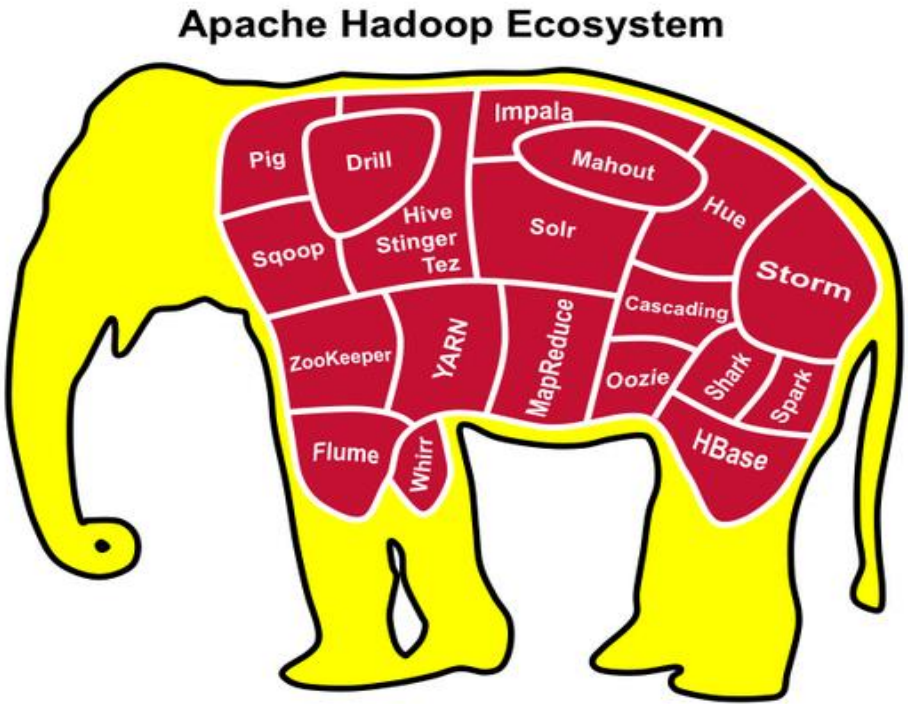


Data Hadoop & Spark Training - ACADGILD

Assignment 3.1

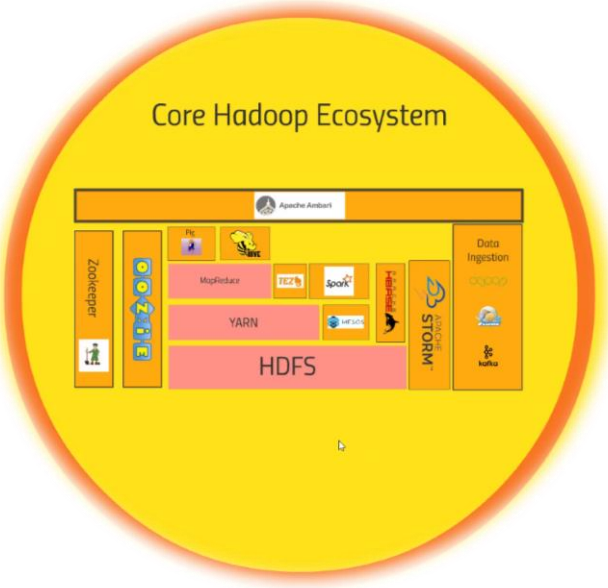
26-Nov-17

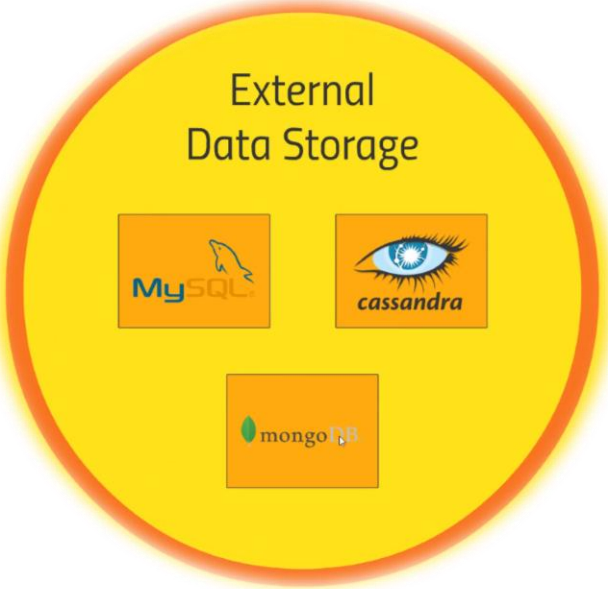
BIG DATA

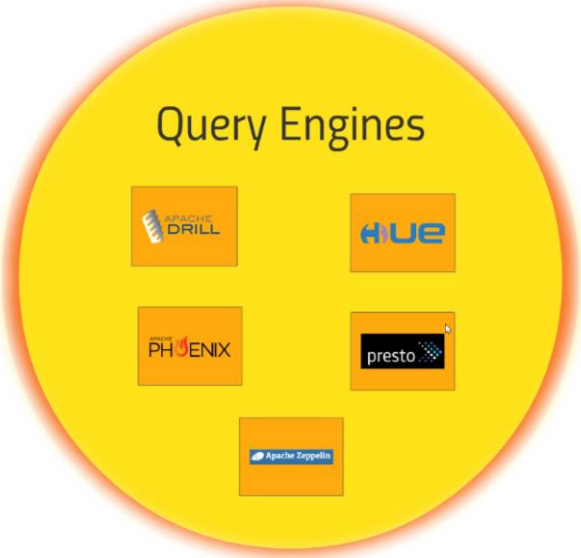
WORLD OF HADOOP

