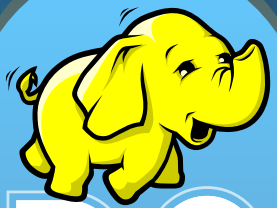


# HADOOP

## BUYER'S GUIDE

Everything you need to know about choosing the right Hadoop distribution for production

Robert D. Schneider
Author of Hadoop for Dummies

ubuntu
Supported by Canonical

## INTRODUCTION

Increasing numbers of enterprises are turning to Hadoop as an indispensable component for the mission-critical applications that drive their core business operations. Hadoop is no longer considered a “science project" or something to be casually spun off to a “skunk works” team. This newfound prominence means that the responsibility of selecting a Hadoop platform is now just as serious as when choosing any other core technology product.

This Buyer’s Guide presents a series of guidelines that you can use when searching for the essential Hadoop infrastructure that will be sustaining your organization for years to come. In fact, this guide is specifically designed to be incorporated into your RFP when it comes to evaluating Hadoop platforms.

The guide starts off with background on Big Data, MapReduce, and Hadoop. Next up is a justification about why selecting a Hadoop platform is so vital, followed by a sequence of recommendations that will help you make the best decision when appraising Hadoop options — including the operating systems that support it — with a particular emphasis on:

- Performance and scalability
- Dependability
- Manageability
- Data access

The intended audience for this guide includes IT leaders, database architects, system administrators, software developers and business analysts: in short, anyone charged with ensuring that Big Data is a success in their organization.

## BIG DATA, MAPREDUCE, AND HADOOP

Before we can explore what to consider when scouting for your Hadoop platform, it’s worth spending a little time understanding how we reached this juncture. To begin,

if you were to ask five people for their definition of Big Data, there's a good chance that you'll get ten different answers — and each of them would probably be correct!

That said, Big Data tends to describe one or more of the following characteristics:

- Encompasses large amounts of information
- Consists of a variety of data types and formats
- Generated by disparate sources
- Retained for long periods
- Utilized by new and innovative applications

Each of these characteristics introduces its own unique challenges — for IT as well as the entire organization — that must be surmounted before you can realize the full benefits of Big Data.

**Encompasses large amounts of information.** There was a time when a gigabyte of data was considered to be imposing. Nowadays, we casually talk about terabytes and petabytes of data.

**Consists of a variety of data types and formats.** Information is no longer constrained to rigid structures of rows and columns. Much of today's data is dynamic and unstructured — or semi-structured — and is proving to be very hard to process by traditional methods.

**Generated by disparate sources.** You still have to deal with information from all of your transactional applications, but you must also adapt to data from a host of new sources such as:

- Wireless devices
- Sensors
- Behavioral indicators, such as clickstreams, or query and beacon logs
- Streaming communication generated by machine-to-machine interactions
- Data exhaust: all of the system and access logs, configuration changes and other miscellaneous results of running computers



The old approach of painstakingly cleansing information from transactional systems and neatly placing it into data warehouses doesn't work in an era where data is arriving in such huge volumes from so many diverse sources. This is especially true since even the format of the data can be changing.

**Retained for long periods.** Partly spurred by legal and regulatory mandates, storage requirements are exploding because retention periods for previously ephemeral data can now extend to years or even decades.

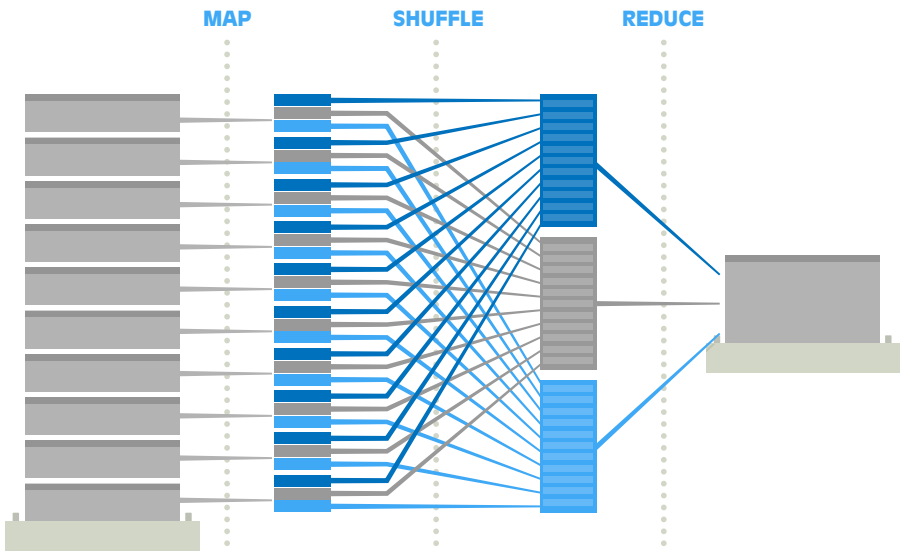
**Utilized by new and innovative applications.** Many enterprises are following a "grab the data first, and then figure out what to do with it later" approach. Frankly, there's just no way for IT to predict and plan for all of the new applications that are being aimed at Big Data. Since much of the value of data is only recognized after the data is available for experimentation, it can be argued that this kind of opportunistic retention is a very good strategy.

