially life-threatening interaction, the pharmacy could hold the prescription until the patient's doctor has been notified.

This example, and the others in this "Big Data in Action" section, are technologies that work alone or in concert. You're limited only by your imagination — and the data you're able to collect. Be sure to check out Chapter 5 for some best practices as you begin your Spark journey.

IN THIS CHAPTER

- **Setting the foundation for a successful Spark initiative**
- **Striving for flexibility**
- **Delivering the fastest performance while safeguarding your data**

Chapter 5 Ten Tips for an Excellent Spark Experience

If you've read the book up to this point, you've seen that deploying Apache Spark has the potential to transform the way your organization does business. In this chapter, we humbly provide a few suggestions to help turn the hypothetical into reality.

These guidelines range from pointing out how crucial it is to assemble a multifaceted team and select a robust Spark implementation to the value of concentrating on operational and administrative efficiency. Aside from common sense staffing and ancillary personnel-related recommendations, what should become readily apparent from our advice is the outsized impact your preferred Spark technology will have on the overall trajectory of your efforts.

Get the Entire Organization Involved

The most successful Big Data and Spark ventures have a number of common attributes. A diverse, well-rounded project team is one of them. Be sure to include

- End-users
- Line-of-business leaders

- Data architects
- Data scientists
- Application architects
- Software developers
- Operations staff

Chapter 4 presents a number of use cases that would certainly require full input from multiple parts of the organization to succeed.

WARNING

Don’t fall into the trap of “accidentally” excluding representatives from key constituencies from the project because you think they’ll just slow things down. If you keep them out of the loop until the end, you run the risk of much further delays. Correcting deficiencies is always more expensive just prior to rollout.

Select a Solid Implementation, but Remain Flexible

As we discuss throughout Chapter 3, choosing your Spark infrastructure is an important decision. A packaged offering from a reputable software vendor can make everything much easier, but you should balance identifying the right supplier with keeping your options open should things change down the road.

Spark — and the entire Big Data open-source universe for that matter — is continually evolving. Because you’ll be basing your new applications on these technologies, it’s vital that you select a supplier with a strong commitment to the open-source community. This can smooth out the rough edges that are inherent in open-source-based initiatives, particularly in projects such as Spark that are advancing so rapidly.

REMEMBER

Your chosen Spark environment must be capable of running multiple versions as well as smoothly interacting with the other components that are already operating in your technology portfolio. It must also be open and flexible enough to incorporate new and evolving technologies like deep learning frameworks such as TensorFlow and PyTorch.

Profit from External Expertise

Spark makes building Big Data solutions simpler, particularly compared with the original Hadoop and MapReduce approach. Nevertheless, it still can be daunting for an organization to launch its first Spark effort. Outsiders with proven success can provide guidance at all phases of a project, and can make it easier to get things off the ground.

TIP

Seek out consultants that help your organization get up to speed but also enthusiastically transfer knowledge to you. Your goal should be to take over maintenance after you go live and then develop new solutions internally.

Pay Attention to Performance

Delivering instant insights to users is one of the biggest selling points for Spark. But your choice of Spark infrastructure — including the hardware it runs on — plays a major role in how quickly everything operates.

The open-source Apache YARN and Apache Mesos resource managers can be bottlenecks during Spark processing (covered more in Chapter 2). In contrast, Chapter 3 tells you about the IBM Spectrum Conductor alternative and how it yields significant performance advantages. Because these components are at the heart of a Spark environment, efficiencies here can add up.

TIP

In terms of hardware, be sure to select servers that are optimized and tuned for the types of workloads found in Spark environments. Critical considerations will include

- Memory bandwidth
- Cache size
- Processor performance
- Availability and support for accelerators like GPUs and FPGAs

Leverage All Existing Assets before Buying New

Plenty of Big Data initiatives are marked by extensive upfront hardware expenditures to store and process all the information that will drive the new solutions. Because money still doesn't grow on trees, keeping costs under control is important.

TIP

Select technology that uses strategies such as dynamic resource allocation to ensure that your Spark environment is fully utilizing all your current processing capabilities. This can translate to big savings on hardware and all ancillary costs. In fact, you may discover that you can get by with what you already have — it may not be necessary to purchase any new resources.

