HDP Overview: Apache Hadoop Essentials

Student Guide

Rev 3.0

HORTONWORKS® UNIVERSITY

Become a ***Hortonworks Certified Professional*** and establish your credentials:

• HDP Certified Developer: for Hadoop developers using frameworks like Pig, Hive, Sqoop and Flume.

• HDP Certified Administrator: for Hadoop administrators who deploy and manage Hadoop clusters.

• HDP Certified Developer: Java: for Hadoop developers who design, develop and architect Hadoop-based solutions written in the Java programming language.

• HDP Certified Developer: Spark: for Hadoop developers who write and deploy applications for the Spark framework.

How to Register: Visit www.examslocal.com and search for "Hortonworks" to register for an exam. The cost of each exam is $250 USD, and you can take the exam anytime, anywhere using your own computer. For more details, including a list of exam objectives and instructions on how to attempt our practice exams, visit http://hortonworks.com/training/certification/

Earn Digital Badges: Hortonworks Certified Professionals receive a digital badge for each certification earned. Display your badges proudly on your résumé, LinkedIn profile, email signature, etc.

Self Paced Learning Library

On Demand Learning

Hortonworks University Self-Paced Learning Library is an on-demand dynamic repository of content that is accessed using a Hortonworks University account. Learners can view lessons anywhere, at any time, and complete lessons at their own pace. Lessons can be stopped and started, as needed, and completion is tracked via the Hortonworks University Learning Management System.

Hortonworks University courses are designed and developed by Hadoop experts and provide an immersive and valuable real world experience. In our scenario-based training courses, we offer unmatched depth and expertise. We prepare you to be an expert with highly valued, practical skills and prepare you to successfully complete Hortonworks Technical Certifications.

Target Audience: Hortonworks University Self-Paced Learning Library is designed for those new to Hadoop, as well as architects, developers, analysts, data scientists, and IT decision makers. It is essentially for anyone who desires to learn more about Apache Hadoop and the Hortonworks Data Platform.

Duration: Access to the Hortonworks University Self-Paced Learning Library is provided for a 12-month period per individual named user. The subscription includes access to over 400 hours of learning lessons.

The online library accelerates time to Hadoop competency. In addition, the content is constantly being expanded with new material, on an ongoing basis.

Visit: http://hortonworks.com/training/class/hortonworks-university-self-paced-learning-library/

Table of Contents

The Case for Hadoop

Lesson Objectives

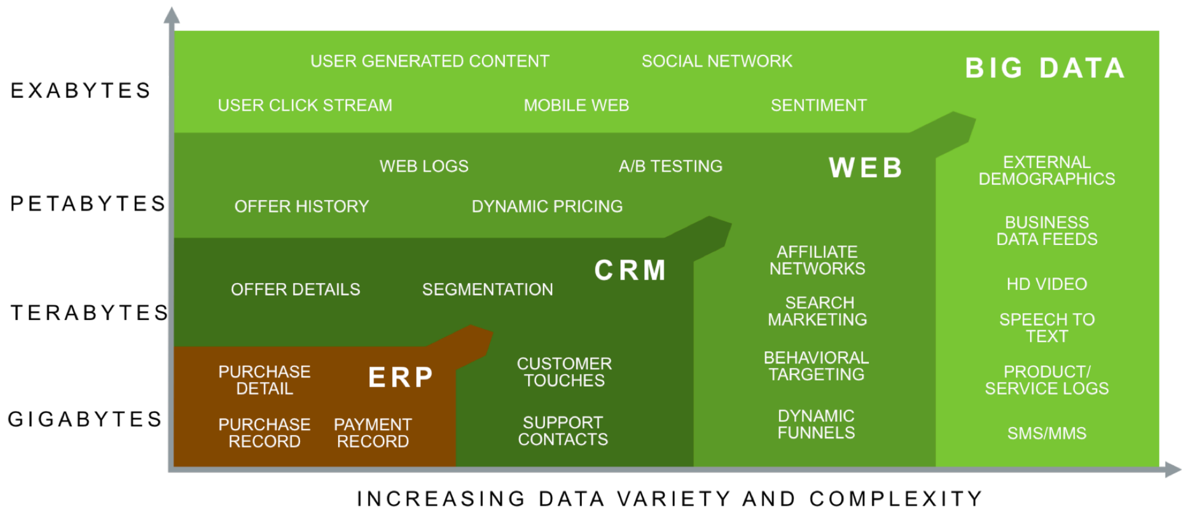
After completing this lesson, students should be able to:

- ✓ Describe data trends of volume, velocity & variety
- ✓ List popular use cases for Hadoop
- ✓ Discuss the importance of Open Enterprise Hadoop
- ✓ Give an overview of Connected Data Platforms powered by Hadoop

Big Data Trends

This lesson discusses the factors driving the exposion of Big Data; what makes data "big"; and how this explosion drives technology and opportunities in today's Enterprise.

3 Big Data Vs Drive Apache Hadoop



Companies like Hortonworks are not driving the case for Hadoop; data **Volume**, **Velocity**, and **Variety** are driving the need for Hadoop. Hadoop is the industry recognized backplane for Connected Data Platforms.

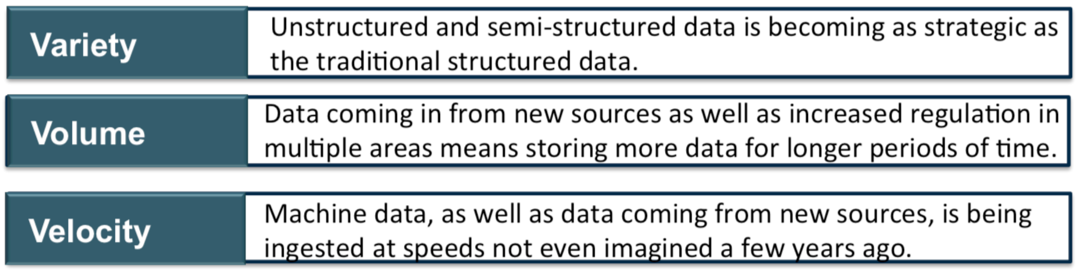
Next up? 1000 Exabytes = 1 Zettabyte

REFERENCE:

In 2012, International Data Corporation estimated that the total size of data in the world would be 2.7 zettabytes. This was an increase of 48% from 2011.

One expert estimated in 2013 that total would grow to 4 zettabytes.

What Makes Big Data Big



Variety	Unstructured and semi-structured data is becoming as strategic as the traditional structured data.
Volume	Data coming in from new sources as well as increased regulation in multiple areas means storing more data for longer periods of time.
Velocity	Machine data, as well as data coming from new sources, is being ingested at speeds not even imagined a few years ago.

3 Vs of Big Data

Hadoop does not just work on data; it was specifically designed to work on **Big Data.**

What what makes data big? Where did the phrase **Big Data** come from and what does it mean?

The term **Big Data** came from the computational sciences. Specifically, it is used to describe scenarios where the volume and variety of data types overwhelm the existing tools to store and process it.

In 2001, the industry analyst Doug Laney described Big Data using the three V's of volume, velocity, and variety.

NOTE:

Instead of "unstructured", currently we tend towards talking in terms of "schema on read".

Volume

Volume refers to the amount of data being generated. Think in terms of gigabytes, terabytes, and petabytes. Many systems and applications are just not able to store, let alone ingest or process, that much data.

Many factors contribute to the increase in data volume including: transaction-based data stored for years, unstructured data streaming in from social media, and the ever-increasing amounts of sensor and machine data being produced and collected.

Volume issues include:

- Storage cost
- Filtering and finding relevant and valuable information in large quantities of data that often contains much information that is not valuable
- The ability to analyze data quickly enough in order to maximize business value today and not just next quarter or next year

Velocity

Velocity refers to the rate at which new data is created. Think in terms of megabytes per second and gigabytes per second.

Data is streaming in at unprecedented speed and must be dealt with in a timely manner in order to extract maximum value from the data. Sources of this data include logs, social media, RFID tags, sensors, and smart metering.

Velocity issues include:

- Not reacting quickly enough to benefit from the data
  For example, data could be used to create a dashboard that could warn of imminent failure or a security breach - failure to react in time could lead to service outages
- Data flows tend to be highly inconsistent with daily, seasonal, or event-triggered changes in peak loads
  For example, a change in political leadership could cause an a peak in social media

Variety

Variety refers to the number of types of data being generated. Data can be gathered from databases, XML or JSON files, text documents, email, video, audio, stock ticker data, and financial transactions.

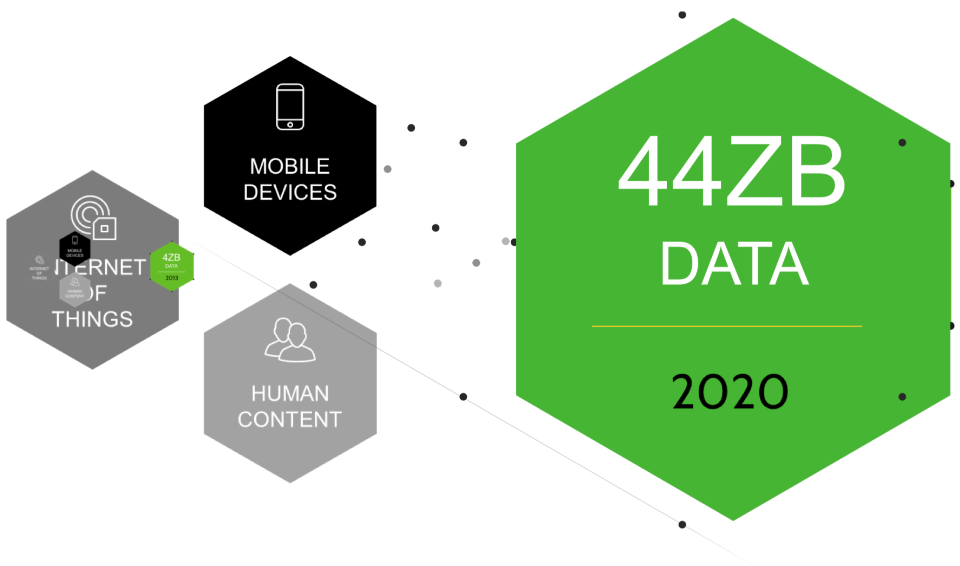
Varieties of data include:

- Structured
- Semi-structured
- Unstructured (schema on read)

Variety issues include:

- How to gather, link, match, cleanse, and transform data across systems
- How to connect and correlate data relationships and hierarchies in order to extract business value from the data

The Information Explosion



Source: http://www.emc.com/leadership/digital-universe/2014iview/executive-summary.htm

The world's data used to double every century, now it doubles every two years. This explosion is driven by the Internet of Things(IoT), by mobile devices, and by our ability to generate more digital content than ever before.

The digital universe will grow from 4 zettabytes of data in 2013 to 44 zetabytes in 2020. 1

Threats

Existing data architectures make data inaccessible, incomplete, irrelevant, and expensive.

This is the largest business innovation cycle in history, and these changes threaten existing data strategies. Many companies have big plans for big data, but existing data architectures make our data inaccessible, incomplete, irrelevant, and expensive. As data streams in at accelerating rates, the cost to store, reformat, and retrieve it grows more quickly than the value it may provide.

We know that big data holds big value, but we also know that we are at risk of being left behind if our competitors capture that value before we do.

Opportunities

Apache™ Hadoop® transforms business, making Big Data easily accessible for advanced analytic applications.

Companies of every size are using new big data opportunities to transform their businesses and the lives of their customers.

Hadoop can help: help . It can help . It can help a And it can help .

1 **Source:** *http://www.emc.com/leadership/digital-universe/2014iview/executive-summary.htm*

- Pharmaceutical manufacturers make better vaccines that save lives
- Doctors prescribe treatments based on data from all previous patients
- Car insurance companies keep drivers safe
- Mobile providers reduce call center wait times

There isn't a single organization that couldn not benefit from better insight into their data, but most are unable to store or even make use of all the data they have.

What is Apache Hadoop

Apache Hadoop

So what is Apache Hadoop?

It is a scalable, fault tolerant, open source framework for the distributed storing and processing of large sets of data on commodity hardware.

But what does all that mean?[2]

Scalability

Hadoop clusters can range from as few as one machine to literally thousands of machines.

Fault Tolerance

Hadoop services become fault tolerant through redundancy. For example, the Hadoop Distributed File System, called HDFS, automatically replicates data blocks to three separate machines, assuming that your cluster has at least three machines in it. Many other Hadoop services are replicated, too, in order to avoid any single points of failure.

Open Source

Hadoop development is a community effort governed under the licensing of the Apache Software Foundation. Anyone can help to improve Hadoop by adding features, fixing software bugs, or improving performance and scalability.

Distributed Storage and Processing

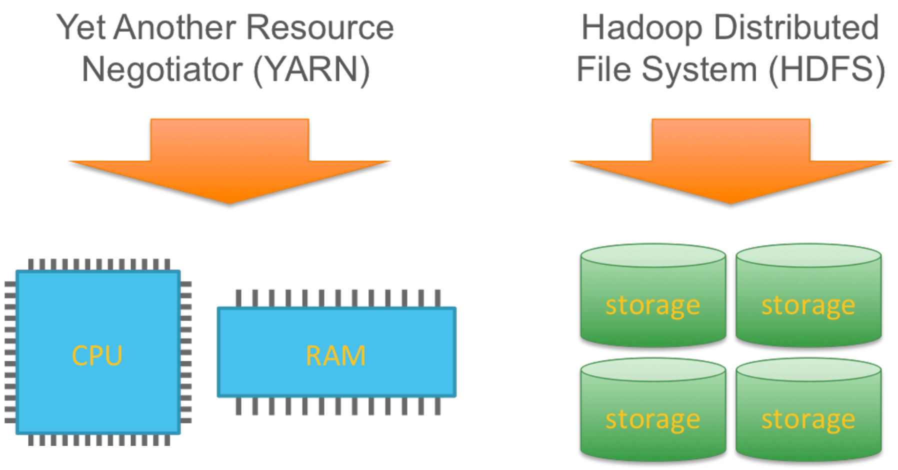
Large datasets are automatically split into smaller chunks, called blocks, and distributed across the cluster machines. Not only that, but each machine processes its local block of data. This means that processing is distributed too, potentially across hundreds of CPUs and hundreds of gigabytes of memory.

Commodity Hardware

All of this occurs on commodity hardware which reduces not only the original purchase price, but also potentially reduces support costs as well.

2 *Source: http://hadoop.apache.org*

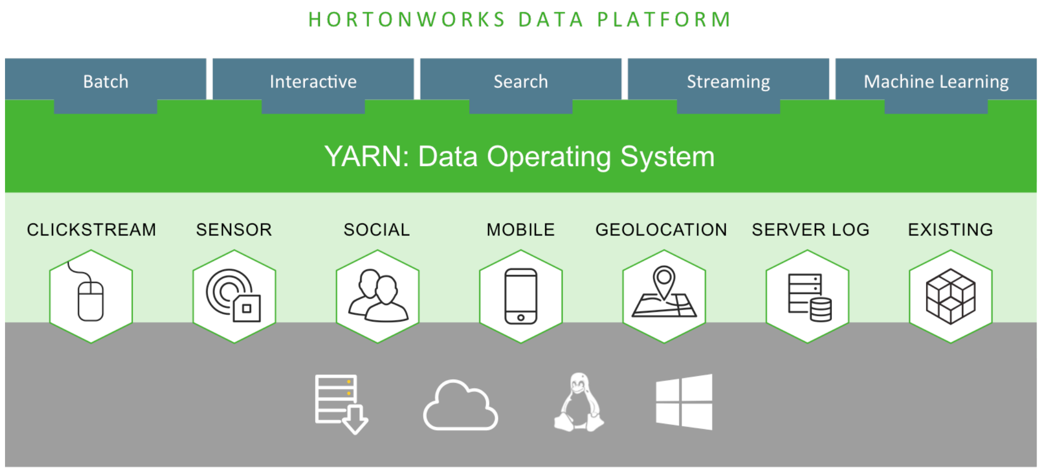
Hadoop Core = Storage + Compute


Yet Another Resource Negotiator (YARN)
Hadoop Distributed File System (HDFS)
CPU
RAM
storage
storage
storage
storage

Hadoop = Storage Plus Compute

At the most granular level, Hadoop is an engine who provides storage via **HDFS** and compute via **YARN** capabilities.

Hortonworks Data Platform



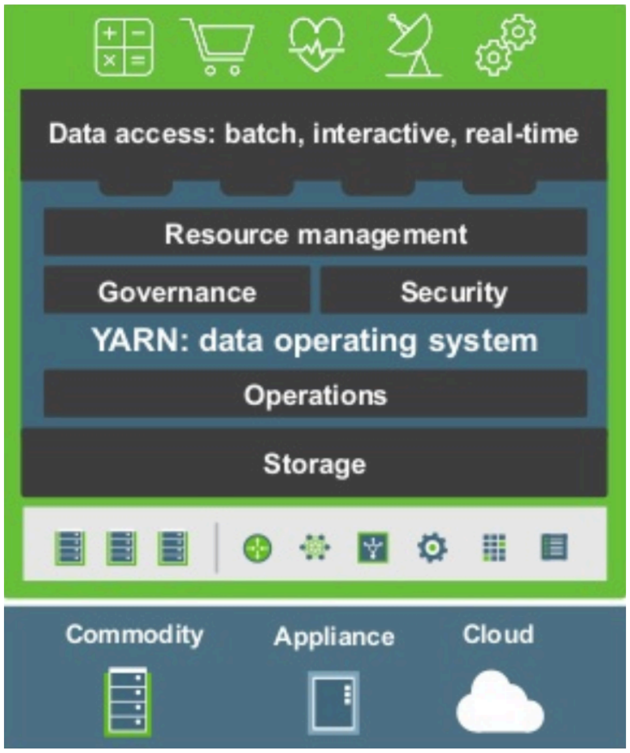
The Hortonworks Data Platform

Hortonworks Data Platform (bottom to top):

- **Deployment options:**
  - On-premises, in the cloud or in a hybrid architecture; on Linux or Windows.
- **It ingests and stores data in its raw form, regardless of its source or format.**
  - This schema-on-read architecture is far more flexible than your familiar schema-on-write databases.

    - ELT, not ETL allows ingestion of not only existing tabular data from your RDBMS, but newer types of less structured data, such as web clickstream, machine and sensors, social media, mobile, geo-location, or server log data.
    - You can also import your existing structured, tabular data into HDP and enrich it with those newer forms of data.
- **The architecture of Hortonworks Data Platform holds YARN at its center.**
    - YARN supports multiple, heterogeneous data access engines that analyze one shared big data set with batch processing, interactive query, search, analysis of streaming data or iterative machine learning approaches. All of these can run simultaneously, centrally managed by YARN.
    - YARN also coordinates cluster resources for enterprise-ready services for cluster operations, data governance and security.

Open Enterprise Hadoop



Open Enterprise Hadoop

Hortonworks leads the emerging category known as Open Enterprise Hadoop. HDP is the Hadoop distribution created and supported by Hortonworks.

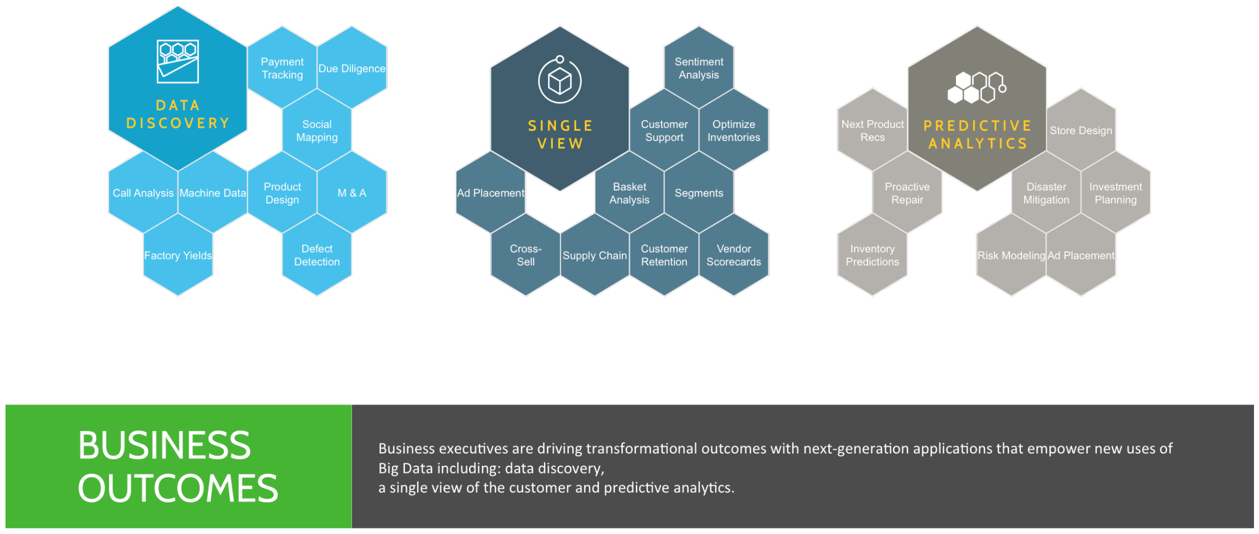
Open Enterprise Hadoop is a new paradigm that scales with the demands of big data applications. It is supported by a rich and growing partner ecosystem that enables enterprises to meet the unique demands of their industries. By making governance, security and operations an integral part of the the platform Open Enterprise Hadoop opens the door for integration with existing enterprise architectures.

All of this is possible because Open Enterprise Hadoop maximizes community innovation by collaborating with developers in open source and within an open community environment. 3

Hadoop Use Cases

Big Data drives a need for new and better analytic applications for new types of data. As businesses embark on a new customer journey to take advantage of these new opportunities, they consider business outcomes and cost savings.

Business Outcomes



BUSINESS OUTCOMES

Business executives are driving transformational outcomes with next-generation applications that empower new uses of Big Data including: data discovery,
a single view of the customer and predictive analytics.

Hadoop Use Case: Business Outcomes

Many turn to Hadoop because they want to pursue business outcomes to transform their businesses. But many feel unsure about how to begin that journey. You can start anywhere you want – no linear path, and it varies from one company to the next.

Data Discovery

Many companies begin their path to Hadoop with data discovery – by exploring data that has never been available to them at scale. With far more data from new sources, accessible via SQL query and other familiar methods, Hortonworks customers quickly uncover new insights.

Single View

Other companies already have more data than they can organize or analyze. These companies use Hadoop to unify all the data and paint a single view of their business. They use Open Enterprise Hadoop to create a single view of the customer; a single view of the patient; a single view of the telecom network; a single view of the supply chain; a single view of the product, or a single view of anything else that the business cares about.

Predictive Analytics

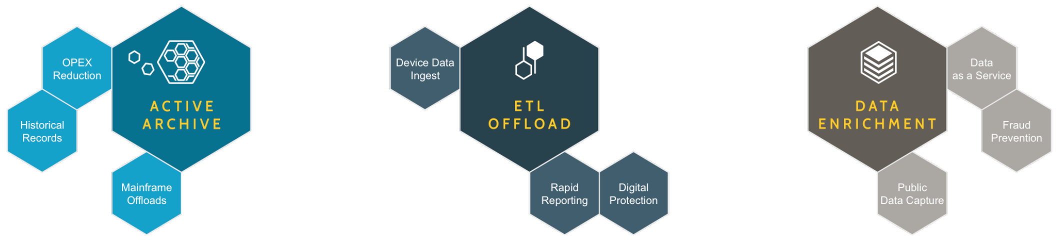
3 *Source: https://www.youtube.com/watch?v=AdOBnCb8Xbg*

The most sophisticated business outcome possible with Open Enterprise Hadoop is predictive analytics. With far more data about what's happened in the past, and advanced machine learning algorithms, companies can to predict future outcomes with far more certainty. They use this capability to: do proactive repairs to equipment, recommend the next product to buy, stock the appropriate amount of inventory, design their stores, or model risk

Cost Savings

COST SAVINGS

IT executives are delivering substantial reductions in operating costs by modernizing their data architectures with Open Enterprise Hadoop. These cost saving innovations include active archive of cold data, offloading ETL processes and enriching existing data.



Hadoop Use Case: Cost Savings

Others begin their journey with a goal of cost savings, most commonly by optimizing their IT architectures.

Active Archive

They move cold data that they hardly ever access into a Hadoop active archive. This helps them save money by lowering their data storage costs, without losing easy access to that data.

ETL Offload

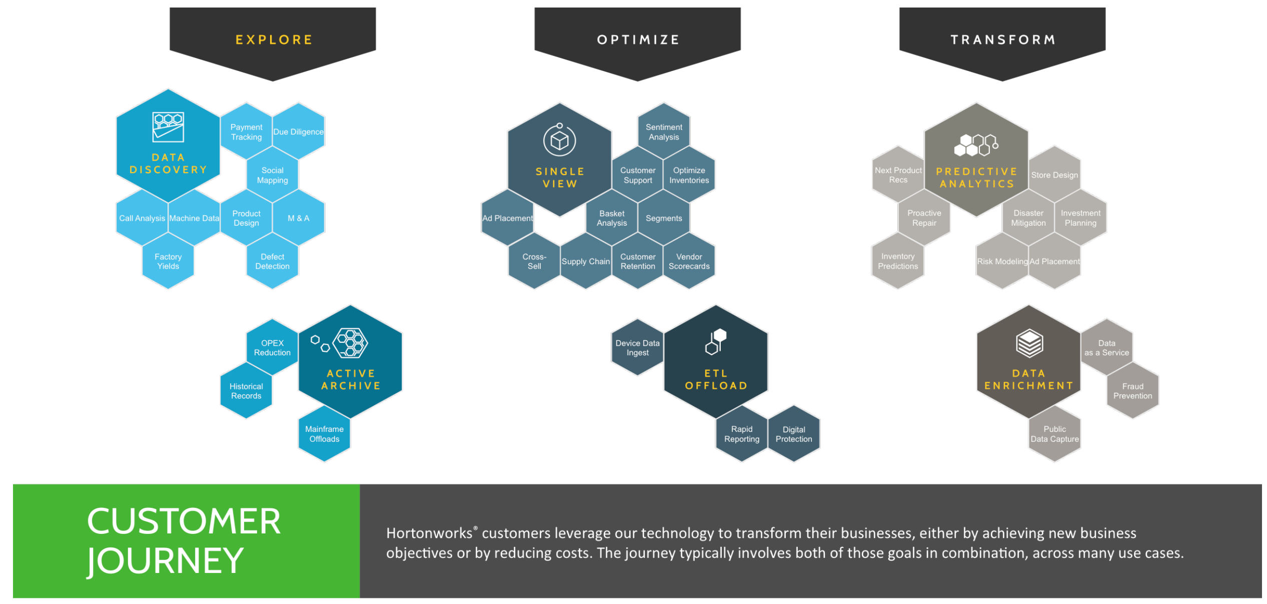
They offload costly and complicated ETL jobs by storing data in its raw form in Hadoop. Hadoop has a schema-on-read architecture, which means you can store data in its native form and then retrieve whatever data you need to analyze, regardless of its source, format or data structure. Since the typical EDW consumes 60% of its cycles just preparing data for a particular schema, ETL offload allows them to use their important EDW resource for what it was intended.

Data Enrichment

The most advanced costs savings use cases are to enrich existing data with newly available data. With these data enrichment capabilities, companies can:

- Incorporate publicly available datasets from sources like Data.gov
- Offer new products for data as a service
- Prevent fraud by finding new data correlations that point to bad behavior

Customer Journey



Hortonworks® customers leverage our technology to transform their businesses, either by achieving new business objectives or by reducing costs. The journey typically involves both of those goals in combination, across many use cases.

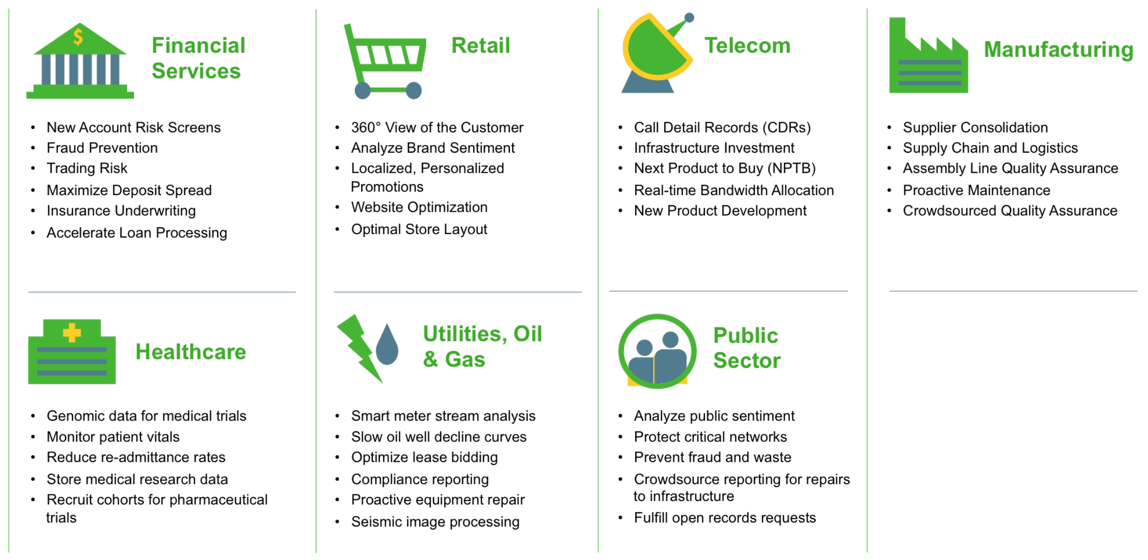
Hadoop Use Case: Customer Journey

Everyone’s Journey to Open Enterprise Hadoop will be Different.

Your journey can start anywhere you want. You can pursue both business outcomes and cost savings. You can start at the most sophisticated use cases if your team is experienced, or you can build your expertise by beginning with less complex use cases that bring quick results. As you build your team’s expertise and comfort with Hadoop, you can then tackle more challenging aspects of your road map.

Hortonworks' customers leverage our technology to transform their businesses, either by achieving new business objectives or by reducing costs. The journay typically involves both of those goals in combination across many use cases.

New Analytic Applications for New Types of Data



Financial Services

- New Account Risk Screens
- Fraud Prevention
- Trading Risk
- Maximize Deposit Spread
- Insurance Underwriting
- Accelerate Loan Processing

Retail

- 360° View of the Customer
- Analyze Brand Sentiment
- Localized, Personalized Promotions
- Website Optimization
- Optimal Store Layout

Telecom

- Call Detail Records (CDRs)
- Infrastructure Investment
- Next Product to Buy (NPTB)
- Real-time Bandwidth Allocation
- New Product Development

Manufacturing

- Supplier Consolidation
- Supply Chain and Logistics
- Assembly Line Quality Assurance
- Proactive Maintenance
- Crowdsourced Quality Assurance

Healthcare

- Genomic data for medical trials
- Monitor patient vitals
- Reduce re-admittance rates
- Store medical research data
- Recruit cohorts for pharmaceutical trials

Utilities, Oil & Gas

- Smart meter stream analysis
- Slow oil well decline curves
- Optimize lease bidding
- Compliance reporting
- Proactive equipment repair
- Seismic image processing

Public Sector

- Analyze public sentiment
- Protect critical networks
- Prevent fraud and waste
- Crowdsource reporting for repairs to infrastructure
- Fulfill open records requests

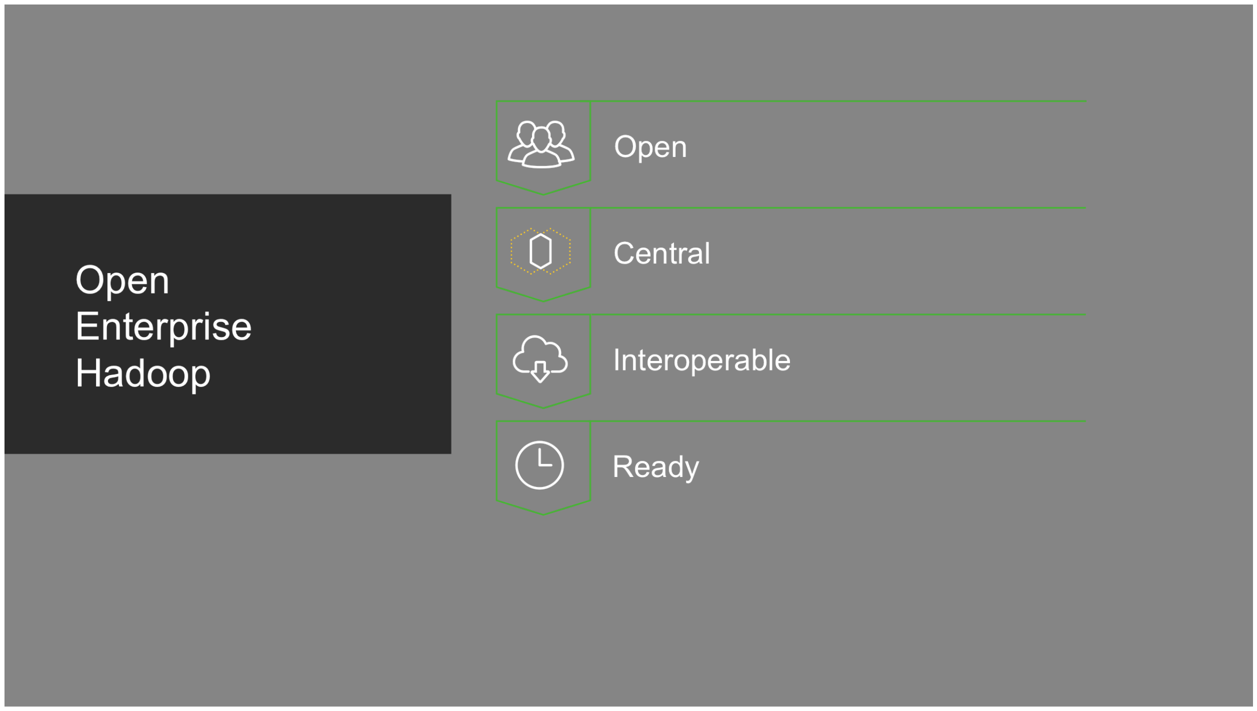
AS the image above shows, Big Data has driven the creation and implementation of new analytical approaches and applications. For example:4

- **Analytics in the cloud**
  Unlimited scalability, ease of use
- **Schema on read**
  Data is applied to a plan (schema) as it is pulled out of a stored location, rather than as it goes in
- **More predictive analytics**
  Traditional analysis depended on samples; now we have the power to process very large amounts of data
- **Faster, better SQL on Hadoop**
  Speed supports iterative analytics - where an analyst asks a question; receives an answer; asks another question based on that answer; and so
- **Deep learning**
  Enables computers to recognize interesting content in huge amounts of unstructured data and deduce relationhips within that data without needing models on which to build instruction
- **In-memory analytics**
  In-memory databases can speed up analytic processessing for some applications

4 8 Big Trends in Big Data Analytics http://www.computerworld.com/article/2690856/big-data/8-big-trends-in-big-data-analytics.html

Open Enterprise Hadoop

Hadoop's characteristics of Open, Central, Interoperable, and Ready makes Hadoop the number one choice for big data solutions.



Hortonworks leads the emerging category known as Open Enterprise Hadoop, with solutions that are:

- 100% Open Source
- Centrally architected with YARN at its core
- Interoperable with existing technology and skills
- Enterprise-ready, with data services for operations, governance and security

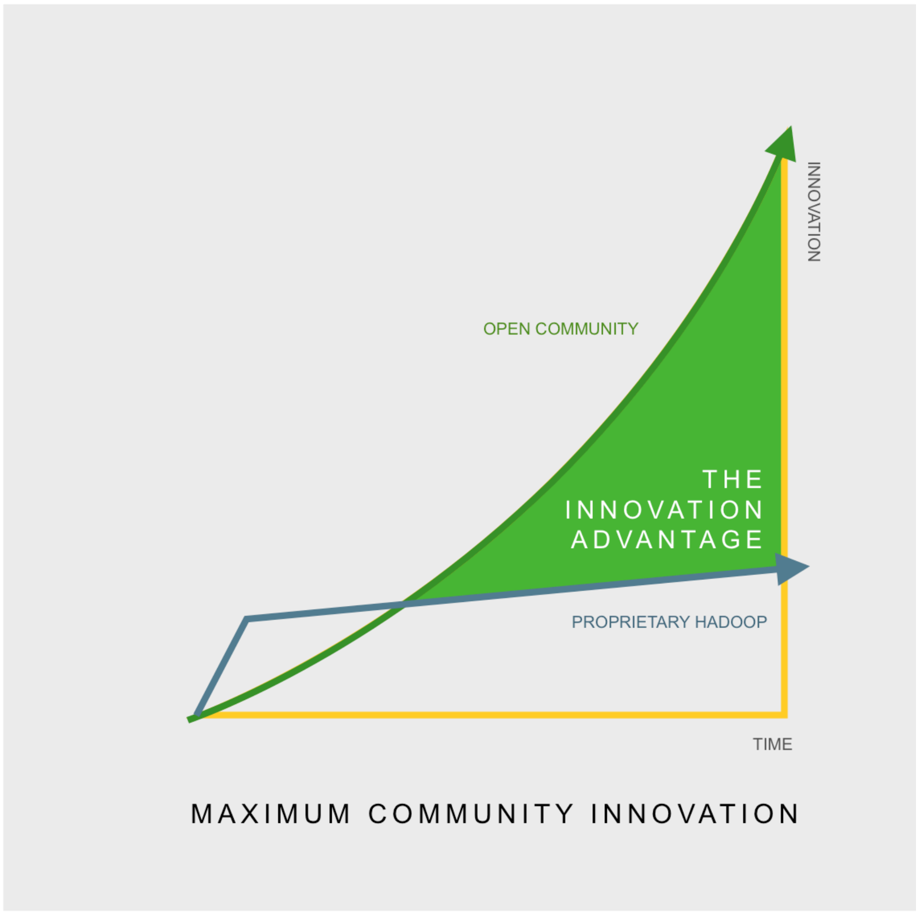
100% Open

The "open" in Open Enterprise Hadoop means that all innovation happens within the open-source processes governed by the Apache Software Foundation.

A 100% open source process is important because:

- It eliminates a customer's risk of becoming locked in to one vendor relationship
- It fosters the fastest, most stable platform innovation
- It allows Hadoop users to influence the Hadoop roadmap

HDP is Genuinely Open

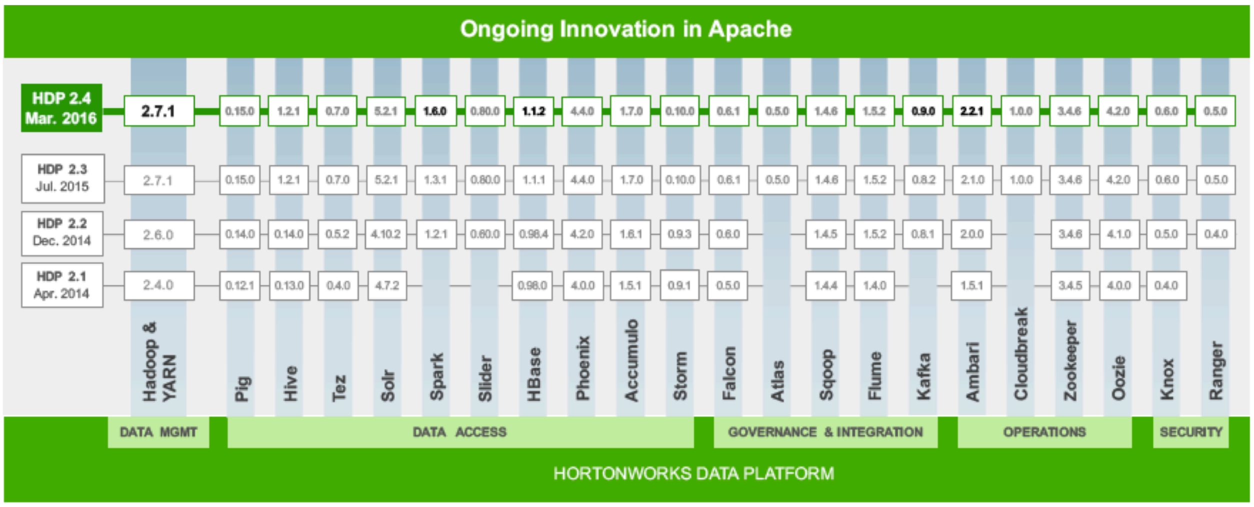


HDP is Open Source

Hortonworks has always followed a 100-percent open approach to Hadoop and Hortonworks Data Platform is the only genuinely open Hadoop distribution. Our open approach:

- Eliminates risk of becoming locked in with one proprietary vendors
- It maximizes innovation by tapping into the ingenuity of the largest number of talented developers with the broadest perspective of the capabilities and areas for improvement
- It integrates seamlessly with other datacenter technologies. Like our customers, our technology partners also want to avoid getting locked in to a proprietary approach.