## THE BIRTH OF MAPREDUCE

Given all of the Big Data-related hurdles, it's been clear for a long time that a better way for working with massive amounts of information was required. Computer scientists have long known that divide-and-conquer algorithms can be very effective for processing large amounts of information. The problem has been that applying divide and conquer to a wide range of problems has historically been quite difficult.

Early in Google's history, developers there codified a style of programming that they called MapReduce that is surprisingly effective at processing very large amounts of data and yet is able to express a wide range of algorithms. In 2004, these developers published an article that described their methods and results.

In this paper, Google summed up MapReduce as follows: "MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key."




*MapReduce*

As implemented, MapReduce is actually a collection of complementary techniques and strategies that include employing commoditized hardware and software, specialized underlying file systems, and parallel processing methodologies. Many of the benefits arise from the fact that computation can be done on the same machines where data resides and from the fact that individual pieces of the overall computation can be recomputed if necessary due to hardware failure or other delays.

This is a revolutionary architectural philosophy that shelters the average developer from the overwhelming complexity that had formerly been required to properly carry out parallel processing. But as we'll see later, the implementation of MapReduce laid the foundation for significant problems now being experienced by many enterprises that are seeking to put it to work.



## ENTER HADOOP

As originally specified, MapReduce was a simple and straightforward architecture, but one that required the formidable resources of a very large and sophisticated organization — such as Google — to correctly implement it. In its most elemental form, MapReduce requires you to discard all of your code and rewrite it in a radically new style that is completely foreign to most programmers. The limitations on what you can express in your code are fundamental for MapReduce's scalability, but also serve as a source of great difficulty in training programmers. Also, since there were no available implementations, adopting MapReduce also meant that you would have to implement the entire framework just to write your first program.

Fortunately, two enterprising engineers — Doug Cutting and Mike Cafarella — had been working on their own web crawling technology named Nutch. After reading the Google research paper, they set out to create the foundations of what would later be known as Hadoop.

Eventually, Cutting joined Yahoo! where Hadoop was extended substantially. As Hadoop grew in sophistication, Yahoo! broadened its usage into a far-reaching suite of critical applications that underpinned much of its daily operations. In early 2008, the Apache Software Foundation (ASF) recognized Hadoop's importance and promoted it to a top-level open source project.

The original implementation of Hadoop was in Java and used the Linux file system to store data. These decisions, both well-founded in the original context of Nutch, later caused difficulties for enterprises aiming to base their operations on Hadoop.

Overall, however, Hadoop has been an enormous success. Here are just a few examples of mainstream production applications:

Companies are using Hadoop to off load data warehouse-bound data. Hadoop, on average, provides at least a 10x cost savings over data warehouse solutions.

Financial institutions are using Hadoop as a critical part of their security architecture — to predict phishing behavior and payments fraud in real time and minimize their impact. They hold on to data for longer periods and run more detailed analytics and forensics.

An online advertising company provides real-time trading technology to its users and relies on Hadoop to store and analyze petabytes worth of data. 90 billion real time ad auctions are processed each day on their Hadoop distribution.

A digital marketing intelligence provider uses Hadoop to process over 1.7 trillion Internet and mobile records per month providing syndicated and custom digital marketing intelligence.

With the right Hadoop distribution, companies can provide new products and services for consumers in real time. Advanced machine learning and statistical techniques are employed over data stored in a highly available Hadoop cluster.