To explain the World of Hadoop I have broadly divided it into 3 catagories

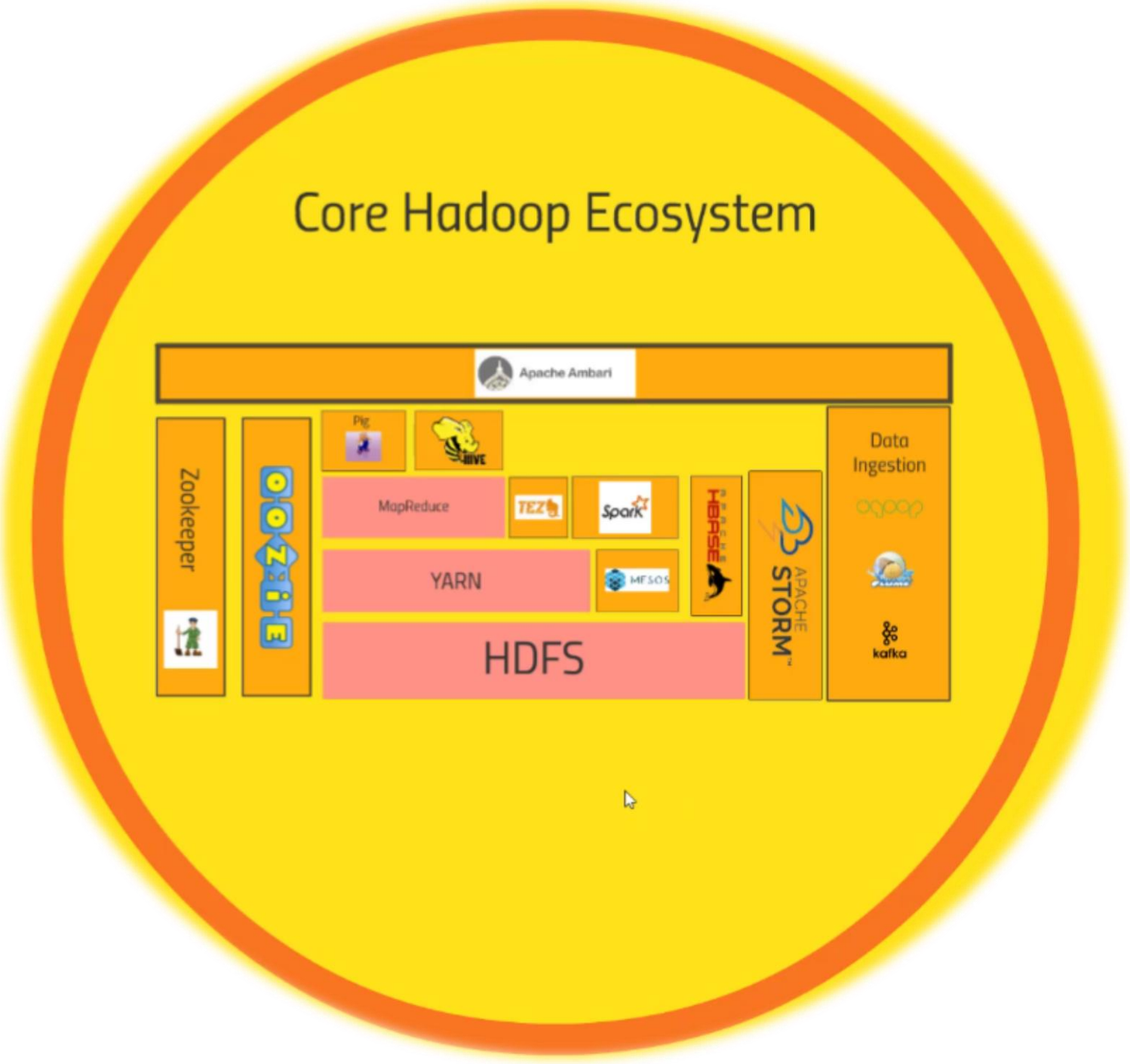
1. Core Hadoop Ecosystem
2. External Data Storage
3. Query Engines


Core Hadoop Ecosystem
Apache Ambari
Pig
Data Ingestion
MapReduce
TEZ
Spark
YARN
HDFS
Zookeeper
OOZIE
STORM
kafka