Fastest Path to Innovation


Ongoing Innovation in Apache
DATA MGMT
DATA ACCESS
GOVERNANCE & INTEGRATION
OPERATIONS
SECURITY
HORTONWORKS DATA PLATFORM

	Hadoop & YARN	Pig	Hive	Tez	Solr	Spark	Slider	HBase	Phoenix	Accumulo	Storm	Falcon	Atlas	Sqoop	Flume	Kafka	Ambari	Cloudbreak	Zookeeper	Oozie	Knox	Ranger
HDP 2.4 Mar. 2016	2.7.1	0.15.0	1.2.1	0.7.0	5.2.1	1.6.0	0.80.0	1.1.2	4.4.0	1.7.0	0.10.0	0.6.1	0.5.0	1.4.6	1.5.2	0.9.0	2.2.1	1.0.0	3.4.6	4.2.0	0.6.0	0.5.0
HDP 2.3 Jul. 2015	2.7.1	0.15.0	1.2.1	0.7.0	5.2.1	1.3.1	0.80.0	1.1.1	4.4.0	1.7.0	0.10.0	0.6.1	0.5.0	1.4.6	1.5.2	0.8.2	2.1.0	1.0.0	3.4.6	4.2.0	0.6.0	0.5.0
HDP 2.2 Dec. 2014	2.6.0	0.14.0	0.14.0	0.5.2	4.10.2	1.2.1	0.60.0	0.98.4	4.2.0	1.6.1	0.9.3	0.6.0		1.4.5	1.5.2	0.8.1	2.0.0		3.4.6	4.1.0	0.5.0	0.4.0
HDP 2.1 Apr. 2014	2.4.0	0.12.1	0.13.0	0.4.0	4.7.2			0.98.0	4.0.0	1.5.1	0.9.1	0.5.0		1.4.4	1.4.0		1.5.1		3.4.5	4.0.0	0.4.0	

HDP = Fastest Path to Innovation

This open approach is the fastest path to innovation.

Here you can see the pace of platform innovation represented by the release cadence across all of the Apache Software Foundation projects that go into HDP.

And remember: these development efforts are broad-based and collaborative with engineers participating from the world's largest, most influential technology companies: Microsoft, Yahoo, Facebook, LinkedIn, Twitter, HP, SAP, SAS and also the mainstream enterprises that use HDP. These include Schlumberger, Target, Merck and Aetna.

Centrally Architected

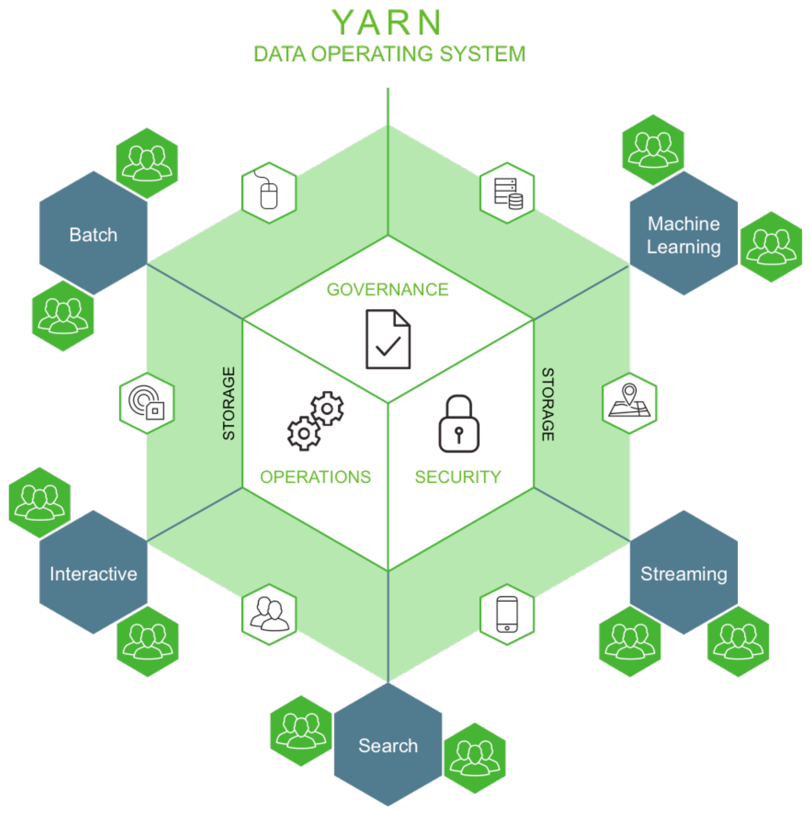
Apache Hadoop YARN is the central Big Data operating system for Open Enterprise Hadoop.

- Many moving parts need to work together for you to make the most of your data.
- YARN orchestrates those moving parts.

Unlike earlier approaches to Hadoop in the enterprise, Open Enterprise Hadoop with YARN at its center:

- Manages your cluster resources with optimal efficiently and
- Provides integrated services for operations, security and data governance that allow you to confidently extend your Big Data assets to the maximum number of users in your company.

Centralized Platform

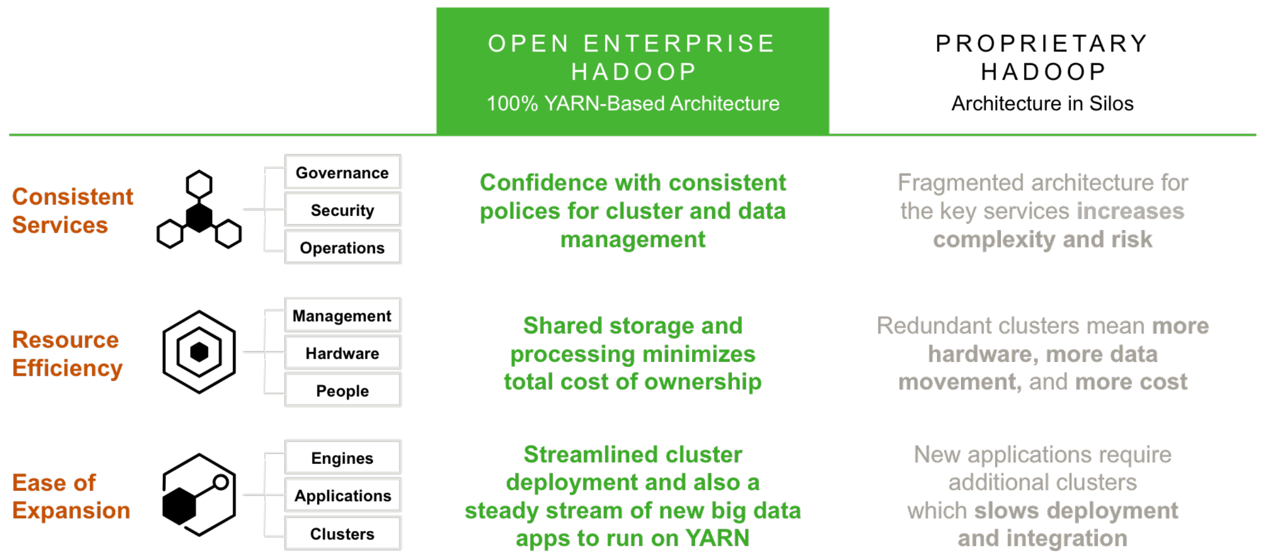


Centralized Platform with YARN-Based Architecture

YARN is the architectural center of Open Enterprise Hadoop. It:

- Coordinates cluster-wide service for operations, data governance and security.
- It allocates resource amongst diverse applications that process the data
- It maximizes your data ingest, by helping ingest all types of data
- And it allows you to confidently extend your big data assets to the largest possible audience within your organization

YARN Architecture Benefits



		OPEN ENTERPRISE HADOOP 100% YARN-Based Architecture	PROPRIETARY HADOOP Architecture in Silos
Consistent Services	Governance Security Operations	**Confidence with consistent polices for cluster and data management**	Fragmented architecture for the key services **increases complexity and risk**
Resource Efficiency	Management Hardware People	**Shared storage and processing minimizes total cost of ownership**	Redundant clusters mean **more hardware, more data movement**, and **more cost**
Ease of Expansion	Engines Applications Clusters	**Streamlined cluster deployment and also a steady stream of new big data apps to run on YARN**	New applications require additional clusters which **slows deployment and integration**

Benefits of the YARN-Based Architecture

Above is a side-by-side comparison of the architectural advantages of Open Enterprise Hadoop as compared to earlier, proprietary approaches to Hadoop in the enterprise

You can see how Open Enterprise Hadoop provides:

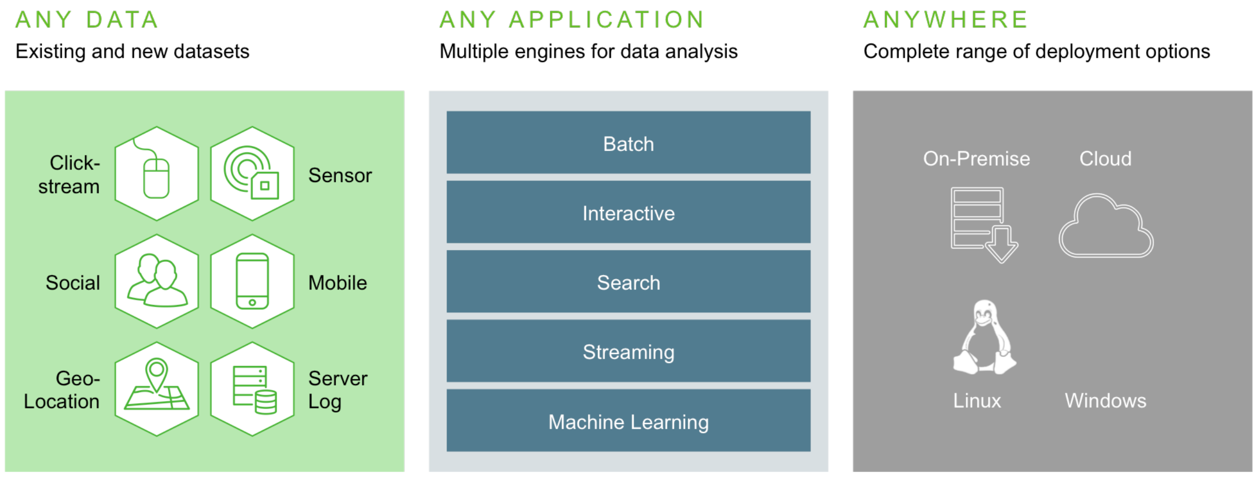
- Consistent services for governance, security and operations versus fragmented policies that increase risk and harm efficiency
- Superior resource efficiency, versus duplicative hardware costs and inefficient data movement
- And ease of expansion as you deploy and grow HDP clusters to meet your business needs.

Interoperable with Existing Technology

Open Enterprise Hadoop is interoperable. It:

- Is flexible to handle any data, for any application, from anywhere
- It evolves in sync with common industry standards for Hadoop
- It integrates seamlessly with preferred data platform technologies, processing engines and BI tools already used by the world's leading businesses.

Offers the Most Flexibility



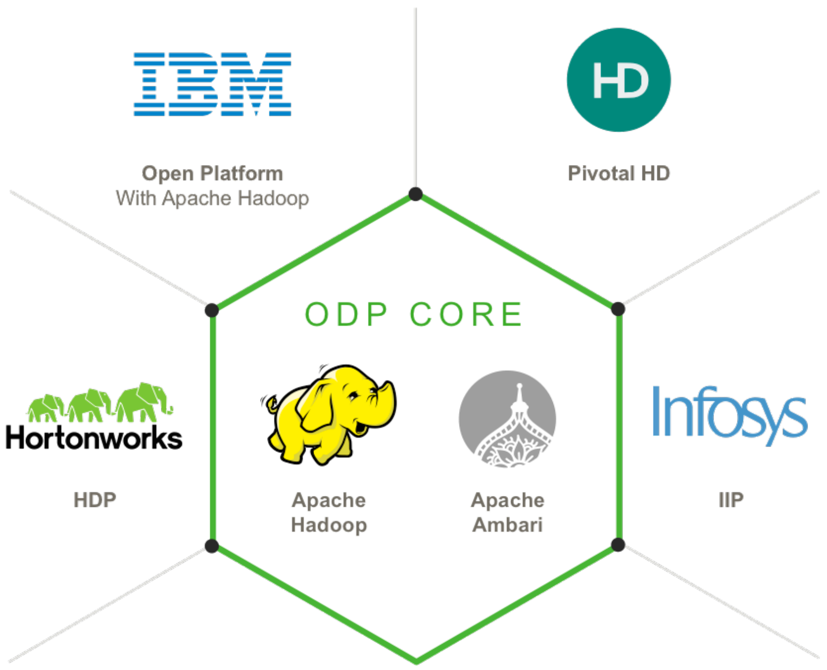
Flexibility for the Customer Journey

On the first day of your Hadoop journey, you can't know exactly what the future will hold. So you need flexibility.

Platform interoperability and flexibility free you to:

- Capture any data and store it for as long as you need it
- Analyze data for any application you already use, or others that you might create in the future
- Explore your data with any combination of batch, interactive, search, streaming analytics and machine learning – all at once
- Deploy these capabilities however you like, and change those deployments whenever it suits your needs

Synchronized with Industry Standards



Enterprise Hadoop is Synchronized with Industry Standards

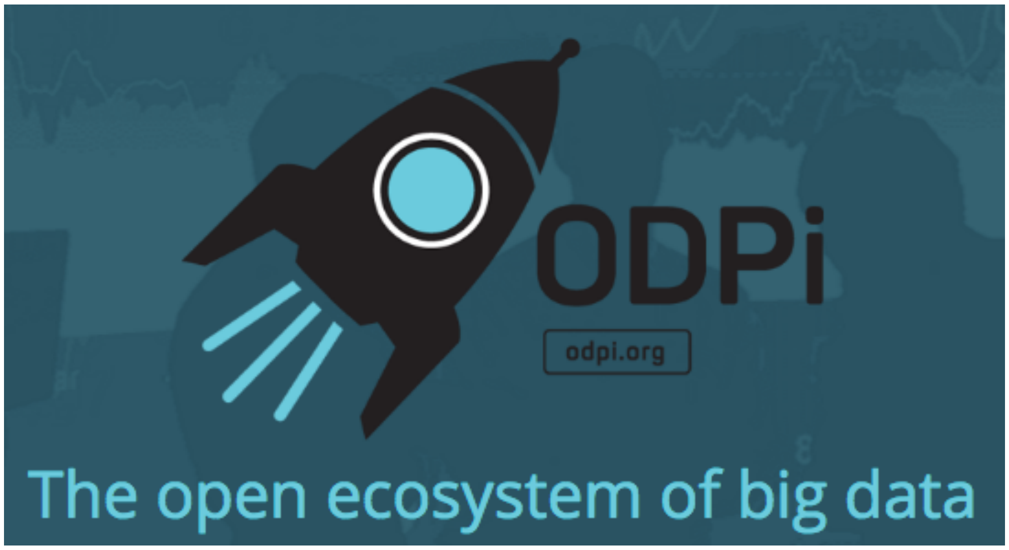
Open Enterprise Hadoop protects that flexibility while you're on your journey, by adhering to core standards defined by the Open Data Platform initiative.

The ODP:

- Improves ecosystem interoperability so that you can pick and choose components from across the ecosystem.
- This unlocks choice and lets you run Pivotal Hawq or IBM BigInsights with Hortonworks Data Platform.
- It also means that your architects don't have to implement difficult integrations across multiple system versions.

Open Data Platform (ODP) Initiative

Open Enterprise Hadoop is synchronized with industry standards through the Open Data Platform (ODP) initiative.



- **Why this matters to our customers:** The ODP standardizes multiple Hadoop deployments on the same core versions of Apache Hadoop and Apache Ambari. This helps the enterprise pick and choose the components that work best in combination.
- **Proof point:** For example Pivotal Hawq can run on HDP and IBM BigInsights can run on HDP. This allows each player to focus on developing the components for which it has the best expertise, which speeds ecosystem innovation. | Source: http://pivotal.io/big-data/press-release/pivotal-hawq-now-certified-on-hortonworks-data-platform | *Source: http://www.prnewswire.co.uk/news-releases/hortonworks-ibm-and-pivotal-harmonize-on-open-data-platform-vision-to-accelerate-big-data-solutions-499672871.html*
- **IBM Citation:** "*IBM is a long time open source leader and community participant. The Open Data Platform continues this heritage by ensuring a strong foundation for the future of Hadoop. As a founding member, IBM is taking a central role to ensure a high degree of compatibility with its portfolio of products including SPSS, Cognos, and BigInsights as well as an expanding ecosystem of Hadoop partners. IBM Open Platform 4.0 is part of the next generation heterogeneous computing platform to make in-time analytics pervasive across the business.*" -- Anjul Bhambhri, Vice President of IBM Analytics Platform, IBM
- **Pivotal Citation:** "*To move forward, Pivotal believes that the big data market requires a standard, predictable, and mature Hadoop-based core platform for modern management solutions. The quick momentum we have around the Open Data Platform is already taking the guesswork out of fragmented and duplicative processes of what works and what doesn't. This will enable enterprises and ecosystem vendors to focus on integrating and building business driven applications and use cases that drive innovation.*" -- Leo Spiegel, Senior Vice President of Corporate Development and Strategy, Pivotal

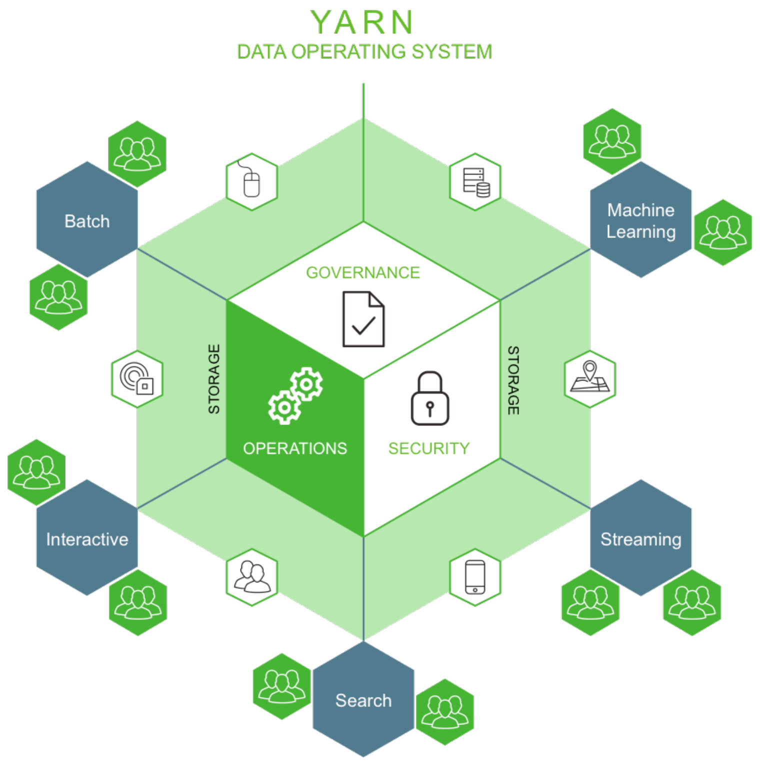
REFERENCE

The open ecosystem of big data was created to complement the Apache Software Foundation and help companies use Apache Hadoop more effectively. See: https://www.odpi.org/

Enterprise Ready

Open Enterprise Hadoop is enterprise-ready with enterprise-grade services for operations, data governance and security.

Provides Consistent Operations

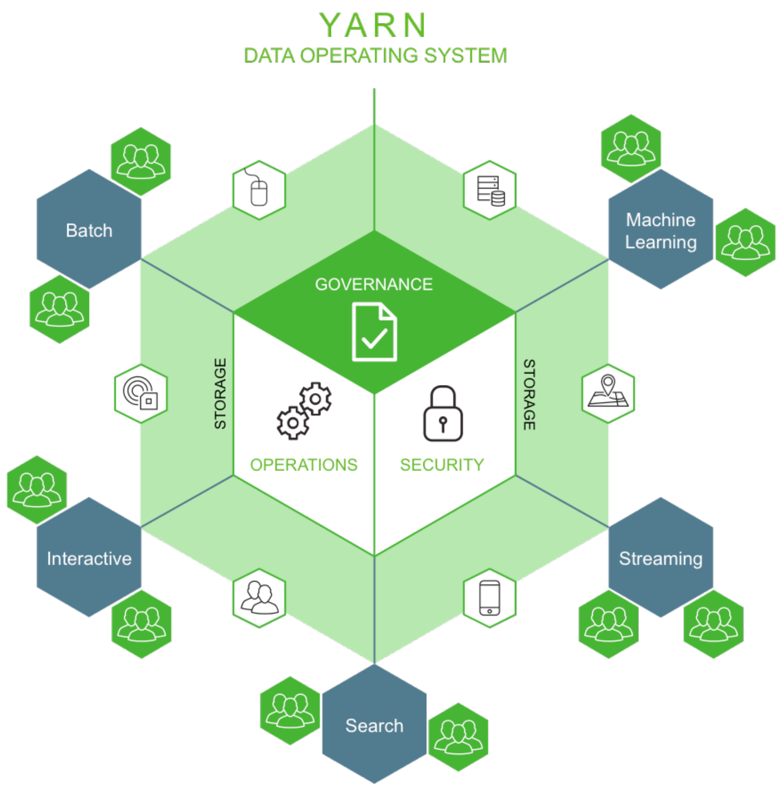


YARN Provides Consistent Operations

Open Enterprise Hadoop provides consistent operations, with:

- **Centralized management and monitoring** of clusters through a single pane of glass
- **Automated provisioning**, either on-premises or in the cloud with the Cloudbreak API. You can mange one huge data lake, or spin up and spin down multiple clusters as needed. You choose. And also,
- **Managed services** to make sure that your cluster is highly available.

Enables Trusted Governance



YARN Enables Trusted Governance

Open Enterprise Hadoop enables trusted governance, with:

- **Data lifecycle management** along the entire lifecycle
- **Modeling with metadata, and**
- **Interoperable solutions** that can access a common metadata store.

Trusted Governance

- Why this matters to our customers: As data accumulates in an HDP cluster, the enterprise needs governance policies to control how that data is ingested, transformed and eventually retired. This keeps those Big Data assets from turning into big liabilities that you can't control.
-
- Proof point: HDP includes 100% open source Apache Atlas and Apache Falcon for centralized data governance coordinated by YARN. These data governance engines provide those mature data management and metadata modeling capabilities, and they are constantly strengthened by members of the Data Governance Initiative. The Data Governance Initiative (DGI) is working to develop an extensible foundation that addresses enterprise requirements for comprehensive data governance. The DGI coalition includes Hortonworks partner SAS and customers Merck, Target, Aetna and Schlumberger. Together, we assure that Hadoop:
    - Snaps into existing frameworks to openly exchange metadata
    - Addresses enterprise data governance requirements within its own stack of technologies

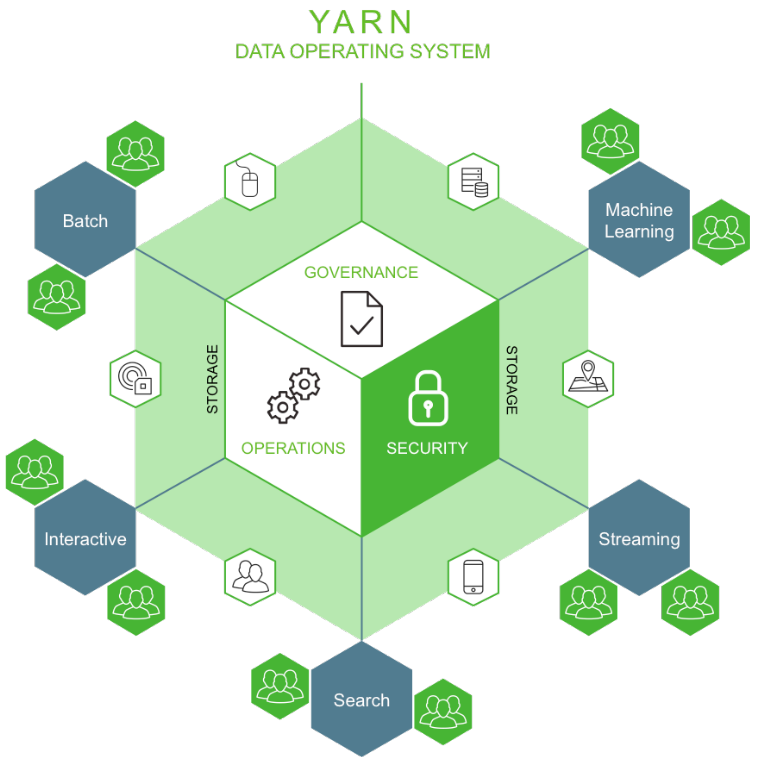
REFERENCE:

"As customers are moving Hadoop into corporate data and processing environments, metadata and data governance are much needed capabilities. SAS participation in this initiative strengthens the integration of SAS data management, analytics and visualization into the HDP environment and more broadly it helps advance the Apache Hadoop project. This additional integration will give customers better ability to manage big data governance within the Hadoop framework."

SAS Vice President of Product Management Randy Guard.

http://hortonworks.com/press-releases/hortonworks-establishes-data-governance-initiative/

Ensures Comprehensive Security



YARN Ensures Comprehensive Security

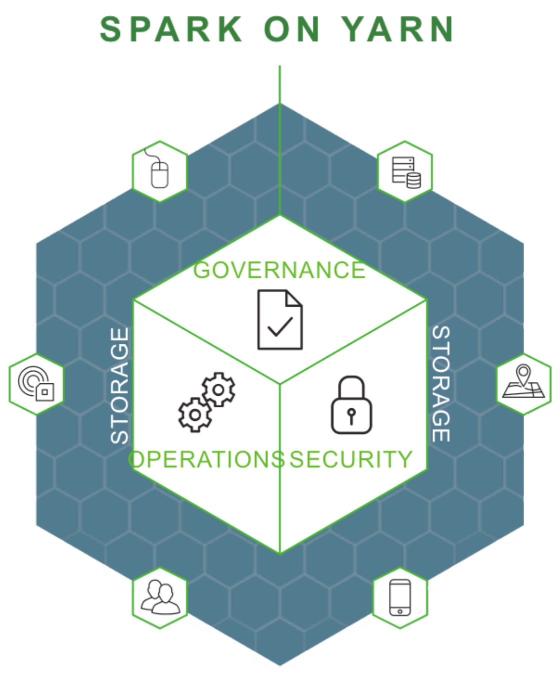
Enterprise-readiness means comprehensive security through a platform approach.

This includes:

- **Encryption of data** at rest and in motion
- **Centralized administration** of polices for authentication, and
- **Fine-grained authorization** to control data access.

These are the four pillars of the emerging solution category known as Open Enterprise Hadoop.

Agile Analytics with Enterprise Spark at Scale

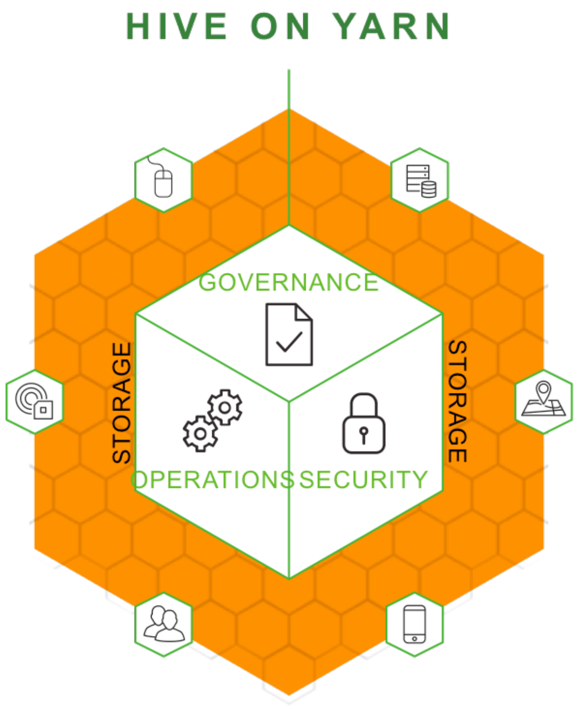


Agile Analytics with Spark

Enterprise Spark powers agile analytics via data science notebooks and automation for most common analytics (including geospatial and entity resolution) and seamless data access across as many data types as possible.

It also provides unmatched economics, combining in-memory processing speed with HDP's cost efficiencies at scale. It is ready for the Enterprise with robust security, governance and operations coordinated centrally by Apache Hadoop and YARN.

Fast SQL with Apache Hive



Achieve Fast SQL with Apache Hive

While Spark at Scale is new and hot, SQL is still the *lingua franca* of data analysis and is critical to the "Enterprise Ready" message.

Spark is a Pluggable Architecture that supports Apache Hive, Pivotal HAWQ and other leading SQL engines which provide familiar SQL Query Semantics and enable transactions and SQL:2011 Analytics for rich reporting.

All this at unprecedented speed at extreme scale that returns query results in interactive time, even as data sets grow to petabytes.

Why Hortonworks?

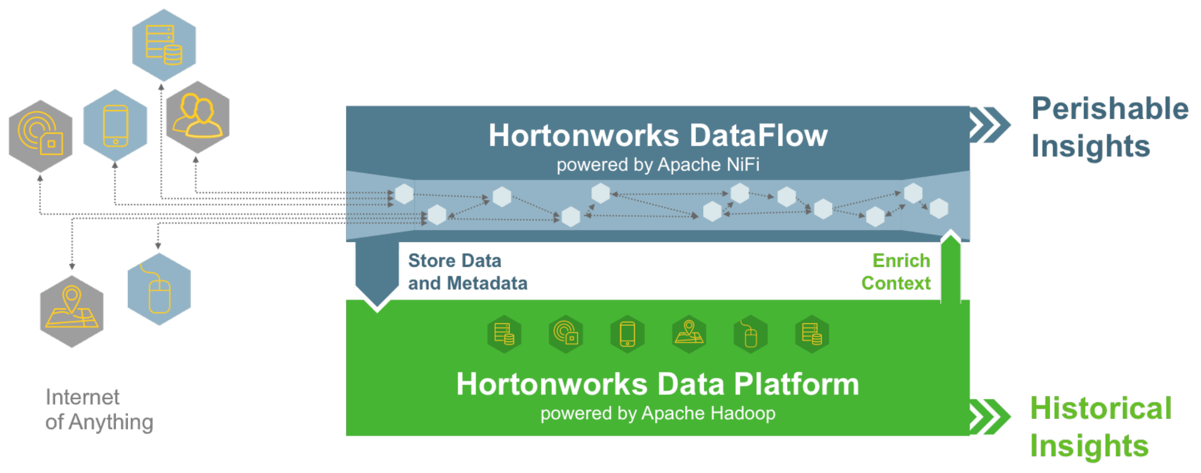
All of these features and technologies add up to why Hortonworks is your best choice for Open Enterprise operations.

Only Hortonworks Delivers Open Enterprise Hadoop

Hortonworks uniquely delivers Open Enterprise Hadoop because Hortonworks Data Platform is:

- 100% Open
- Centrally Architected around YARN
- Interoperable with top technologies and prevailing skills, and
- It is ready for broad deployment in the enterprise.

Hortonworks Data Flow



Hortonworks Data Flow (HDF)

Hortonworks DataFlow and Hortonworks Data Platform – referred to as Connected Data Platforms by Hortonworks – deliver the industry's most complete solution for Big Data management.

Summary

- The 3V's of Big Data are driving the adoption of Apache Hadoop (44 ZB by 2020)
- Existing data architectures make data inaccessible, incomplete, irrelevant, and expensive
- Hadoop is a scalable, fault tolerant, open source framework for the distributed storing and processing of large sets of data on commodity hardware
- Six common use case families have emerged
  - Data Discovery
  - Single View
  - Predictive Analytics
  - Active Archive
  - ETL Offload
  - Data Enrichment
- YARN-centralized HDP = Open Enterprise Hadoop

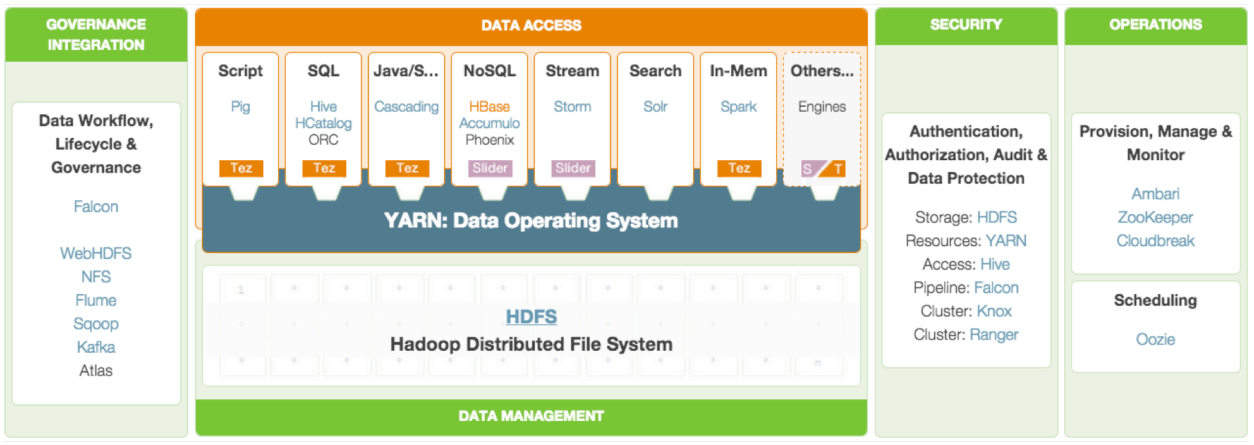
The Hadoop Ecosystem

Lesson Objectives

After completing this lesson, students should be able to:

- ✓ Describe the Hadoop ecosystem frameworks
- ✓ Deploy Hadoop scalability in a datacenter

Hadoop Ecosystem Frameworks



Hadoop Frameworks

Hadoop is not a monolithic piece of software. It is a collection of architectural pillars that contain software frameworks. Most of the frameworks are part of the Apache software ecosystem. The picture illustrates the Apache frameworks that are part of the Hortonworks Hadoop distribution.

So why does Hadoop have so many frameworks and tools? The reason is that each tool is designed for a specific purpose. The functionality of some tools overlap but typically one tool is going to be better than others when performing certain tasks.

For example, both Apache Storm and Apache Flume ingest data and perform real-time analysis. But Storm has more functionality and is more powerful for real-time data analysis.

Data Management Frameworks

Framework	Description
Hadoop Distributed File System (HDFS)	A Java-based, distributed file system that provides scalable, reliable, high-throughput access to application data stored across commodity servers
Yet Another Resource Negotiator (YARN)	A framework for cluster resource management and job scheduling

Hadoop Data Management Frameworks

There are two data management frameworks: HDFS and YARN.

HDFS is a Java-based distributed file system that provides scalable, reliable, high-throughput access to application data stored across commodity servers. HDFS is similar to many conventional file systems. For example, it shares many similarities to the Linux file system. HDFS supports operations to read, write, and delete files. It supports operations to create, list, and delete directories.

YARN is a framework for cluster resource management and job scheduling. YARN is the architectural center of Hadoop and enables multiple data processing engines such as interactive SQL, real-time streaming, data science, and batch processing to co-exist on a single cluster.

Operations Frameworks

Framework	Description
Ambari	A Web-based framework for provisioning, managing, and monitoring Hadoop clusters
ZooKeeper	A high-performance coordination service for distributed applications
Cloudbreak	A tool for provisioning and managing Hadoop clusters in the cloud
Oozie	A server-based workflow engine used to execute Hadoop jobs

Hadoop Operations Frameworks

The four operations frameworks are Apache Ambari, Apache ZooKeeper, Cloudbreak, and Apache Oozie.

Ambari is a completely open operational framework for provisioning, managing, and monitoring Hadoop clusters. It includes an intuitive collection of operator tools and a set of RESTful APIs that mask the complexity of Hadoop, simplifying the operation of clusters. The most visible Ambari component is the Ambari Web UI, a Web-based interface used to provision, manage, and monitor Hadoop clusters. The Ambari Web UI is the “face” of Hadoop management.

ZooKeeper is a coordination service for distributed applications and services. Coordination services are hard to build correctly, and are especially prone to errors such as race conditions and deadlock. In addition, a distributed system must be able to conduct coordinated operations while dealing with such things as scalability concerns, security concerns, consistency issues, network outages, bandwidth limitations, and synchronization issues. ZooKeeper is designed to help with these issues.

Cloudbreak is a cloud agnostic tool for provisioning, managing, and monitoring of on-demand clusters. It automates the launching of elastic Hadoop clusters with policy-based autoscaling on the major cloud infrastructure platforms including Microsoft Azure, Amazon Web Services, Google Cloud Platform, OpenStack, and Docker containers.

Oozie is a server-based workflow engine used to execute Hadoop jobs. Oozie enables Hadoop users to build and schedule complex data transformations by combining MapReduce, Apache Hive, Apache Pig, and Apache Sqoop jobs into a single, logical unit of work. Oozie can also perform Java, Linux shell, distcp, SSH, email, and other operations.

Data Access Frameworks

Framework	Description
Pig	A high-level platform for extracting, transforming, or analyzing large datasets
Hive	A data warehouse infrastructure that supports ad hoc SQL queries
HCatalog	A table information, schema, and metadata management layer supporting Hive, Pig, MapReduce, and Tez processing
Cascading	An application development framework for building data applications, abstracting the details of complex MapReduce programming
HBase	A scalable, distributed NoSQL database that supports structured data storage for large tables
Phoenix	A client-side SQL layer over HBase that provides low-latency access to HBase data
Accumulo	A low-latency, large table data storage and retrieval system with cell-level security
Storm	A distributed computation system for processing continuous streams of real-time data
Solr	A distributed search platform capable of indexing petabytes of data
Spark	A fast, general purpose processing engine use to build and run sophisticated SQL, streaming, machine learning, or graphics applications

Hadoop Data Access Frameworks

Apache Pig is a high-level platform for extracting, transforming, or analyzing large datasets. Pig includes a scripted, procedural-based language that excels at building data pipelines to aggregate and add structure to data. Pig also provides data analysts with tools to analyze data.

Apache Hive is a data warehouse infrastructure built on top of Hadoop. It was designed to enable users with database experience to analyze data using familiar SQL-based statements. Hive includes support for SQL:2011 analytics. Hive and its SQL-based language enable an enterprise to utilize existing SQL skillsets to quickly derive value from a Hadoop deployment.

Apache HCatalog is a table information, schema, and metadata management system for Hive, Pig, MapReduce, and Tez. HCatalog is actually a module in Hive that enables non-Hive tools to access Hive metadata tables. It includes a REST API, named WebHCat, to make table information and metadata available to other vendors' tools.

