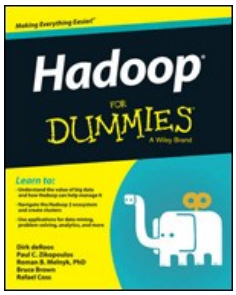

Hadoop for Dummies

by Dirk deRoos et al.

Chapter 19: Ten Reasons to Adopt Hadoop

In This Chapter

- The price is right
- The (open source) community is there
- Companies love Hadoop — they really do
- Scalability isn't a problem
- Hadoop plays nicely with traditional tools
- Hadoop has broad tastes in data types
- Hadoop can face (almost) any analytical challenge
- Full data sets are the norm (no sampling)
- Hardware's ability to deal with Hadoop improves every day
- Flexible workloads? No problem!

Hadoop is a powerful and flexible platform for large-scale data analysis. This statement alone is a compelling reason to consider using Hadoop for your analytics projects, especially for solutions involving the use cases we describe in Chapter 2. To help further tip the scales, this chapter lists ten compelling reasons to deploy Hadoop as part of your big data solution.

REMEMBER Though we're excited about Hadoop and we want to promote its adoption, in some cases other software solutions are more appropriate. For example, replacing an online transaction processing database system with Hadoop is almost never a good idea. Architecture decisions come down to requirements, which may include performance thresholds, fine-grained access control, data column masking, or a host of other data governance-related considerations. If your project's criteria align with the characteristics and capabilities of Hadoop that we describe throughout this book, the reasons in this chapter apply to you!

Hadoop is Relatively Inexpensive

At the time we wrote this book, the cost per terabyte to implement a Hadoop cluster was cheaper than the per-terabyte cost to set up a tape backup system. Granted, a Hadoop system costs more to operate, because the disk drives holding the data are all online and powered, unlike tape drives. But this interesting metric still shows the tremendous potential value of an investment in Hadoop.

The primary reason Hadoop is inexpensive is its reliance on commodity hardware. Traditional solutions in enterprise data management depend on expensive resources to ensure high availability and fast performance. Storage is an excellent example, because the typical relational data warehouse uses expensive SAS disk drives arranged in RAID arrays. By contrast, Hadoop was designed to run with inexpensive SATA drives, where availability is provided by replicating individual data blocks three times. The assumption that all hardware fails is a core principle for Hadoop, so it was designed to run on less expensive hardware.

REMEMBER You may look at this section's heading and say, "Of course Hadoop is inexpensive. Open source software is free!" Well, if you're a hobbyist programmer, then yes, you may download and play with Hadoop for free. But if you're an enterprise that's deploying Hadoop in places where it's delivering business value, you can't get by with a hobbyist mentality; you need an enterprise-ready software license and a support contract to boot. The bottom line is that although a respectable Hadoop distribution will cost you license and support fees, these expenses for Hadoop are far lower than for large relational database technologies.

On the three-legged stool of IT solution costs, we've covered only the hardware and software legs. One other critical ingredient is services, or *skills*. When Hadoop was younger, fewer people had Hadoop skills, so folks were seeing major shortages of trained personnel. Also, Hadoop was a more difficult platform to use a few years ago, which made the skills shortage even more acute. The open source community has made progress in improving Hadoop's usability — most significantly with Hive. People with SQL skills — a large contingent of IT professionals — can now query data using a SQL dialect that's becoming increasingly compatible with SQL-92, which reduces the dependency on, for example, MapReduce skills.

Hadoop Has an Active Open Source Community

Whenever an organization invests in a software package, a key consideration is the long-term relevance of the software it bought. No business wants to purchase software licenses and build specific skills around technologies that will be either obsolete or irrelevant in the coming months and years.

In that regard, you don't need to worry about Hadoop. The Apache Hadoop project is on the path of long-term adoption and relevance. Its key projects have dozens of *committers* (see below) and hundreds of developers contributing code. Though a few of these people are academics or hobbyists, the majority of them are paid by enterprise software companies to help grow the Hadoop platform.

TECHNICAL STUFF Since the Hadoop community projects are part of the Apache Software Foundation (ASF), here's a bit of background. The ASF provides the key ingredients for a community to manage the development and release of a software project. For example, the ASF

features a governance structure to ensure an open and democratic approach to evolving the project; an issue tracking framework to manage bugs and new feature development; and a software license that encourages easy adoption and future innovation.

Anyone can be a contributor for an Apache project. In fact, projects with large numbers of developers representing diverse interests contributing code are considered the healthiest. To ensure code integrity and that development is being done according to the project's democratically agreed upon direction, it's only the project's committers that have write access to the project's code repository. A *committer* is a special role that the Project Management Committee (PMC) votes to assign to contributors who have shown both deep expertise and personal investment. The PMC itself is made up of committers who are effectively the stewards of the Apache project, voting on its overall direction, major features, and releases.

In addition to the large numbers of individual people contributing to Hadoop projects, a significant number of software companies are actively investing top development talent in growing the Apache Hadoop ecosystem, including larger IT companies such as IBM, Intel, Microsoft, and Yahoo! but also a multitude of smaller and younger companies — most notably, Cloudera, Hortonworks, Facebook, and MapR.

Along with the number of committers, contributors, and companies funding open source development work, the number of recently filed bug reports is an excellent indicator of technology uptake. All reasonably sophisticated software will inevitably have bugs, so in general the more people using the software, the more bugs will surface. Apache projects make bug reporting highly visible via the JIRA interface. If you search on the Internet for *Hadoop JIRA*, you'll quickly see dozens of bug reports opened for the Hadoop ecosystem project and others.

Hadoop is Being Widely Adopted in Every Industry

As with the adoption of relational database technology from the 1980s and onward, Hadoop solutions are springing up in every industry. Looking at the generic use cases we describe in Chapter 2, you can easily imagine most of them having a specific application for a business in any industry.