External Data Storage
MySQL
cassandra
mongoDB


Query Engines
APACHE DRILL
HUE
PHOENIX
presto
Apache Zeppelin

Core Hadoop Ecosystem

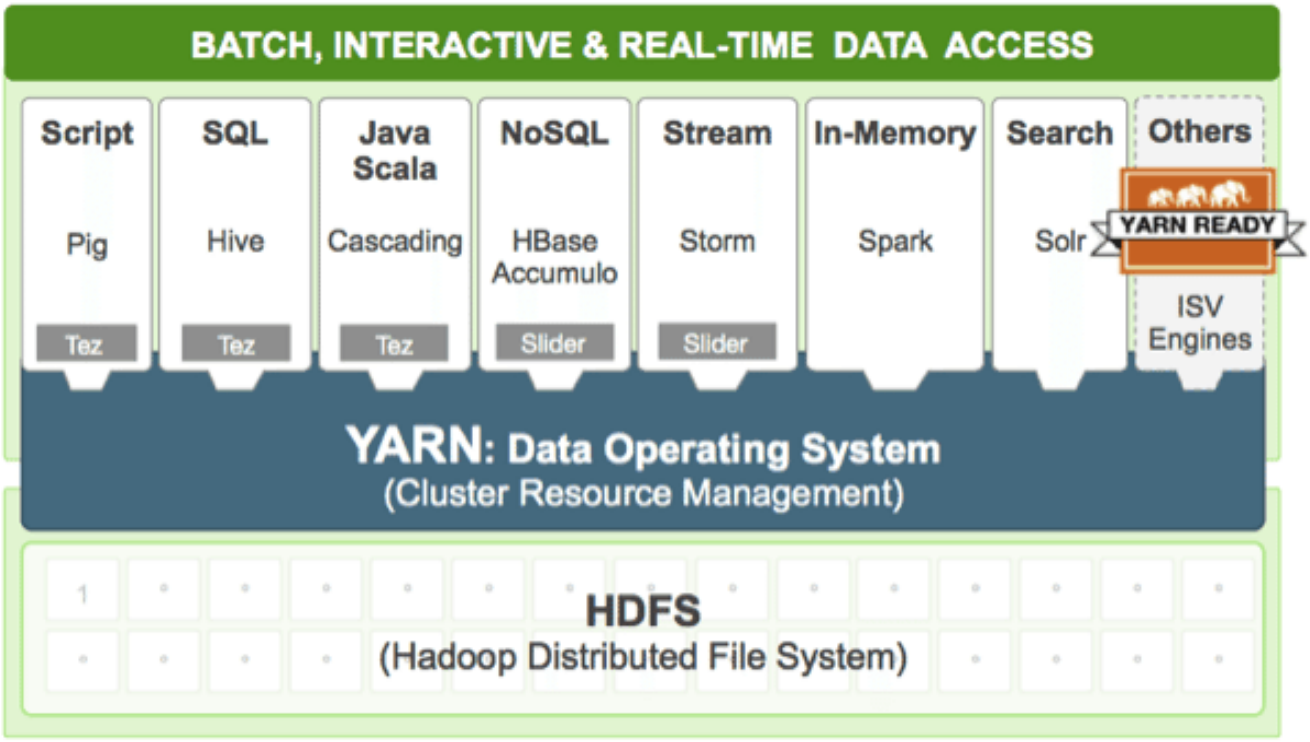


1. Hadoop Distributed File System (HDFS)

The Hadoop Distributed File System (HDFS) is designed to store very large data sets reliably, and to stream those data sets at high bandwidth to user applications. In a large cluster, thousands of servers both host directly attached storage and execute user application tasks. By distributing storage and computation across many servers, the resource can grow with demand while remaining economical at every size. We describe the architecture of HDFS and report on **experience** using HDFS to manage 40 petabytes of enterprise data at Yahoo.

Features:

a. Rack awareness allows consideration of a node's physical location, when allocating storage and scheduling tasks
b. Minimal data motion. MapReduce moves compute processes to the data on HDFS and not the other way around. Processing tasks can occur on the physical node where the data resides. This significantly reduces the network I/O patterns and keeps most of the I/O on the local disk or within the same rack and provides very high aggregate read/write bandwidth.
c. Utilities diagnose the health of the files system and can rebalance the data on different nodes
d. Rollback allows system operators to bring back the previous version of HDFS after an upgrade, in case of human or system errors
e. Standby NameNode provides redundancy and supports high availability
f. Highly operable. Hadoop handles different types of cluster that might otherwise require operator intervention. This design allows a single operator to maintain a cluster of 1000s of nodes.



2. Yet Another Resource Negotiator (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.

Features:

a. Multi-tenancy - YARN allows multiple access engines (either open-source or proprietary) to use Hadoop as the common standard for batch, interactive and real-time engines that can simultaneously access the same data set.
b. Multi-tenant data processing improves an enterprise's return on its Hadoop investments.
c. Cluster utilization - YARN's dynamic allocation of cluster resources improves utilization over more static MapReduce rules used in early versions of Hadoop
d. Scalability - Data center processing power continues to rapidly expand. YARN's ResourceManager focuses exclusively on scheduling and keeps pace as clusters expand to thousands of nodes managing petabytes of data.
e. Compatibility - Existing MapReduce applications developed for Hadoop 1 can run YARN without any disruption to existing processes that already work

3. MapReduce

Hadoop MapReduce is a software framework for easily writing applications which process vast amounts of data (multi-terabyte data-sets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner. The MapReduce framework consists of a single master JobTracker and one slave TaskTracker per cluster-node. The master is responsible for scheduling the jobs' component tasks on the slaves, monitoring them and re-executing the failed tasks. The slaves execute the tasks as directed by the master.

Features:

a. Scale-Out Architecture – Add servers to increase capacity
b. High Availability – Serve mission-critical workflows and applications
c. Fault Tolerance – Automatically and seamlessly recover from failures
d. Flexible Access – Multiple and open frameworks for serialization and file system mounts
e. Load Balancing – Place data intelligently for maximum efficiency and utilization
f. Tunable Replication – Multiple copies of each file provide data protection and computational performance
g. Security – POSIX-based file permissions for users and groups with optional LDAP integration