Cascading is an application development framework for building data applications. Acting as an abstraction layer, Cascading converts applications built on Cascading into MapReduce jobs that run on top of Hadoop.

Apache HBase is a non-relational, or NoSQL, database. HBase was created to host very large tables with billions of rows and millions of columns. HBase provides random, real-time access to data. It adds some transactional capabilities to Hadoop, allowing users to conduct table inserts, updates, scans, and deletions.

Apache Phoenix is a client-side SQL skin over HBase that provides direct, low-latency access to HBase. Entirely written in Java, Phoenix enables querying and managing HBase tables using SQL commands.

Apache Accumulo is a low-latency, large table data storage and retrieval system with cell-level security. Accumulo is based on Google's Bigtable but it runs on YARN.

Apache Storm is a distributed computation system for processing continuous streams of real-time data. Storm augments the batch processing capabilities provided by MapReduce and Tez by adding reliable, real-time data processing capabilities to a Hadoop cluster.

Apache Solr is a distributed search platform capable of indexing petabytes of data. Solr provides user-friendly, interactive search to help businesses find data patterns, relationships, and correlations across petabytes of data. Solr ensures that all employees in an organization, not just the technical ones, can take advantage of the insights that Big Data can provide.

Apache Spark is an open source, general purpose processing engine that allows data scientists to build and run fast, sophisticated applications on Hadoop. Spark provides a set of simple and easy-to-understand programming APIs that are used to build applications at a rapid pace in Scala. The Spark Engine supports a set of high-level tools that support SQL-like queries, streaming data applications, complex analytics such as machine learning, and graph algorithms.

Governance and Integration Frameworks

Framework	Description
Falcon	A data governance tool providing workflow orchestration, data lifecycle management, and data replication services.
WebHDFS	A REST API that uses the standard HTTP verbs to access, operate, and manage HDFS
HDFS NFS Gateway	A gateway that enables access to HDFS as an NFS mounted file system
Flume	A distributed, reliable, and highly-available service that efficiently collects, aggregates, and moves streaming data
Sqoop	A set of tools for importing and exporting data between Hadoop and RDBM systems
Kafka	A fast, scalable, durable, and fault-tolerant publish-subscribe messaging system
Atlas	A scalable and extensible set of core governance services enabling enterprises to meet compliance and data integration requirements

Hadoop Governance and Integration Frameworks

Apache Falcon is a data governance tool. It provides a workflow orchestration framework designed for data motion, coordination of data pipelines, lifecycle management, and data discovery. Falcon enables data stewards and Hadoop administrators to quickly onboard data and configure its associated processing and management on Hadoop clusters.

WebHDFS uses the standard HTTP verbs GET, PUT, POST, and DELETE to access, operate, and manage HDFS. Using WebHDFS, a user can create, list, and delete directories as well as create, read, append, and delete files. A user can also manage file and directory ownership and permissions.

The **HDFS NFS Gateway** allows access to HDFS as though it were part of an NFS client's local file system. The NFS client mounts the root directory of the HDFS cluster as a volume and then uses local command-line commands, scripts, or file explorer applications to manipulate HDFS files and directories.

Apache Flume is a distributed, reliable, and available service that efficiently collects, aggregates, and moves streaming data. It is a distributed service because it can can be deployed across many systems. The benefits of a distributed system include increased scalability and redundancy. It is reliable because its architecture and components are designed to prevent data loss. It is highly-available because it uses redundancy to limit downtime.

Apache Sqoop is a collection of related tools. The primary tools are the import and export tools. Writing your own scripts or MapReduce program to move data between Hadoop and a database or an enterprise data warehouse is an error prone and non-trivial task. Sqoop import and export tools are designed to reliably transfer data between Hadoop and relational databases or enterprise data warehouse systems.

Apache Kafka is a fast, scalable, durable, and fault-tolerant publish-subscribe messaging system. Kafka is often used in place of traditional message brokers like Java Messaging Service (JMS) or Advance Message Queuing Protocol (AMQP) because of its higher throughput, reliability, and replication.

Apache Atlas is a scalable and extensible set of core foundational governance services that enable and enterprise to meet their compliance requirements within Hadoop and enables integration with the complete enterprise data ecosystem.

Security Frameworks

Framework	Description
HDFS	A storage management service providing file and directory permissions, even more granular file and directory access control lists, and transparent data encryption
YARN	A resource management service with access control lists controlling access to compute resources and YARN administrative functions
Hive	A data warehouse infrastructure service providing granular access controls to table columns and rows
Falcon	A data governance tool providing access control lists that limit who may submit Hadoop jobs
Knox	A gateway providing perimeter security to a Hadoop cluster
Ranger	A centralized security framework offering fine-grained policy controls for HDFS, Hive, HBase, Knox, Storm, Kafka, and Solr

Hadoop Security Frameworks

HDFS also contributes security features to Hadoop. HDFS includes file and directory permissions, access control lists, and transparent data encryption. Access to data and services often depends on having the correct HDFS permissions and encryption keys.

YARN also contributes security features to Hadoop. YARN includes access control lists that control access to cluster memory and CPU resources, along with access to YARN administrative capabilities.

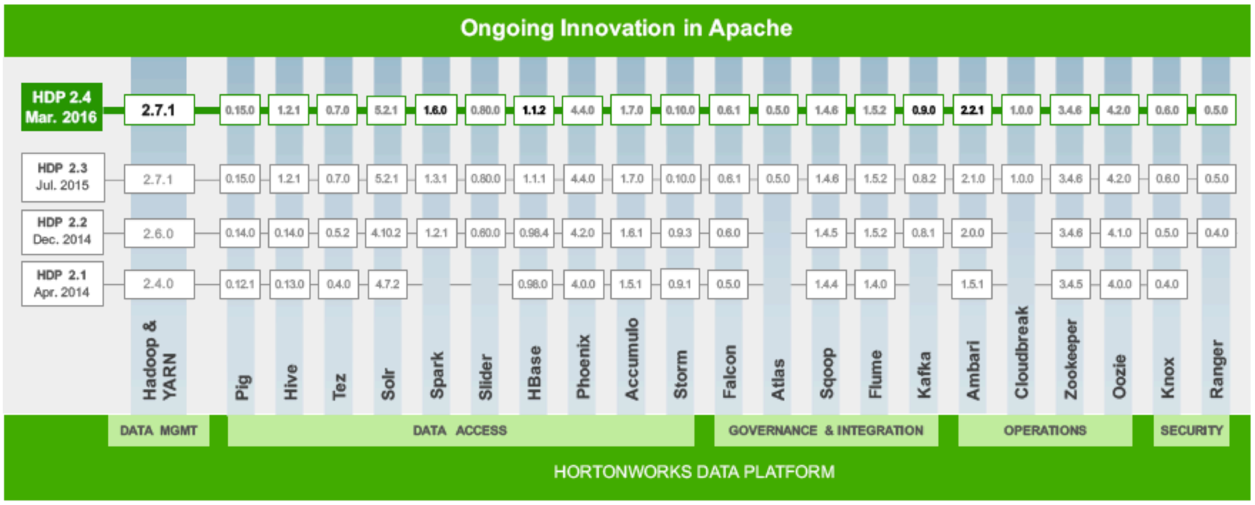
Apache Hive can be configured to control access to table columns and rows.

Apache Falcon is a data governance tool that also includes access controls that limit who may submit automated workflow jobs on a Hadoop cluster.

Apache Knox is a perimeter gateway protecting a Hadoop cluster. It provides a single point of authentication into a Hadoop cluster.

Apache Ranger is a centralized security framework offering fine-grained policy controls for HDFS, Hive, HBase, Knox, Storm, Kafka, and Solr. Using the Ranger Console, security administrators can easily manage policies for access to files, directories, databases, tables, and columns. These policies can be set for individual users or groups and then enforced within Hadoop.

Ecosystem Component Versions



Ongoing Innovation in Apache

	DATA MGMT	DATA ACCESS									
	Hadoop & YARN	Pig	Hive	Tez	Solr	Spark	Slider	HBase	Phoenix	Accumulo	Storm
HDP 2.4 Mar. 2016	2.7.1	0.15.0	1.2.1	0.7.0	5.2.1	1.6.0	0.80.0	1.1.2	4.4.0	1.7.0	0.10.0
HDP 2.3 Jul. 2015	2.7.1	0.15.0	1.2.1	0.7.0	5.2.1	1.3.1	0.80.0	1.1.1	4.4.0	1.7.0	0.10.0
HDP 2.2 Dec. 2014	2.6.0	0.14.0	0.14.0	0.5.2	4.10.2	1.2.1	0.60.0	0.98.4	4.2.0	1.6.1	0.9.3
HDP 2.1 Apr. 2014	2.4.0	0.12.1	0.13.0	0.4.0	4.7.2			0.98.0	4.0.0	1.5.1	0.9.1

	GOVERNANCE & INTEGRATION					OPERATIONS				SECURITY	
	Falcon	Atlas	Sqoop	Flume	Kafka	Ambari	Cloudbreak	Zookeeper	Oozie	Knox	Ranger
HDP 2.4 Mar. 2016	0.6.1	0.5.0	1.4.6	1.5.2	0.9.0	2.2.1	1.0.0	3.4.6	4.2.0	0.6.0	0.5.0
HDP 2.3 Jul. 2015	0.6.1	0.5.0	1.4.6	1.5.2	0.8.2	2.1.0	1.0.0	3.4.6	4.2.0	0.6.0	0.5.0
HDP 2.2 Dec. 2014	0.6.0		1.4.5	1.5.2	0.8.1	2.0.0		3.4.6	4.1.0	0.5.0	0.4.0
HDP 2.1 Apr. 2014	0.5.0		1.4.4	1.4.0		1.5.1		3.4.5	4.0.0	0.4.0	

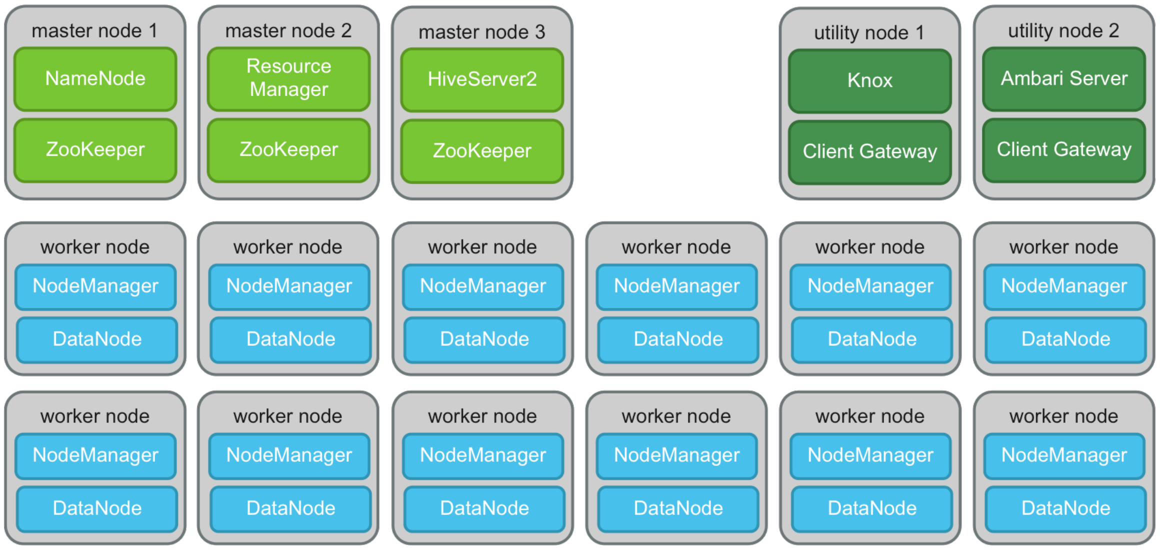
HORTONWORKS DATA PLATFORM

For someone evaluating Hadoop, the considerably large list of components in the Hadoop ecosystem can be overwhelming. Above is a reference table with keywords you may have heard in discussions concerning Hadoop as well as a brief description.

Hadoop Scalability in the Datacenter

Hadoop integrates easily with Datacenters, providing effective workload management, great scalability, monitoring, data integration, and security.

Distinct Masters and Scale-Out Workers



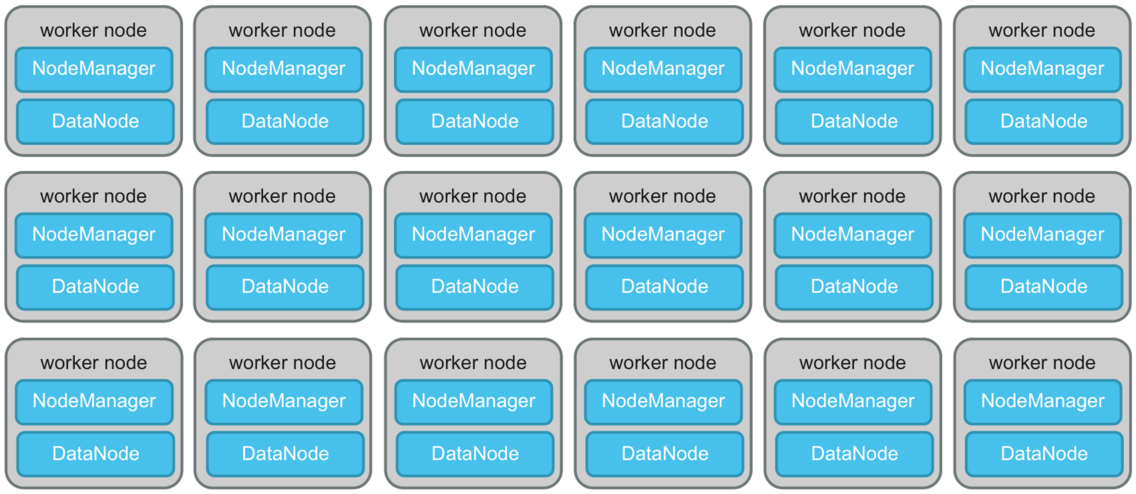
Masters and Workers

Here is an example cluster with three master nodes, 12 worker nodes, and two utility nodes. The cluster is running various services, like YARN and HDFS. Services can be implemented by one or more service components.

- The three master nodes are running service master components.
- The 12 worker nodes are running service worker components, sometimes called slave components.
- The two utility nodes are running service components that provide access, security, and management services for the cluster.

This illustration does not illustrate all services, service master, or service worker components.

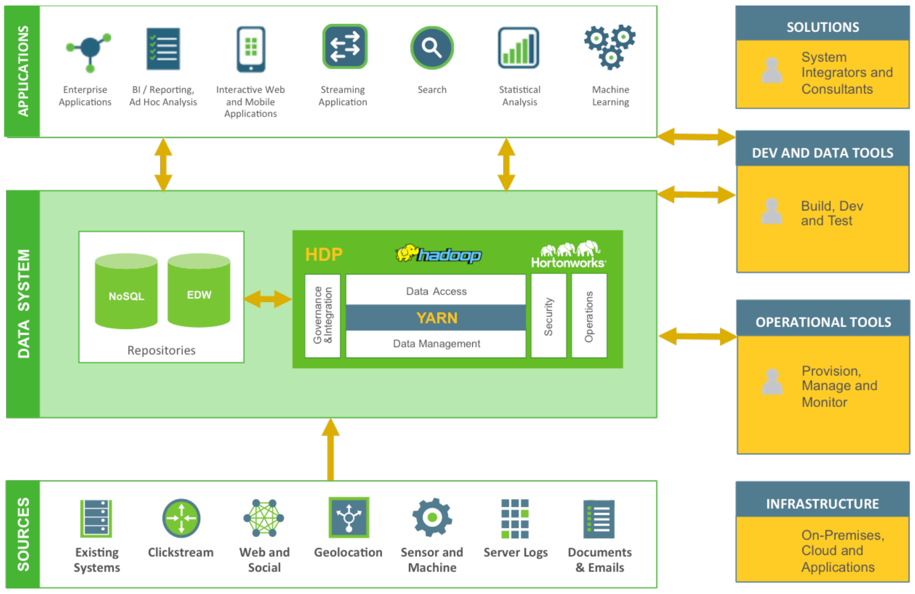
Worker Nodes Scale to the Thousands



Worker Nodes Scale to the Thousands

Hadoop clusters keep scaling out into the thousands of nodes. With all this hardware, failures will occur and Hadoop is designed to account for them and self-heal.

Connected Data Platforms



Connected Data Platforms

With the continued growth in scope and scale of analytics applications using Hadoop and other data sources, the vision of an enterprise data lakeData Lakecan become a reality.

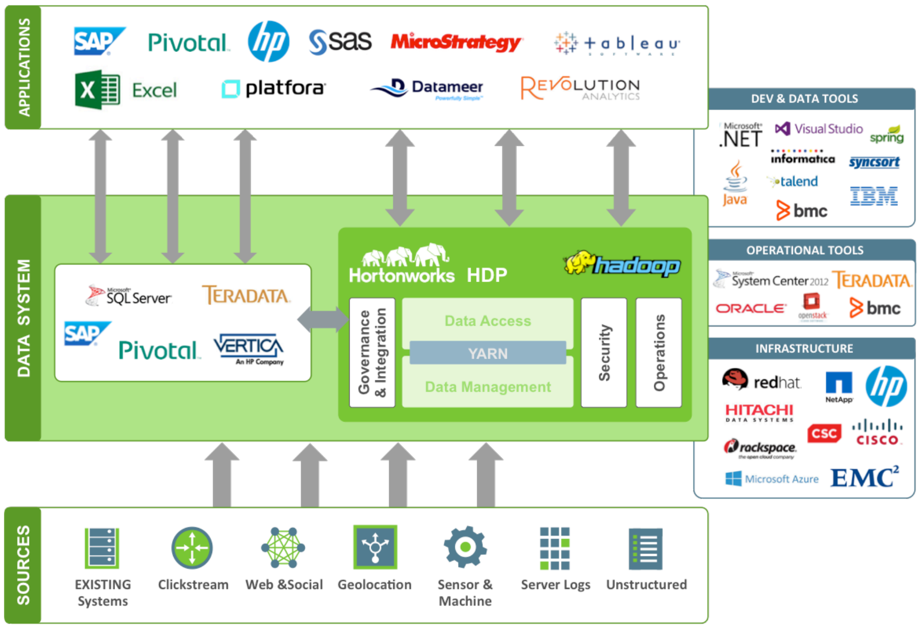
In a practical sense, a data lake is characterized by three key attributes:

- **Collect everything**
  A data lake contains all data, both raw sources over extended periods of time as well as any processed data.
- **Dive in anywhere**
  A data lake enables users across multiple business units to refine, explore and enrich data on their terms.
- **Flexible access**
  A data lake enables multiple data access patterns across a shared infrastructure: batch, interactive, online, search, in-memory and other processing engines.

The result is a repository that delivers maximum scale and insight with the lowest possible friction and cost.

As data continues to grow exponentially, then Enterprise Hadoop and EDW investments can provide a strategy for both efficiency in connected data platforms, and opportunity in an enterprise data lake.

Hadoop as a +1 Architecture



Hadoop as a +1 Architecture

Apache Hadoop is integrated with existing data systems.

Even if it could technologically displace all other data systems, the deep embedded nature built up over years would make it cost prohibitive to completely decommission most enterprises' utilization of existing data systems.

Summary

- Hadoop ecosystem frameworks fall into the following five categories:
  - Data Management
  - Data Access
  - Governance & Integration
  - Security
  - Operations
- Primary server stereotypes are:
  - Master nodes
  - Worker nodes
- Hadoop complements existing systems and is the foundation of Connected Data Platforms

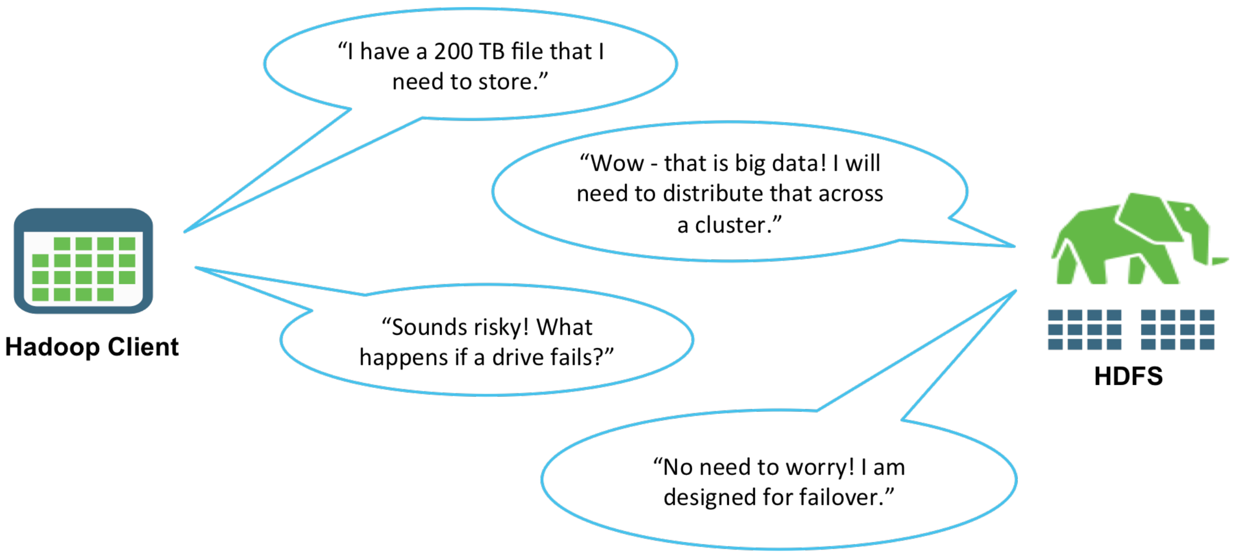
The HDFS Architecture

Lesson Objectives

After completing this lesson, students should be able to:

- ✓ Present an overview of the Hadoop Distributed File System (HDFS)
- ✓ Detail the major architectural components and their interactions
- ✓ Discuss important additional features and interactions

HDFS Overview

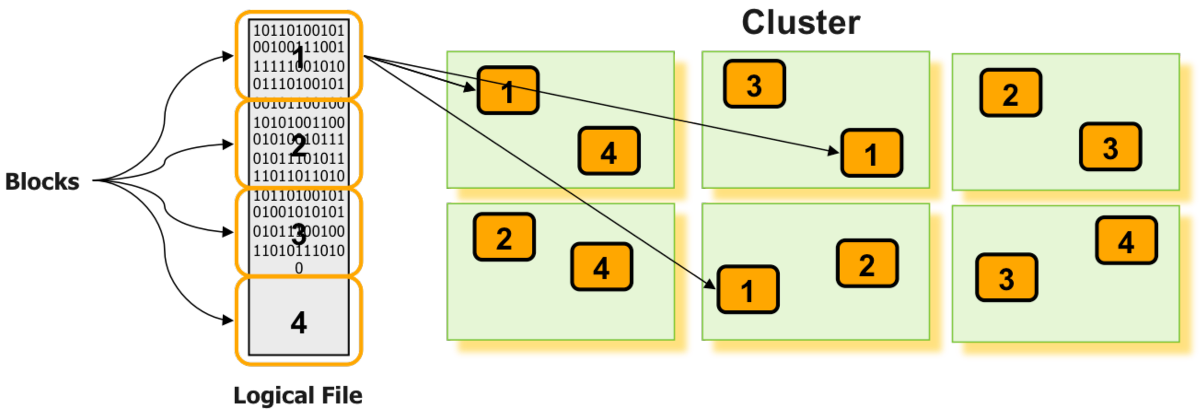

"I have a 200 TB file that I need to store."
"Wow - that is big data! I will need to distribute that across a cluster."
"Sounds risky! What happens if a drive fails?"
"No need to worry! I am designed for failover."
Hadoop Client
HDFS

HDFS

HDFS is a Java-based file system that provides scalable and reliable data storage, and it was designed to span large clusters of commodity servers. HDFS has demonstrated production scalability of up to 200 PB of storage and a single cluster of 4500 servers, supporting close to a billion files and blocks. When that quantity and quality of enterprise data is available in HDFS, and YARN enables multiple data access applications to process it, Hadoop users can confidently answer questions that eluded previous data platforms.

HDFS is a scalable, fault-tolerant, distributed storage system that works closely with a wide variety of concurrent data access applications, coordinated by YARN. HDFS will "just work" under a variety of physical and systemic circumstances. By distributing storage and computation across many servers, the combined storage resource can grow linearly with demand while remaining economical at every amount of storage.

Key HDFS Concepts

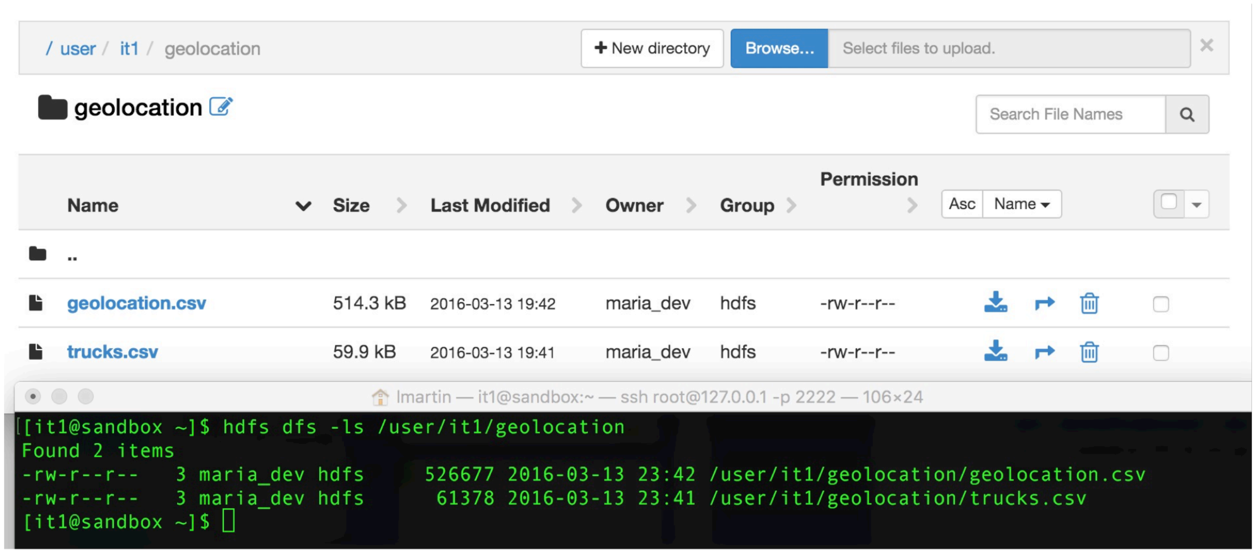


Key HDFS Concepts

Key concepts of HDFS include:

- Write Once, Read Many times (WORM)
- Divide files into big blocks and distribute across the cluster
- Store multiple replicas of each block for reliability
- Programs can ask "where do the pieces of my file live?" in order to find replicas for each block and try to gain data locality

HDFS Looks Like a File System



Name	Size	Last Modified	Owner	Group	Permission
..					
geolocation.csv	514.3 kB	2016-03-13 19:42	maria_dev	hdfs	-rw-r--r--
trucks.csv	59.9 kB	2016-03-13 19:41	maria_dev	hdfs	-rw-r--r--

```
lmartin — it1@sandbox:~ — ssh root@127.0.0.1 -p 2222 — 106×24
[it1@sandbox ~]$ hdfs dfs -ls /user/it1/geolocation
Found 2 items
-rw-r--r--   3 maria_dev hdfs     526677 2016-03-13 23:42 /user/it1/geolocation/geolocation.csv
-rw-r--r--   3 maria_dev hdfs      61378 2016-03-13 23:41 /user/it1/geolocation/trucks.csv
[it1@sandbox ~]$ 
```

HDFS as a File System

Common tools such as a web-based file system browser and basic CLI represent the basic expectations of a file system.

However, the differences between HDFS and other distributed file systems are significant:

- HDFS is highly fault-tolerant
- It is designed to be deployed on low-cost hardware
- It provides high throughput access to application data and is suitable for applications that have large data sets
- HDFS allows streaming access to file system data

HDFS Acts like a File System

```
hdfs dfs –command [args]
```

HDFS as a File System

Concerted effort has been made to move from “`hadoop fs`” to “`hdfs dfs`”.

TIP:

http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/FileSystemShell.html states that “`hdfs dfs`” is a synonym for “`hadoop fs`” when HDFS is being used (i.e. the “hadoop fs” command can interact with other file systems that Hadoop supports such as local FS, HFTP FS, S3 FS, and “others”) which makes sense with HDP.

HDFS Components and Interactions

The HDFS NameNode and DataNodes form the basis of the HDFS architecture and interact to disribute and replicate data across the cluster.

HDFS Components

NameNode

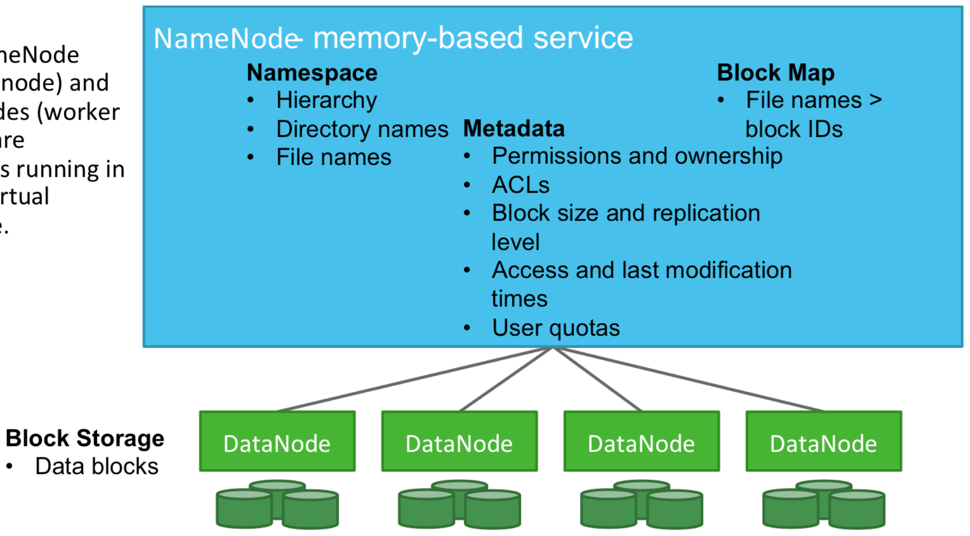
The NameNode is the master service of HDFS. It determines and maintains how the chunks of data are distributed across the DataNodes. A Namenode represents a single namespace. Data never reside on a NameNode.

DataNode

DataNodes store the chunks of data, and are responsible for replicating the chunks across other DataNodes. The Default block size in HDP 128MB. The default replication factor is 3.

HDFS Architecture

- The NameNode (master node) and DataNodes (worker nodes) are daemons running in a Java virtual machine.



HDFS Architecture

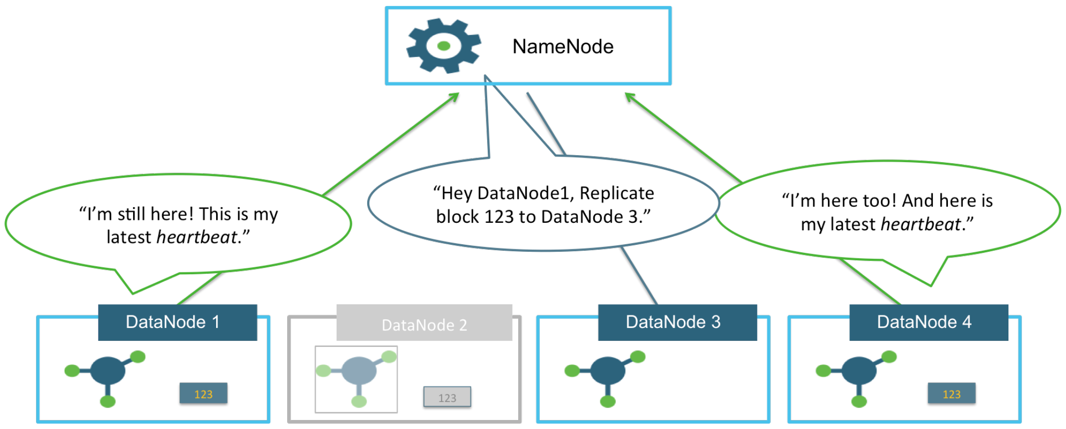
NameNode and DataNodes are components of the HDFS service. The NameNode is an HDFS **master** component while a DataNode is an HDFS **worker** component. The NameNode and DataNode are implemented as daemons running inside a Java virtual machine.

The NameNode maintains critical HDFS information. To enhance HDFS performance, it maintains and serves this information from memory. The memory-based information includes:

- Namespace information
- Metadata information
- Journaling information
- Block map information

Because the NameNode maintains all file system state information in memory, it is critical to ensure that the NameNode has sufficient memory.

Resolving Missing or Corrupted Blocks


NameNode
"I'm still here! This is my latest *heartbeat*."
"Hey DataNode1, Replicate block 123 to DataNode 3."
"I'm here too! And here is my latest *heartbeat*."
DataNode 1
123
DataNode 2
123
DataNode 3
DataNode 4
123

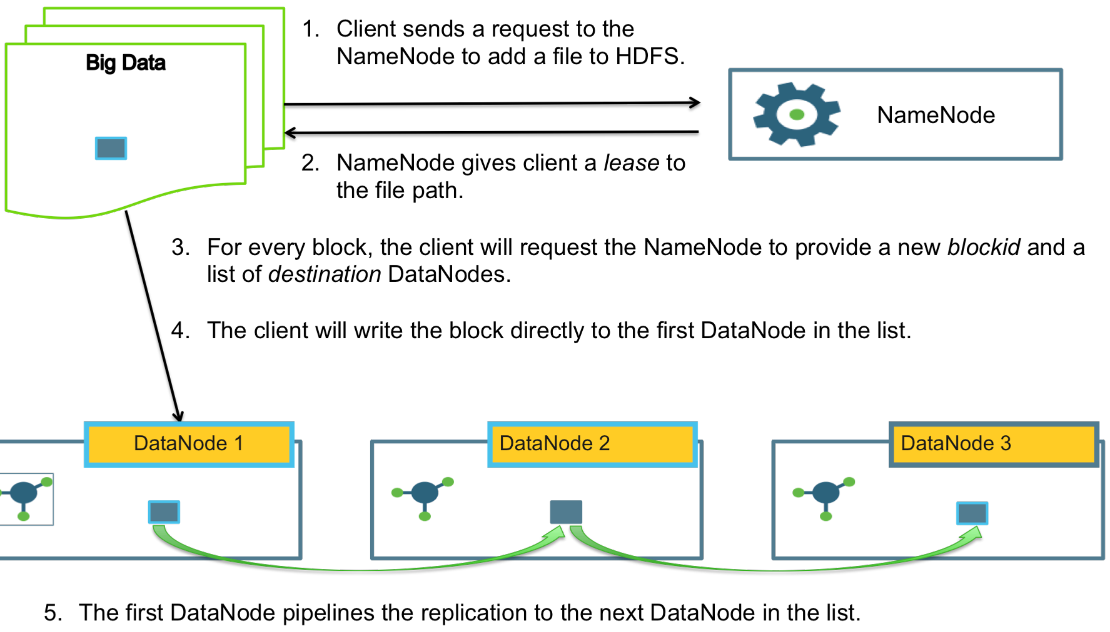
NameNode is Responsible for Resolving Missing or Corrupted Blocks

DataNodes periodically send a Heartbeat message to the NameNode. If a DataNode loses connectivity with the NameNOde, the NameNode detects this condition by the absence of the Heartbeat. The NameNode then marks silent DataNode as dead and will not forward any new IO requests to it. Data registered to a dead DataNode will no longer be available to HDFS.

It is the NameNode's job to track which blocks need to be replicated and initiate replication. A need for re-replication may occur for several reasons:

- A DataNode may become unavailable
- A replica could become corrupted
- A hard disk on a DataNode may fail
- The Replication factor of a file may be increased

NameNode and DataNodes Interaction


Big Data
1. Client sends a request to the NameNode to add a file to HDFS.
NameNode
2. NameNode gives client a *lease* to the file path.
3. For every block, the client will request the NameNode to provide a new *blockid* and a list of *destination* DataNodes.
4. The client will write the block directly to the first DataNode in the list.
DataNode 1
DataNode 2
DataNode 3
5. The first DataNode pipelines the replication to the next DataNode in the list.

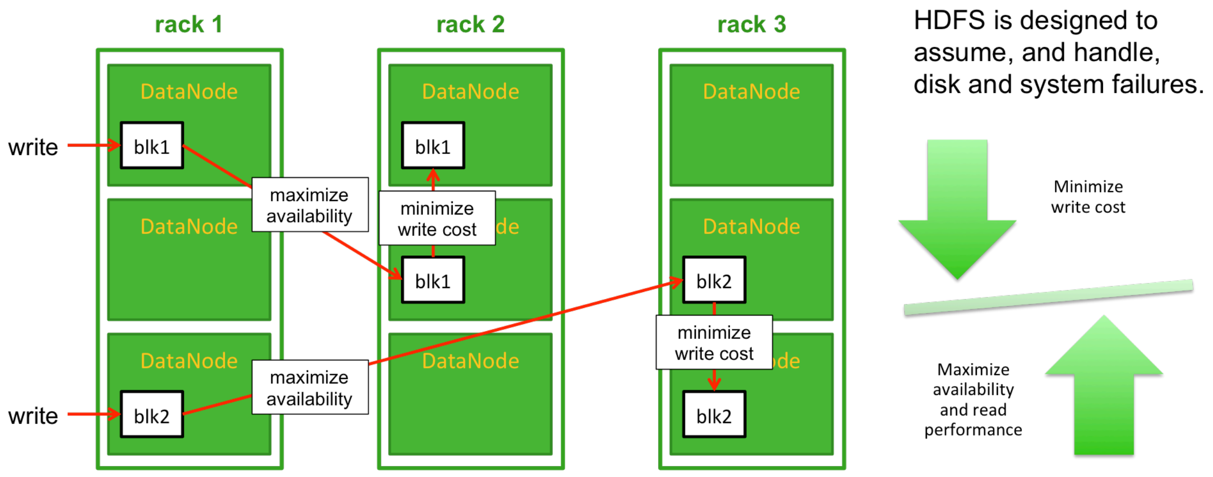
NameNode and DataNodes Interaction

HDFS is a master/slave architecture.

An HDFS cluster has a single NameNode that manages the namespace and regulates access to files by clients. There are a number of DataNodes, usually one per node in the cluster, which manage storage.

HDFS allows user data to be stored in files. DataNodes are responsible for serving read and write requests from the file system's clients. The DataNodes also perform block creation, deletion, and replication upon instruction from the NameNode.

Replication and Block Placement


rack 1
rack 2
rack 3
DataNode
write
blk1
blk2
maximize availability
minimize write cost
HDFS is designed to assume, and handle, disk and system failures.
Minimize write cost
Maximize availability and read performance

HDFS Replication and Block Placement

HDFS is designed to assume that disk, system, and network failures will occur. As a result, HDFS is also designed to automatically and transparently handle disk failures. It does this by automatically replicating data across different DataNodes.

HDFS stores a file as a sequence of blocks; all blocks in a file except the last block are the same size. Block size and replication factor are configurable per file and an application can specify the number of replicas of a file.

Additional Important HDFS Interactions

High availability and multi-tenancy are two additional, but important, features of HDFS,

NameNode High Availability

In Hadoop prior to version 2.0, the NameNode was a single point of failure. The entire cluster would become unavailable if the NameNode failed or became unreachable. Even maintenance events such as software or hardware upgrades on the NameNode machine would result in periods of cluster downtime.

The HDFS NameNode High Availability (HA) feature eliminates the NameNode as a single point of failure. It enables a cluster to run redundant NameNodes in an Active/Standby configuration.