### RELATED HADOOP PROJECTS

Hadoop has also spawned an entire ecosystem of ancillary initiatives. Here are just a few examples of other Apache projects:

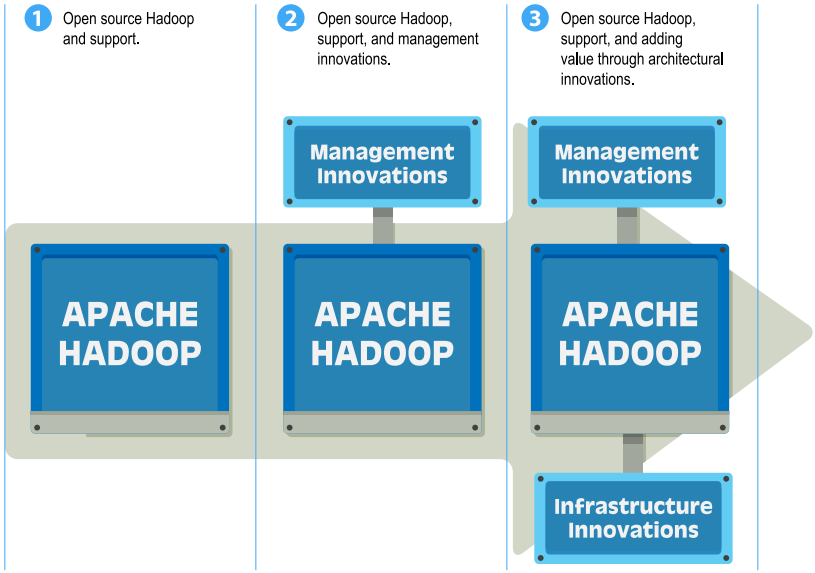
| Workflow Scheduling | Resource Scheduling | Data Interaction and Analysis |
|---|---|---|
| Oozie | YARN | Pig |
| **Databases** | **Data Warehousing** | Drill |
| HBase | Hive | **Extract, Transform, and Load (ETL)** |
| **Scalable Machine Learning** | | Flume |
| Mahout | | Sqoop |
| | | Scribe |

## WHY YOUR CHOICE OF HADOOP INFRASTRUCTURE IS IMPORTANT

Acquiring, deploying, and properly integrating all of the moving parts that constitute the Hadoop ecosystem has proven to be a hardship for many IT organizations, which would much rather focus on their primary business responsibilities than the care and feeding of a hand-crafted Hadoop environment. To further muddy the water, not only is Hadoop continually evolving, but so are all of the related projects in its ecosystem.

In an effort to ease the task of rolling out a complete Hadoop implementation, a number of vendors are offering comprehensive distributions that generally fall into one of three models:

1. **Open source Hadoop and support.** This pairs bare-bones open source with paid professional support and services. Hortonworks is a good example of this strategy.
2. **Open source Hadoop, support, and management innovations.** This goes a step further by combining open source Hadoop with IT-friendly tools and utilities that make things easier for mainline IT organizations. Cloudera is an instance of this model.
3. **Open source Hadoop, support, and adding value through architectural innovations.** Hadoop is architected with a component model down to the file system level. Innovators can then replace one or more components and package the rest of the open source components and maintain compatibility with Hadoop. MapR is an instance of this model.




*3 models of Hadoop distributions*

## CRITICAL CONSIDERATIONS WHEN SELECTING A HADOOP PLATFORM

Adopting a Hadoop distribution is a vital decision that has far-reaching ramifications for the entire organization, in ways that you can't fully anticipate when you create your initial appraisal. This is particularly true since we're only at the dawn of Big Data in the enterprise.

Hadoop infrastructure is just that: infrastructure, and it requires the same level of attention and scrutiny as your organization expends when choosing other critical assets, such as application servers, storage, and databases. Thus, you shouldn't be surprised that your Hadoop environment will be subject to the same requirements as the rest of your IT portfolio, in terms of:

- Service Level Agreements (SLAs)
- Data protection
- Security
- Integration with other applications
- Professional services
- Training

To begin, don't think of Hadoop as a single solution, but rather as a platform with a collection of applications on top. These elements must work together to derive maximum value. Secondly, don't force your enterprise to conform to your chosen Hadoop technology; instead, find solutions that adapt to the way you operate your business.