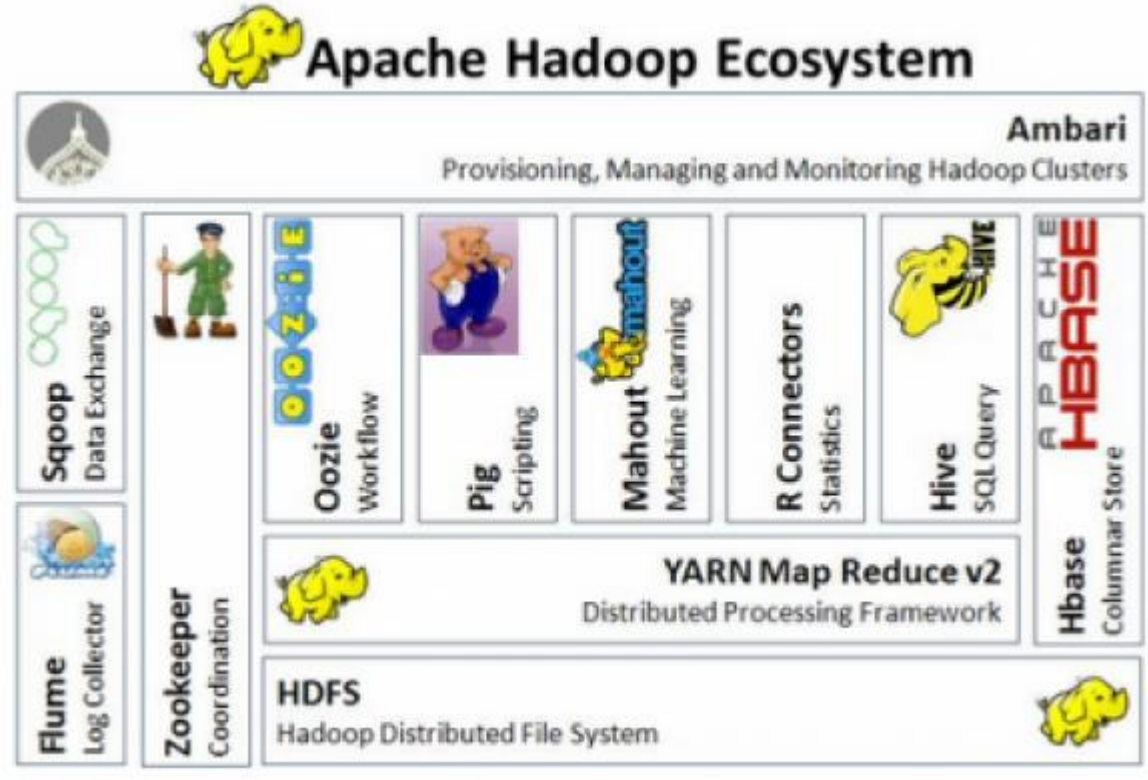
Detail of Hadoop Framework

The Apache Hadoop framework is composed of the following modules

- Hadoop Common: contains libraries and utilities needed by other Hadoop modules
- Hadoop Distributed File System (HDFS): a distributed file-system that stores data on the commodity machines, providing very high aggregate bandwidth across the cluster
- Hadoop YARN: a resource-management platform responsible for managing compute resources in clusters and using them for scheduling of users' applications
- Hadoop MapReduce: a programming model for large scale data processing

All the modules in Hadoop are designed with a fundamental assumption that hardware failures (of individual machines, or racks of machines) are common and thus should be automatically handled in software by the framework. Apache Hadoop's MapReduce and HDFS components originally derived respectively from Google's MapReduce and Google File System (GFS) papers.

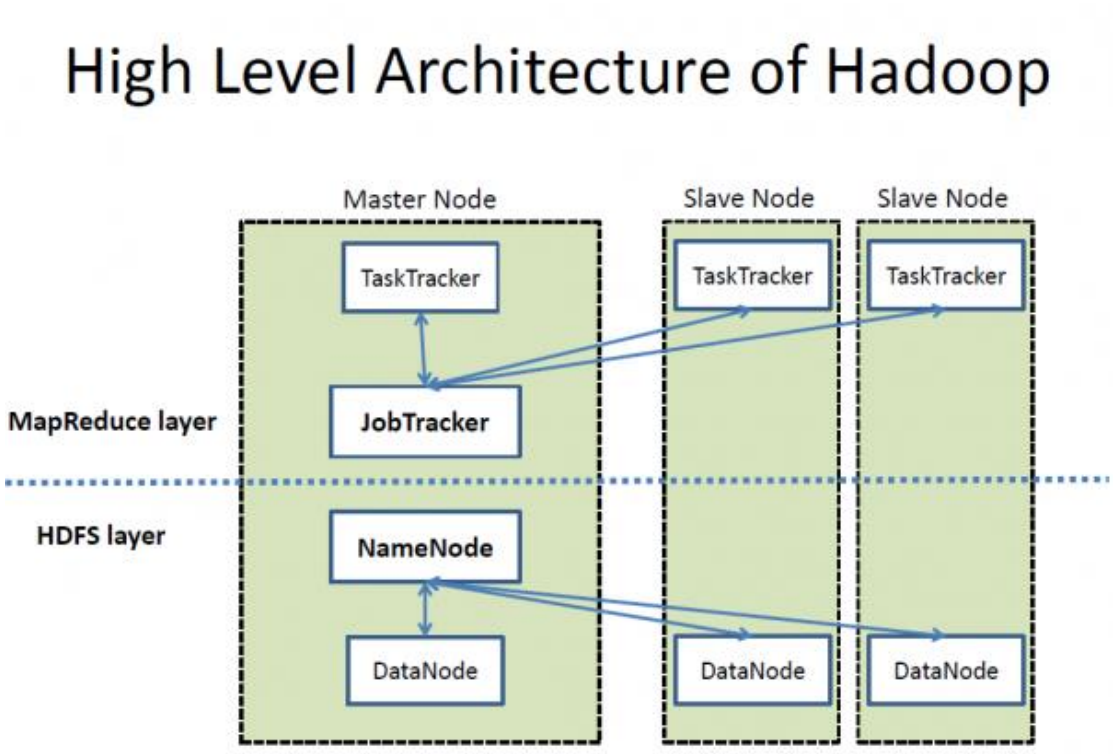
Beyond HDFS, YARN and MapReduce, the entire Apache Hadoop "platform" is now commonly considered to consist of a number of related projects as well: Apache Pig, Apache Hive, Apache HBase, and others.



For the end-users, though MapReduce Java code is common, any programming language can be used with "Hadoop Streaming" to implement the "map" and "reduce" parts of the user's program. Apache Pig and Apache Hive, among other related projects, expose higher level user interfaces like Pig latin and a SQL variant respectively. The Hadoop framework itself is mostly written in the Java programming language, with some native code in C and command line utilities written as shell-scripts.