NameNode HA enables fast failover to the Standby NameNode in response to a failure, or a graceful administrator-initiated failover for planned maintenance.

There are two ways of configuring NameNode HA. Using the Ambari Web UI is the easiest way. Manually editing the configuration files and starting or restarting the necessary daemons is also possible. However, manual configuration of NameNode HA is not compatible with Ambari administration. Any manual edits to the hdfs-site.xml file would be overwritten by information in the Ambari database when the HDFS service is restarted.

HDFS Multi-Tenant Controls

Security

- Classic POSIX permissioning (ex: -rwxr-xr--)
- Extended Access Control Lists (ACL) for richer scenarios
- Centralized authorization policies and audit available via Ranger plug-in

Quotas

- Easy to understand data size quotas
- Additional option for controlling the number of files

Summary

- HDFS breaks files into blocks and replicates them for reliability and processing data locality
- The primary components are the master NameNode service and the worker DataNode service
- The NameNode is a memory-based service
- The NameNode automatically takes care of recovery missing and corrupted blocks
- Clients interact with the NameNode to get a list, for each block, of DataNodes to write data to

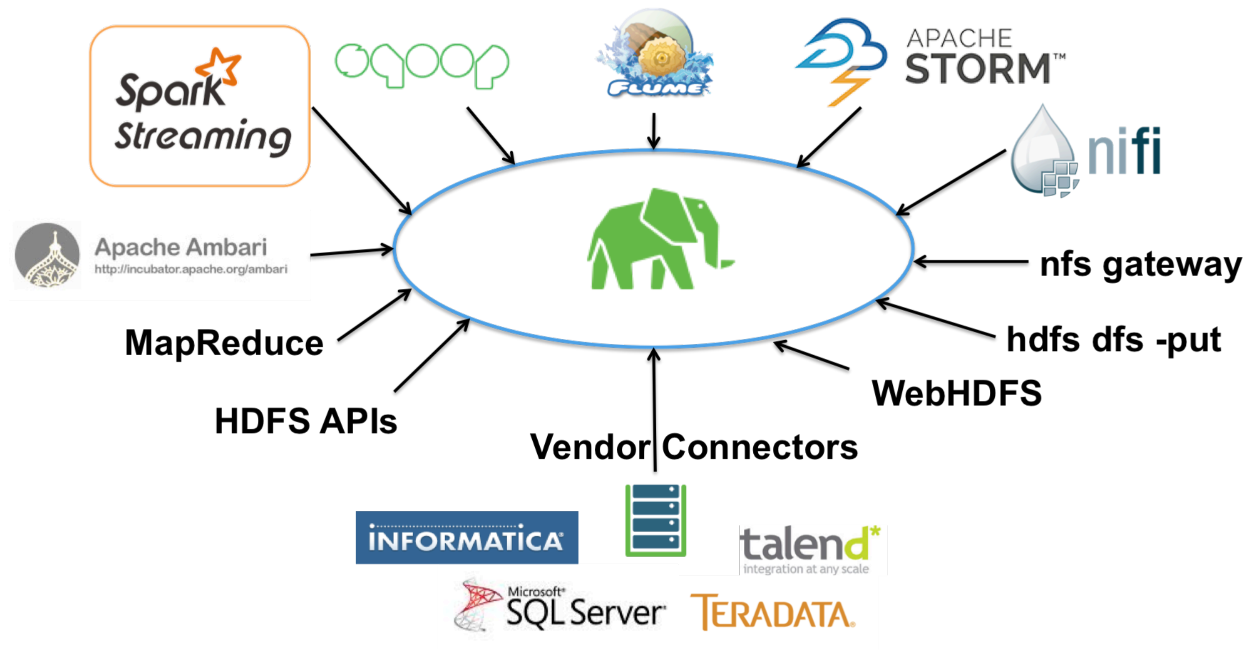
Ingesting Data

Lesson Objectives

After completing this lesson, students should be able to:

- ✓ Describe data ingestion
- ✓ Describe batch and bulk data ingestion options
- ✓ List streaming framework alternatives

Data Ingestion



Many Ways of Ingesting Data into HDFS

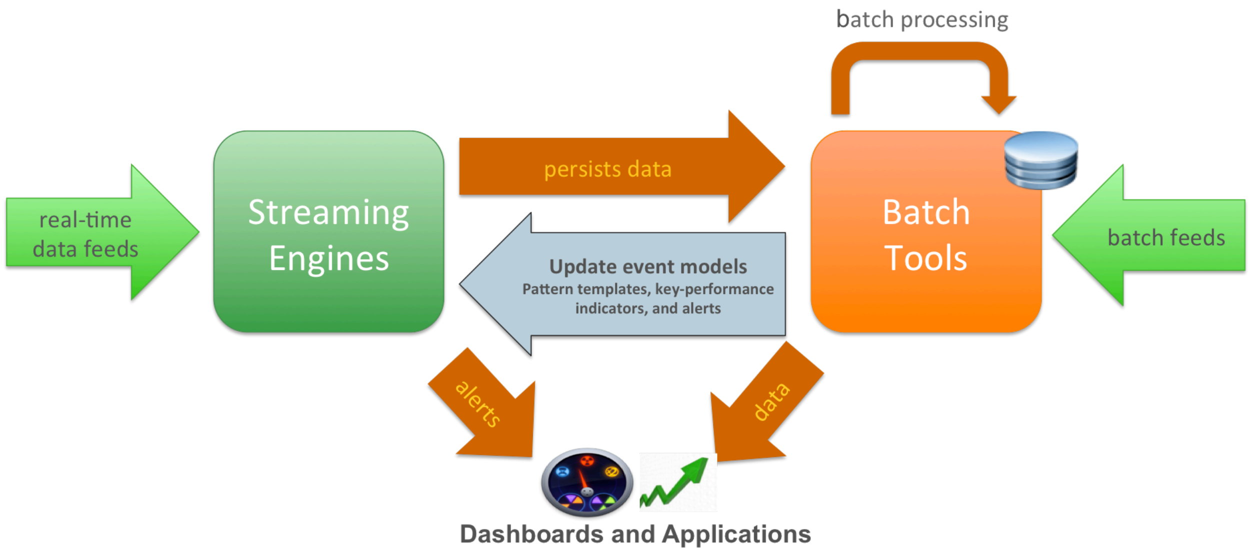
There are many different data sources with different formats that need to be input into HDFS. Just as there are many vendors that your organization may have, there are many different mechanisms to get your data loaded into HDFS. There are many tools for assisting in the ease of getting the data into the correct structure to suit your specific goals.

Call out Vendor Connectors and HDFS APIs which are available for Java and C. As you see in this slide the list of vendors is growing quickly just as corporate needs change.

Best Practice:

Keep your big data in its raw format and worry about applying structure and schema to it later when you transform and analyze the data.

Multiple Ingestion Workflows



Lambda Arhictecture

LAMBDA Architecture is a data-processing architecture designed to handle massive quantities of data by taking advantge of both batch and stream-based architecture methods. The approach balances uses batch processing to provide comprehensive and accurate views of batch data while real-time stream processing to provide views of online data. The inception of the lambda architecture has paralleled the growth of big data, real-time analytics, and the drive to mitigate the latencies of map-reduce. (Schuster, Werner. "Nathan Marz on Storm, Immutability in the Lambda Architecture, Clojure". www.infoq.com. Interview with Nathan Marz, 6 April 2014)

Streaming in Hadoop helps capture new business opportunities with low-latency dashboards, security alerts, and operational enhancements integrated with other applications running in a Hadoop cluster.

To realize these benefits, an enterprise should integrate real-time processing with normal Hadoop batch processing. Data derived from batch processing can be used to inform real-time processing dashboards and applications.

For example, data derived from batch processing is commonly used to create the event models used by the real-time system. These event models define the schemas of incoming event data, such as records of calls into the customer contact center, copies of customer order transactions, or external market data that might affect any action taken.

Real-Time vs. Batch Ingestion

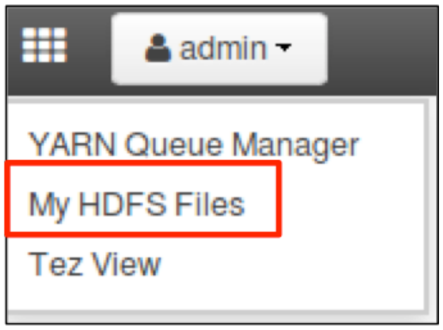
Factors		Real-Time	Batch
Data	**Age**	Real-time – usually less than 15 minutes old	Historical – usually more than 15 minutes old
	Location	Primarily in memory – moved to disk after processing	Primarily on disk – moved to memory for processing
Processing	**Speed**	Sub-second to few seconds	Few seconds to hours
	Frequency	Always running	Sporadic to periodic
Clients	**Who**	Automated systems only	Human & automated systems
	Type	Primarily operational applications	Primarily analytical applications

Differences Between Real-Time and Batch Ingestion

Batch and real-time data processing are very different. The differences include the characteristics of the data, the requirements for processing the data, and the clients who use or consume the data.

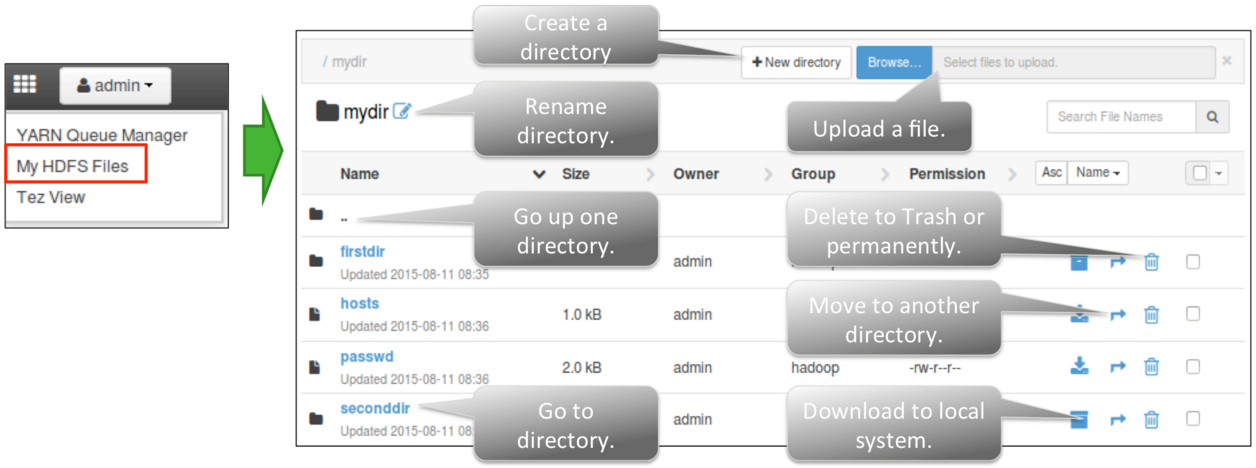
One of the primary differences between batch and real-time processing is that real-time systems are always running and therefore typically require automated applications or dashboards to consume the data. These applications or dashboards are used to effect current operations while batch processing is commonly used for historical data analysis.

Batch and Bulk Ingestion Options



Ambari Files View Menu Selection

The Ambari Files View is an Ambari Web UI plug-in providing a graphical user interface to HDFS files and directories.



Ambari Files View Functions

The Files View can create a directory, rename a directory, browse to and list a directory, download a zipped directory, and delete a directory. It can also upload a file, list files, open files, download files, and delete a file.

The Hadoop Client

Hadoop Client machines are neither Master nor Slave. Their role is to load data into the cluster, submit jobs describing how that data should be processed, and then retrieve or view the results of the job when its finished.

They:

- Use the `put` command to upload data to HDFS
- Are perfect for inputting local files into HDFS
- Are useful in batch scripts

Usage:

```
hdfs dfs -put mylocalfile /some/hdfs/path
```

Typically, local files are loaded into the HDFS environment by writing or creating scripts In the language of choice.

In the above usage example:

- The file system is **hdfs**
- The type of file system is **dfs** (distributed file system)
- The command being used to load data is `-put` which loads data
- The name of the file is `mylocalfile`
- An absolute file path name is then given
  - The the leading "/" denotes an absolute versus current directory where file can be found `/some hdfs/path`

WebHDFS

WebHDFS provides an external client that does not necessarily run on the Hadoop cluster itself a standard method of executing Hadoop filesystem operations.

Using WebHDFS provides a method to identify the host that must be connected to in the cluster. It is based on based on HTTP operations such as:

- `GET`
- `PUT`
- `POST`
- `DELETE`

Operations such as `OPEN, GETFILESTATUS, LISTSTATUS` use `HTTP GET`; other operations such as `CREATE, MKDIRS, RENAME, SETPERMISSIONS` use `HTTP PUT`. `APPEND` operations are based on `HTTP POST`.

Sample Commands

```
http://host:port/webhdfs/v1/test/mydata.txt?op=OPEN
http://host:port/webhdfs/v1/user/train/data?op=MKDIRS
http://host:port/webhdfs/v1/test/mydata.txt?op=APPEND
```

The above example shows **http://host:port** and identifies the server that contains the information for each one of the files.

- `OPEN` – request is redirected to a datanode where the file can be read
- `MKDIRS` – request is redirected to a datanode where the directory can be found
- `APPEND` – request is redirected to a datanode where the file mydata.txt can is to be be appended.

NFS Gateway

The NFS Gateway for HDFS allows clients to mount HDFS and interact with it through NFS, as if it were part of their local file system. The gateway supports NFSv3.

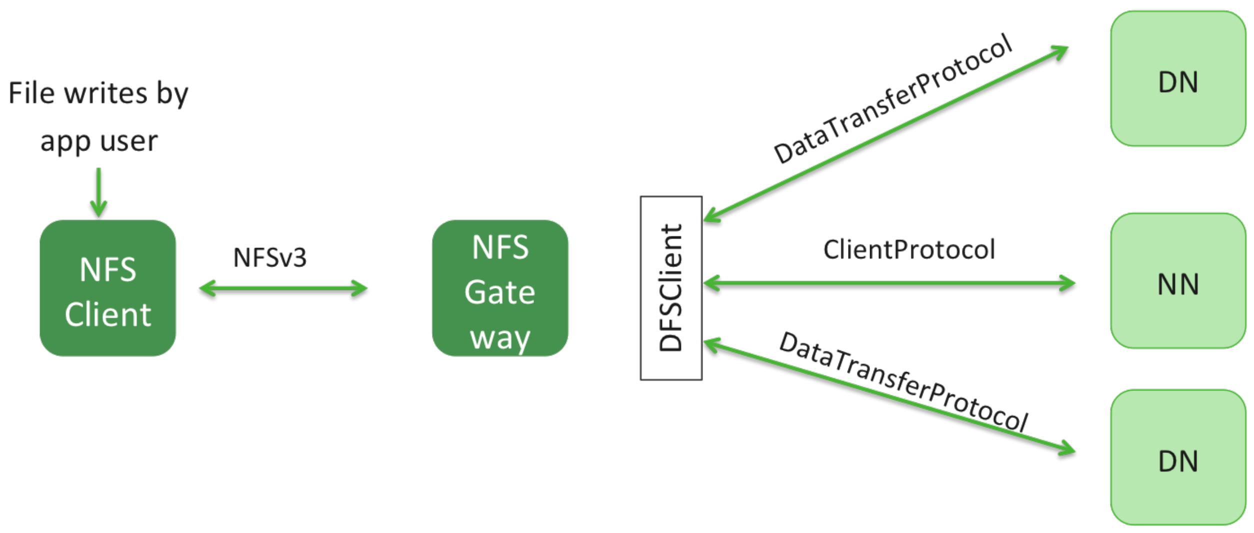
After mounting HDFS, a user can:

- Browse the HDFS file system through their local file system on NFSv3 client-compatible operating systems.
- Upload and download files between the HDFS file system and their local file system.
- Stream data directly to HDFS through the mount point. (File append is supported, but random write is not supported.)

Prerequisites

The NFS Gateway machine must be running all components that are necessary for running an HDFS client, such as a Hadoop core JAR file and a HADOOP_CONF directory.

The NFS Gateway can be installed on any DataNode, NameNode, or HDP client machine. Start the NFS server on that machine.

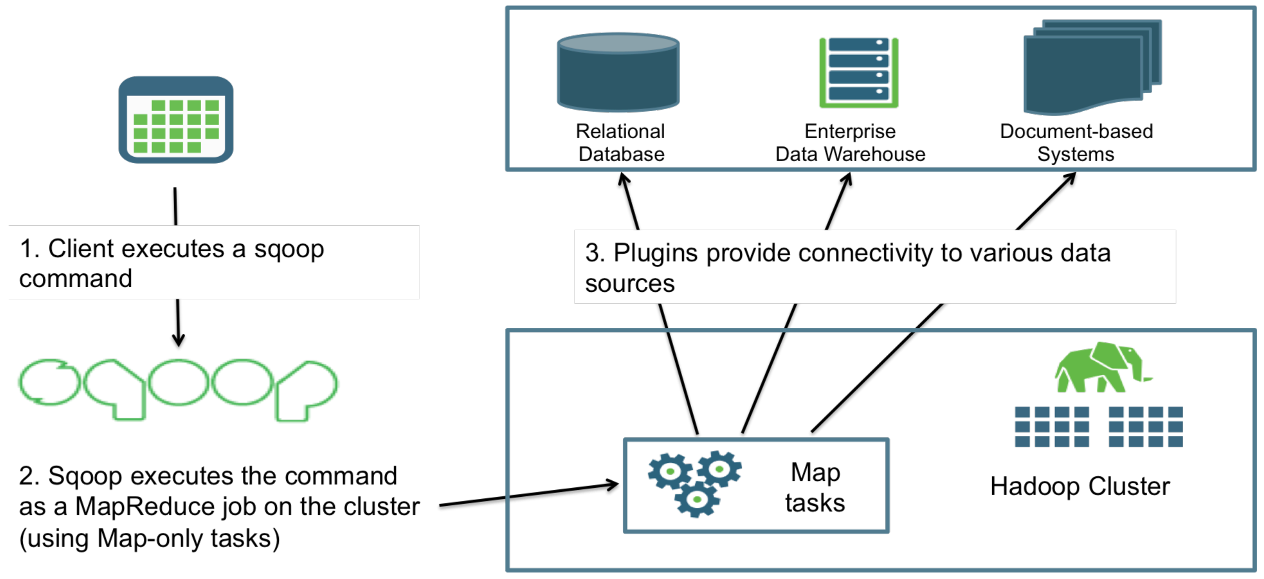


NFS Gateway Supports all HDFS Commands

As the diagram above shows files are written by an application user to the NFS Client. The NFS v3 protocol is used as the mechanism to contact the NFS gateway. On the the right side of the diagram the DFS (Distributive File System) Client is used as traffic cop to send/receive data to the:

- Domain name via the data transfer protocol
- Node Name via the Client protocol

Sqoop


Relational Database
Enterprise Data Warehouse
Document-based Systems
1. Client executes a sqoop command
3. Plugins provide connectivity to various data sources
sqoop
2. Sqoop executes the command as a MapReduce job on the cluster (using Map-only tasks)
Map tasks
Hadoop Cluster

Database Import and Export using Sqoop

Apache Sqoop efficiently transfers bulk data between Apache Hadoop and structured datastores such as relational databases. Sqoop helps offload certain tasks (such as ETL processing) from the EDW to Hadoop for efficient execution at a much lower cost. Sqoop can also be used to extract data from Hadoop and export it into external structured datastores. Sqoop works with relational databases such as Teradata, Netezza, Oracle, MySQL, Postgres, and HSQLDB

Apache Sqoop does the following to integrate bulk data movement between Hadoop and structured datastores:

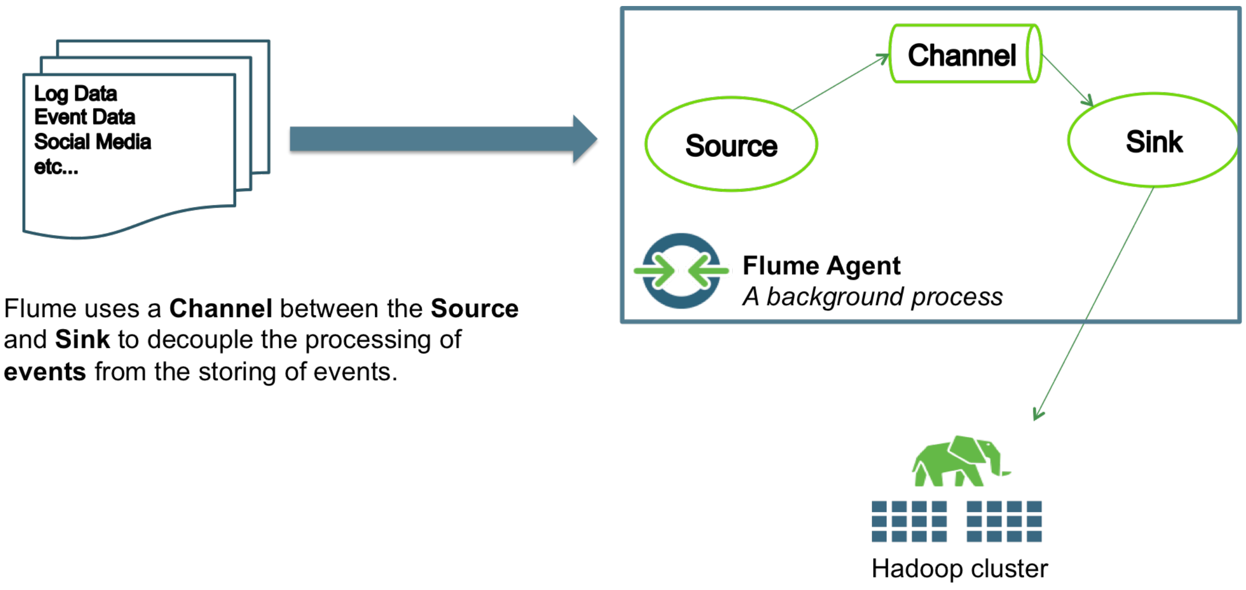
- **Import sequential datasets from mainframe** Satisfies the growing need to move data from mainframe to HDFS
- **Import direct to ORCFiles** Improved compression and light-weight indexing for improved query performance
- **Data imports** Moves certain data from external stores and EDWs into Hadoop to optimize cost-effectiveness of combined data storage and processing
- **Parallel data transfer** For faster performance and optimal system utilization
- **Fast data copies** From external systems into Hadoop
- **Efficient data analysis** Improves efficiency of data analysis by combining structured data with unstructured data in a schema-on-read data lake
- **Load balancing** Mitigates excessive storage and processing loads to other systems YARN coordinates data ingest from Apache Sqoop and other services that deliver data into the Enterprise Hadoop cluster.

Streaming Framework Frameworks

Streaming alternatives for ingestion of data include:

- Flume
- Storm
- Spark Streaming
- Hortonworks Data Flow (HDF)

Flume



Flume uses a **Channel** between the **Source** and **Sink** to decouple the processing of **events** from the storing of events.

Data Streaming with Flume

Apache Flume is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of streaming data into the Hadoop Distributed File System (HDFS). It has a simple and flexible architecture based on streaming data flows; and is robust and fault tolerant with tunable reliability mechanisms for failover and recovery.YARN coordinates data ingest from Apache Flume and other services that deliver raw data into an Enterprise Hadoop cluster.

Flume lets Hadoop users ingest high-volume streaming data into HDFS for storage. Specifically, Flume allows users to:

- **Stream data** Ingest streaming data from multiple sources into Hadoop for storage and analysis
- **Insulate systems** Buffer storage platform from transient spikes, when the rate of incoming data exceeds the rate at which data can be written to the destination
- **Guarantee data delivery** Flume NG uses channel-based transactions to guarantee reliable message delivery. When a message moves from one agent to another, two transactions are started, one on the agent that delivers the event and the other on the agent that receives the event. This ensures guaranteed delivery semantics
- **Scale horizontally** To ingest new data streams and additional volume as needed Enterprises use Flume's powerful streaming capabilities to land data from high-throughput streams in the Hadoop Distributed File System (HDFS).

Typical sources of these streams are application logs, sensor and machine data, geo-location data and social media. These different types of data can be landed in Hadoop for future analysis using interactive queries in Apache Hive. Or they can feed business dashboards served ongoing data by Apache HBase.

Storm

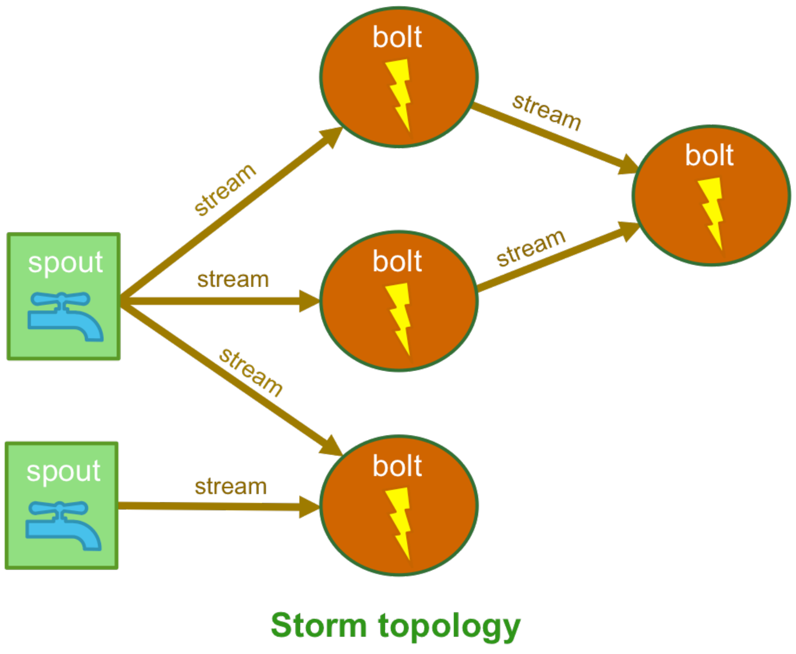
Apache™ Storm adds reliable real-time data processing capabilities to Enterprise Hadoop. Storm on YARN is powerful for scenarios requiring real-time analytics, machine learning and continuous monitoring of operations.

Storm integrates with YARN via Apache Slider, YARN manages Storm while also considering cluster resources for data governance, security and operations components of a modern data architecture.

Storm is a distributed real-time computation system for processing large volumes of high-velocity data. Storm is extremely fast, with the ability to process over a million records per second per node on a cluster of modest size. Enterprises harness this speed and combine it with other data access applications in Hadoop to prevent undesirable events or to optimize positive outcomes.

Some of specific new business opportunities include: real-time customer service management, data monetization, operational dashboards, or cyber security analytics and threat detection.

Storm Abstractions



Storm Data Processing

There are four abstractions in Storm: spouts, bolts, streams, and topologies. Storm data processing occurs in a topology. A topology consists of spout and bolt components. Spouts and bolts run on the systems in a Storm cluster. Multiple topologies can co-exist to process different data sets in different ways.

Streams

The core abstraction in Storm is the "stream". A stream is an unbounded sequence of tuples. Storm provides the primitives for transforming a stream into a new stream in a distributed and reliable way. For example, you may transform a stream of tweets into a stream of trending topics.

The basic primitives Storm provides for doing stream transformations are "spouts" and "bolts". Spouts and bolts have interfaces that you implement to run your application-specific logic.

Spouts

A spout is a source of streams. For example, a spout may read tuples off of a Kestrel queue and emit them as a stream. Or a spout may connect to the Twitter API and emit a stream of tweets.

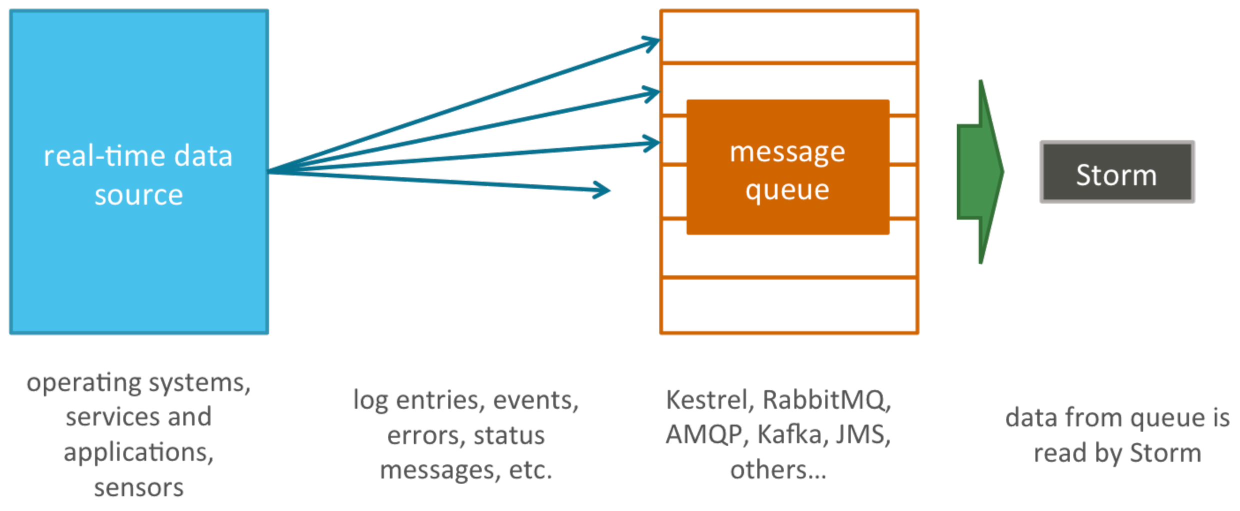
Bolts

A bolt consumes any number of input streams, does some processing, and possibly emits new streams. Complex stream transformations, like computing a stream of trending topics from a stream of tweets, require multiple steps and thus multiple bolts. Bolts can do anything from run functions, filter tuples, do streaming aggregations, do streaming joins, talk to databases, and more.

Topologies

Networks of spouts and bolts are packaged into a "topology" which is the top-level abstraction that you submit to Storm clusters for execution. A topology is a graph of stream transformations where each node is a spout or bolt. Edges in the graph indicate which bolts are subscribing to which streams. When a spout or bolt emits a tuple to a stream, it sends the tuple to every bolt that subscribed to that stream.

Message Queues

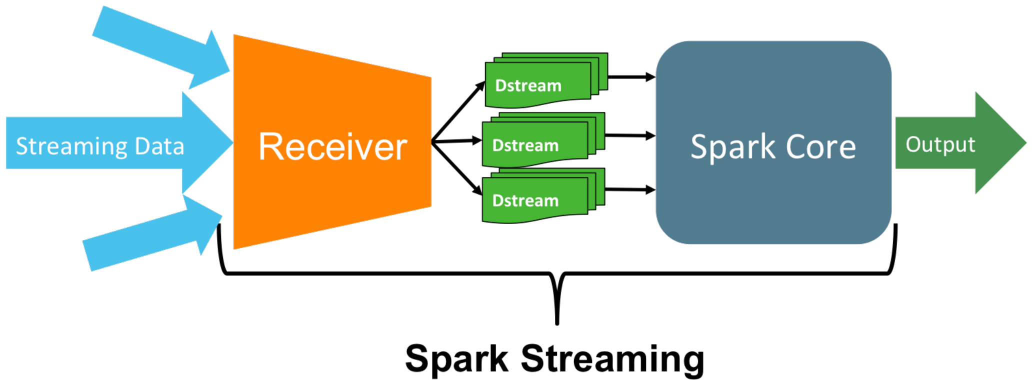

real-time data source
message queue
Storm
operating systems, services and applications, sensors
log entries, events, errors, status messages, etc.
Kestrel, RabbitMQ, AMQP, Kafka, JMS, others...
data from queue is read by Storm

Message Queues are Often a Source of Real-Time Data Processing

The source of Storm data is often a message queue. For example, an operating system, service, or application will send log entries, event information, or other messages to a queue. The queue is read by Storm.

Storm integrates with many queuing systems. Example queue integrations include Kestrel, RabbitMQ, Advanced Message Queuing Protocol (AMQP), Kafka, Java Message Service (JMS), and Amazon Kinesis. Storm's abstractions make it easy to integrate with a new queuing system.

Spark Streaming



Spark Streaming Application

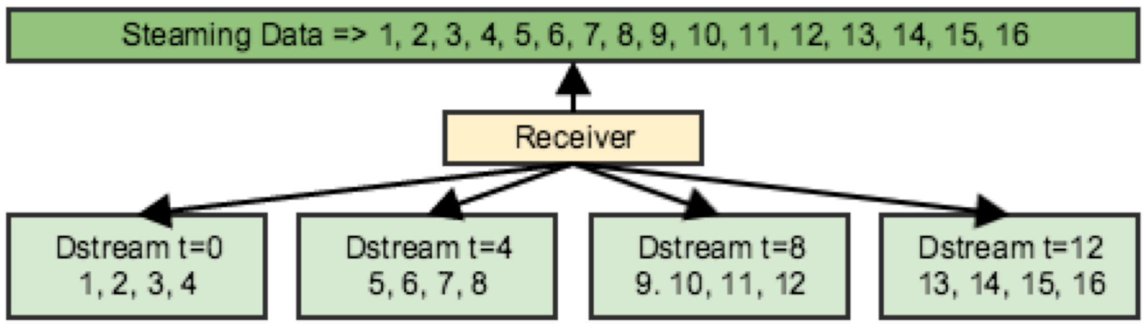
As a general rule, new frameworks don't introduce very many new concepts. Spark Streaming is an exception, as the new concepts of a **receiver** and **Dstream** are introduced. A streaming application is composed of a **receiver**, **Core Spark**, and a **Dstream.**

In Spark Streaming:

- Receivers listen to data streams, and create batches of data called Dstreams
- Dstreams are then processed by the Spark Core engine

Once data is ingested, you can use all the Spark frameworks; you are not limited to using only Spark streaming functionalities.

Micro-Batches


Steaming Data => 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16
Receiver
Dstream t=0 1, 2, 3, 4
Dstream t=4 5, 6, 7, 8
Dstream t=8 9. 10, 11, 12
Dstream t=12 13, 14, 15, 16

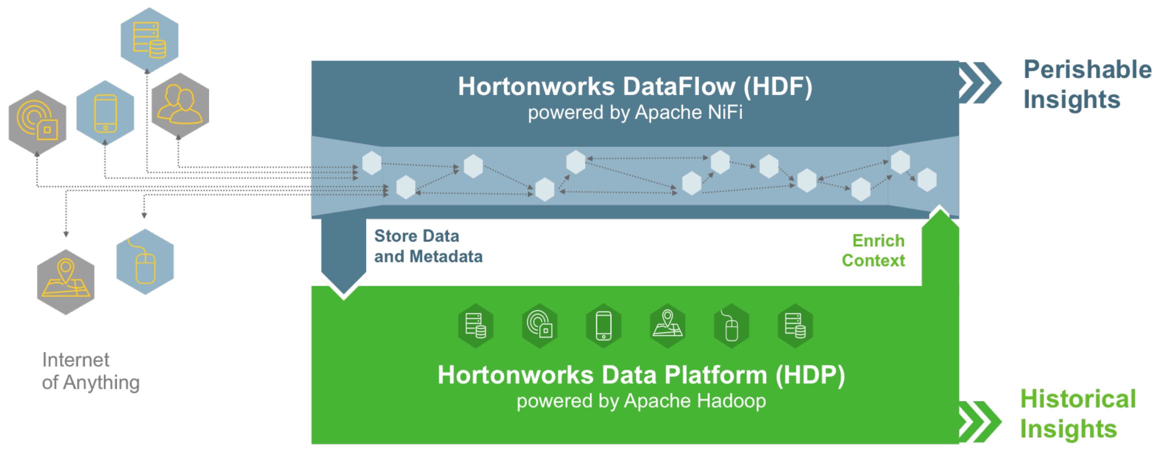
Spark Streaming Micro-Batches

Dstreams are created as batches of data from a streaming source by the receiver at regular time intervals.

1) When a streaming source begins communicating with the Spark streaming application, the receiver begins filling up a bucket.

2) At a predetermined time interval, the bucket is shipped off to be processed.
Each of these buckets is a single Dstream. Once the Dstream is created it is conceptually very similar to an RDD.

HDF



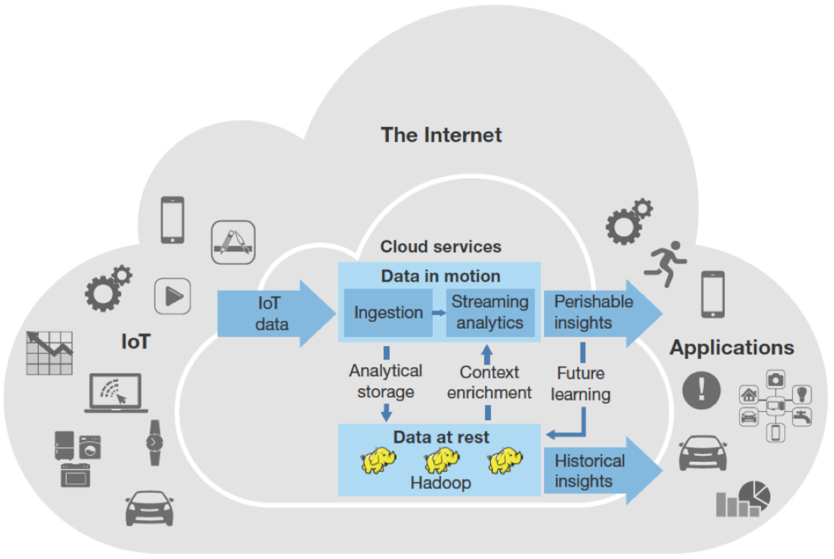
Hortonworks Data Flow – A Complete Big Data Solution

Hortonworks DataFlow and Hortonworks Data Platform both work independently, but when they work together they deliver the industry's most complete Big Data solution.

HDF allows you to identify and then quickly act on perishable insights. For example: It is more helpful to know about disease outbreaks, trading windows, fraud or equipment failures when they occur – not days after.

Then HDF can pass the data and metadata from those perishable insights into HDP for historical storage and analysis. As a result, the deeper analysis in HDP can be pushed back to HDF to improve data flows by continuously tuning the data coming in and the decisions being made in the perimeter.

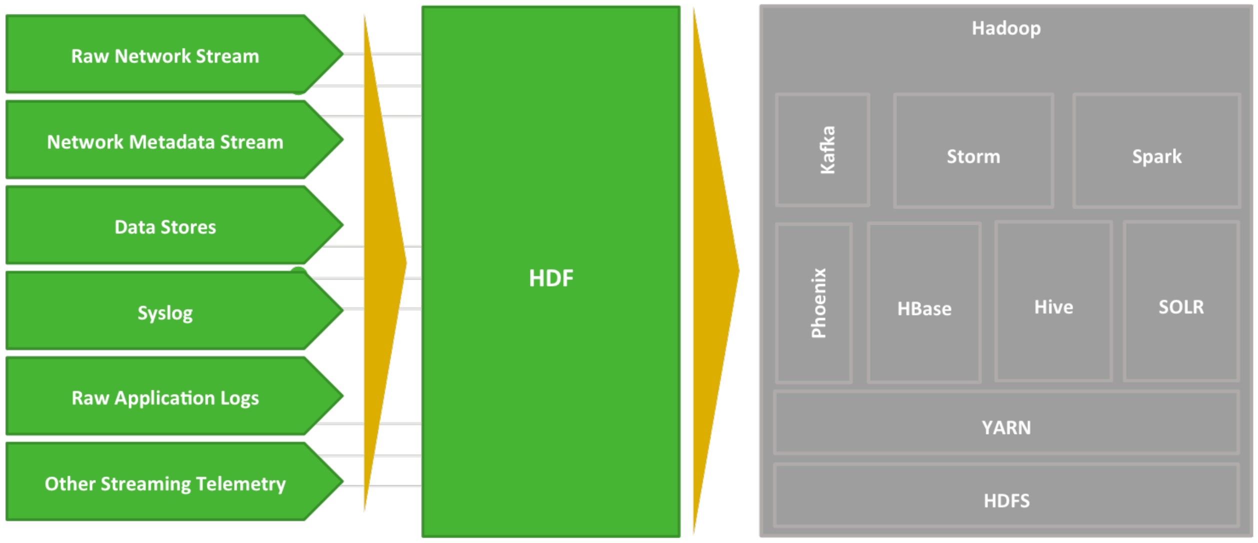
Figure 1 IoT App Design Must Simultaneously Support Analytics In Motion And Analytics At Rest



Source: Forrester - Internet Of Things Applications Hunger For Hadoop And Real-Time Analytics In The Cloud, March 2015

For example: A NiFi instance can do its own analytics to generate perishable data, but for collective Information and to persist it you may need to feed it to a central application such as Spark Streaming or Storm – and store in HDFS.