Consider using the guidelines in this section to help you construct an RFP, just like you would when identifying any other fundamental software product. For clarity, these are grouped into four major categories:

- Performance and Scalability
- Dependability
- Manageability
- Data Access

For each recommendation, we explain what it is and what to look for in a Hadoop distribution. Also, provided are several examples that demonstrate how these capabilities add value in real-world situations.

## PERFORMANCE AND SCALABILITY

In its earliest days, Hadoop was primarily used to crawl and index the Web, which was less sensitive from that standpoint than many current use cases. Today, growing numbers of Hadoop projects are being tasked with delivering actionable results in real-time, or near real-time. Not surprisingly, the definition of Hadoop's performance has evolved in lockstep: in the earliest days, fast throughput was the primary metric; now, it includes low latency.

This recent emphasis on low latency places intense focus on two major attributes of any Hadoop platform:

**Its raw performance.** This refers to everything from how quickly it ingests information and whether this data is immediately available for analysis, to its latency for real-time applications and MapReduce speeds.

**Its ability to scale.** This describes how easily it can expand in all relevant dimensions, such as number of nodes, tables, files, and so on. Additionally, this shouldn't impose heavy administrative burdens, any changes to application logic, or excessive costs.

This section explores a number of capabilities that are directly related to how well your Hadoop implementation will be able to perform — and scale.

### ARCHITECTURAL FOUNDATIONS FOR PERFORMANCE AND SCALABILITY

For specific features that should be present in your Hadoop environment, have a look at Table 1, which itemizes a number of critical architecture preconditions that can have a positive impact on performance and scalability.

| Prerequisite | Why It's Important |
|---|---|
| **Build key components with a systems language like C/C++** | The Apache open source Hadoop distribution is written in Java. While this is helpful for portability, it also introduces a number of Java-related issues that can degrade performance, such as Java's inherent overhead vs. languages that are "closer to the iron," and its well-documented unpredictability and latency of garbage collection. Using C/C++ is consistent with almost all other enterprise-grade software. |
| **Minimal software layers** | In general, any system benefits from a reduced number of "moving parts." In the case of Hadoop and the related Apache HBase project, performance and reliability can be obstructed by the inefficiencies that come from having to navigate a series of separate layers such as the HBase Master and RegionServer, the Java Virtual Machine, and the local Linux file system. |
| **A single environment/ platform for all Big Data applications** | Largely due to performance necessities, many Hadoop implementations have required administrators to create separate instances. It's much better if your Hadoop technology delivers sufficient throughput for the full spectrum of workloads that you're likely to experience. This also eliminates/reduces duplication of data, which in turn improves scalability. |
| **Leverage the elasticity and scalability of popular public cloud platforms** | Developers and administrators alike need the flexibility to run Hadoop inside their firewall as well as on widely adopted cloud environments such as Amazon Web Services and Google Compute Engine. |

*Table 1: Performance and scalability preconditions*

### STREAMING WRITES

Given that Hadoop typically is meant to work with massive amounts of information, the job of loading and unloading data must be as efficient as possible. Yet many Hadoop distributions require complex and cumbersome batch or semi-streaming processes using technologies such as Flume and Scribe. To make things worse,

these deep-seated inefficiencies are magnified when data volumes are in the gigabytes to terabytes and beyond.

A better technique is for your Hadoop implementation to expose a standard file interface that lets your applications access the Hadoop cluster as if it was traditional Network Attached Storage (NAS). Application servers are then able to directly write information into the Hadoop cluster, as opposed to first staging it on local disks. Data bound for Hadoop can also be automatically compressed on the fly as it arrives, and it's immediately available for random read and write access by applications through multiple parallel, concurrent connections. These immediate interactions permit the real-time Hadoop-based decision-making described earlier.

Consider an online gaming company that's relying on Hadoop to track millions of users and billions of events. There are very short windows of opportunity to introduce virtual goods to players, because these users tend to come and go very quickly. Fortunately, real-time or near real-time analysis on streaming data helps increase revenue by making it possible to offer timely suggestions. Although projects like Apache Drill are meant to facilitate rapid decision-making, this isn't possible unless the raw data itself arrives in the Hadoop cluster as speedily as possible.