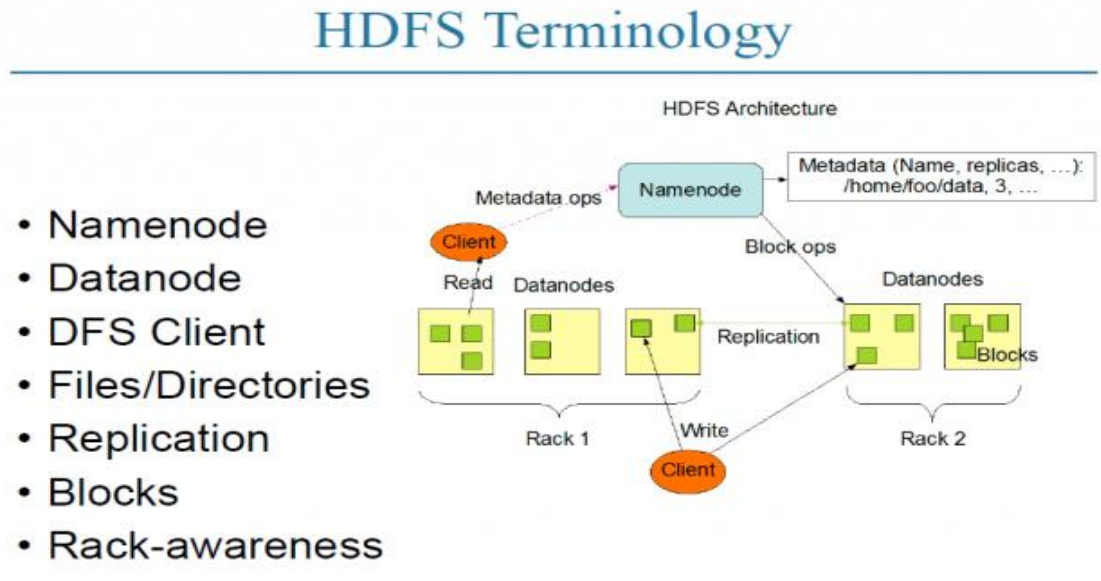
HDFS and MapReduce

There are two primary components at the core of Apache Hadoop 1.x: the Hadoop Distributed File System (HDFS) and the MapReduce parallel processing framework. These are both open source projects, inspired by technologies created inside Google.



Hadoop distributed file system

The Hadoop distributed file system (HDFS) is a distributed, scalable, and portable file-system written in Java for the Hadoop framework. Each node in a Hadoop instance typically has a single namenode, and a cluster of datanodes form the HDFS cluster. The situation is typical because each node does not require a datanode to be present. Each datanode serves up blocks of data over the network using a block protocol specific to HDFS. The file system uses the TCP/IP layer for communication. Clients use Remote procedure call (RPC) to communicate between each other.



HDFS stores large files (typically in the range of gigabytes to terabytes) across multiple machines. It achieves reliability by replicating the data across multiple hosts, and hence does not require RAID storage on hosts. With the default replication value, 3, data is stored on three nodes: two on the same rack, and one on a different rack. Data nodes can talk to each other to rebalance data, to move copies around, and to keep the replication of data high. HDFS is not fully POSIX-compliant, because the requirements for a POSIX file-system differ from the target goals for a Hadoop application. The tradeoff of not having a fully POSIX-compliant file-system is increased performance for data throughput and support for non-POSIX operations such as Append.

HDFS added the high-availability capabilities for release 2.x, allowing the main metadata server (the NameNode) to be failed over manually to a backup in the event of failure, automatic fail-over.

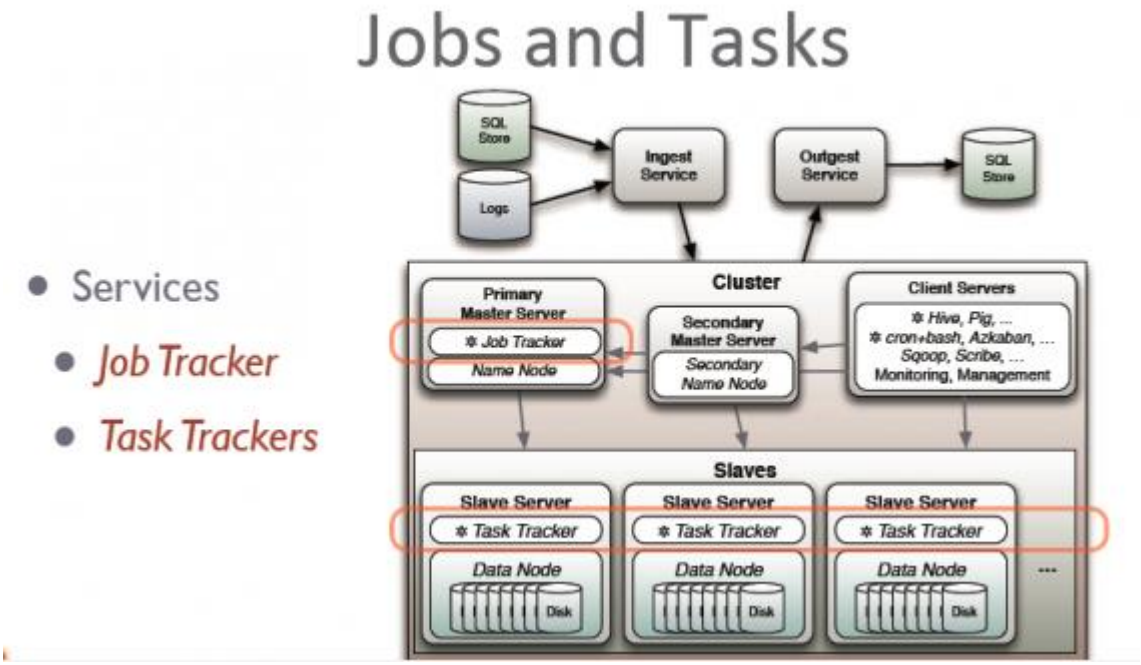
The HDFS file system includes a so-called secondary namenode, which misleads some people into thinking that when the primary namenode goes offline, the secondary namenode takes over. In fact, the secondary namenode regularly connects with the primary namenode and builds snapshots of the primary namenode's directory information, which the system then saves to local or remote directories. These checkpointed images can be used to restart a failed primary namenode without having to replay the entire journal of file-system actions, then to edit the log to create an up-to-date directory structure. Because the namenode is the single point for storage and management of metadata, it can become a bottleneck for supporting a huge number of files, especially a large number of small files. HDFS Federation, a new addition, aims to tackle this problem to a certain extent by allowing multiple name-spaces served by separate namenodes.