Coupling HDF with Storm/Spark



HDF Workflows and Storm/Spark Can be Coupled

HDF is used for perishable insights that can be gained from data-in-motion; Storm and Spark streaming are used to gather historical insights from data-at-rest. Together they securely and easily collect, conduct, and curate dynamic Internet of Anything data into actionable insights.

Summary

- There are many different ways to ingest data including customer solutions written via HDFS APIs as well as vendor connectors
- Streaming and batch workflows can work together in a holistic system to analyze perishable and historic data
- The NFS Gateway allows clients to mount HDFS and interact with it through NFS as if it were part of the local file system
- Sqoop transfers bulk data between Hadoop and structured datastores such as relational databases
- The following are streaming frameworks:
    - Flume
    - Storm
    - Spark Streaming
    - Hortonworks Data Flow (HDF)

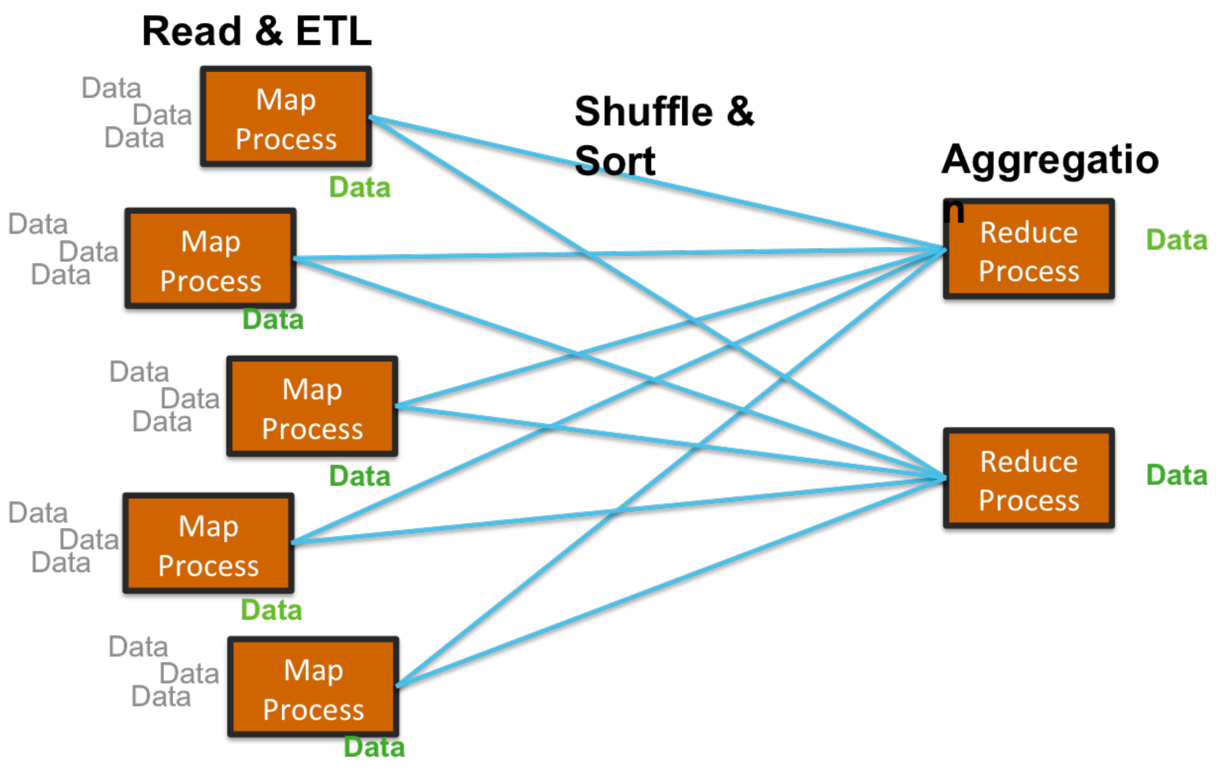
Parallel Processing with MapReduce

Lesson Objectives

After completing this lesson, students should be able to:

- ✓ Describe how the MapReduce Framework works

MapReduce Framework



MapReduce Implements Two Processes: Map and Reduce

MapReduce enables programmers to process huge amounts of data in parallel across distributed processors. It handles details such as parallelization, fault tolerance, data distribution and load balancing so programmers do not have to worry about them.

MapReduce provides a clear abstraction for programmers. They only have to use two functions, map and reduce:

1) Data are fed into the map function as key value pairs to produce intermediate key/value pairs

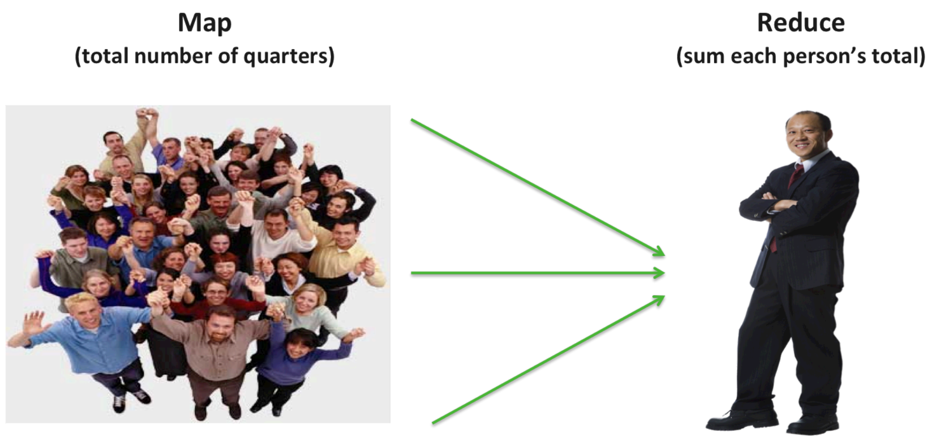
2) Once the mapping is done, intermediate results from various nodes are reduced to create the final output

The number of mappers is aligned with the number of blocks that the input data takes up. The number of reducers is set by the application developer.

Simple MapReduce Algorithm Example

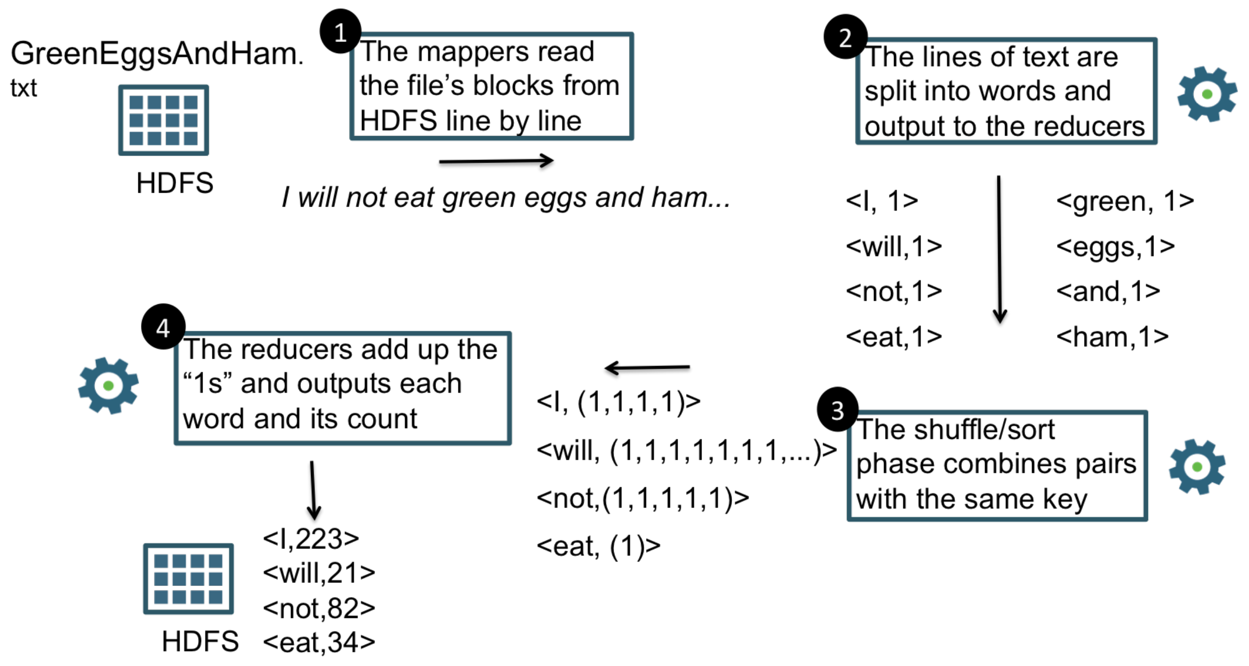
In our example, we want to review a stack of quarters and count only the quarters that were minted in even years. We could easily just count out this stack by hand.

But what happens when we have to do the same task at scale?



We might want to call on our friends to help process this huge pile. So in this scenario, we give each of our friends part of the pile and have them count them – then we will reduce those subtotals and combine them to find the final tally.

Word Count Process


GreenEggsAndHam.
txt
HDFS
1
The mappers read the file's blocks from HDFS line by line
I will not eat green eggs and ham...
2
The lines of text are split into words and output to the reducers
<I, 1>
<will,1>
<not,1>
<eat,1>
<green, 1>
<eggs,1>
<and,1>
<ham,1>
3
The shuffle/sort phase combines pairs with the same key
<I, (1,1,1,1)>
<will, (1,1,1,1,1,1,1,...)>
<not,(1,1,1,1,1)>
<eat, (1)>
4
The reducers add up the "1s" and outputs each word and its count
<I,223>
<will,21>
<not,82>
<eat,34>
HDFS

The Mapper

The Mapper reads data in the form of key/value pairs (KVPs). It outputs zero or more KVPs. The Mapper may use or completely ignore the input key. For example, a standard pattern is to read a line of a file at a time:

- The key is the byte offset into the file at which the line starts
- The value is the contents of the line itself
- Typically the key is considered irrelevant with this pattern

If the Mapper writes anything out, it must in the form of KVPs. This "intermediate data" is NOT stored in HDFS (local storage only without replication).

The Reducer

After the Map phase is over, all intermediate values for a given intermediate key are combined together into a list. This list is given to a single or multiple Reducers.

- All values associated with a particular intermediate key are guaranteed to go to the same Reducer
- The intermediate keys, and their value lists, are passed in sorted order

The Reducer outputs zero or more KVPs which are written to HDFS. In practice, the Reducer often emits a single KVP for each input key.

MapReduce Word Count Example

Goal:

To count the number of occurrences of each word in a large amount of data.

```
map(String input_key, String input_value)
        foreach word in input_value:
                emit(word,1)
```

```
reduce(String output_key, Iter<int> intermediate_vals)
        set count = 0
        foreach val in intermediate_vals:
                count += val
        emit(output_key, count)
```

The Map Phase:

Input to Mapper

```
(8675, 'I will not eat green
eggs and ham')

(8709, 'I will not eat them Sam
I am')
```

Output from Mapper

```
('I', 1), ('will', 1), ('not', 1),
('eat', 1), ('green', 1),
('eggs', 1), ('and', 1),
('ham', 1), ('I', 1), ('will', 1),
('not', 1), ('eat', 1),
('them', 1), ('Sam', 1),
('I', 1), ('am', 1)
```

Each mapper takes a line as input and breaks it into words. It then outputs a key/value pair of the word and 1.

The Reduce Phase:

Input to Reducer

```
('I', [1, 1, 1])
('Sam', [1])
('am', [1])
('and', [1])
('eat', [1, 1])
('eggs', [1])
('green', [1])
('ham', [1])
('not', [1, 1])
('them', [1])
('will', [1, 1])
```

Output from Reducer

```
('I', 3)
('Sam', 1)
('am', 1)
('and', 1)
('eat', 2)
('eggs', 1)
('green', 1)
('ham', 1)
('not', 2)
('them', 1)
('will', 2)
```

Each reducer sums the counts for each word and outputs a single key/value with the word and sum.

Summary

- MapReduce is the foundational framework for processing data at scale because of its ability to break a large problem into any smaller ones
- Mappers read data in the form of key/value pairs (KVPs) and each call to a Mapper is for a single KVP; it can return 0..m KVPs
- The framework shuffles and sorts the Mappers' outputted KVPs with the guarantee that only one Reducer will be asked to process a given Key's data
- Reducers are given a list of Values for a specific Key; they can return 0..m KVPs

Apache Hive Overview

Lesson Objectives

After completing this lesson, students should be able to:

- ✓ Describe Hive features and functions
- ✓ Explain how to perform classic query operations in Hive
- ✓ List performance improvements using Stinger.next and Tez

Hive Overview

Apache Hive is a data warehouse infrastructure built on top of Apache Hadoop that provides data summarization, ad-hoc query, and analysis of large datasets. It projects structure onto the data in Hadoop and queries that data using an SQL-like language called HiveQL (HQL).

Tables in Hive are similar to tables in a relational database, and data units are organized in a taxonomy from larger to more granular units. Databases are comprised of tables, which are made up of partitions. Data can be accessed via a simple query language and Hive supports overwriting or appending data.

Within any particular database, data in the tables is serialized and each table has a corresponding Hadoop Distributed File System (HDFS) directory. Each table can be sub-divided into partitions that determine how data is distributed within sub-directories of the table directory. Data within partitions can be further broken down into buckets.

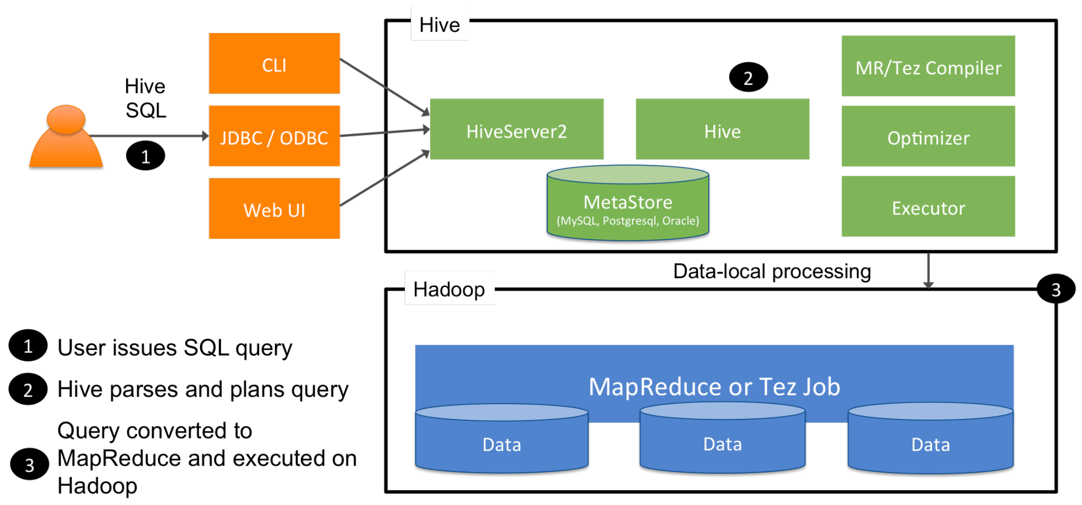
Hive supports all the common primitive data formats such as BIGINT, BINARY, BOOLEAN, CHAR, DECIMAL, DOUBLE, FLOAT, INT, SMALLINT, STRING, TIMESTAMP, and TINYINT. In addition, analysts can combine primitive data types to form complex data types, such as structs, maps and arrays.

Hive Alignment with SQL

SQL Datatypes	SQL Semantics
INT	SELECT, LOAD, INSERT from query
TINYINT/SMALLINT/BIGINT	Expressions in WHERE and HAVING
BOOLEAN	GROUP BY, ORDER BY, SORT BY
FLOAT	CLUSTER BY, DISTRIBUTE BY
DOUBLE	Sub-queries in FROM clause
STRING	GROUP BY, ORDER BY
BINARY	ROLLUP and CUBE
TIMESTAMP	UNION
ARRAY, MAP, STRUCT, UNION	LEFT, RIGHT and FULL INNER/OUTER JOIN
DECIMAL	CROSS JOIN, LEFT SEMI JOIN
CHAR	Windowing functions (OVER, RANK, etc.)
VARCHAR	Sub-queries for IN/NOT IN, HAVING
DATE	EXISTS / NOT EXISTS
	INTERSECT, EXCEPT

Hive offers semantics similar to RDBMS tools.

Hive Query Process

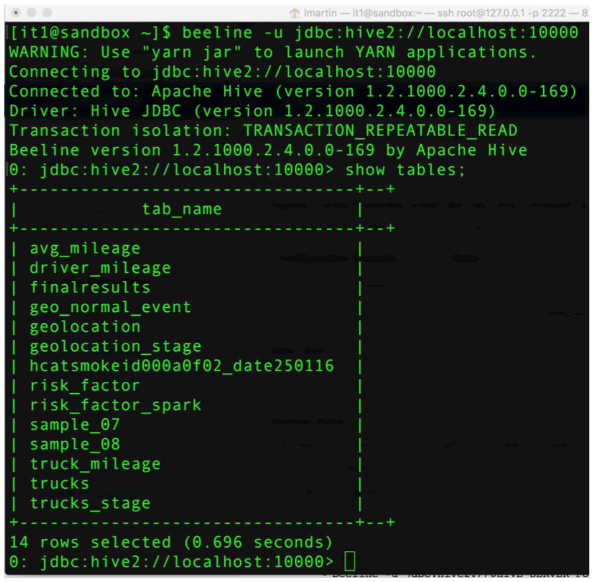


The Hive Query Process

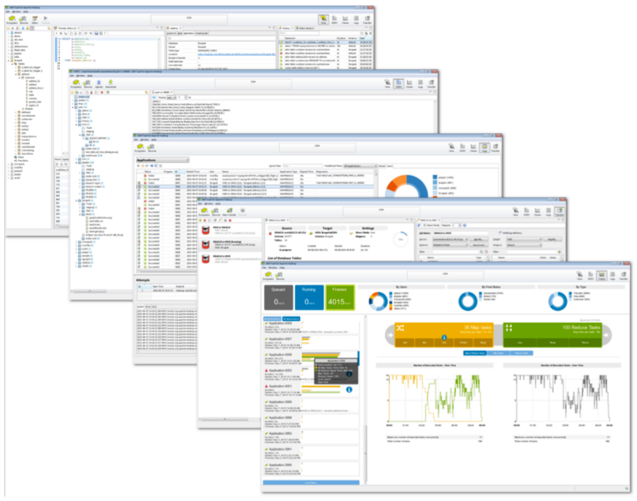
The steps called out in the above process are maintained by HiveServer 2, but in reality all data processing is happening on the worker nodes in the cluster.

Hive SQL is converted to Map/Reduce jobs and run on Hadoop. The SQL is optimized, just as in an RDBMS, for best performance.

Query Submission Tools



```
[it1@sandbox ~]$ beeline -u jdbc:hive2://localhost:10000
WARNING: Use "yarn jar" to launch YARN applications.
Connecting to jdbc:hive2://localhost:10000
Connected to: Apache Hive (version 1.2.1000.2.4.0.0-169)
Driver: Hive JDBC (version 1.2.1000.2.4.0.0-169)
Transaction isolation: TRANSACTION_REPEATABLE_READ
Beeline version 1.2.1000.2.4.0.0-169 by Apache Hive
0: jdbc:hive2://localhost:10000> show tables;
+-----------------------------------+--+
|             tab_name              |
+-----------------------------------+--+
| avg_mileage                       |
| driver_mileage                    |
| finalresults                      |
| geo_normal_event                  |
| geolocation                       |
| geolocation_stage                 |
| hcatsmokeid000a0f02_date250116    |
| risk_factor                       |
| risk_factor_spark                 |
| sample_07                         |
| sample_08                         |
| truck_mileage                     |
| trucks                            |
| trucks_stage                      |
+-----------------------------------+--+
14 rows selected (0.696 seconds)
0: jdbc:hive2://localhost:10000>
```



Use familiar command-line and SQL GUI tools just as with “normal” RDBMS technologies.

Beeline

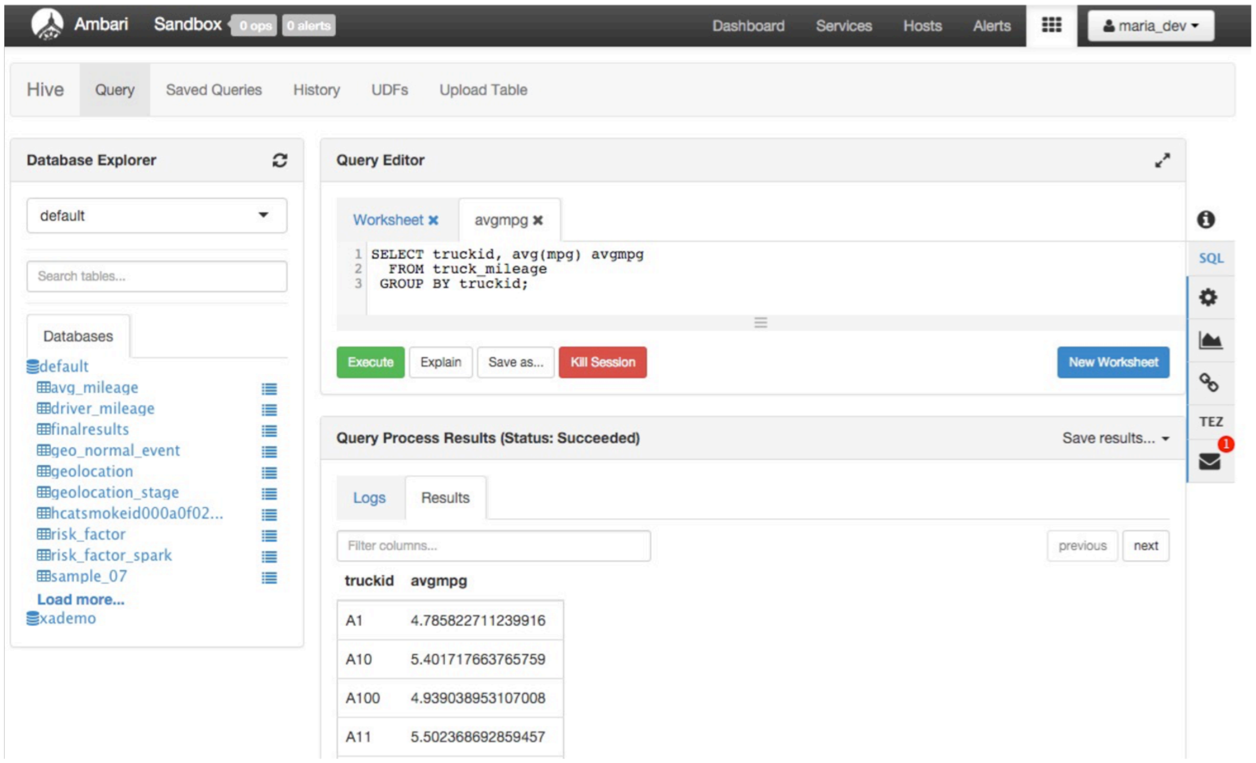
HiveServer2 (introduced in Hive 0.11) has its own CLI called Beeline. HiveCLI is now deprecated in favor of Beeline, as it lacks the multi-user, security, and other capabilities of HiveServer2. Beeline is started with the JDBC URL of the HiveServer2, which depends on the address and port where HiveServer2 was started. By default, it will be (localhost:10000), so the address will look like jdbc:hive2://localhost:10000.

GUI Tools

Open source SQL tools used to query Hive include:

- Ambari Hive View (http://docs.hortonworks.com/HDPDocuments/Ambari-2.2.0.0/bk_ambari_views_guide/content/)
- Zeppelin (https://zeppelin.incubator.apache.org/)
- DBVisualizer (https://www.dbvis.com/download/)
- Dbeaver (http://dbeaver.jkiss.org/)
- SquirrelSQL (http://squirrel-sql.sourceforge.net/)
- SQLWorkBench (http://www.sql-workbench.net/)

Ambari Hive View



```
SELECT truckid, avg(mpg) avgmpg
  FROM truck_mileage
 GROUP BY truckid;
```

truckid	avgmpg
A1	4.785822711239916
A10	5.401717663765759
A100	4.939038953107008
A11	5.502368692859457

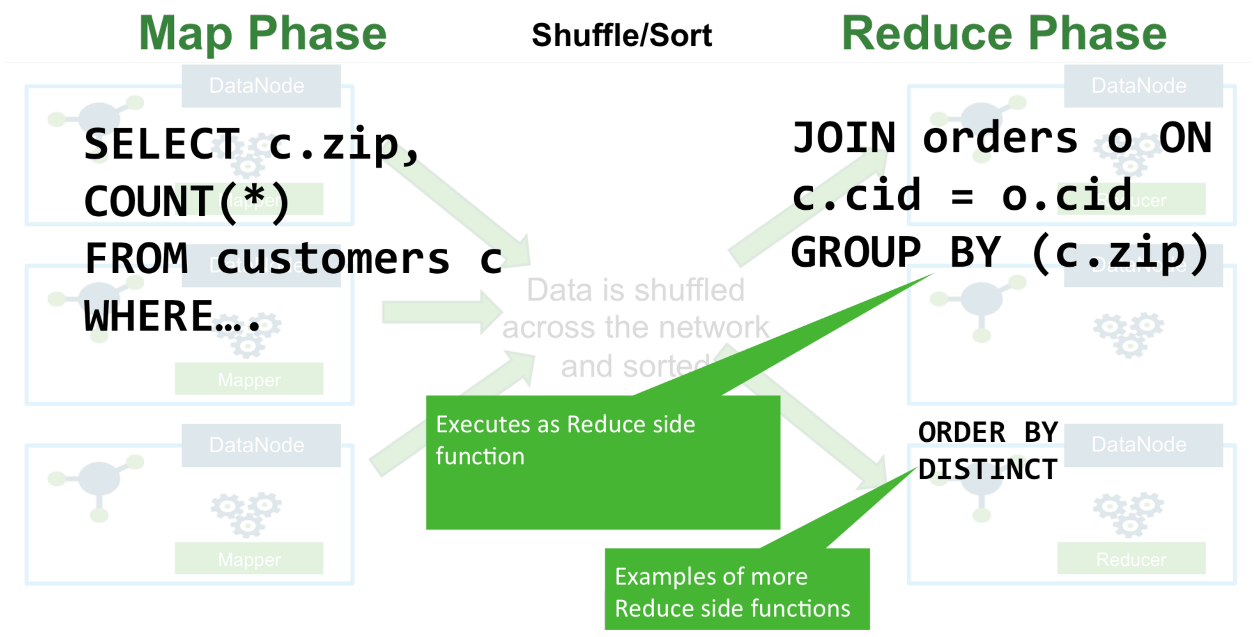
Ambari Hive View

Ambari includes a built-in set of Views that are pre-deployed for you to use with your cluster. The Hive View is designed to help you author, execute, understand, and debug Hive queries.

From the Ambari Hive View you can:

- Browse databases
- Write and execute queries
- Manage query execution jobs and history

Performing Queries in Hive



Hive Remains MapReduce "Under the Covers"

Hive still thinks in MaprReduce terms – even with Tez.

Defining a Hive-Managed Table

```
CREATE TABLE customer (
          customerID INT,
          firstName STRING,
          lastName STRING,
          birthday TIMESTAMP,
  ) ROW FORMAT DELIMITED
    FIELDS TERMINATED BY ',';
```

Defining a Hive-Managed Table

The main difference between Hive and SQL table creation is the specification of how it is stored, such as each row's field delimiter. If the table definition is dropped, the underlying data is lost.

Defining an External Table

```
CREATE EXTERNAL TABLE salaries (
   gender string,
   age int,
   salary double,
   zip int
 ) ROW FORMAT DELIMITED
   FIELDS TERMINATED BY ',';
```

Defining an External Hive Table

An external data uses existing Hadoop data; it makes existing data look like a relational table. If the table definition is dropped, the underlying data will remain.

Defining a Table LOCATION

```
CREATE EXTERNAL TABLE SALARIES (
   gender string,
   age int,
   salary double,
   zip int
 ) ROW FORMAT DELIMITED
   FIELDS TERMINATED BY ','
   LOCATION '/user/train/salaries/';
```

Defining a Table LOCATION

If a LOCATION is not supplied, the table's data will reside in `/apps/hive/warehouse/db_name/table_name`.

LOCATION is commonly to map EXTERNAL tables, but can be used for other types.

Loading Data into Hive

```
LOAD DATA LOCAL INPATH '/tmp/customers.csv' OVERWRITE INTO TABLE
customers;
```

```
LOAD DATA INPATH '/user/train/customers.csv' OVERWRITE INTO TABLE
customers;
```

```
INSERT INTO birthdays
    SELECT firstName, lastName, birthday
    FROM customers
    WHERE birthday IS NOT NULL;
```

Loading Data into Hive

Data can be loaded into Hive tables from external files or from other tables. The table definition (FIELDS TERMINATED BY ',') allows proper identification of each column.

The OVERWRITE clause replaces all existing table data instead of appending to it.

The last example shows the population of a table with the results of a SELECT statement using data from an existing table.

A pair of CREATE TABLE and INSERT statements can be combined into one Create Table As Select statement.

Performing a Query

```
SELECT * FROM customers;
```

```
FROM customers
    SELECT firstName, lastName, address, zip
    WHERE orderID > 0
    GROUP BY zip;
```

```
SELECT customers.*, orders.*
    FROM customers
    JOIN orders ON
    (customers.customerID = orders.customerID);
```

Performing a Hive Query

HIVE SELECT statements are just like those in SQL.

Ambari Hive Views

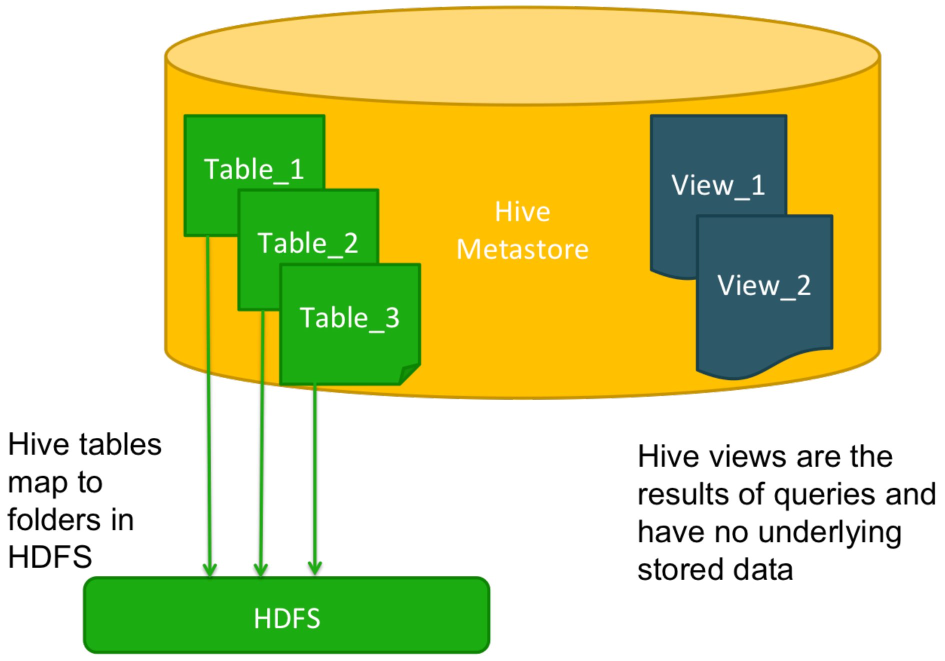


Table_1
Table_2
Table_3
Hive Metastore
View_1
View_2
Hive tables map to folders in HDFS
HDFS
Hive views are the results of queries and have no underlying stored data

Hive Tables

A view is a "synthetic" table. It is essentially a predefined SELECT statement that can be referenced as if it were a table.

Hive views are not "materialized", that is, they are created by running the SELECT action when they are accessed, so they always have current data.

Defining a Hive View

```
CREATE VIEW 2010_visitors AS
  SELECT fname, lname,
     time_of_arrival, info_comment
  FROM wh_visits
  WHERE
  cast(substring(time_of_arrival,6,4)
AS int) >= 2010
  AND
  cast(substring(time_of_arrival,6,4)
AS int) < 2011;
```

Defining a Hive View

A view is defined using the CREATE VIEW statement.

You can run the DESCRIBE command on a view to see its schema, and also see it in the output of "show tables;"

Using Views

```
from 2010_visitors
  select *
  where info_comment like "%CONGRESS%"
  order by lname;
```

Using a Hive View

When you access a view, HIVE will parse, optimize and execute the query just as if a physical table was accessed, substituting the view's definition into the query plan in place of the table name. Hive will determine the best way to convert the above command into one or more MapReduce jobs at runtime .

Performance Improvements

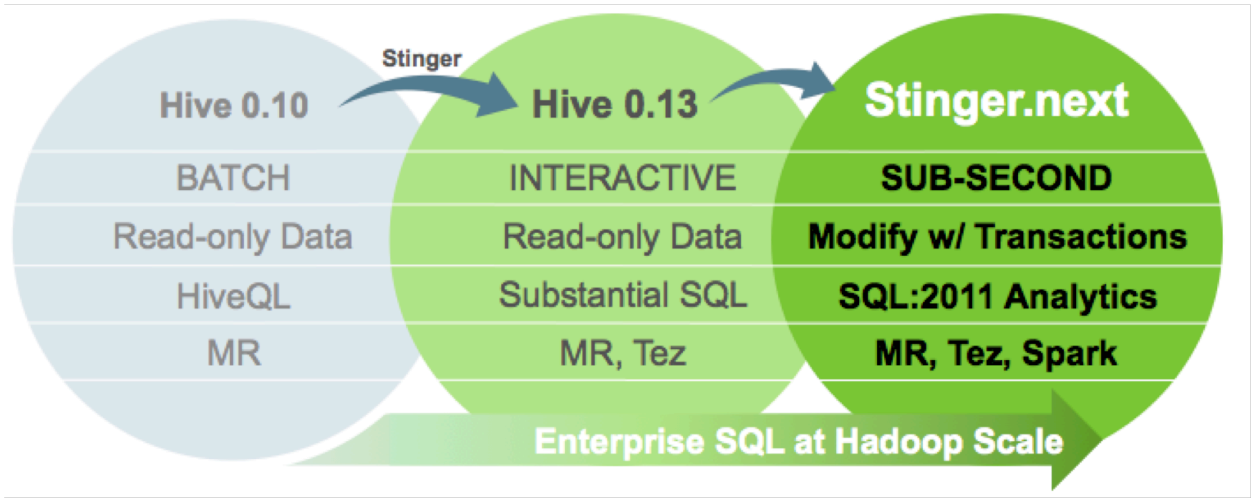
Both Stinger and Tez have increased the performance of Hive queries.

The **Stinger Initiative** enables Hive to support an even broader range of use cases at truly Big Data scale: bringing it beyond its Batch roots to support interactive queries – all with a common SQL access layer.

Tez improves the MapReduce paradigm by dramatically improving its speed, while maintaining MapReduce's ability to scale to petabytes of data.

Hive has always been the defacto standard for SQL in Hadoop and these advances will surely accelerate the production deployment of Hive across a much wider array of scenarios.

Singer.next



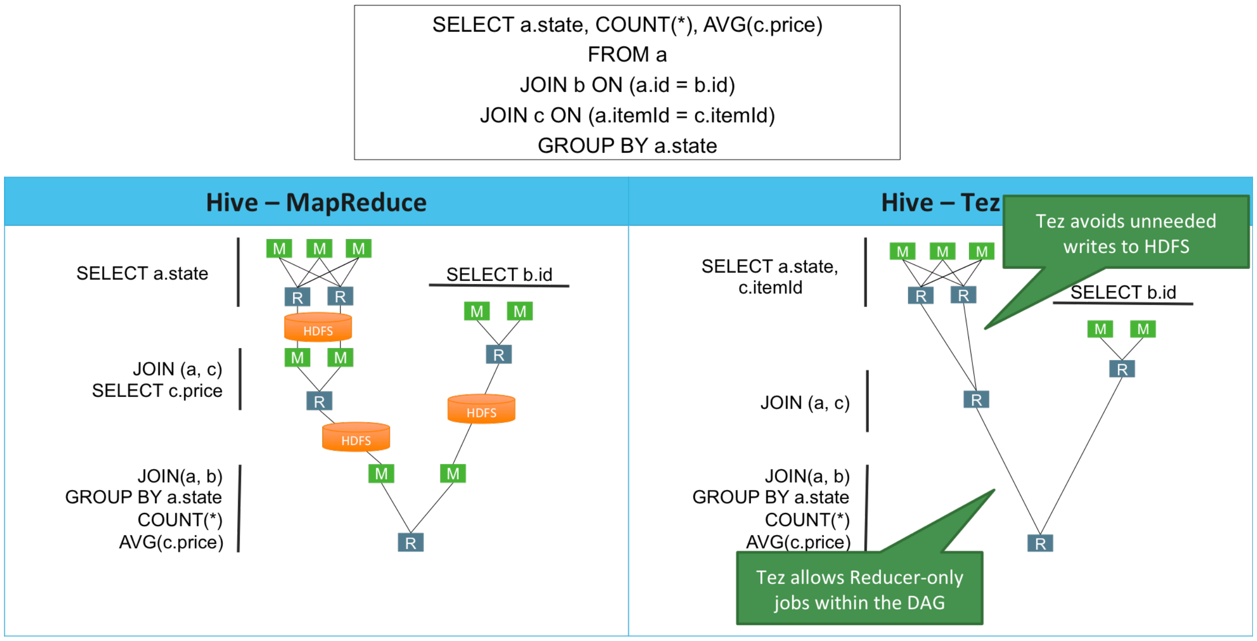
The Stringer Initative

Stinger.next is a continuation of the Stinger initiative focused on even further enhancing the speed, scale and breadth of SQL support to enable truly real-time access in Hive while also bringing support for transactional capabilities. And just as the original Stinger initiative did, this will be addressed through a familiar three-phase delivery schedule and developed completely in the open Apache Hive community.

Stinger.next Goals

- **Speed**
  Deliver sub-second query response times.
- **Scale**
  The only SQL interface to Hadoop designed for queries that scale from Gigabytes, to Terabytes and Petabytes.
- **SQL**
  Enable transactions and SQL:2011 Analytics for Hive.

Tez



Apache Tez

Apache™ Tez is an extensible framework for building high performance batch and interactive data processing applications, coordinated by YARN in Apache Hadoop. Important Hadoop ecosystem projects like Apache Hive and Apache Pig use Apache Tez, as do a growing number of third party data access applications developed for the broader Hadoop ecosystem.

Apache Tez provides a developer API and framework to write native YARN applications that bridge the spectrum of interactive and batch workloads. It allows those data access applications to work with petabytes of data over thousands nodes. The Apache Tez component library allows developers to create Hadoop applications that integrate natively with Apache Hadoop YARN and perform well within mixed workload clusters.