## SCALABILITY

IT organizations eager to capitalize on Hadoop are often faced with a conundrum: either acquire more hardware and other resources than will ever be necessary and thus wastefully expend scarce funds and administrator time, or try to squeeze as much as possible from a relatively limited set of computing assets and potentially miss out on fully capitalizing on their Big Data.

A scalable Hadoop platform can help balance these choices and thus make it easier to meet user needs while staying on budget.



Recall from earlier that a given Hadoop instance's scalability isn't measured on a single scale. Instead, you should take several factors into consideration:

**Files.** Hadoop's default architecture consists of a single NameNode. This constrains Hadoop clusters to a (relatively) paltry 100 million to 150 million files, a number that's also impacted by the amount of memory available for file metadata. And in small clusters, ceilings on the number of blocks on each data node further constrain the number of available files. Look for a Hadoop platform that avoids the single NameNode bottleneck and has distributed metadata architecture, and can thus scale to billions — or even trillions — of files and tables.

**Number of nodes.** Another dimension of scale is the number of physical nodes. Depending on the processing or data storage requirements your selected Hadoop implementation might need to scale to 1,000 nodes and beyond.

**Node capacity/density.** In addition, for storage intensive use cases you need to scale through nodes with higher disk densities. This serves to reduce the overall number of nodes required to store a given volume of data.

## REAL-TIME NOSQL

More enterprises than ever are relying on NoSQL-based solutions to drive critical business operations. The only way for these new applications to achieve the reliability and adoption of RDBMS-based solutions is for them to conform to the same types of rigorous SLAs that IT expects from applications built on relational databases. For example, NoSQL solutions with wildly fluctuating response times would not be candidate solutions for core business operations that require consistent low latency.

Apache HBase is a key-value based NoSQL database solution that is built on top of Hadoop. It provides storage and real-time analytics for Big Data with the added advantage of MapReduce operations using Hadoop. About 30-40% of Hadoop installs today are estimated to be using HBase. Despite its advantage of integrating with Hadoop, HBase has not reached its true adoption potential because of several limitations in its performance and dependability.

Fortunately, there are a number of innovations that can transform HBase applications to meet the stringent needs for most online applications and analytics. These include:

- Reducing the overall number of layers
- Eliminating the need for Java garbage collection
- Eliminating the need for manually pre-splitting tables
- Distributing metadata across the cluster, rather than on a single NameNode
- Avoiding compactions and the related I/O storms that these trigger

## DEPENDABILITY

You can expect Hadoop to be subject to the same dependability expectations as every other type of enterprise software system. You can also anticipate that the same IT administrators who are caring for the rest of your IT assets will also manage your Hadoop implementations.

To reduce the overall burden on users and administrators alike, the most successful Hadoop infrastructure will be capable of coping with the inevitable problems encountered by all production systems. Many of these reactions should be automated to further enhance dependability. This section reviews several traits of Hadoop platforms that have been architected to thrive in the most stressful environments.

## ARCHITECTURAL FOUNDATIONS FOR DEPENDABILITY

Table 2 depicts several foundational principles that help increase the dependability of your Hadoop implementation.

| Prerequisite | Why It's Important |
|---|---|
| **Fewer moving parts** | Your HBase environment shouldn't require RegionServers or HBase Masters. Eliminating the Java virtual machine is also very helpful. |
| **Fewer manual tasks** | Many Hadoop administrators are burdened by tasks such as compactions, manual administration, and manual pre-splitting. Eradicating these responsibilities helps increase overall dependability. |
| **Data integrity** | This should be heightened through internal checksums, replication and through data protection features such as snapshots, which are described later. |
| **Job optimization** | Your runtime environment should be optimized to ensure that small tasks — such as ad-hoc queries — do not get stuck behind pre-scheduled large jobs. |

*Table 2: Architectural foundations for dependability*

## HIGH AVAILABILITY

High availability (HA) refers to the propensity of a Hadoop system to continue to service users even when confronted with the inevitable hardware, network, and other issues that are characteristic to distributed computing environments of this size and complexity.

To deliver the availability that you'll need for mission-critical production applications, your Hadoop environment should incorporate each of these HA capabilities:

**HA is built-in.** First and foremost, it shouldn't be necessary to perform any special steps to take advantage of HA; instead, it should be default behavior for the platform itself.