From what we're seeing firsthand as we work with clients on building Hadoop solutions, most businesses with large-scale information management challenges are seriously exploring Hadoop. Broad consensus from media stories and analyst reports now indicate that almost every Fortune 500 company has embarked on a Hadoop project.

Hadoop Can Easily Scale Out as Your Data Grows

Rising data volumes are a widespread big data challenge now faced by organizations. In highly competitive environments where analytics is increasingly becoming the deciding factor in determining winners and losers, being able to analyze those increasing volumes of data is becoming a high priority. Even now, most traditional data processing tools, such as databases and statistical packages, require larger scale hardware (more memory, disk, and CPU cores) to handle the increasing data volumes. This scale-up approach is limiting and cost-ineffective, given the need for expensive components.

In contrast to the scale-up model, where faster and higher capacity hardware is added to a single server, Hadoop is designed to *scale out* with ease by adding data nodes. These data nodes, representing increased cluster storage capacity and processing power, can easily be added on the fly to an active cluster. There are some software solutions using a scale-out model, but they often have complex dependencies and require application logic to change when resources are added or subtracted. Hadoop applications have no dependencies on the layout of racks or data nodes and require no changes as the numbers of active nodes change.

Traditional Tools are Integrating with Hadoop

With increased adoption, businesses are coming to depend on Hadoop and are using it to store and analyze critical data. With this trend comes an appetite for the same kinds of data management tools that people are accustomed to having for their traditional data sources, such as a relational database. Here are some of the more important application categories where we're seeing integration with Hadoop:

- **Business analysis tools**: Analysts can build reports against data stored in HDFS and cataloged using Hive. (Cognos, Microstrategy, and Tableau support this tack, for example.)
- **Statistical analysis packages**: Statisticians can apply their models on large data sets stored in HDFS and have that processing be pushed down to the Hadoop cluster to be run on the data nodes, where the data is stored. (For example, both SAS and SPSS have enabled limited push-down to MapReduce, as we discussed in Chapter 9.)
- **Data integration tools**: Data architects can enable high-speed data exchange between Hadoop and relational databases, and varying degrees of being able to push down transformation logic to the Hadoop cluster. (For example, both IBM DataStage and Informatica have parallel connectors to Hadoop enabling high speed data transfer and varying degrees of ability to have custom data transformation algorithms execute on the data nodes.)

Hadoop Can Store Data in Any Format

One feature of Hadoop reflects a key NoSQL principle: Store data first, and apply any schemas after it is queried. (For more on the ideas behind NoSQL, check out Chapter 11.) One major benefit that accrues to Hadoop from acting in accordance with this principle is that you can literally store any kind of data in Hadoop: completely unstructured, binary formats, semistructured log files, or relational data. But along with this flexibility comes a curse: After you store data, you eventually want to analyze it — and analyzing messy data can be difficult and time consuming. The good news here is that increasing numbers of tools can mitigate the analysis challenges commonly seen in large, messy data sets.

Hadoop is Designed to Run Complex Analytics

You can not only store just about anything in Hadoop but also run just about any kind of algorithm against that data. The machine learning models and libraries included in Apache Mahout are prime examples, and they can be used for a variety of sophisticated problems, including classifying elements based on a large set of training data.

Hadoop Can Process a Full Data Set (As Opposed to Sampling)

For fraud-analysis types of use cases (see Chapter 2), industry data from multiple sources indicates that less than 3 percent of all returns and claims are audited. Granted, in many circumstances, such as election polling, analyzing small sample sets of data is useful and sufficient. But when 97 percent of returns and claims are unaudited, even with good sampling rules, many fraudulent returns still occur. By being able to run fraud analysis against the entire corpus of data, you now get to decide whether to sample.

Hardware is Being Optimized for Hadoop

One of the more interesting Hadoop-related news items we've recently read is that Intel is now a player in the Hadoop distribution market. This new strategy raised many eyebrows: "What's a hardware manufacturer doing selling software?" This move by Intel was a shrewd one because its distribution work shows the seriousness and commitment behind its open source integration efforts. With Hadoop, Intel sees a tremendous opportunity to sell more hardware. After all, Hadoop clusters can feature hundreds of nodes, all requiring processors, motherboards, RAM, and hard disk drives. Intel has been investing heavily in understanding Hadoop so that it can build Intel-specific hardware optimizations that its Hadoop contributors can integrate into open source Hadoop projects. Other major hardware vendors (such as IBM, Dell, and HP) are also actively bringing Hadoop-friendly offerings to market.

Hadoop Can Increasingly Handle Flexible Workloads (No Longer Just Batch)

During the four-year lead-up to the release of Hadoop 2, a great deal of attention was directed at solving the problem of having a single point of failure (SPOF) with the HDFS NameNode (see Chapter 4). Though this particular success was no doubt an important improvement, since it did much to enable enterprise stability, we would argue that YARN is a far more significant development (see Chapter 7). Until Hadoop 2, the only processing that could be done on a Hadoop cluster was restricted to the MapReduce framework. This was acceptable for the log analytics use cases that Hadoop was originally built for, but with increased adoption came the real need for increased flexibility.

By decoupling resource management and scheduling responsibilities and implementing them in a generic framework, YARN can provision and manage a wider variety of processing models. At the time Hadoop 2 was released, MapReduce was still the only production-ready framework available. But active projects exist for in-memory processing, streaming data analysis, graph analysis, and much more.

The following statement is, for us, the perfect closing note for this book: We're about to see Hadoop become a truly multipurpose, flexible data processing engine. Where it once could support only batch workloads, it can now support real-time queries, and even a completely different processing approach by analyzing streaming data while it's still in motion.