Summary

- Hive is the data warehouse system for Hadoop and uses the familiar table and SQL metaphors that are used with classic RDBMS solutions
- Hive can create, populate and query tables
- Views are supported, but are not materialized
- Significant performance improvements have surfaced from the Stinger initiative including the use of the ORC file format and Tez as the execution engine

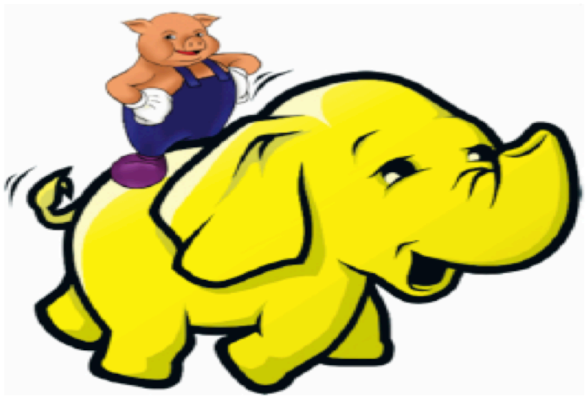
Apache Pig Overview

Lesson Objectives

After completing this lesson, students should be able to:

- ✓ Describe Pig features and functions

Apache Pig



Apache Pig

Apache Pig is a scripting platform for processing and analyzing large data sets.

Pig was designed to perform long series of data operations, making it ideal for three categories of Big Data jobs:

- **Extract-transform-load** (ETL) data pipelines
- **Research** on raw data
- **Iterative data processing**

Pig's design goals can be described as:

- **Pigs eat anything**
  Pig can process any data, structured or unstructured
- **Pigs live anywhere**
  Pig can run on any parallel data processing framework, so Pig scripts do not have to run just on Hadoop
- **Pigs are domestic animals**
  Pig is designed to be easily controlled and modified by its users
- **Pigs fly**
  Pig is designed to process data quickly

Pig Latin

The language for the platform is called Pig Latin, which abstracts from the Java MapReduce idiom into a form similar to SQL. While SQL is designed to query the data, Pig Latin allows you to write a data flow that describes how your data will be transformed (such as aggregate, join and sort).

Pig executes in a unique fashion:

- During execution, each statement is processed by the Pig interpreter
- If a statement is valid, it gets added to a logical plan built by the interpreter
- The steps in the logical plan do not actually execute until a DUMP or STORE command is used

Since Pig Latin scripts can be graphs (instead of requiring a single output) it is possible to build complex data flows involving multiple inputs, transforms, and outputs. Users can extend Pig Latin by writing their own functions, using Java, Python, Ruby, or other scripting languages. Pig Latin is sometimes extended using UDFs (User Defined Functions), which the user can write in any of those languages and then call directly from the Pig Latin.

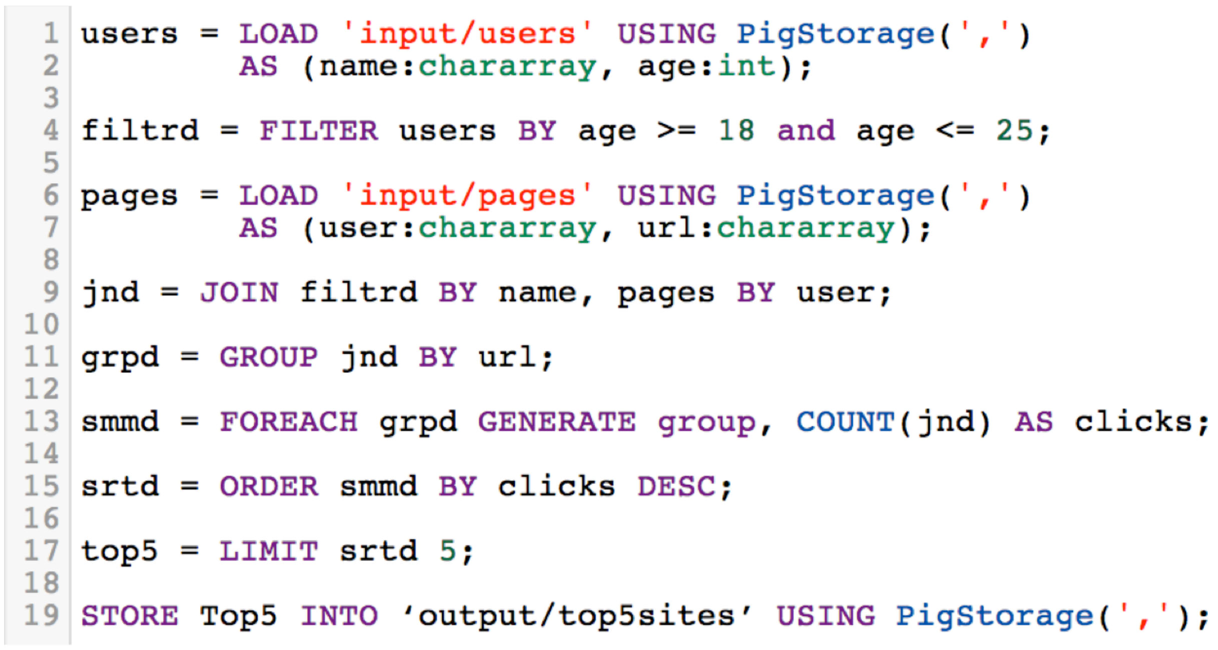
The user can run Pig in two modes, using either the “pig” command or the “java” command:

MapReduce Mode. This is the default mode, which requires access to a Hadoop cluster.

Local Mode. With access to a single machine, all files are installed and run using a local host and file system.

Why Use Pig

Apache Pig allows Apache Hadoop users to write complex MapReduce transformations using a simple scripting language called Pig Latin. Pig translates the Pig Latin script into MapReduce so that it can be executed within YARN for access to a single dataset stored in the Hadoop Distributed File System (HDFS).



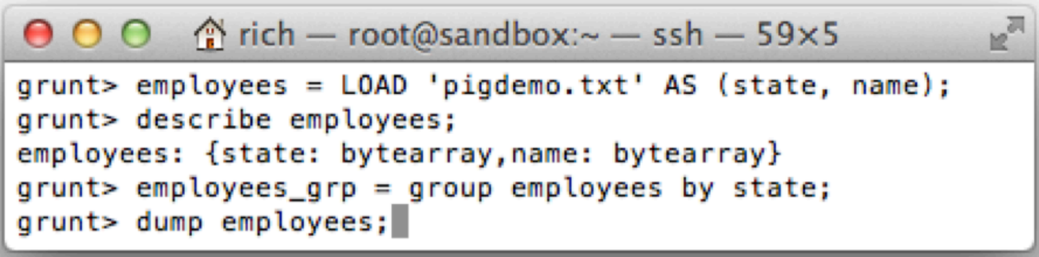
```
users = LOAD 'input/users' USING PigStorage(',')
        AS (name:chararray, age:int);

filtrd = FILTER users BY age >= 18 and age <= 25;

pages = LOAD 'input/pages' USING PigStorage(',')
        AS (user:chararray, url:chararray);

jnd = JOIN filtrd BY name, pages BY user;

grpd = GROUP jnd BY url;

smmd = FOREACH grpd GENERATE group, COUNT(jnd) AS clicks;

srtd = ORDER smmd BY clicks DESC;

top5 = LIMIT srtd 5;

STORE Top5 INTO ‘output/top5sites’ USING PigStorage(',');
```

Pig Simplifies the Developer's Job

The Grunt Shell

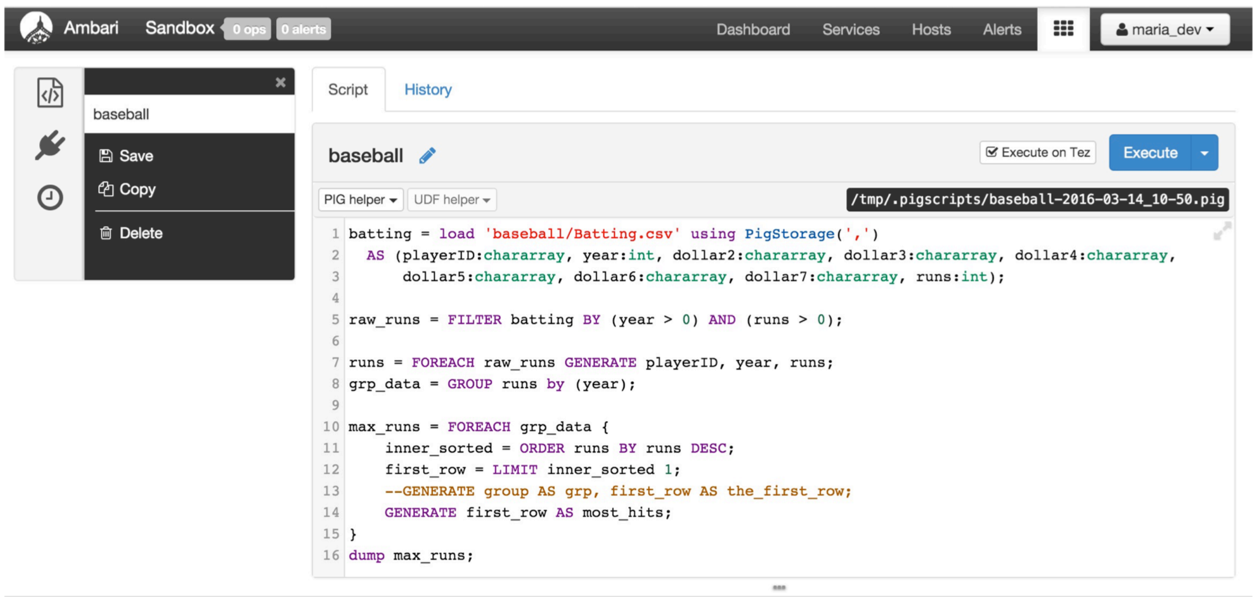
Grunt is Pig's interactive shell. Users can enter commands interactively and interact with HDFS. Grunt provides:

- A command-line history
- Editing
- Tab completion



```
grunt> employees = LOAD 'pigdemo.txt' AS (state, name);
grunt> describe employees;
employees: {state: bytearray,name: bytearray}
grunt> employees_grp = group employees by state;
grunt> dump employees;
```

Ambari Pig View



```
batting = load 'baseball/Batting.csv' using PigStorage(',')
  AS (playerID:chararray, year:int, dollar2:chararray, dollar3:chararray, dollar4:chararray,
      dollar5:chararray, dollar6:chararray, dollar7:chararray, runs:int);

raw_runs = FILTER batting BY (year > 0) AND (runs > 0);

runs = FOREACH raw_runs GENERATE playerID, year, runs;
grp_data = GROUP runs by (year);

max_runs = FOREACH grp_data {
    inner_sorted = ORDER runs BY runs DESC;
    first_row = LIMIT inner_sorted 1;
    --GENERATE group AS grp, first_row AS the_first_row;
    GENERATE first_row AS most_hits;
}
dump max_runs;
```

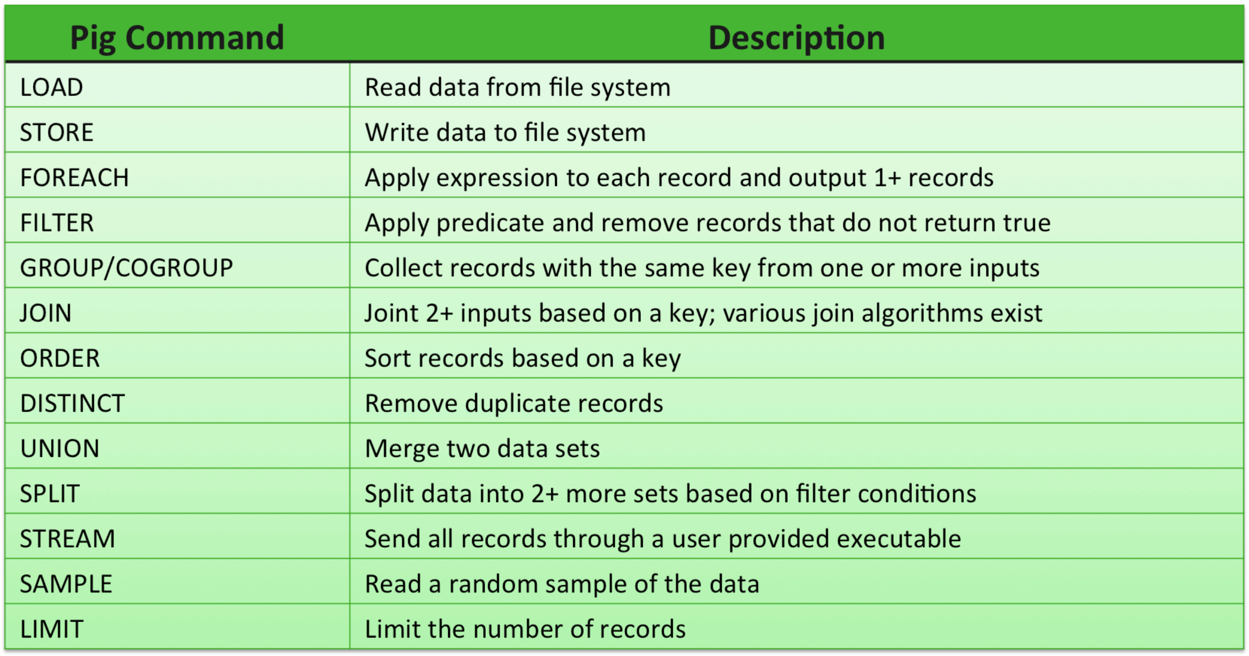
Executing a Script in Ambari Pig View

The **Pig View** provides a web-based interface to compose, edit, and submit Pig scripts, download results, and view logs and the history of job submissions.

You can use Pig View to:

- Write Pig scripts
- Execute Pig scripts
- Add user-defined functions (UDFs) to Pig scripts
- View the history of all Pig scripts run by the current user

Pig Latin Commands



Pig Command	Description
LOAD	Read data from file system
STORE	Write data to file system
FOREACH	Apply expression to each record and output 1+ records
FILTER	Apply predicate and remove records that do not return true
GROUP/COGROUP	Collect records with the same key from one or more inputs
JOIN	Joint 2+ inputs based on a key; various join algorithms exist
ORDER	Sort records based on a key
DISTINCT	Remove duplicate records
UNION	Merge two data sets
SPLIT	Split data into 2+ more sets based on filter conditions
STREAM	Send all records through a user provided executable
SAMPLE	Read a random sample of the data
LIMIT	Limit the number of records

Pig Latin Commands

DataFu Library

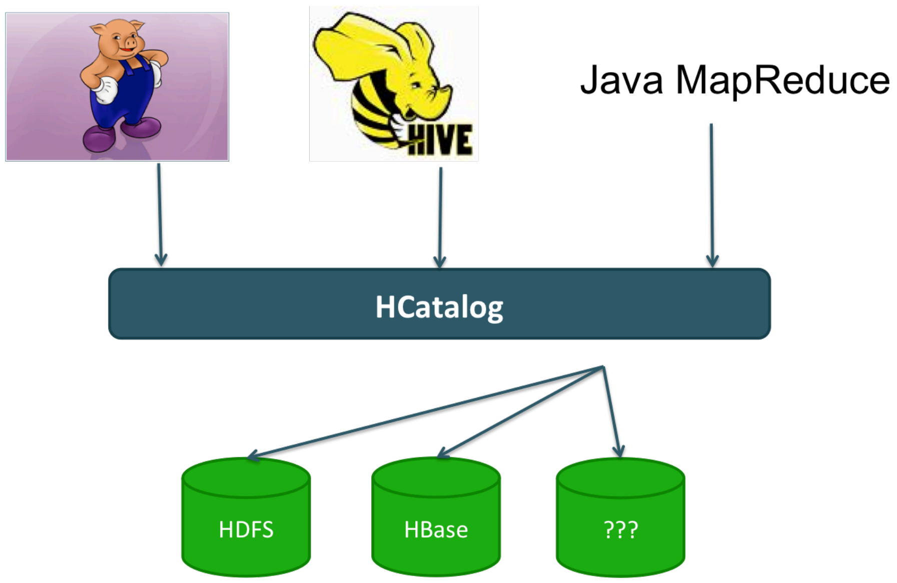
The DataFu Library is a collection of Pig UDFs for data analysis on Hadoop. Started by LinkedIn it is no open source under the Apache 2.0 license.

DataFu includes functions for:

- Bag and set operations
- PageRank
- Quantiles
- Variance
- Sessionization

To use the functions in the DataFu library, you need to register the DataFu JAR file, just like you would with any other Pig UDF library: `register datafu-1.2.0.jar;`

HCatalog



HCatalog in the Hadoop Ecosystem

Apache™ HCatalog is a table management layer that exposes Hive metadata to other Hadoop applications. HCatalog's table abstraction presents users with a relational view of data in the Hadoop Distributed File System (HDFS) and ensures that users need not worry about where or in what format their data is stored. HCatalog displays data from RCFile format, text files, or sequence files in a tabular view. It also provides REST APIs so that external systems can access these tables' metadata.

HCatalog:

- Frees the user from having to know where the data is stored, with the table abstraction
- Enables notifications of data availability
- Provides visibility for data cleaning and archiving tools

Summary

- Pig is a high-level data-flow scripting language
- Scripts do not execute until an I/O operation like DUMP or STORE are reached
- Can be run via the interactive shell or as a script
- Pig has a comprehensive set of commands available to programmers
- DataFu library is a collection of Pig UDFs for data analysis on Hadoop
- HCatalog provides a consistent data model for the various tools that use Hadoop

Apache Spark Overview

Lesson Objectives

After completing this lesson, students should be able to:

- ✓ Describe Spark with special focus on
- ✓ Discuss performance considerations
- ✓ Describe modules that layer on top of Spark Core

Apache Spark Overview

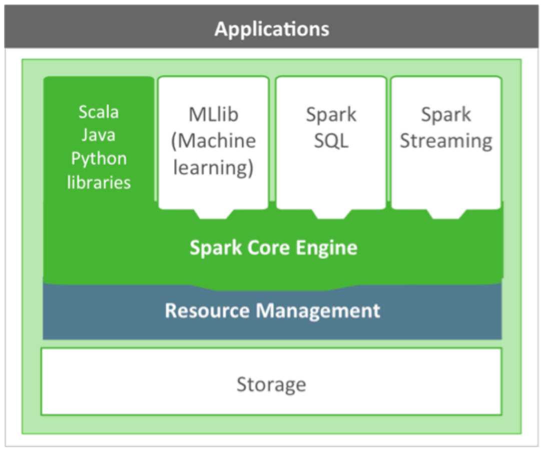
What is Spark?

Spark was created to be a general purpose data processing engine, focused on in-memory distributed computing use cases. Spark took many concepts from MapReduce and implemented them in new ways. Spark uses API's written in Scala, that can be called with Scala, Python, Java and more recently R. The core Spark API's focus on using key/value pairs for data manipulation.

In addition to the Core Spark, a few extraction frameworks have been developed on top of the core API. Most notable is the Spark SQL module. The Spark SQL module allows developers to seamlessly mix SQL queries within their Spark applications, unlocking more doors to potential data processing applications.

Spark is build around the concept of an RDD. RDD stands for Resilient Distributed Dataset. Resilient in the sense that data can be recreated on the fly in the event of a machine failure. Distributed in the sense that operations can happen in parallel on multiple machines working on blocks of data allowing Spark to scale very easily as data and machine network size grows.

Apache Spark



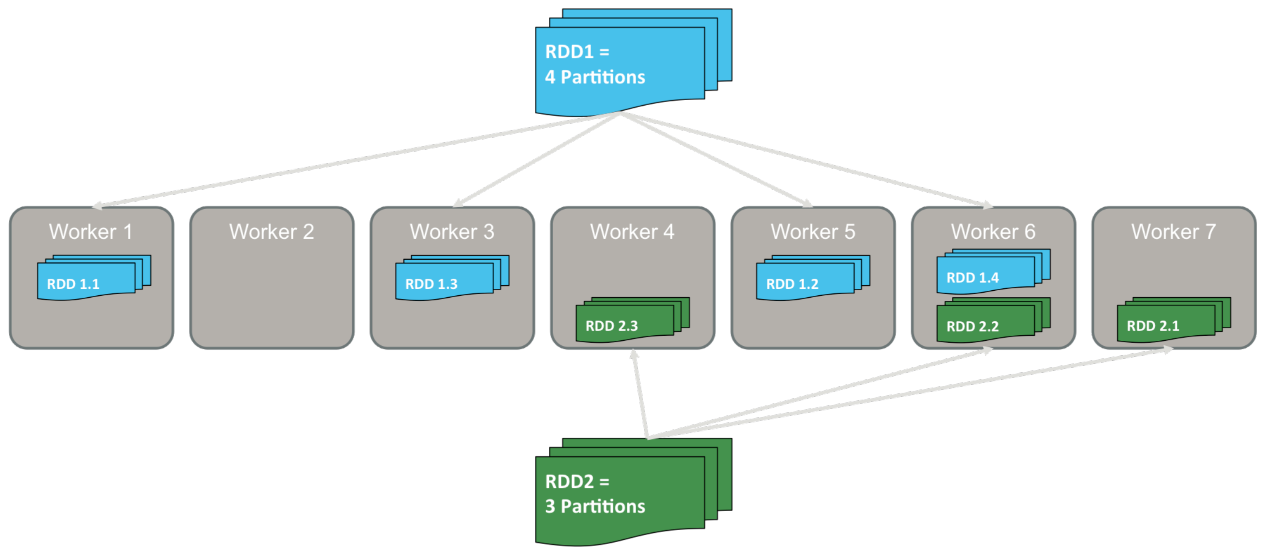
Apache Spark

Spark is a data access engine for fast, large-scale data processing designed for iterative in-memory computations and interactive data mining. It provides expressive multi-language APIs for Scala, Java and Python.

Data workers can use built-in libraries to rapidly iterate over data for:

- ETL
- Machine learning
- SQL workloads
- Stream processing

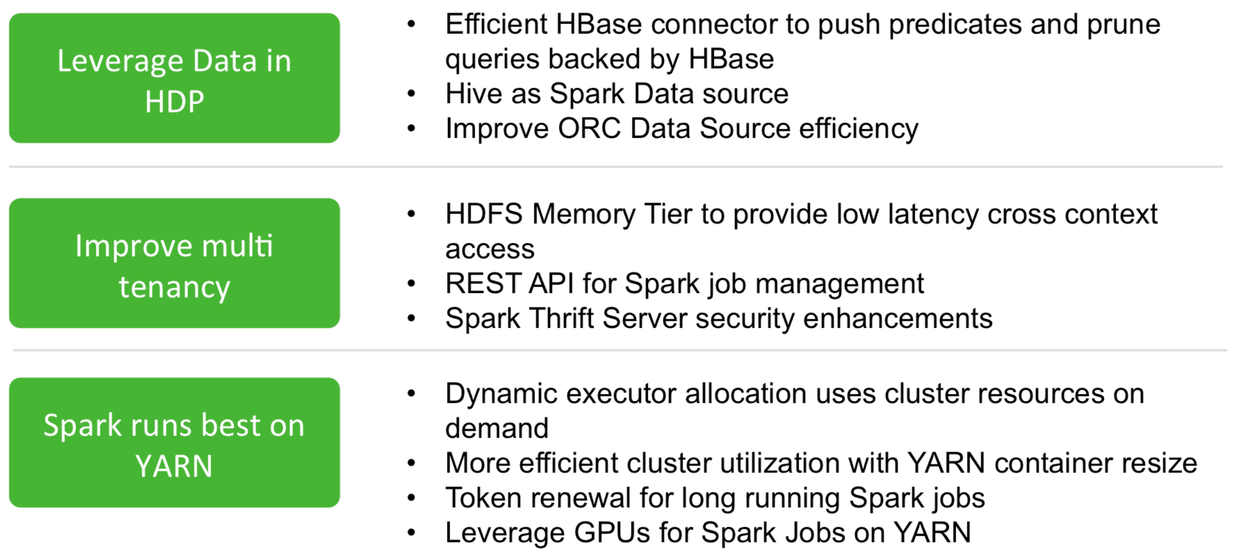
Cluster with Two RDDs



Cluster with Two RDDs

This diagram shows a hypothetical cluster that has two RDD's. Each RDD is composed of multiple partitions, and the partitions are distributed across the cluster.

Spark Focus



Leverage Data in HDP

- Efficient HBase connector to push predicates and prune queries backed by HBase
- Hive as Spark Data source
- Improve ORC Data Source efficiency

Improve multi tenancy

- HDFS Memory Tier to provide low latency cross context access
- REST API for Spark job management
- Spark Thrift Server security enhancements

Spark runs best on YARN

- Dynamic executor allocation uses cluster resources on demand
- More efficient cluster utilization with YARN container resize
- Token renewal for long running Spark jobs
- Leverage GPUs for Spark Jobs on YARN

Apache Spark Focus

Spark was able to solve these issues. Spark is able to leverage data in HDP, HBase included. An Hbase connector exists, which allows the pushing of predicates and pruning of data, that is backed by Hbase.

Spark is able to leverage Hive tables and the HiveMetastore to use data already stored using Hive. In addition, Spark can do some data pruning in conjunction with ORC Data format to improve efficiency even more.

Spark provides a rest API for Spark Job management, allowing a way to kick off jobs remotely.

Spark has been focusing on Security and can leverage the security enhancement in the Spark Thrift Server

Spark is getting better at playing nicely when working on a multi tenant cluster. Spark can be submitted using YARN, and recently dynamic executor allocation. This allows for Spark applications to utilize cluster resources more efficiently.

Spark Shell vs. Spark Applications

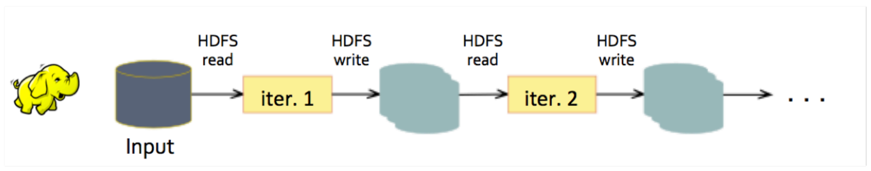
The difference between a Spark Application and running a job in the Spark is minimal. This is another example of Spark being extremely friendly. We can move from ad-hoc data analysis to production-scheduled applications with only minimal differences.

Spark Shell allows interactive manipulation/exploration of data for use in data discovery and building pipelines interactively.

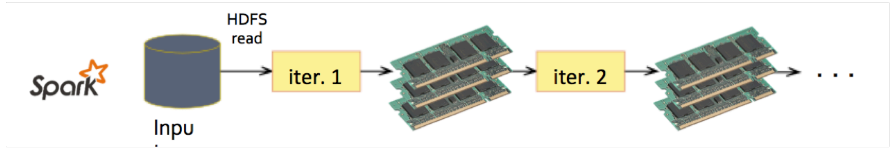
Spark Applications run as independent programs that can be scheduled. They are commonly used for ETL processing, streaming and model building.

Performance Considerations

- **MapReduce** – involves *lots of disk I/O* (Disk I/O is very slow)



- **Spark** – Keep more data in memory



Spark Motivations

The initial motivation for Spark was that iterative applications didn't work well with MapReduce. Iterative applications in MapReduce require intermediate data to be written to the HDFS and then read again for every iteration. Spark's goal was to keep more data in memory, to minimize the expensive read/writes that plague developers. Reading from memory is measured in nanoseconds, while reading from disk is measured in milliseconds.

Spark RDD Persist Options

Storage Level	Where?	Storage format	Comments
MEMORY_ONLY	RAM	Deserialized	Default level
MEMORY_AND_DISK	RAM and DISK	Deserialized	Disk is backup for partitions that don't fit in memory
MEMORY_ONLY_SER	RAM	Serialized	Reduced RAM but more CPU intensive
MEMORY_AND_DISK_SER	RAM AND DISK	Serialized	Reduced RAM but more CPU intensive
DISK_ONLY	DISK	Deserialized	
MEMORY_ONLY_2	RAM	Deserialized	Stores each partition on two cluster nodes
MEMORY_AND_DISK_2	RAM AND DISK	Deserialized	Stores each partition on two cluster nodes
OFF_HEAP	TACHYON	Serialized	Experimental

Spark RDD Persist Options

Hortonworks recommends using the persist API when building an application The reason being is that it forces the developer to be conscious of what level of storage they're using and remain aware of what is going on as well. There are many different types of options for persisting, called Storage Levels, and these can alter the performance of the application. There are a combination of ways data can be stored, which come down to three main parts: serialized vs. un-serialized, memory, and disk. All the persistence levels are made up of combinations of these.

MEMORY : Data that is persisted will be persisted in RAM

DISK : Data that is persisted will be persisted to local disk

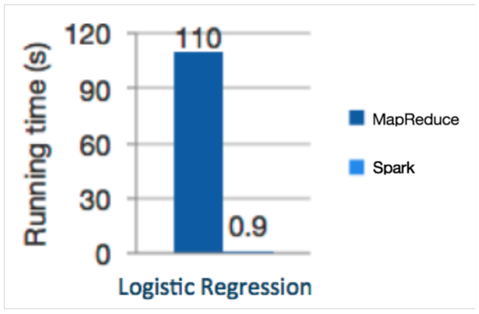
SER: Data will be serialized before writing.

So in the case of MEMORY_AND_DISK_SER, the data will be serialized before being persisted. The data will then be stored in RAM until the application runs out of RAM when the rest of the persisted data will be spilled to local disk.

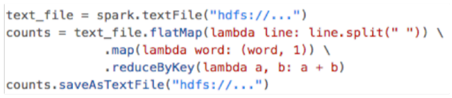
There is one other StorageLevel that hasn't been discussed and that is **OFF_HEAP.** This is an experimental API that takes advantage of OFF_HEAP storage and has potential to allow developers to persist quite a bit more data. It is, however, not ready for use in a production environment.

Spark vs. MapReduce

- Higher level API
- In-memory data storage
  - Up to 100x performance improvement

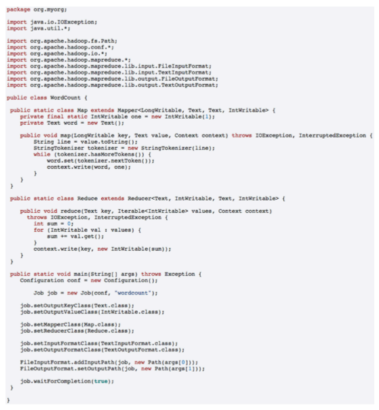


Pyspark



```
text_file = spark.textFile("hdfs://...")
counts = text_file.flatMap(lambda line: line.split(" ")) \
             .map(lambda word: (word, 1)) \
             .reduceByKey(lambda a, b: a + b)
counts.saveAsTextFile("hdfs://...")
```

Java MapReduce



```
package org.myorg;

import java.io.IOException;
import java.util.*;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

public class WordCount {

 public static class Map extends Mapper<LongWritable, Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
        String line = value.toString();
        StringTokenizer tokenizer = new StringTokenizer(line);
        while (tokenizer.hasMoreTokens()) {
            word.set(tokenizer.nextToken());
            context.write(word, one);
        }
    }
 }

 public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable> {

    public void reduce(Text key, Iterable<IntWritable> values, Context context)
      throws IOException, InterruptedException {
        int sum = 0;
        for (IntWritable val : values) {
            sum += val.get();
        }
        context.write(key, new IntWritable(sum));
    }
 }

 public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();

        Job job = new Job(conf, "wordcount");

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

    job.setMapperClass(Map.class);
    job.setReducerClass(Reduce.class);

    job.setInputFormatClass(TextInputFormat.class);
    job.setOutputFormatClass(TextOutputFormat.class);

    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));

    job.waitForCompletion(true);
 }

}
```

Spark vs. MapReduce: WordCount

Pictured above are a WordCount implementation in Spark using Python API and Java MapReduce. We can see immediately how much plumbing is removed when we implement a simple word count. Spark can perform wordcount in ~3 lines of code, where as MapReduce requires over 50.

Also, as the diagram shows, when taking advantage of in-memory storage, Spark performs over 100x faster then tradition MapReduce for some use cases.

Why is Spark Faster?

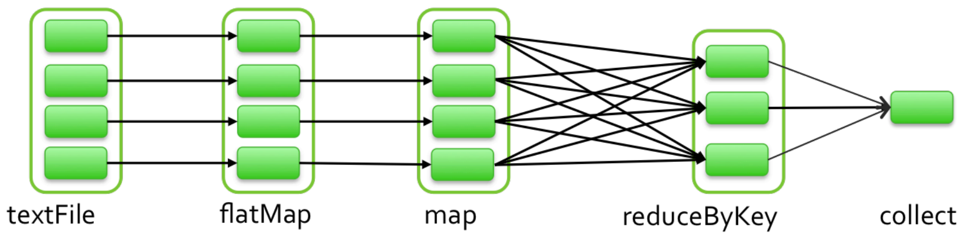
Spark is faster than MapReduce for several reasons.

First and foremost, Spark can cache data into memory.

In addition to the gains of reading data from disk vs memory, task scheduling in Spark has greatly decreased from MapReduce. Spark has dedicated resources, so task scheduling doesn't require a resource request. Because of this, scheduling has gone from 15-20s to 15-20ms. In Spark, you can have multiple reduces and maps in a row. You do not need a map phase for every reduce phase; skipping this extra map saves reading and writing data to disk.

Still Based on MapReduce Principles

```
sc.textFile("/some-hdfs-data") \                        RDD[String]
 .flatMap(lambda line: line.split(" ")) \               RDD[List[String]]
 .map(lambda line: (word, 1))) \                        RDD[(String, Int)]
 .reduceByKey(lambda a,b : a+b, \                       RDD[(String, Int)]
       numPartition=3) \                                Array[(String, Int)]
 .collect()
```



Spark is Based on MapReduce

The code on the left of this picture should look familiar: it is the wordcount application that has been referenced many times. On the bottom is an example of what is going on with the partitions of data. In the first line of code, a file is read in. The number of partitions defaults to the number of blocks the file takes up on the HDFS. Here, the file takes up 4 blocks on the HDFS, as represented by the 4 green rectangles above the textFile syntax.

Spark Modules

Various modules are layered on top of the Spark Core. These include Spark SQL, Spark Streaming, and Spark MLlib. Sparl also integrates with Apache Zeppelin, a web-based notebook server that supports Data Science.

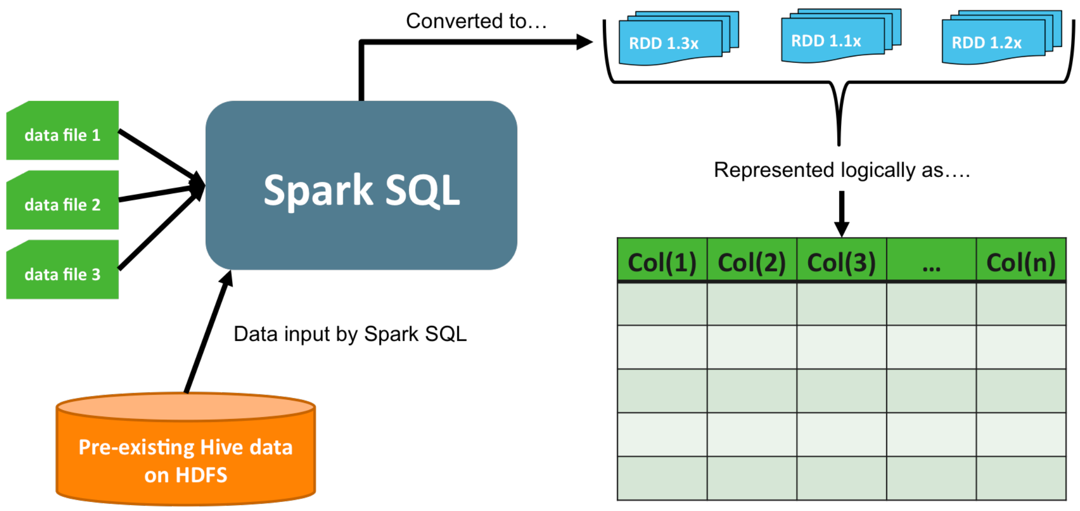
Spark SQL Overview

Spark SQL is a module that is built on top of Spark Core for structured data processing. Spark SQL provides another level of abstraction for declarative programming on top of Spark.

In Spark SQL, data is described as a DataFrame with rows, columns and a schema. Data manipulation in Spark SQL is available via SQL queries, the DataFrames API, and the Datasets APUI.

Spark SQL is being used more and more at the enterprise level. Spark SQL allows the developer to focus even more on the business logic and even less on the plumbing, and even some of the optimizations.

DataFrames



The Spark DataFrame Visually

The image above shows what a data frame looks like visually. Much like Hive, a DataFrame is a set of metadata that sits on top of an RDD. The RDD can be created from many file types. A DataFrame is conceptually equivalent to a table in traditional data warehousing.

Datasets

Datasets are new experimental interfaces added in Spark 1.6 for the purpose of providing the benefits of RDDs (strong typing, ability to use powerful lambda functions) with the benefits of Spark SQL's optimized execution engine. A Dataset can be constructed from JVM objects and then manipulated using functional transformations (map, flatMap, filter, etc.).

The unified Dataset API can be used both in Scala and Java. Python does not yet have support for the Dataset API, but due to its dynamic nature many of the benefits are already available (i.e. you can access the field of a row by name naturally row.columnName). Full python support will be added in a future release.

REFERENCE:

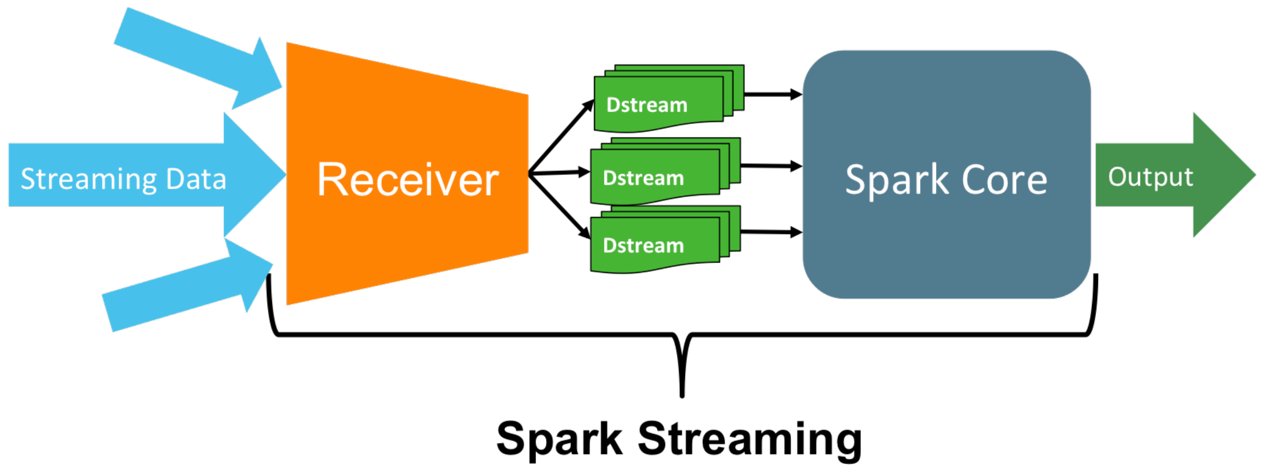
See the "Spark SQL, DataFrames and Datasets Guide" at
http://spark.apache.org/docs/latest/sql-programming-guide.html#overview

Spark Streaming Overview

Spark Streaming is a library built on top of the Spark Core framework that enables scalable, high throughput, fault-tolerant stream processing. Spark Streaming is able to reuse many of the same API's that Spark Core uses. Spark Streaming utilizes a micro-batch architecture, allowing it to process data with as low as 1-2s of latency.

Spark introduces a new abstraction of data called **Discretized Streams** (Dstreams). Dstreams are batches of data that share very little physically with RDD's, but are used very similarly to RDD's.

Input Dstreams and Receivers



Spark Architecture

Spark usually does not introduce many new concepts with new modules. Spark Streaming is the closest to an exception, as the new concepts of a receiver and Dstream are introduced. They are introduced because streaming is a bit different than batch processing. A streaming application is composed of a receiver, Core Spark, and this concept of a Dstream.