**Meta data.** A single NameNode that contains all meta data for the cluster represents a single point of failure and an exposure for HA. A solution that distributes the meta data coupled with failover has HA advantages and as an added benefit, there's no practical limit on the number of files that be supported.

**MapReduce HA.** One important aspect of HA is how MapReduce jobs are impacted by failures. A failure in a job or task tracker can impact the ability to meet SLAs. Determine whether MapReduce HA includes automated failover and the ability to continue with no manual restart steps.

**NFS HA.** This offers high throughput and resilience for NFS-based data ingestion and access.

**Recovery time from multiple failures.** One of the areas of differentiation across Hadoop distributions is the time and process it takes to recover files in case of a hardware, user or application error, including the ability to recover from multiple failures. How soon are files and tables accessible after a node failure or cluster restart? Seconds? Minutes? Longer?

**Rolling upgrades.** As Hadoop and its complementary technologies evolve, you should be able to upgrade your implementation without needing to incur any downtime.

## DATA PROTECTION

For growing numbers of organizations, Hadoop is driving crucial business decisions that directly impact the bottom line. This is placing heightened emphasis on safeguarding the data that Hadoop processes. Fortunately, well-proven techniques such as replication and snapshots have long been fundamental building blocks for protecting relational data, and they each have a role to play in shielding Hadoop's information as well.

**Replication.** This helps defend Hadoop's data from the periodic failures you can expect when conducting distributed processing of huge amounts of data on commodity hardware. Your chosen platform should automatically replicate — at least 3X — Hadoop's file chunks, table regions, and metadata, with at least one replica sent to a different rack.

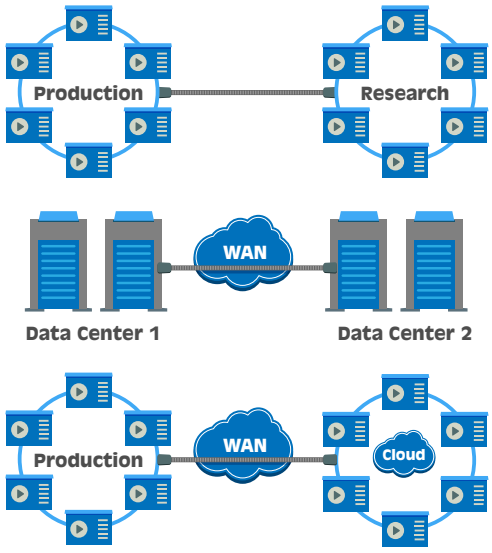
**Snapshots.** By offering point-in-time recovery without data duplication, Hadoop Snapshots provide additional insurance from user and application errors. If possible, your Hadoop platform's capabilities should permit snapshots to share the same storage as live information, all without having impact on performance or scalability. You should also be able to read files and tables directly from a snapshot. Snapshots go beyond mere data protection. For example, data scientists can use a snapshot to aid in the process of creating a new model. Different models can be run against the same snapshot, isolating results to model changes.

## DISASTER RECOVERY

Hadoop is particularly prone to events that have the potential to significantly disrupt business operations, because:

- It's commonly deployed on commoditized hardware
- It stores enormous amounts of information
- Its data is distributed, and networks are prone to sporadic outages
- Today's IT environments are routinely subject to attack

Since there's such a good chance that you'll encounter an emergency, you would be wise to employ mirroring as a preventative measure that can help you recover from even the most dire situations.




*Mirroring synchronizes from production to research clusters, across data centers and from on-premise to cloud.*

**Mirroring.** Your Hadoop mirroring should be asynchronous and perform auto-compressed, block-level data transfer of differential changes across the WAN. It should mirror data as well as its meta data, while maintaining data locality and data consistency at all times. This ensures that applications can restart immediately upon site failure. It should also have the following characteristics:

| Efficient | Safe | Easy |
|---|---|---|
| Block-level (8 KB) deltas | Point-in-time consistency | Graceful handling of network issues |
| Automatic compression | End-to-end checksums | Access mirror volume directly (i.e. not a cold standby) |
| No performance impact | | Scheduling at the volume level |

## MANAGEABILITY

Early in Hadoop's history, it was fairly common for sophisticated developers with source code-level understanding of Hadoop to manage multiple Hadoop environments. This could work because these developers detailed knowledge of Hadoop internals and because they had combined developmental and operational responsibilities as is typical in startups. This clearly won't translate into mainline IT usage because it simply is not feasible for an operations team to handle many different systems in addition to Hadoop. Total cost of ownership (TCO) is always a major consideration when IT compares solutions, and it's especially relevant in Hadoop environments.