An advantage of using HDFS is data awareness between the job tracker and task tracker. The job tracker schedules map or reduce jobs to task trackers with an awareness of the data location. For example, if node A contains data (x, y, z) and node B contains data (a, b, c), the job tracker schedules node B to perform map or reduce tasks on (a,b,c) and node A would be scheduled to perform map or reduce tasks on (x,y,z). This reduces the amount of traffic that goes over the network and prevents unnecessary data transfer. When Hadoop is used with other file systems, this advantage is not always available. This can have a significant impact on job-completion times, which has been demonstrated when running data-intensive jobs. HDFS was designed for mostly immutable files and may not be suitable for systems requiring concurrent write-operations.

Another limitation of HDFS is that it cannot be mounted directly by an existing operating system. Getting data into and out of the HDFS file system, an action that often needs to be performed before and after executing a job, can be inconvenient. A filesystem in Userspace (FUSE) virtual file system has been developed to address this problem, at least for Linux and some other Unix systems.

File access can be achieved through the native Java API, the Thrift API, to generate a client in the language of the users' choosing (C++, Java, Python, PHP, Ruby, Erlang, Perl, Haskell, C#, Cocoa, Smalltalk, or OCaml), the command-line interface, or browsed through the HDFS-UI web app over HTTP.

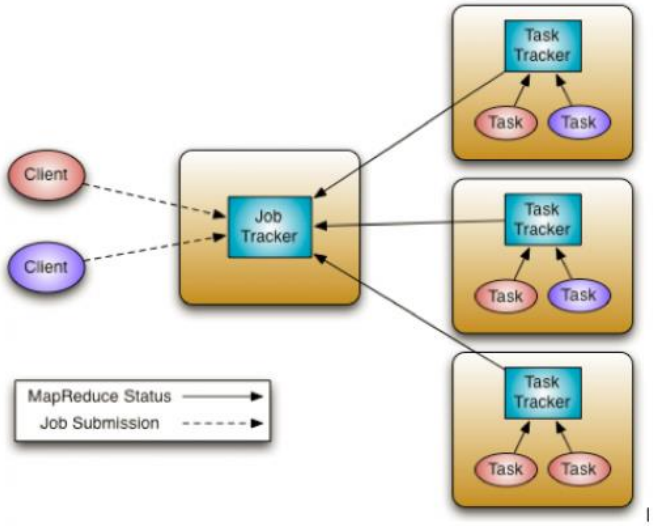
JobTracker and TaskTracker: The MapReduce engine



Above the file systems comes the MapReduce engine, which consists of one JobTracker, to which client applications submit MapReduce jobs. The JobTracker pushes work out to available TaskTracker nodes in the cluster, striving to keep the work as close to the data as possible.

With a rack-aware file system, the JobTracker knows which node contains the data, and which other machines are nearby. If the work cannot be hosted on the actual node where the data resides, priority is given to nodes in the same rack. This reduces network traffic on the main backbone network.

If a TaskTracker fails or times out, that part of the job is rescheduled. The TaskTracker on each node spawns off a separate Java Virtual Machine process to prevent the TaskTracker itself from failing if the running job crashes the JVM. A heartbeat is sent from the TaskTracker to the JobTracker every few minutes to check its status. The Job Tracker and TaskTracker status and information is exposed by Jetty and can be viewed from a web browser.



If the JobTracker failed on Hadoop 0.20 or earlier, all ongoing work was lost. Hadoop version 0.21 added some checkpointing to this process. The JobTracker records what it is up to in the file system. When a JobTracker starts up, it looks for any such data, so that it can restart work from where it left off.