Input Dstreams represent the stream of input data received from streaming sources. There are two types of input Dstreams: **Basic sources** that are directly available in the SgtreamingContext API and **Advanced sources** that are available through extra utility classes.

Receivers listen to data streams, and create batches of data called Dstreams that are then processed by the Spark Core engine. There are two types of receivers: **reliable receivers** which send acknowledgment when data has been received and **unreliable receivers** that do not send ackowledgements.

Once the data is ingested, we have the liberty to use all Spark frameworks - we are not limited to using only Spark streaming functionalities.

REFERENCE:

See "Spark Streaming Programming Guide"
http://spark.apache.org/docs/latest/streaming-programming-guide.html#discretized-streams-dstreams

MLlib Overview

MLlib provides Spark a library of common machine learning algorithms and utilities. This includes:

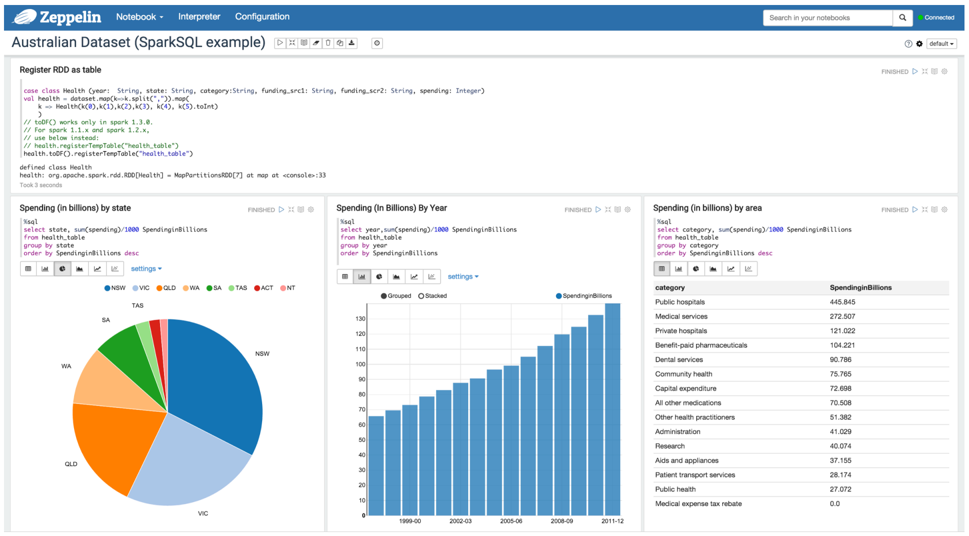
- Classification
- Regression
- Clustering
- Collaborative filtering
- Dimensionality reduction

MLlib is composed of algorithms that scale well and perform well in a parallel processing world. MLlib allows data scientist the ability to easily scale machine learning algorithms on a Hadoop cluster.

Apache Zeppelin

Features	Use Cases
A web-based notebook for interactive analytics —Ad-hoc experimentation with Spark, Hive, Shell, Flink, Tajo, Ignite, Lens, etc.	Data exploration and discovery Visualization—tables, graphs and charts
Deeply integrated with Spark and Hadoop —Can be managed via Ambari Stacks	Interactive snippet-at-a-time experience Collaboration and publishing
Supports multiple language backends —Pluggable "Interpreters"	"Modern Data Science Studio"
Incubating at Apache —100% open source and open community	

Interactive Zeppelin browser-based notebooks enable data engineers, data analysts and data scientists to be more productive by developing, organizing, executing, and sharing data code and visualizing results without referring to the command line or needing the cluster details. Notebooks allow these users not only allow to execute but to interactively work with long workflows.



Australian Dataset (SparkSQL example)

Register RDD as table

```
case class Health (year:  String, state: String, category:String, funding_src1: String, funding_scr2: String, spending: Integer)
val health = dataset.map(k=>k.split(",")).map(
    k => Health(k(0),k(1),k(2),k(3), k(4), k(5).toInt)
    )
// toDF() works only in spark 1.3.0.
// For spark 1.1.x and spark 1.2.x,
// use below instead:
// health.registerTempTable("health_table")
health.toDF().registerTempTable("health_table")
```

```
defined class Health
health: org.apache.spark.rdd.RDD[Health] = MapPartitionsRDD[7] at map at <console>:33
Took 3 seconds
```

Spending (in billions) by state

```
%sql
select state, sum(spending)/1000 SpendinginBillions
from health_table
group by state
order by SpendinginBillions desc
```

Spending (In Billions) By Year

```
%sql
select year,sum(spending)/1000 SpendinginBillions
from health_table
group by year
order by SpendinginBillions
```

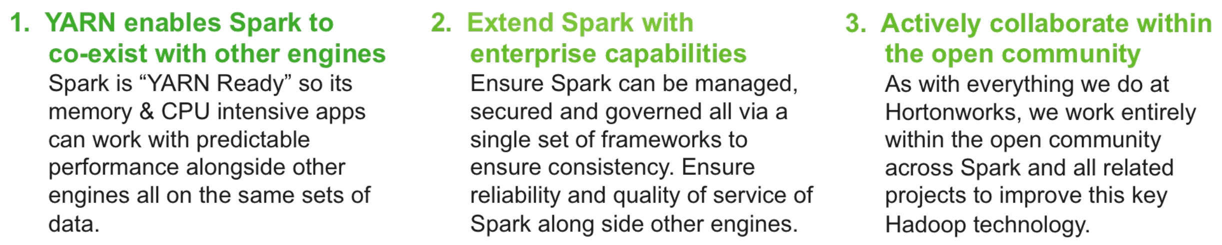
Spending (in billions) by area

```
%sql
select category, sum(spending)/1000 SpendinginBillions
from health_table
group by category
order by SpendinginBillions desc
```

category	SpendinginBillions
Public hospitals	445.845
Medical services	272.507
Private hospitals	121.022
Benefit-paid pharmaceuticals	104.221
Dental services	90.786
Community health	75.765
Capital expenditure	72.698
All other medications	70.508
Other health practitioners	51.382
Administration	41.029
Research	40.074
Aids and appliances	37.155
Patient transport services	28.174
Public health	27.072
Medical expense tax rebate	0.0

Apache Zeppelin

Hortonworks' Commitment to Spark



1. **YARN enables Spark to co-exist with other engines**
   Spark is "YARN Ready" so its memory & CPU intensive apps can work with predictable performance alongside other engines all on the same sets of data.
2. **Extend Spark with enterprise capabilities**
   Ensure Spark can be managed, secured and governed all via a single set of frameworks to ensure consistency. Ensure reliability and quality of service of Spark along side other engines.
3. **Actively collaborate within the open community**
   As with everything we do at Hortonworks, we work entirely within the open community across Spark and all related projects to improve this key Hadoop technology.

Hortonworks' Commitment to Spark

Hortonworks has made a large investment in Spark to ensure enterprise readiness for mission critical applications.

Summary

- Spark houses data in an RDD structure and allows re-parallelization as needed
- The “sweet spot” is iterative in-memory computations and interactive data modeling
- Python, Scala, Java and R are supported languages
- Provides data processing, ETL, machine learning, stream processing, SQL querying
- In-memory caching is not a default setting and there are many options to choose from
- Maintains dedicated resources and its task scheduler is lightning fast
- Spark SQL has a DataFrame API In addition to classical SQL querying
- Spark Streaming uses micro-batches that are much like RDDs loaded from disk
- MLlib allows data scientists the ability to easily scale machine learning algorithms
- Apache Zeppelin is considered the “Modern Data Science Studio”

YARN Overview

Lesson Objectives

After completing this lesson, students should be able to:

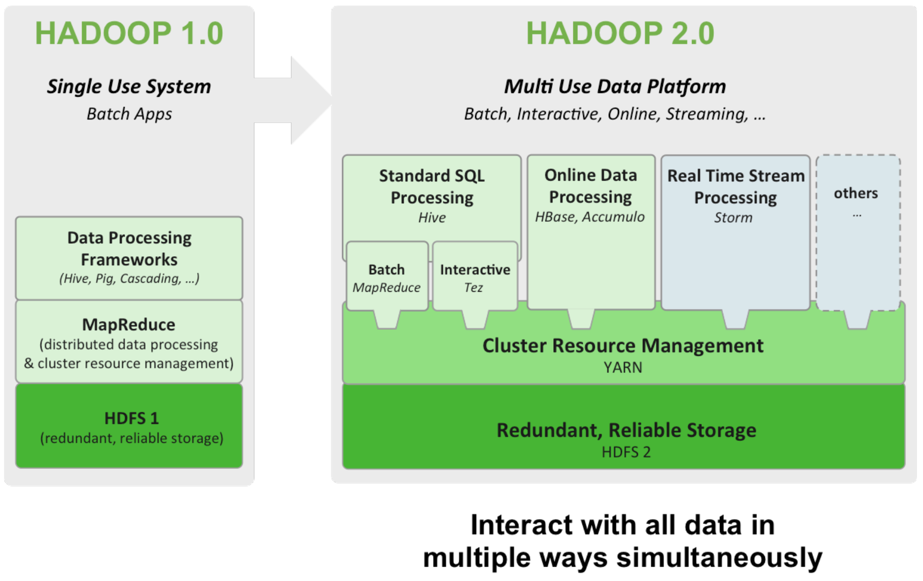
- ✓ Describe the purpose and components of YARN
- ✓ List YARN Components and Interactions
- ✓ Describe additional YARN Features

Introduction to YARN

YARN is the prerequisite for Enterprise Hadoop, providing resource management and a central platform to deliver consistent operations, security, and data governance tools across Hadoop clusters.

YARN also extends the power of Hadoop to incumbent and new technologies found within the data center so that they can take advantage of cost effective, linear-scale storage and processing. It provides ISVs and developers a consistent framework for writing data access applications that run IN Hadoop.

Enabling Multiple Workloads


HADOOP 1.0
Single Use System
Batch Apps
Data Processing Frameworks
(Hive, Pig, Cascading, ...)
MapReduce
(distributed data processing & cluster resource management)
HDFS 1
(redundant, reliable storage)
HADOOP 2.0
Multi Use Data Platform
Batch, Interactive, Online, Streaming, ...
Standard SQL Processing
Hive
Batch
MapReduce
Interactive
Tez
Online Data Processing
HBase, Accumulo
Real Time Stream Processing
Storm
others
...
Cluster Resource Management
YARN
Redundant, Reliable Storage
HDFS 2
Interact with all data in multiple ways simultaneously

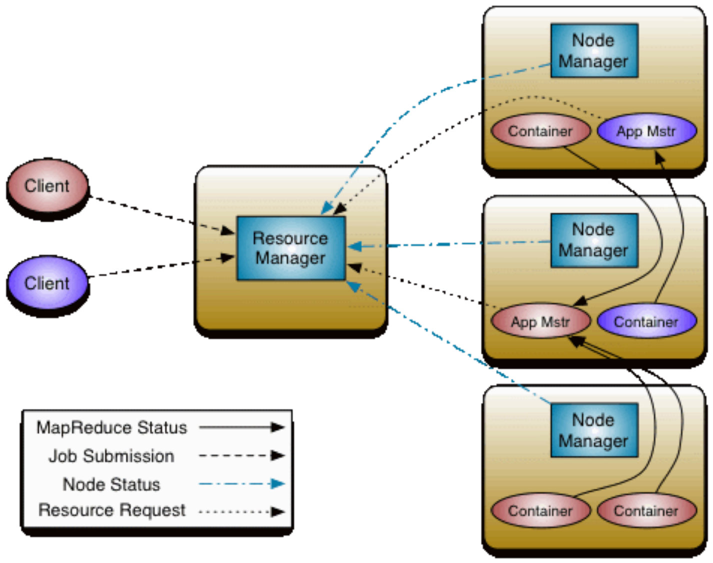
YARN Enables Multiple Workloads

Hadoop 1.0 was mainly used for MapReduce jobs in more of a project level adoption of Hadoop for big data analysis.

Hadopp 2.0 with YARN as its architectural center makes a mixed load data lake a reality by enabling the running of mixed workloads in the same Hadoop cluster. This makes it easy to build a data lake.

Different workloads interact with data in different ways, simultaneously and seamlessly.

YARN Architectural Components



YARN Achitecture

The MapReduce framework was decomposed to generalize resource management from a single workload type. The ApplicationMaster allows clusters to scale beyond the 2000-4000 node range due to the decentralization of tracking independent jobs

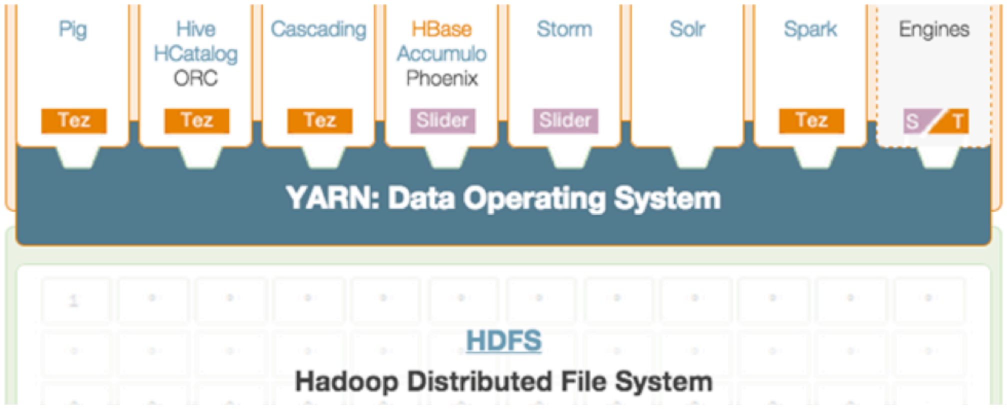
YARN Components and Interactions

As the architectural center of Hadoop, YARN enhances a Hadoop compute cluster through multi-tenancy, dyanmic utilization, scalability and compatibility with MapReduce applications developed for Hadoop 1.x.

YARN Resource Management

YARN

YARN (unofficially "Yet Another Resource Negotiator") is the computing framework for Hadoop. If you think about HDFS as the cluster file system for Hadoop, YARN would be the cluster operating system. It is the architectural center of Hadoop.



YARN as an Operating System

A computer operating system, such as Windows or Linux, manages access to resources, such as CPU, memory, and disk, for installed applications. In similar fashion, YARN provides a managed framework that allows for multiple types of applications – batch, interactive, online, streaming, and so on – to execute on data across your entire cluster.

Just like a computer operating system manages both resource allocation (which application gets access to CPU, memory, and disk now, and which one has to wait if contention exists) and security (does the current user have permission to perform the requested action), YARN manages resource allocation for the various types of data processing workloads, prioritizes and schedules jobs, and enables authentication and multi-tenancy.

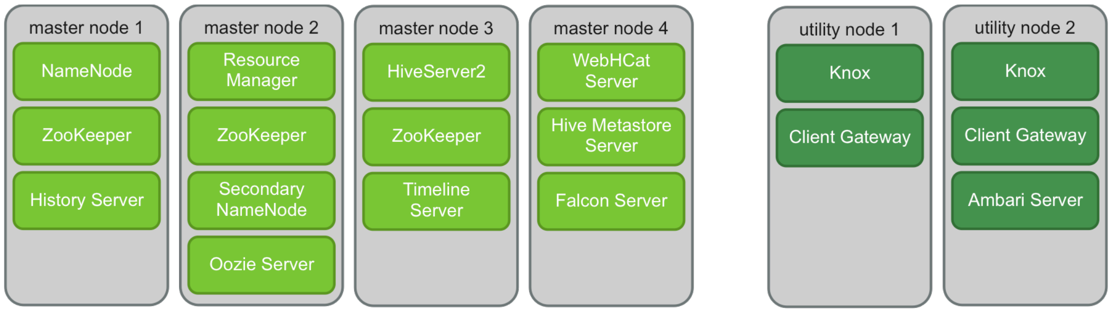
Multi-Tenancy

Software ***multi-tenancy*** is achieved when a single instance of an application serves multiple groups of users, or "tenants." Each tenant shares common access to an application, hardware, and underlying resources (including data), but with specific and potentially unique privileges granted by the application based on their identification. A typical example of a multi-tenant application architecture would be SaaS cloud computing, where multiple users and even multiple companies are accessing the same instance of an application at the same time (for example, Salesforce CRM).

This is in contrast with ***multi-instance*** architectures, where each user gets a unique instance of an application, and the application then competes for resources on behalf of its tenant. A typical example of a multi-instance architecture would be applications running in virtualized or IaaS environments (for example, applications running in KVM virtual machines).

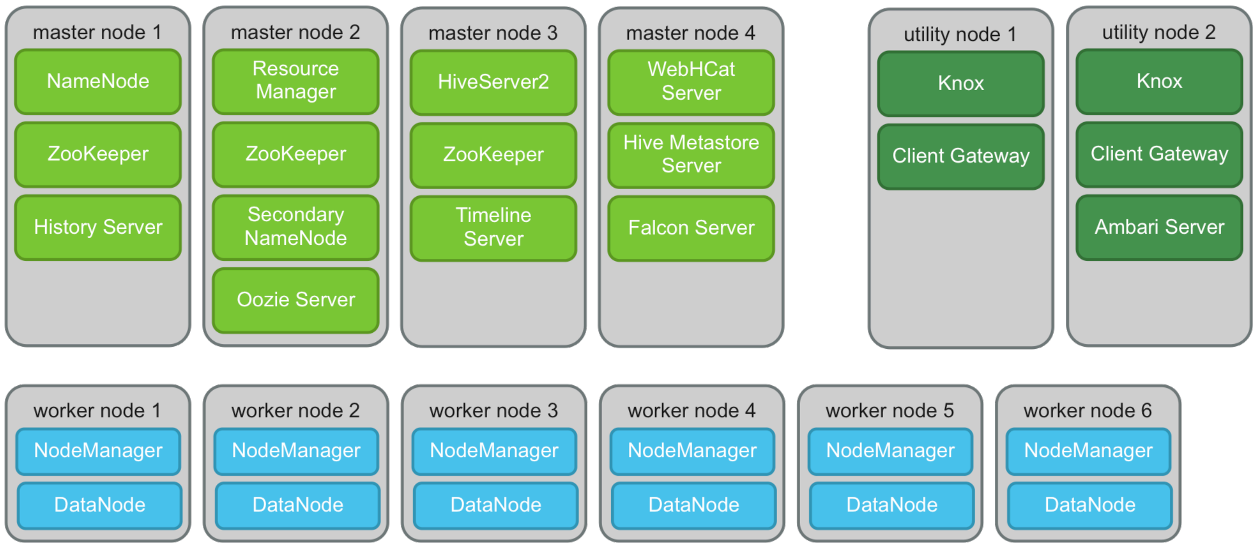
NOTE: In prior versions of Hadoop, resource management was part of the MapReduce process. In this scenario, you had a single application handling both job scheduling and running data processing jobs at the same time. Starting with Hadoop 2.0, MapReduce is simply another data processing application running on top of the YARN framework.

YARN – The Big Picture View



YARN Master and Utility Nodes

The YARN Master node component centrally manages cluster resources for all YARN applications.

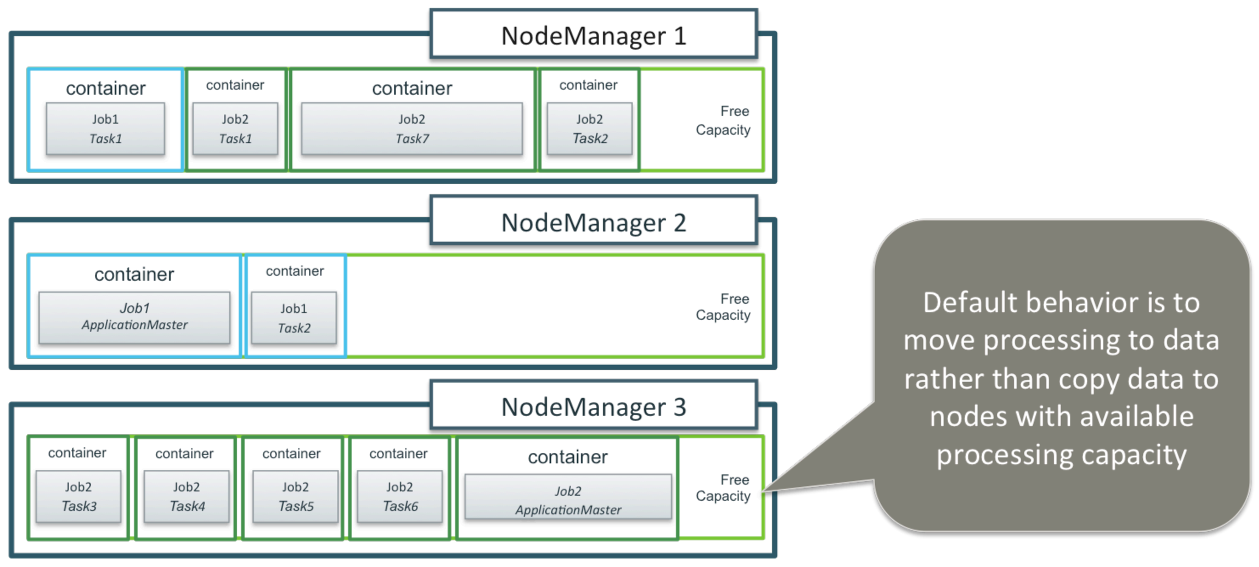


YARN Master, Utility, and Worker Nodes

The YARN Worker node component manages local resources at the direction of the ResourceManager.

YARN Multi-Node Resource Allocation Example

The following graphic illustrates how containers, ApplicationMasters, and job tasks might be spread across a 3-node cluster.



YARN Multi-Node Resource Allocation Scenario

In this example, the Job1 ApplicationMaster was started on NodeManager 2. The first task for Job1 was started and NodeManager 1, and the second Job1 task was started on NodeManager 2. This completed all the tasks required for Job1.

The Job2 ApplicationMaster was launched on NodeManager 3. The first two Job2 tasks were launched on NodeManager 1, Job2 tasks 3 through 6 were launched on NodeManager 3, and the final Job2 task was launched back on NodeManager 1.

The main point to this is to illustrate that the ApplicationMaster can initiate the creation of containers on any appropriate NodeManager in the cluster. The default behavior is for all jobs to be collocated where data blocks already exist, even if more processing power is available on a node without those data blocks whenever possible.

YARN Features

YARN's original purpose was to split up the two major responsibilities of the JobTracker/TaskTracker into separate entities:

- A global ResourceManager
- A per-application ApplicationMaster
- A per-node slave NodeManager
- A per-application Container running on a NodeManager

Resource Manager High Availability

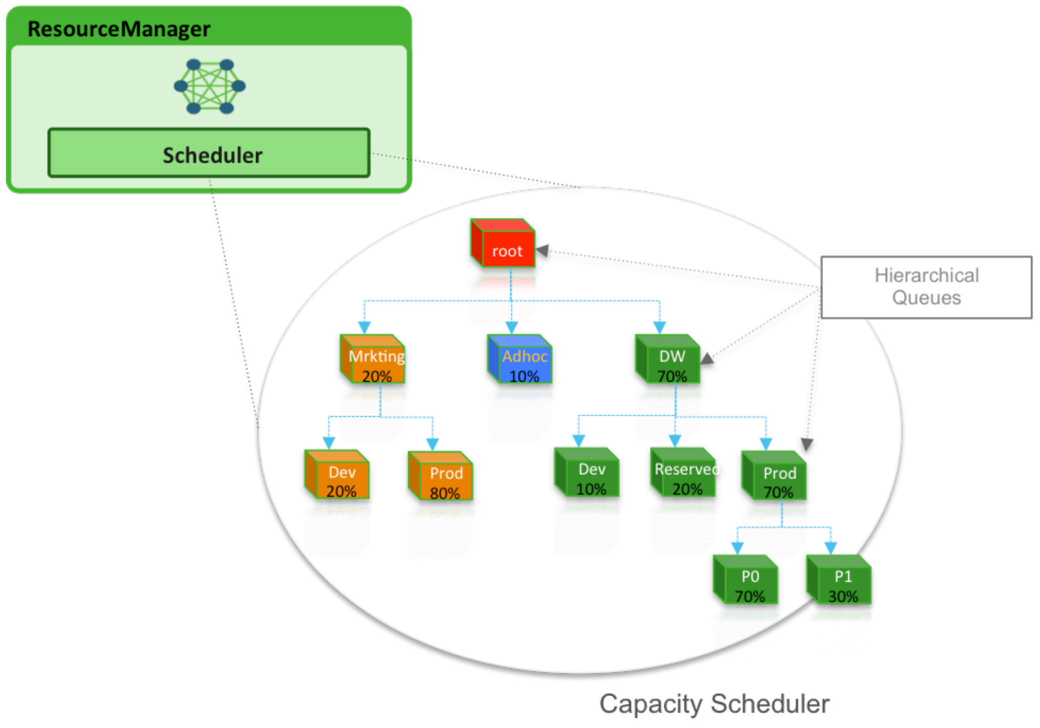
In HDP prior to version 2.1, the ResourceManager was a single point of failure. The entire cluster would become unavailable if the ResourceManager failed or became unreachable. Even maintenance events such as software or hardware upgrades on the ResourceManager machine would result in periods of cluster downtime.

The YARN ResourceManager High Availability (HA) feature eliminates the ResourceManager as a single point of failure. It enables a cluster to run one or more ResourceManagers in an Active/Standby configuration.

ResourceManager HA enables fast failover to the Standby ResourceManager in response to a failure, or a graceful administrator-initiated failover for planned maintenance.

There are two ways of configuring ResourceManager HA. Using Ambari is the easiest way. Manually editing the configuration files and starting or restarting the necessary daemons is also possible. However, manual configuration of ResourceManager HA is not compatible with Ambari administration. Any manual edits to the yarn-site.xml file would be overwritten by information in the Ambari database when the YARN service is restarted.

Multi-Tenancy with Capacity Scheduler



YARN Capacity Scheduler

Traditionally, organizations have had their own compute resources with sufficient capacity to meet SLAs under peak or near peak conditions. This often results in poor average utilization and the increased overhead of managing multiple independent clusters.

While the concept of sharing clusters is logically a cost-effective manner of running large Hadoop installations, individual organizations may have concerns about sharing a cluster, fearing other organizations may use resources critical to their SLAs.

The CapacityScheduler is designed to enable sharing of compute resources in a large cluster while guaranteeing each organization a minimum capacity. Available resources in the cluster are partitioned between multiple organizations based on computing needs. In addition, any organization can access excess capacity at any given point in time. This provides cost-effective elasticity.

Implementation of the CapacityScheduler is based on the concept of queues which are set up by administrators to reflect organizational economics of the shared cluster.

Resource Isolation

Resource isolation in provided on Linux by Control Groups (CGroups) and on Windows through Job Control.

CGroups

Cgroups enable an administrator to allocate resources among processes running on a system. Administrators can:

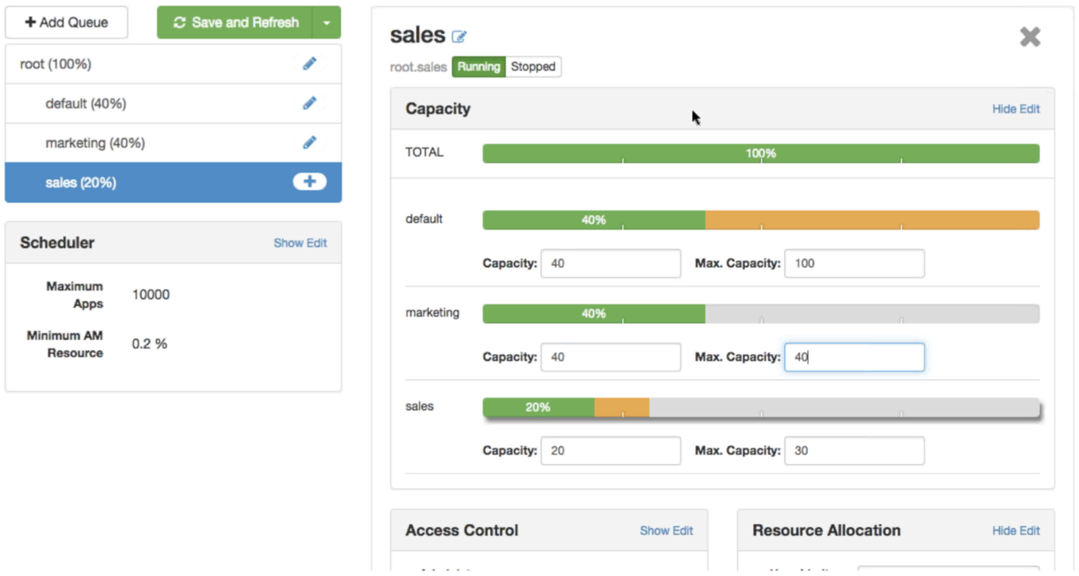
- Monitor cgroups
- Deny cgroups access to certain resources
- Reconfigure cgroups dynamically on a running system

Cgroups can be made persistent across reboots by configuring the **cgconfig** service to run at boot time to reestablish cgroups.

Windows Job Control

Isolation controls similar to Linux CGroups have been implemented on Windows to perform default job control actions. Job control messages can only be processed by customized applications.

Managing Queue Limits with Ambari



root (100%)
default (40%)
marketing (40%)
sales (20%)

Scheduler

Maximum Apps	10000
Minimum AM Resource	0.2 %

sales
root.sales

Capacity

TOTAL 100%

Queue		Capacity	Max. Capacity
default	40%	40	100
marketing	40%	40	40
sales	20%	20	30

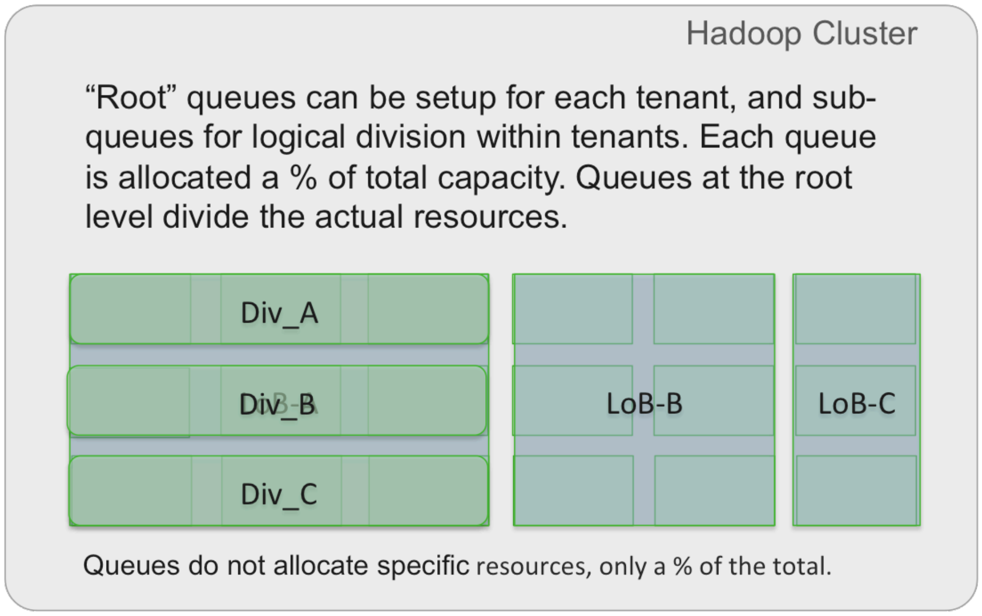
Ambari Queue Limits

Ambari is the Apache project that allows single pane of glass administration of clusters (including multiple clusters). Ambari provides a standard set of tools, APIs and processes to be leveraged across Hadoop instances.

To configure a queue in Ambari, click on the queue name. From here you can set queue parameters:

- **Capacity**
  - Capacity
    The percentage of cluster resources available to the queue. For a sub-queue, the percentage of parent queue resources.
  - Max Capacity
    The maximum percentage of cluster resources available to the queue. Setting this value tends to restrict elasticity, as the queue will be unable to utilize idle cluster resources beyond this setting.
  - Enable Node Labels
    Select this check box to enable node labels for the queue.
- **Access Control and Status**
  - State
    Running is the default state. Setting this to Stopped lets you gracefully drain the queue of jobs (for example, before deleting a queue).
  - Administer Queue
    Clicking Custom will restrict administration of the queue to specific users and groups.
  - Submit Applications
    Clicking Custom will restrict the ability to run applications in the queue to specific users and groups.
- **Resources**
  - User Limit Factor
    A measure of the maximum any user can occupy in a queue. Setting this variable to "1" results in a maximum equal to the queue's configured capacity. The setting is used to prevent any one user from monopolizing resources across all queues in a cluster.
  - Minimum User Limit
    A measure of the minimum percentage of resources allocated to each queue user. For example, to enable equal sharing of the queue capacity among four users, you would set this property to 25%.
  - Maximum Applications
    Enables the administrator to override the Scheduler Maximum Applications setting.
  - Maximum AM Resource
    Enables an administrator to override the Scheduler Maximum AM Resource setting
  - Ordering Policy
    Enables setting FIFO (First In, First Out) or fair (Fair Scheduler where applications get a fair share of capacity regardless of the order in which they were submitted).

Policy-Based Use of Computing Resources


Hadoop Cluster
“Root” queues can be setup for each tenant, and sub-queues for logical division within tenants. Each queue is allocated a % of total capacity. Queues at the root level divide the actual resources.
Div_A
Div_B
Div_C
LoB-B
LoB-C
Queues do not allocate specific resources, only a % of the total.

Policy-Based Computer Resources Use

The use of queues limits access to resources. Sub-queues are possible allowing capacity to be shared within a tenant. Each queue has ACLs associated with users and groups. Capacity guarantees can be set to provide minimum resource allocations and soft and hard limits can be placed on queues.

Summary

- YARN enables multiple workloads to execute simultaneously in the cluster
- The ResourceManager is the master process responsible for fulfilling resource requests and the NodeManager resides on the worker nodes along with the actual Containers that fulfill job functions
- The ApplicationMaster resides within a Container and is the process responsible for running a job (batch or long-lived service) and making appropriate resource requests
- The Capacity Scheduler allows for resource sharing that enables SLA-enabled multi-tenancy

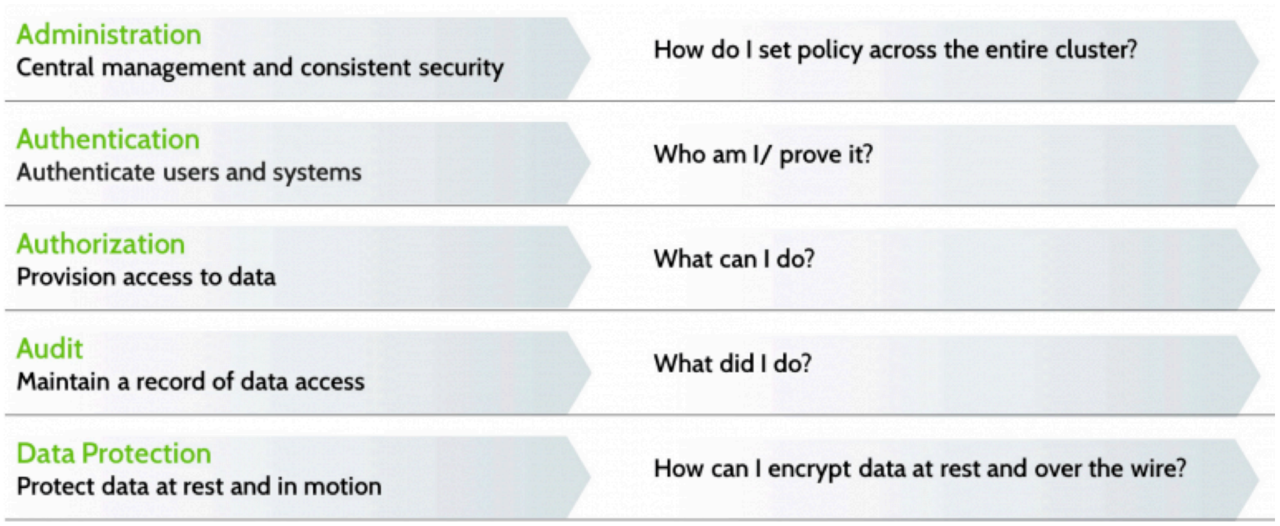
Hadoop Security

Lesson Objectives

After completing this lesson, students should be able to:

- ✓ Describe how Hadoop addresses enterprise security concerns
- ✓ Describe Hortonworks' commitment to enterprise-ready security
- ✓ Describe the high-level architecture of Apache Ranger

Hadoop Security Overview



Administration Central management and consistent security	How do I set policy across the entire cluster?
Authentication Authenticate users and systems	Who am I/ prove it?
Authorization Provision access to data	What can I do?
Audit Maintain a record of data access	What did I do?
Data Protection Protect data at rest and in motion	How can I encrypt data at rest and over the wire?

Five Pillars of Enterprise Security

Effective Hadoop security depends on a holistic approach. The Hortonworks framework for comprehensive security revolves around five pillars: administration, authentication, authorization, audit and data protection.

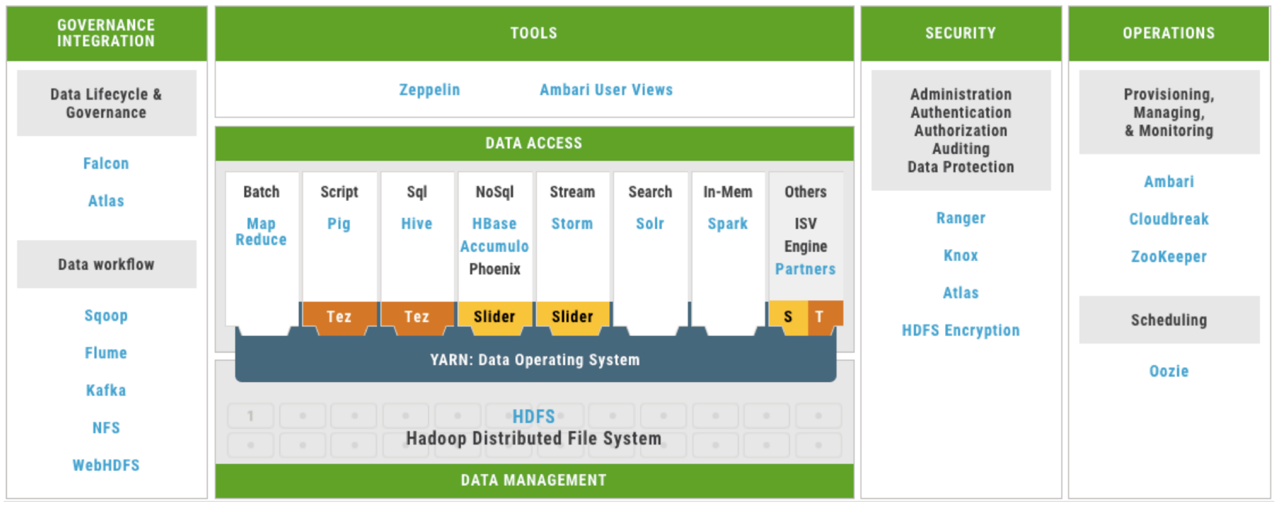
Enterprise infrastructures must provide enterprise-grade coverage across each of these pillars because if any are weak, they introduce thread vectors into the entire data lake.

Security in HDP is the most comprehensive and complete available for Hadoop.