Seek out a Hadoop platform that supplies comprehensive, intelligently designed tooling that eases administrative burdens. As Hadoop continues to mature, Hadoop distributions will compete on the quality and depth of their management tools in each of these critical areas:

### Administration

- Volume-based data and user management
- Centralized node administration and troubleshooting
- Adding and removing disk drives directly through a graphical user interface (GUI)
- Rolling upgrades that permit staggered software upgrades over a period of time without disrupting the service
- Automated and scheduled administrative tasks
- Multi-tenant user access with data and job placement control

### Monitoring

- End-to-end monitoring of the Hadoop cluster, from the application to the hardware level, including detecting disk failures
- Alerts, alarms, and heat maps that provide a color-coded, real-time view of the nodes, including their health, memory, CPU, and other metrics
- Integration, via a REST API, into different open source and commercial tools as well as the ability to build custom dashboards
- Visibility through standard tools like Ganglia and Nagios

Finally, Hadoop implementations routinely scale to hundreds — or thousands — of nodes. Attempting to manage the configuration, deployment, and administration of all these nodes is a chore that should be automated as much as possible. Fortunately, leading operating system vendors are continually refining their automated configuration and service orchestration solutions. For example, Juju from Canonical offers both a graphical user interface and a command line interface that lets administrators automate all facets of their distributed processing environments.

## DATA ACCESS

Gobbling up colossal arrays of information is only the beginning of your Hadoop story. To unlock all of your data's potential value, you need a Hadoop platform that makes it easy to ingest and extract this information quickly and securely, and then lets your developers build fully capable applications using well-proven tools and techniques. It's even more auspicious if your existing applications can easily connect to Hadoop's data.

This section is all about making sure that your appointed Hadoop platform will interact smoothly with the rest of your IT environment.

### ARCHITECTURE FOUNDATIONS FOR DATA ACCESS

Before reading the suggestions for enhancing data access in Hadoop, take a look at Table 3 for some fundamentals.

| Prerequisite | Why It's Important |
|---|---|
| **Full access to the Hadoop file system API** | Complete functionality for application development, along with easy portability to other Hadoop implementations. The result is no vendor lock-in. |
| **Full POSIX read/write/ update access to files** | Greater software development flexibility, plus existing applications can work directly with Hadoop data. |
| **Direct developer control over key resources** | No need to summon administrators to create permanent or temporary tables to optimize application workflows. |
| **Secure, enterprise-grade search** | Applications can include vital capabilities like click scoring, search alerts and search relevancy tuning. |
| **Comprehensive data access tooling** | Administrators and developers can select the right implement for the job at hand, such as:<br>···> Apache Flume for log collection and aggregation<br>···> Apache Sqoop for parallel import/export between Hadoop and relational databases and data warehouses<br>···> distcp for distributed copying between clusters and between remote data sources and Hadoop<br>···> ODBC and JDBC to export data from Hadoop (via Hive) |

*Table 3: Architectural foundations for data access*

### STANDARD FILE SYSTEM INTERFACE AND SEMANTICS (POSIX)

A POSIX file system that supports random read/write operations on Hadoop as well as providing NFS access opens up Hadoop to much broader usage than is commonly found with the default HDFS. This also simplifies tasks that would otherwise have required much more complex processes.

Administrators and users should be able to mount the cluster over the network like enterprise NAS. Browsers such as Windows Explorer, Mac Finder, IDEs, and standard Linux file interaction commands like ls, grep and tail — will thus all be able to work directly with the cluster. With the Hadoop cluster treated like part of the file system, users can drag and drop data into Hadoop or hit the Tab key to auto-complete instructions on its command line interface.

Going beyond new Hadoop-based applications, other solutions — legacy or new, and written in your choice of programming language — can use the file system to access and write data on Hadoop. A POSIX file system also makes it straightforward

to import and export information to/from relational databases and data warehouses using the standard tools without a need for special connectors.

For example, a retailer could quickly load data in parallel using standard tools through NFS. Data will be streamed in directly, and won't require creating sequential writes that will slow down the entire process.