Seek Diverse Storage Management

We introduce you to the Hadoop Distributed File System (HDFS) in Chapter 1. It remains the most popular repository for the data that drives Hadoop applications. However, much of the structured and unstructured data that power Spark solutions is originally found in other locations.

Given this reality, make sure that your Spark implementation will let you interact with multiple types of information repositories. To derive maximum value from your entire portfolio, make sure that you select Spark technology that can interact with other Portable Operating System (POSIX) compliant file systems.

IBM Spectrum Scale represents a great example of a proven, enterprise-grade yet cost-effective alternative to HDFS. It offers a variety of data storage selections, ranging from physical disk to pure in-memory to combinations of both technologies; you can even use tape if you like! Regardless of your chosen storage topography, you're then free to consume Spectrum Scale–hosted data for your MapReduce and Spark applications.

Protect Your Data

The information stored in your Big Data environment is a tempting target for an attacker. Thus, it's essential to implement comprehensive yet easily maintained security processes, ideally powered by the built-in capabilities of your Spark technology. For example, because different users often manage varying activities, role-based access control can minimize the risk of a single user deliberately or inadvertently causing damage to the entire system.

Additionally, preserving cross-instance isolation means that users who are authorized to view one of your Spark solutions will be prevented from seeing the data meant for other Spark applications.

Handle Multiple Concurrent Spark Versions

In Chapter 2, we present Spark, including telling you about its heritage as an open-source initiative. Spark continues to be a moving target with rapid updates. With this pace of change, make sure that you can readily bring on board new versions of Spark and run different versions of Spark simultaneously in a shared environment. This gives you the freedom to match the Spark version with the specific application and to roll out new versions at your discretion. You can then migrate your applications at your leisure once you attain confidence in the newer version.

Be Ready to Run Multiple Concurrent Spark Instances

Cost control is a hallmark of a well-managed Spark implementation. Buying superfluous hardware is one way that expenses can quickly get out of hand, and frankly unnecessary in today's modern data centers where intelligent software can provide guidance on acquiring the optimal amount of resources for the job at hand.

Spark workloads tend to vary: Application A may see peak activities in the morning, while application B gets busy in the afternoon. Likewise, some applications tend to be busier on different days of the week or even different times of the month. Running instances simultaneously across shared infrastructure helps balance things out, and takes advantage of resources that would otherwise be idle.

TIP

Try to refrain from the outmoded approach of deploying multiple Spark implementations across a constellation of isolated silos. Instead, seek out infrastructure, such as shared servers and storage, that enables sharing across the organization. This will go a long way toward fine-tuning your hardware investments.

Offer Enterprise-Level Support to Your Spark Administrators

Few organizations have idle system administrators just sitting around waiting to take on new responsibilities. In reality, they're usually overloaded trying to look after what's already in place. To make their lives simpler, provide them with tools that will boost their efficiency and streamline their workloads. You can attain these goals in a Spark environment via capabilities such as graphical user interface (GUI) administrative aids and a single user interface for configuring and managing Spark workloads.

TIP

Select technology that delivers a complete Spark solution, including options for storage, from a provider who's making continued investments in Spark technology and has demonstrated experience delivering enterprise-level services and support.

Notes

Notes

Notes

Notes

Discover the exciting open-source initiative

Spark represents a new approach to designing, developing, and distributing solutions capable of processing the massive amounts of Big Data that enterprises like yours are accumulating each day. Big Data is pervasive and the rise of specialized, first-generation information analytics strategies and technologies such as MapReduce and Apache Hadoop may no longer be your best option. Newer technologies such as Spark protect your existing customer investments and are the path forward to a competitive advantage.

Inside…

- See how Spark works
- Know what Spark means for you
- Discover IBM Spectrum Conductor
- Browse industry-specific use cases
- Fit MapReduce, Hadoop, and Spark together

IBM

Robert D. Schneider is a Silicon Valley–based technology consultant and author. He writes about topics such as Big Data, cloud, software architecture, and APIs. http://rdschneider.com. **Jeff Karmiol** is an Offering Manager at IBM Spectrum Computing focusing on Big Data, Grid, HPC, and cloud computing.

ISBN: 978-1-119-57697-6
Part #: DCM03021USEN-01
Not For Resale

9 781119 576976

for dummies®
A Wiley Brand