HDP:

- Ensures comprehensive enforcement of security policy across the entire Hadoop stack
- Provides functionality across the complete set of security requirements
- Is the only solution to provide a single simple interface for security policy definition and maintenance

Security Built In to the Platform

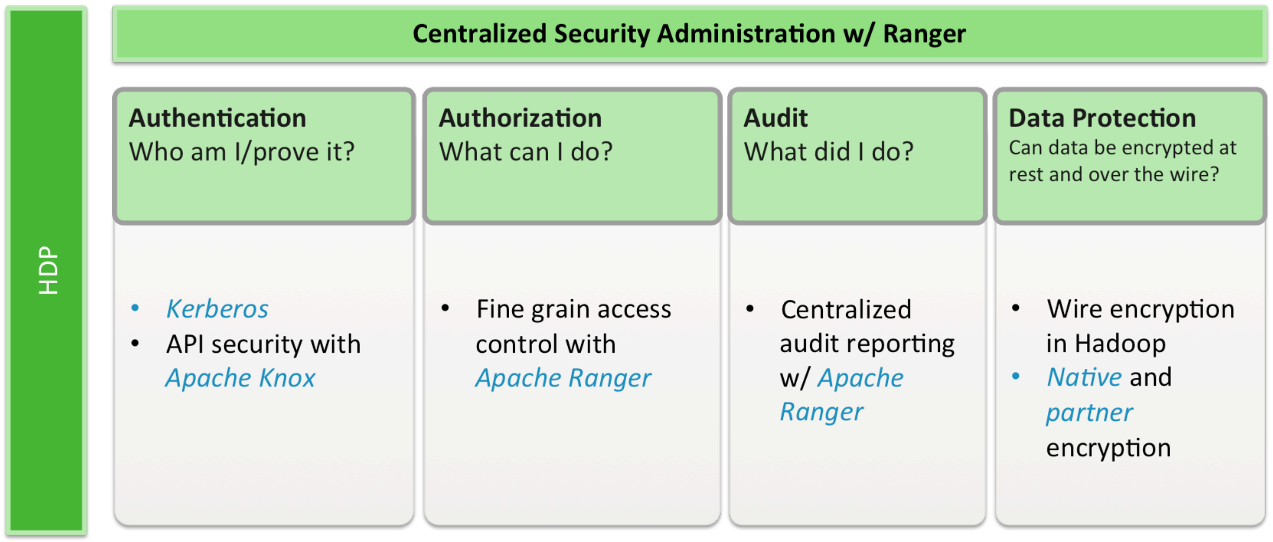


Enforcing Security Across the Hadoop Stack

Security must be an integral part of the platform on which an enterprise's Data Lake is built. The combination of bottom-up and top down approach makes it possible to manage and enforce security across the stack through a central point of administration that prevents gaps and inconsistencies. This approach is especially effective for Hadoop implementations where a dynamic scenario can quickly exacerbate vulnerabilities.

By implementing security at the platform level, Hortonworks ensures that security is consistently administered to any application built on top of the data platform, and makes it easier to build or retire data applications without impacting security.

Hortonworks' Commitment to Security

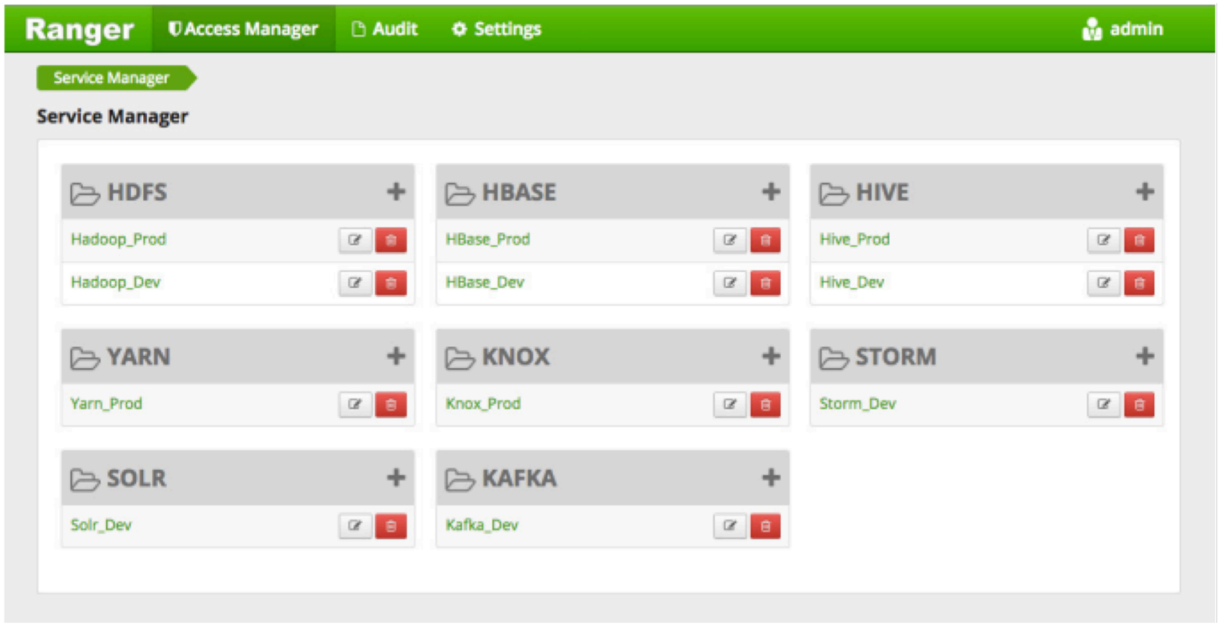

HDP
Centralized Security Administration w/ Ranger
Authentication
Who am I/prove it?
Kerberos
API security with Apache Knox
Authorization
What can I do?
Fine grain access control with Apache Ranger
Audit
What did I do?
Centralized audit reporting w/ Apache Ranger
Data Protection
Can data be encrypted at rest and over the wire?
Wire encryption in Hadoop
Native and partner encryption

Hortonworks' Commitment to Enterprise-Ready Security

Enterprise-ready security must include:

- Central administration
- Authentication
- Authorization
- Audit
- Data protection

Administration



Centralized Security through Ranger

Hadoop administrators need a centralized user interface to deliver security administration and management that can be used to define, administer and manage security policies consistently across all components of the Hadoop stack.

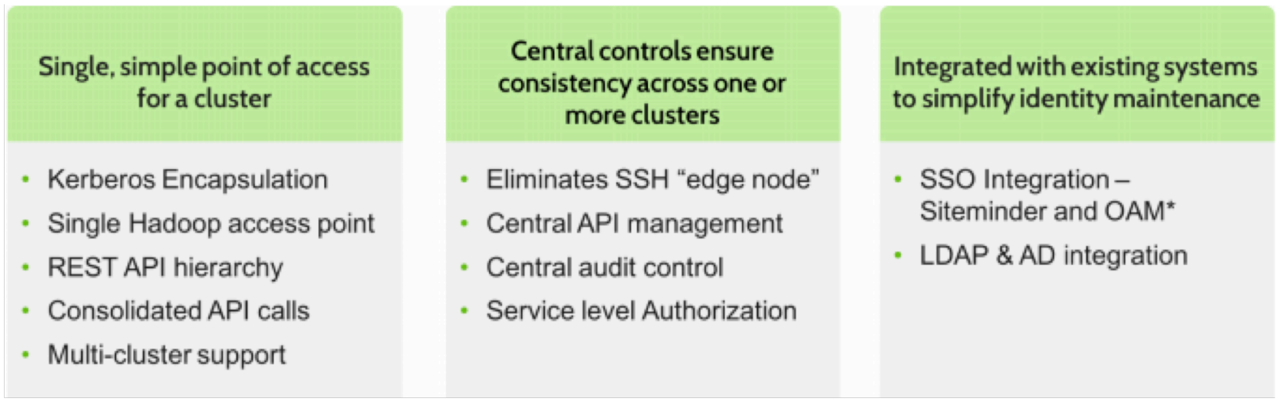
Ranger enhances the productivity of security administrators and reduces potential errors by empowering them to define security policy once and apply it to all the applicable components across the Hadoop stack from a central location.

Authentication and Perimeter Security

Authentication with Kerberos

Using strong authentication to establish user identity is the basis for secure access in Hadoop. Once a user is identified that identity is propagated throughout the Hadoop cluster and used to access resources (i.e.: files and directories) and to perform tasks such as running jobs. Hortonworks uses Kerberos, an industry standard, to authenticate users and resources within a Hadoop cluster. Hortonworks has also simplified Kerberos setup, configuration and maintenance through Ambari 2.0.

Perimeter Security with Knox



Single, simple point of access for a cluster	Central controls ensure consistency across one or more clusters	Integrated with existing systems to simplify identity maintenance
• Kerberos Encapsulation • Single Hadoop access point • REST API hierarchy • Consolidated API calls • Multi-cluster support	• Eliminates SSH "edge node" • Central API management • Central audit control • Service level Authorization	• SSO Integration – Siteminder and OAM* • LDAP & AD integration

Perimeter Security with Knox

With Knox, enterprises can confidently extend the Hadoop REST API to new users without Kerberos complexities, while also maintaining compliance with enterprise security policies. Knox provides a central gateway for Hadoop REST APIs that have varying degrees of authorization, authentication, SSL and SSO capabilities to enable a single access point for Hadoop

Authorization

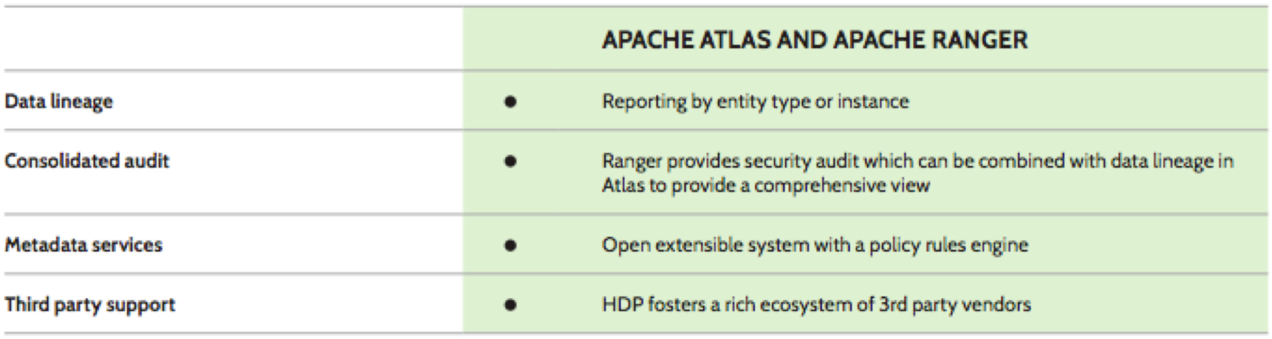
Ranger defines centralized security policies for the following components:

- Apache Hadoop HDFS
- Apache Hadoop YARN
- Apache Hive
- Apache Hbase
- Apache Storm
- Apache Knox
- Apache Solr
- Apache Kafka

Ranger manages fine-grained access control through a rich user interface that ensures consistent policy administration across Hadoop data access components. Security administrators have the flexibility to define security policies for a database, table and column or a file, and administer permissions for specific LDAP based groups or individual users. Rules based on dynamic conditions such as time or geography can also be added to an existing policy rule. The Ranger authorization model is highly pluggable and can be easily extended to any data source using a service-based definition.

Ranger works with standard authorization APIs in each Hadoop component and is able to enforce centrally administered policies for any method of accessing the data lake.

Audit



		APACHE ATLAS AND APACHE RANGER
Data lineage	●	Reporting by entity type or instance
Consolidated audit	●	Ranger provides security audit which can be combined with data lineage in Atlas to provide a comprehensive view
Metadata services	●	Open extensible system with a policy rules engine
Third party support	●	HDP fosters a rich ecosystem of 3rd party vendors

Apache Atlas and Apache Ranger

The new open source project: Apache Atlas, is a set of core foundational governance services that enable enterprises to meet their compliance requirements within Hadoop and allow integration with the complete enterprise data ecosystem. These services include:

- Search and lineage for datasets
- Metadata-driven data access control
- Indexed and searchable centralized auditing operational events
- Data lifecycle management from ingestion to disposition
- Metadata interchange with other tools

Ranger provides a centralized framework for collecting access audit history and easily reporting on this data, including the ability to filter data based on various parameters. Working together, Ranger and Atlas make it possible to gain a comprehensive view of data lineage and access audit, with an ability to query and filter audit based on data classification, users or groups, and other filters.

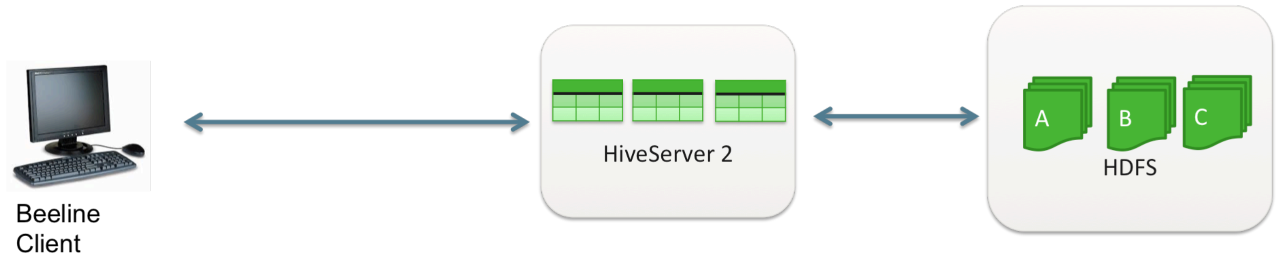
Data Protection

Protection for:

- **Data in motion**
  - Encrypts network traffic
  - Data is unreadable over the network
  - Over RPC, HTTP, Data Transfer Protocol (DTP) and JDBC
- **Data at rest**
  - Encrypts files stored in Hadoop
  - Includes an open source Hadoop key management store (KMS)

Hortonworks is working with its encryption partners to integrate HDFS encryption with enterprise grade key management frameworks. Encryption in HDFS, combined with KMS access policies maintained by Ranger, prevents rogue Linux or Hadoop administrators from accessing data.

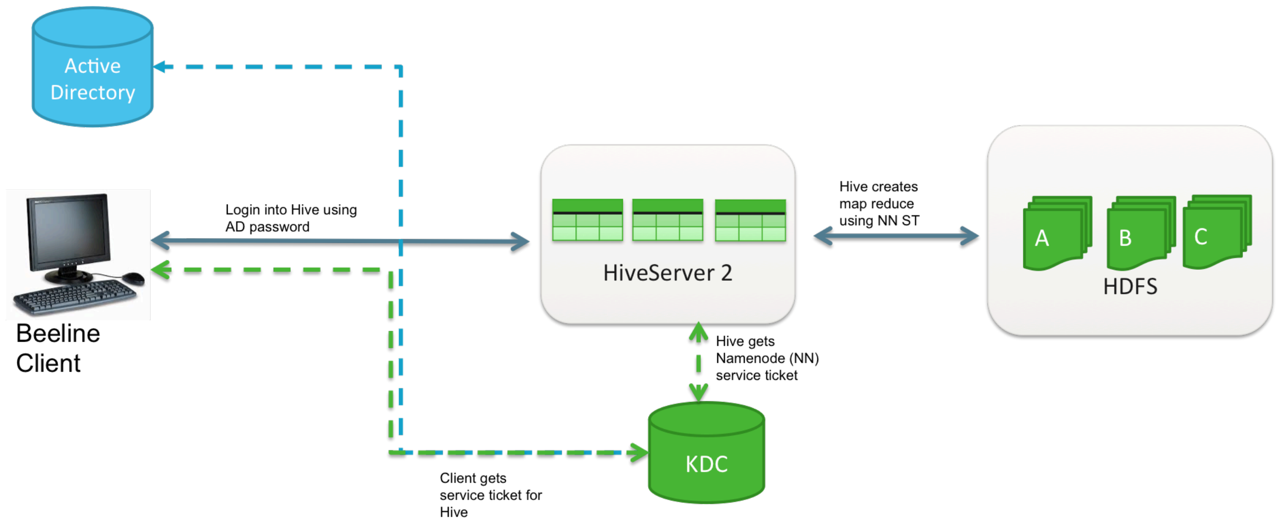
A Typical Data Flow



A Typical Data Flow

To illustrate security for a typical data flow, we will focus on a common use case: Hive access via an xDBC connection to HiveServer 2.

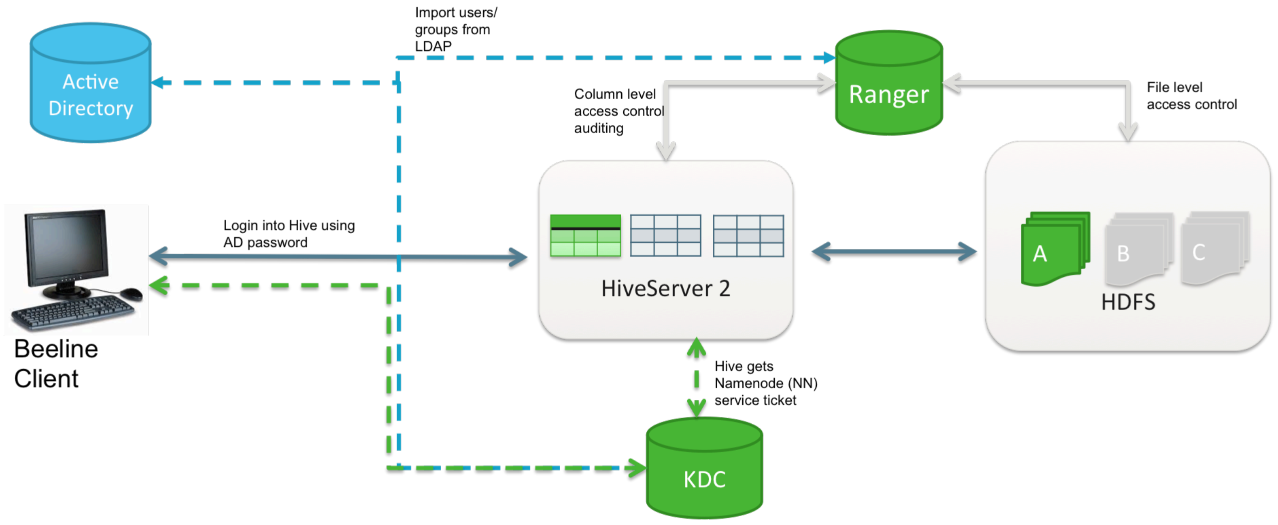
1) The first layer is AUTHENTICATION, provided through Kerberos.


Active Directory
Login into Hive using AD password
Beeline Client
HiveServer 2
Hive creates map reduce using NN ST
A
B
C
HDFS
Hive gets Namenode (NN) service ticket
KDC
Client gets service ticket for Hive

Kerberos establishes a **Service Ticket** (ST) that the client provides when connecting to HiveServer 2.

This strongly associates the client request with a specific user ID.

2) Next, we introduce centralized AUTHORIZATION with Ranger.

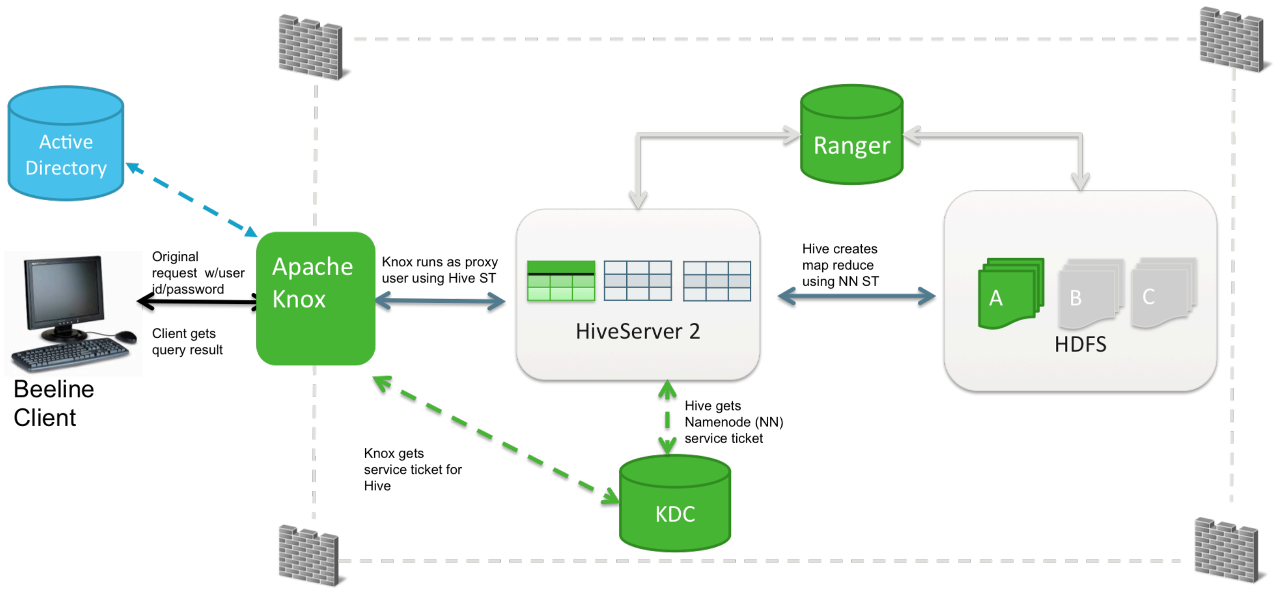

Import users/ groups from LDAP
Active Directory
Ranger
Column level access control auditing
File level access control
Login into Hive using AD password
HiveServer 2
A
B
C
HDFS
Beeline Client
Hive gets Namenode (NN) service ticket
KDC

Once the user ID is established, each request must be checked against what this user is allowed to do. This Authorization is performed by Apache Ranger.

Ranger synchronizes with the user/group repository (Microsoft Windows Active Directory is shown in the above example), which contains detailed policies by data type (hive table, hdfs files, etc.).

HiveServer 2 then passes the user ID and request to Ranger's policy server to determine if the request Authorization can be honored.

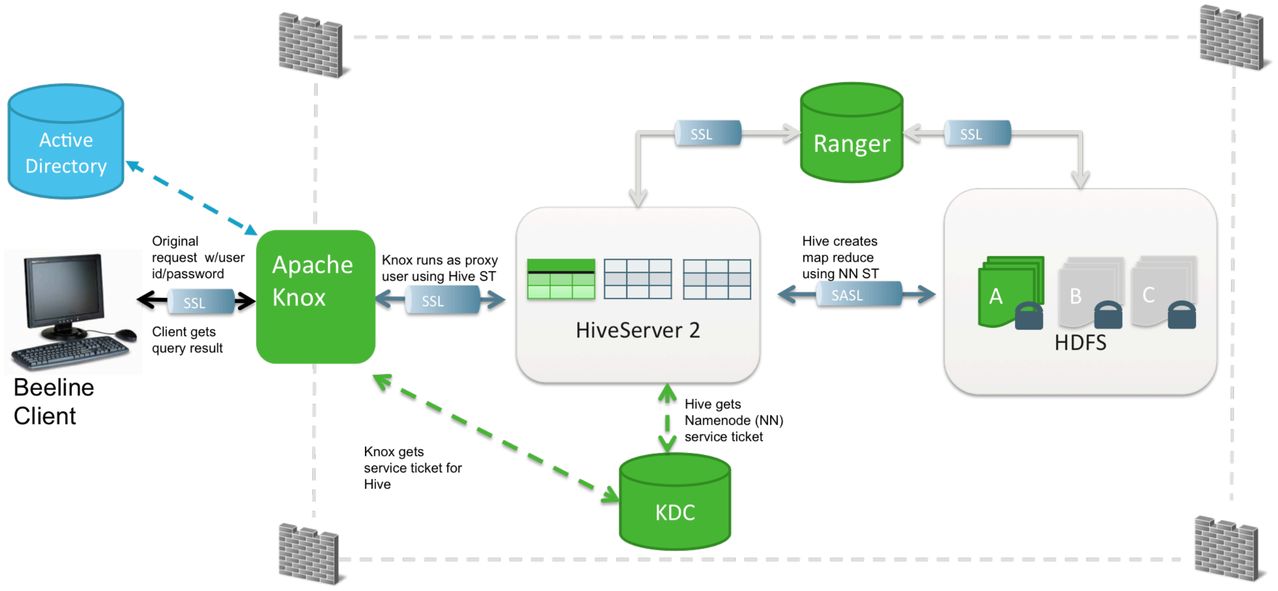
3) The next layer is a strong perimeter technology, Knox, which provides a gateway that all clients to pass through to get to the cluster.


Active Directory
Ranger
Original request w/user id/password
Client gets query result
Apache Knox
Knox runs as proxy user using Hive ST
HiveServer 2
Hive creates map reduce using NN ST
A
B
C
HDFS
Beeline Client
Hive gets Namenode (NN) service ticket
Knox gets service ticket for Hive
KDC

This creates a centralized AUTHENTICATION point.

To simplify client access, all clients connect to Apache Knox, which then contacts Kerberos for authentication. Individual clients do not need to worry about interfacing to Kerberos, as Knox does this for them. This also provides a central audit point. Clients (such as this HiveServer 2 example) connect to Knox using a standard userid/password combination and let Knox create the necessary Kerberos ST based on those credentials.

4) Finally, we can introduce encryption for data in-motion (in the network) and/or at-rest (on disk).


Active Directory
Original request w/user id/password
SSL
Client gets query result
Beeline Client
Apache Knox
Knox runs as proxy user using Hive ST
SSL
SSL
Ranger
SSL
HiveServer 2
Hive creates map reduce using NN ST
SASL
A
B
C
HDFS
Hive gets Namenode (NN) service ticket
Knox gets service ticket for Hive
KDC

Hadoop's included encryption at-rest is called "Transparent Encryption"

REFERENCE:

"Solving Hadoop Security – A Holistic Approach to a Secure Data Lake"

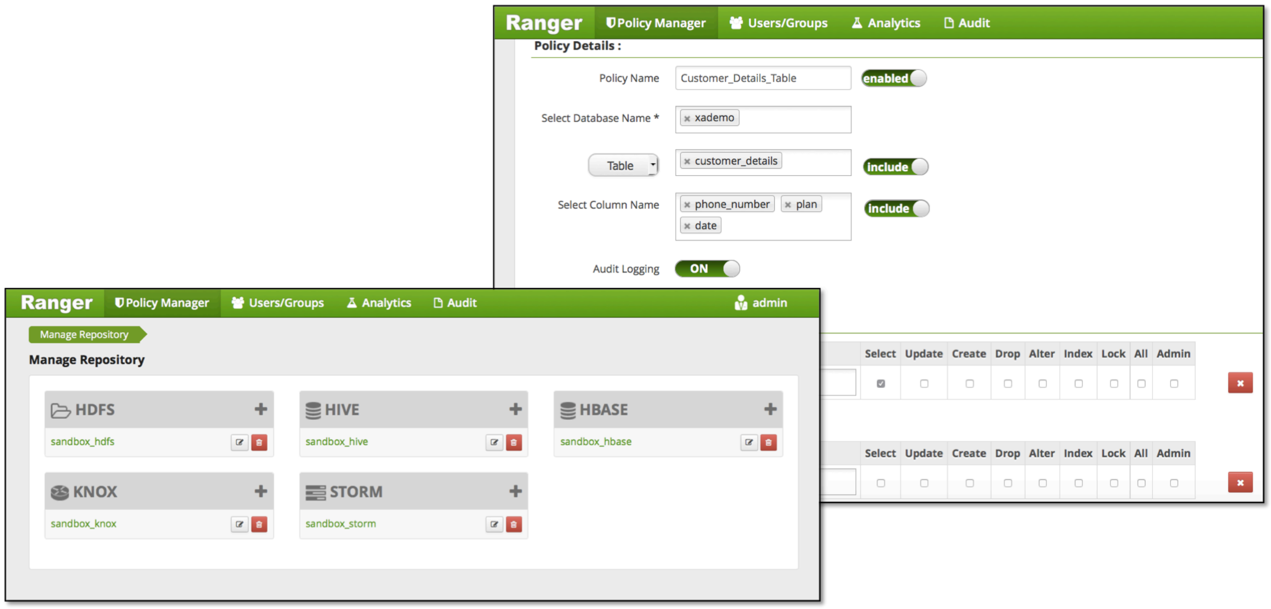
http://hortonworks.com/wp-content/uploads/2015/07/Security_White_Paper.pdf

Apache Ranger

Apache Ranger:

- Delivers a 'single pane of glass' for the security administrator
- Centralizes administration of security policy
- Ensures consistent coverage across the entire Hadoop stack

Central Security Administration



Policy Details :

Policy Name	Customer_Details_Table	enabled
Select Database Name *	xademo	
Table	customer_details	include
Select Column Name	phone_number, plan, date	include
Audit Logging	ON	

Select	Update	Create	Drop	Alter	Index	Lock	All	Admin
✓								

Manage Repository

HDFS	sandbox_hdfs
HIVE	sandbox_hive
HBASE	sandbox_hbase
KNOX	sandbox_knox
STORM	sandbox_storm

Apache Ranger Central Security Administration

Ranger provides:

Authorization flexibility

Fine-grained access control for:

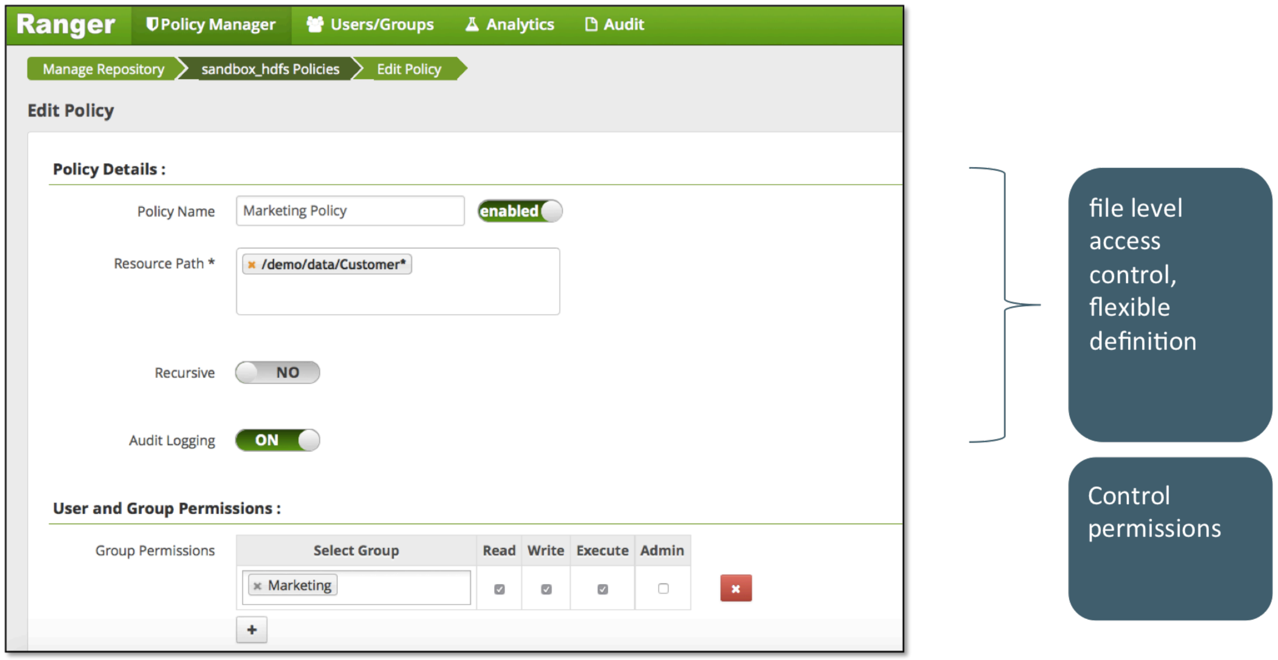
- HDFS – Folder, File
- Hive – Database, Table, Column
- HBase – Table, Column Family, Column
- Storm, Knox and more

Audit control and access

Extensive user access auditing in HDFS, Hive and Hbase for:

- IP Address
- Resource type/ resource
- Timestamp
- Access granted or denied

Setting Up Authorization Policies



Edit Policy

Policy Details :

Policy Name	Marketing Policy (enabled)
Resource Path *	/demo/data/Customer*
Recursive	NO
Audit Logging	ON

User and Group Permissions :

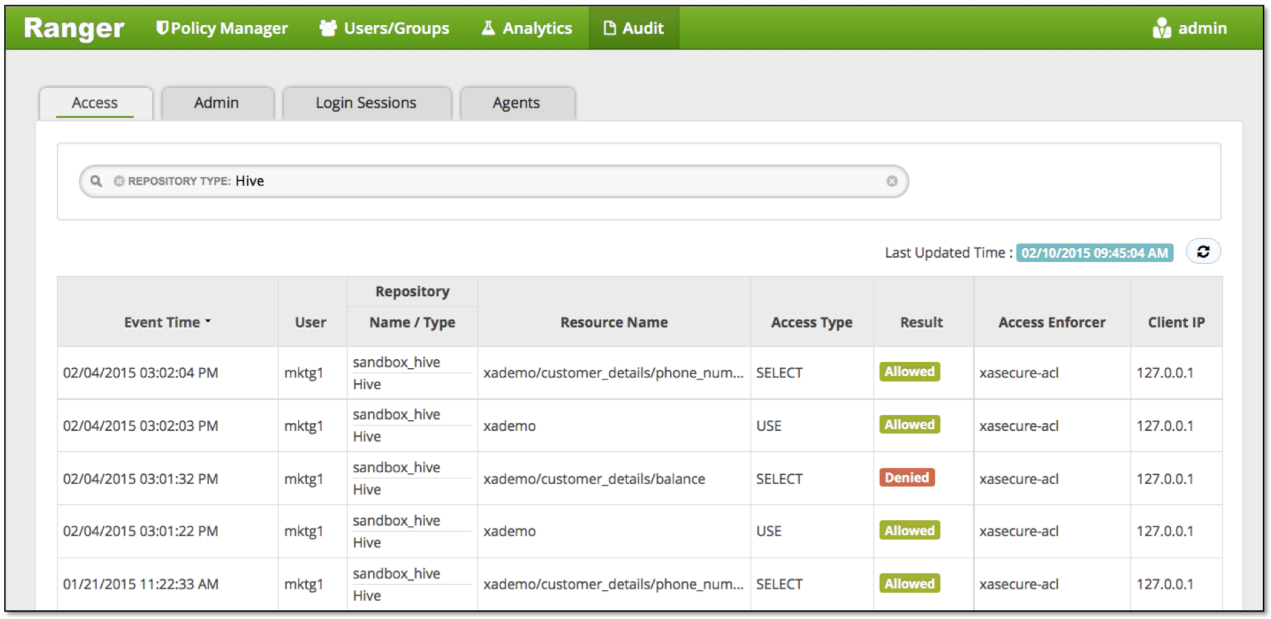
Group Permissions

Select Group	Read	Write	Execute	Admin
Marketing	✓	✓	✓	

Authorization Policies Define Access

Ranger can set up Authorization policies, descriptions of which users may access what data, and how that data can be accessed.

Monitoring Through Auditing



REPOSITORY TYPE: Hive

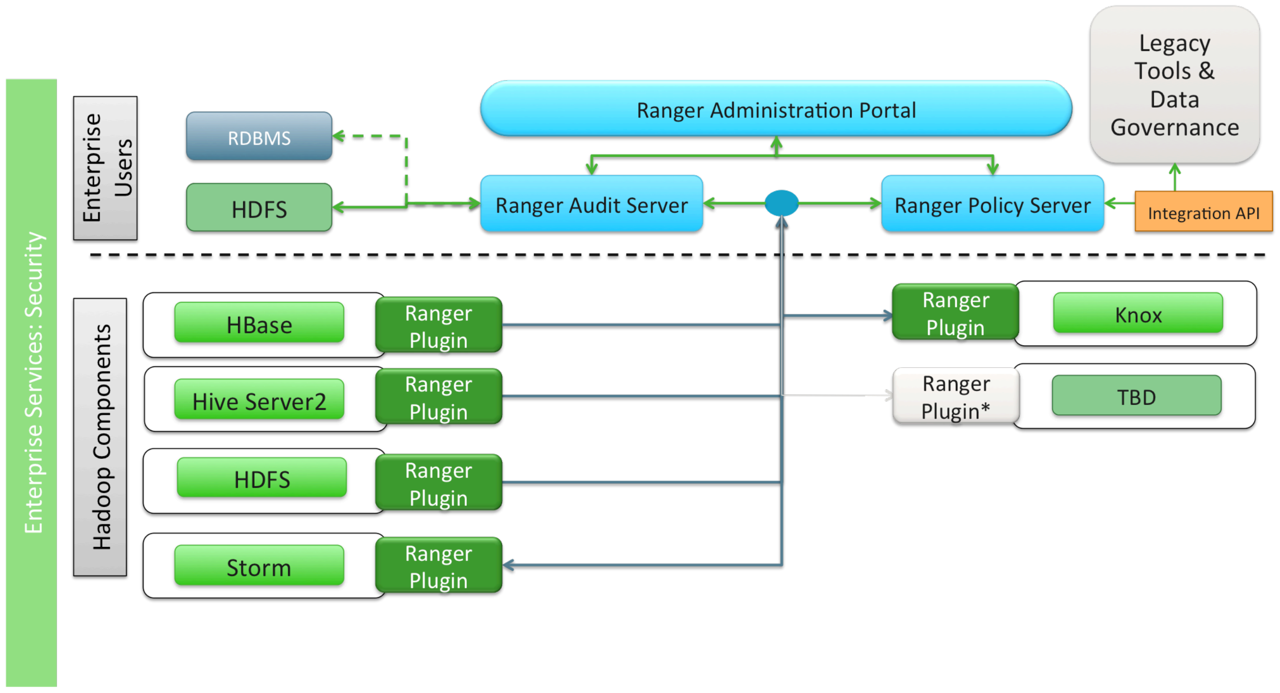
Last Updated Time : 02/10/2015 09:45:04 AM

Event Time	User	Repository Name / Type	Resource Name	Access Type	Result	Access Enforcer	Client IP
02/04/2015 03:02:04 PM	mktg1	sandbox_hive Hive	xademo/customer_details/phone_num...	SELECT	Allowed	xasecure-acl	127.0.0.1
02/04/2015 03:02:03 PM	mktg1	sandbox_hive Hive	xademo	USE	Allowed	xasecure-acl	127.0.0.1
02/04/2015 03:01:32 PM	mktg1	sandbox_hive Hive	xademo/customer_details/balance	SELECT	Denied	xasecure-acl	127.0.0.1
02/04/2015 03:01:22 PM	mktg1	sandbox_hive Hive	xademo	USE	Allowed	xasecure-acl	127.0.0.1
01/21/2015 11:22:33 AM	mktg1	sandbox_hive Hive	xademo/customer_details/phone_num...	SELECT	Allowed	xasecure-acl	127.0.0.1

Auditing Access through Ranger

Ranger auditing shows the requests made to access Hadoop resources, and whether or not access was granted.

Authorization and Auditing with Ranger



Ranger Connects with Hadoop Services and Components

Ranger connects into various Hadoop services and components.

Summary

- HDP ensures comprehensive enforcement of security requirements across the entire Hadoop stack.
- Kerberos is the key to strong authentication.
- Ranger provides a single simple interface for security policy definition and maintenance.
- Encryption options available for data at-rest and in-motion.

Classes Available Worldwide Through Our Partners

Study Options Worldwide

In combination with our partner providers, classes are often available in numerous locations across the world.

Private On-site Training

Hortonworks training in-house covers all of our basic coursework, and provides a more intimate setting for 6 or more students.
Contact us for more details

HORTONWORKS UNIVERSITY

Learn from the company focused solely on Hadoop.

What Makes Us Different?

1. Our courses are designed by the **leaders and committers** of Hadoop
2. We provide an **immersive** experience in **real-world** scenarios
3. We prepare you to **be an expert** with highly valued, **fresh skills**
4. Our courses are available **near you**, or accessible **online**

Hortonworks University courses are designed by the leaders and committers of Apache Hadoop. We provide immersive, real-world experience in scenario-based training. Courses offer unmatched depth and expertise available in both the classroom or online from anywhere in the world. We prepare you to be an expert with highly valued skills and for Certification.