### DEVELOPER TOOLS

Developers have decades of experience employing popular tools and methodologies for interacting with relational databases. While Hadoop introduces new paradigms and concepts, you should seek out platforms that boost developer productivity by:

- Offering open source components on public GitHub for download and customization
- Making binaries available through Maven repositories for faster application builds
- Providing a workflow engine for building applications more quickly
- Enabling standard development tools to work directly with data on the cluster
- Permitting existing non-Hadoop applications and libraries written in any programming language to be able to access and write data on Hadoop
- Supplying SQL-like interactive query capabilities

### SECURITY

Scarcely a day goes by without a news headline about a data breach or other major security violation, often involving Big Data. Given the amount of information stored in Hadoop — and the broad range of this data — it's essential that you take proactive steps to protect your data before your organization is featured on the news. Sadly, some Hadoop implementations are so challenging to secure that customers avoid the subject entirely, and never actually enable security.

Rather than referring to a single capability, your Hadoop security should be far-reaching, and encompass each of the following safeguards:

- Fine-grained permissions on files, directories, jobs, queues, and administrative operations
- Access control lists (ACLs) for tables, columns and column families
- Wire-level encryption between the Hadoop cluster and all external cluster access points both natively and through third parties
- Standard authentication protocols such as Kerberos, LDAP, Active Directory, NIS, local users and groups, and other 3rd party authentication and identity systems
- Simple yet secure access to the cluster through a "gateway node," while blocking direct interaction with all other nodes

## COMPARING MAJOR HADOOP DISTRIBUTIONS

Just about every organization is seeking ways to profit from Big Data, and Hadoop is increasingly serving as the most capable conduit to unlock its inherent value. This means that Hadoop is likely to have a major role to play in your enterprise. Given this probability, you should carefully consider your choice of Hadoop implementation, and pay particular attention to its performance/scalability, dependability, and ease of data access. In particular, make sure that your selection conforms to the way you operate, and not the other way around.

On the next page is a quick comparison chart of some of the differences across the major Hadoop distributions.

| | Hortonworks | Cloudera | MapR |
|---|---|---|---|
| **Performance and Scalability** | | | |
| Data Ingest | Batch | Batch | Batch and streaming writes |
| Metadata Architecture | Centralized | Centralized | Distributed |
| HBase Performance | Latency spikes | Latency spikes | Consistent low latency |
| NoSQL Applications | Mainly batch applications | Mainly batch applications | Batch and online/real-time applications |
| **Dependability** | | | |
| High Availability | Single failure recovery | Single failure recovery | Self healing across multiple failures |
| MapReduce HA | Restart jobs | Restart jobs | Continuous without restart |
| Upgrading | Planned downtime | Rolling upgrades | Rolling upgrades |
| Replication | Data | Data | Data + metadata |
| Snapshots | Consistent only for closed files | Consistent only for closed files | Point-in-time consistency for all files and tables |
| Disaster Recovery | No | File copy scheduling (BDR) | Mirroring |
| **Manageability** | | | |
| Management Tools | Ambari | Cloudera Manager | MapR Control System |
| Volume Support | No | No | Yes |
| Heat map, Alarms, Alerts | Yes | Yes | Yes |
| Integration with REST API | Yes | Yes | Yes |
| Data and Job Placement Control | No | No | Yes |
| **Data Access** | | | |
| File System Access | HDFS, read-only NFS | HDFS, read-only NFS | HDFS, read/write NFS (POSIX) |
| File I/O | Append only | Append only | Read/write |
| Security: ACLs | Yes | Yes | Yes |
| Wire-level Authentication | Kerberos | Kerberos | Kerberos, Native |

## ABOUT THE AUTHOR

*Robert D. Schneider is a Silicon Valley–based technology consultant and author. He has provided database optimization, distributed computing, and other technical expertise to a wide variety of enterprises in the financial, technology, and government sectors.* He has written eight books — including *Hadoop For Dummies*, published by IBM — and numerous articles on database technology and other complex topics such as cloud computing, Big Data, data analytics, and Service Oriented Architecture (SOA). He is a frequent organizer and presenter at technology industry events, worldwide. Robert blogs at www.rdschneider.com.

ubuntu
Supported by Canonical

# HADOOP

## BUYERS GUIDE