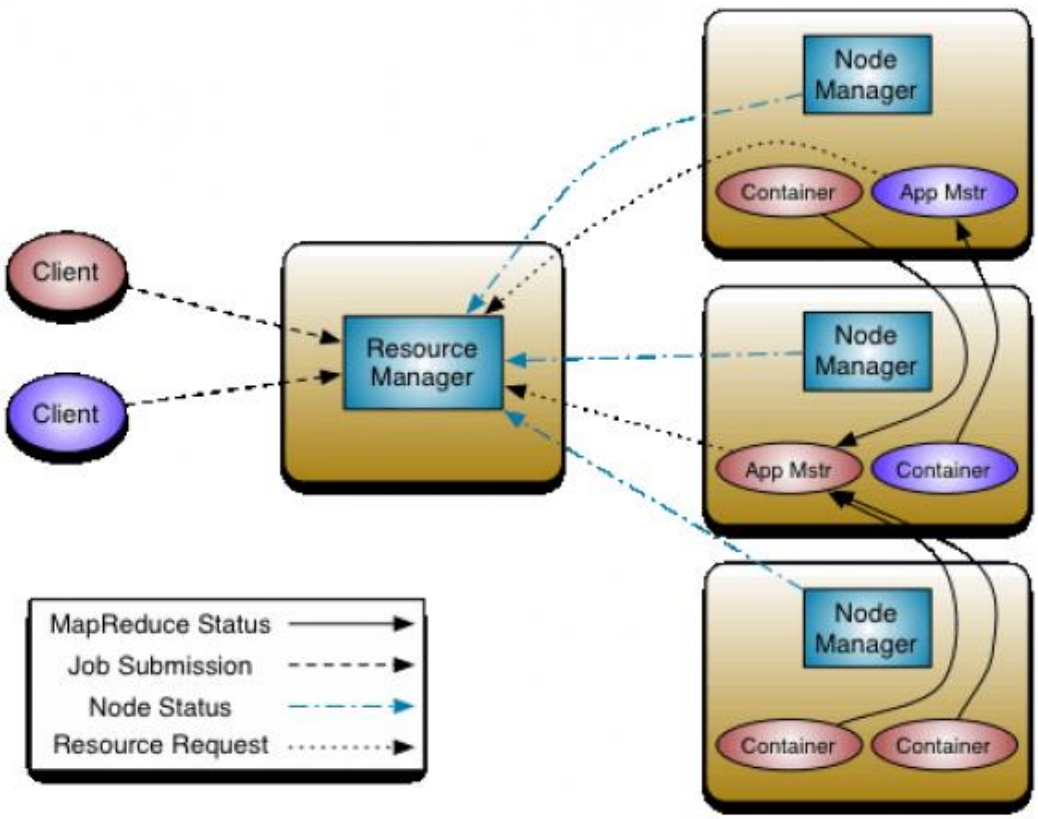
Known limitations of this approach in Hadoop 1.x

The allocation of work to TaskTrackers is very simple. Every TaskTracker has a number of available slots (such as "4 slots"). Every active map or reduce task takes up one slot. The Job Tracker allocates work to the tracker nearest to the data with an available slot. There is no consideration of the current system load of the allocated machine, and hence its actual availability. If one TaskTracker is very slow, it can delay the entire MapReduce job—especially towards the end of a job, where everything can end up waiting for the slowest task. With speculative execution enabled, however, a single task can be executed on multiple slave nodes.

Apache Hadoop NextGen MapReduce (YARN)

MapReduce has undergone a complete overhaul in hadoop-0.23 and we now have, what we call, MapReduce 2.0 (MRv2) or YARN.

Apache™ Hadoop® YARN is a sub-project of Hadoop at the Apache Software Foundation introduced in Hadoop 2.0 that separates the resource management and processing components. YARN was born of a need to enable a broader array of interaction patterns for data stored in HDFS beyond MapReduce. The YARN-based architecture of Hadoop 2.0 provides a more general processing platform that is not constrained to MapReduce.



The fundamental idea of MRv2 is to split up the two major functionalities of the JobTracker, resource management and job scheduling/monitoring, into separate daemons. The idea is to have a global ResourceManager (RM) and per-application ApplicationMaster (AM). An application is either a single job in the classical sense of Map-Reduce jobs or a DAG of jobs.

The ResourceManager and per-node slave, the NodeManager (NM), form the data-computation framework. The ResourceManager is the ultimate authority that arbitrates resources among all the applications in the system.

The per-application ApplicationMaster is, in effect, a framework specific library and is tasked with negotiating resources from the ResourceManager and working with the NodeManager(s) to execute and monitor the tasks.

As part of Hadoop 2.0, YARN takes the resource management capabilities that were in MapReduce and packages them so they can be used by new engines. This also streamlines MapReduce to do what it does best, process data. With YARN, you can now run multiple applications in Hadoop, all sharing a common resource management. Many organizations are already building applications on YARN in order to bring them IN to Hadoop. When enterprise data is made available in HDFS, it is important to have multiple ways to process that data. With Hadoop 2.0 and YARN organizations can use Hadoop for streaming, interactive and a world of other Hadoop based applications.
What YARN does

YARN enhances the power of a Hadoop compute cluster in the following ways:

- Scalability: The processing power in data centers continues to grow quickly. Because YARN ResourceManager focuses exclusively on scheduling, it can manage those larger clusters much more easily.
- Compatibility with MapReduce: Existing MapReduce applications and users can run on top of YARN without disruption to their existing processes.
- Improved cluster utilization: The ResourceManager is a pure scheduler that optimizes cluster utilization according to criteria such as capacity guarantees, fairness, and SLAs. Also, unlike before, there are no named map and reduce slots, which helps to better utilize cluster resources.
- Support for workloads other than MapReduce: Additional programming models such as graph processing and iterative modeling are now possible for data processing. These added models allow enterprises to realize near real-time processing and increased ROI on their Hadoop investments.
- Agility: With MapReduce becoming a user-land library, it can evolve independently of the underlying resource manager layer and in a much more agile manner.

How YARN works

The fundamental idea of YARN is 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 and
- a per-application container running on a NodeManager

The ResourceManager and the NodeManager form the new, and generic, system for managing applications in a distributed manner. The ResourceManager is the ultimate authority that arbitrates resources among all the applications in the system. The per-application ApplicationMaster is a framework-specific entity and is tasked with negotiating resources from the ResourceManager and working with the NodeManager(s) to execute and monitor the component tasks. The ResourceManager has a scheduler, which is responsible for allocating resources to the various running applications, according to constraints such as queue capacities, user-limits etc. The scheduler performs its scheduling function based on the resource requirements of the applications. The NodeManager is the per-machine slave, which is responsible for launching the applications' containers, monitoring their resource usage (cpu, memory, disk, network) and reporting the same to the ResourceManager. Each ApplicationMaster has the responsibility of negotiating appropriate resource containers from the scheduler, tracking their status, and monitoring their progress. From the system perspective, the ApplicationMaster runs as a normal container.

4. Pig

Pig is a platform for analyzing large data sets that consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs. The salient property of Pig programs is that their structure is amenable to substantial parallelization, which in turns enables them to handle very large data sets. At the present time, Pig's infrastructure layer consists of a compiler that produces sequences of Map-Reduce programs, for which large-scale parallel implementations already exist (e.g., the Hadoop subproject). Pig's language layer currently consists of a textual language called Pig Latin

Features:

a. Ease of programming.
b. It is trivial to achieve parallel execution of simple, "embarrassingly parallel" data analysis tasks. Complex tasks comprised of multiple interrelated data transformations are explicitly encoded as data flow sequences, making them easy to write, understand, and maintain.
c. Optimization opportunities.
d. The way in which tasks are encoded permits the system to optimize their execution automatically, allowing the user to focus on semantics rather than efficiency.
e. Extensibility. Users can create their own functions to do special-purpose processing.

5. Hive

The Apache Hive data warehouse software facilitates querying and managing large datasets residing in distributed storage. Hive provides a mechanism to project structure onto this data and query the data using a SQL-like language called HiveQL. At the same time this language also allows traditional map/reduce programmers to plug in their custom mappers and reducers when it is inconvenient or inefficient to express this logic in HiveQL. Support for exporting metrics via the Hadoop metrics subsystem to files or Ganglia; or via JMX.

Features:

a. Indexing to provide acceleration, index type including compaction and Bitmap index as of 0.10, more index types are planned.
b. Different storage types such as plain text, RCFile, HBase, ORC, and others.
c. Metadata storage in an RDBMS, significantly reducing the time to perform semantic checks during query execution.
d. Operating on compressed data stored into Hadoop ecosystem, algorithm including gzip, bzip2, snappy, etc.
e. Built-in user defined functions (UDFs) to manipulate dates, strings, and other data-mining tools. Hive supports extending the UDF set to handle use-cases not supported by built-in functions.
f. SQL-like queries (Hive QL), which are implicitly converted into map-reduce jobs.

6. HBase

HBase is a column-oriented database management system that runs on top of HDFS. It is well suited for sparse data sets, which are common in many big data use cases. Unlike relational database systems, HBase does not support a structured query language like SQL; in fact, HBase isn't a relational data store at all. HBase applications are written in Java much like a typical MapReduce application. HBase does support writing applications in Avro, REST, and Thrift.

Features:

a. Linear and modular scalability.
b. Strictly consistent reads and writes.
c. Automatic and configurable sharding of tables
d. Automatic failover support between RegionServers.
e. Convenient base classes for backing Hadoop MapReduce jobs with Apache HBase tables.
f. Easy to use Java API for client access.
g. Block cache and Bloom Filters for real-time queries.
h. Query predicate push down via server side Filters

7. Spark

Apache Spark is an open-source data analytics cluster computing framework originally developed in the AMPLab at UC Berkeley. Spark fits into the Hadoop open-source community, building on top of the Hadoop Distributed File System (HDFS). However, Spark is not tied to the two-stage MapReduce paradigm, and promises performance up to 100 times faster than Hadoop MapReduce for certain applications. Spark provides primitives for in-memory cluster computing that allows user programs to load data into a cluster's memory and query it repeatedly, making it well suited to machine learning algorithms

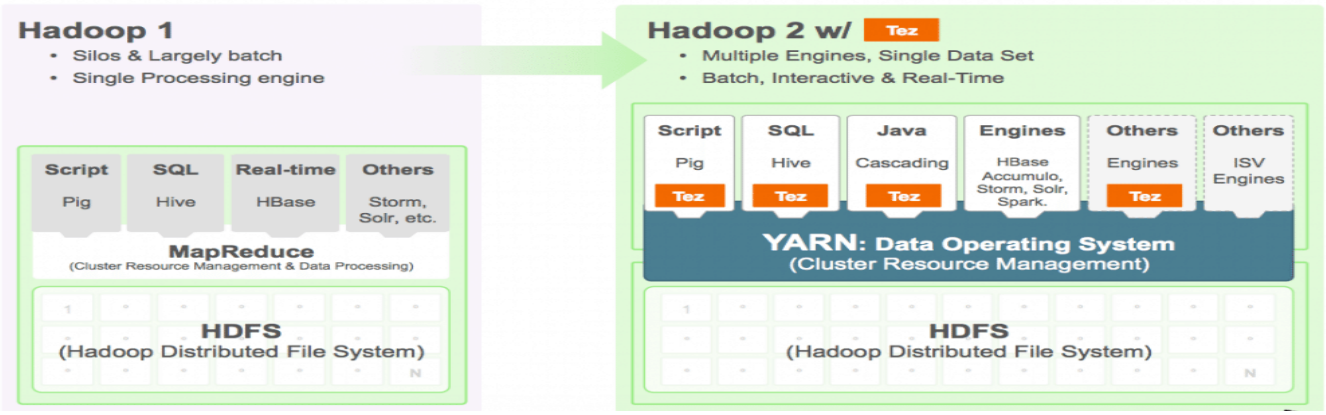
Features:

a. Proven scalability to 100 nodes in the research lab and 80 nodes in production at Yahoo
b. Ability to cache datasets in memory for interactive data analysis: extract a working set, cache it, and query it repeatedly.
c. Interactive command line interface (in Scala or Python) for low-latency data exploration at scale.
d. Higher level library for stream processing, through Spark Streaming.
e. Higher level libraries for machine learning and graph processing that because of the distributed memory-based Spark architecture are ten times as fast as Hadoop disk-based Apache Mahout and even scale better than Vow pal Wabbit

8. Tez

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.

Since Tez is extensible and embeddable, it provides the fit-to-purpose freedom to express highly optimized data processing applications, giving them an advantage over end-user-facing engines such as MapReduce and Apache Spark. Tez also offers a customizable execution architecture that allows users to express complex computations as dataflow graphs, permitting dynamic performance optimizations based on real information about the data and the resources required to process it.

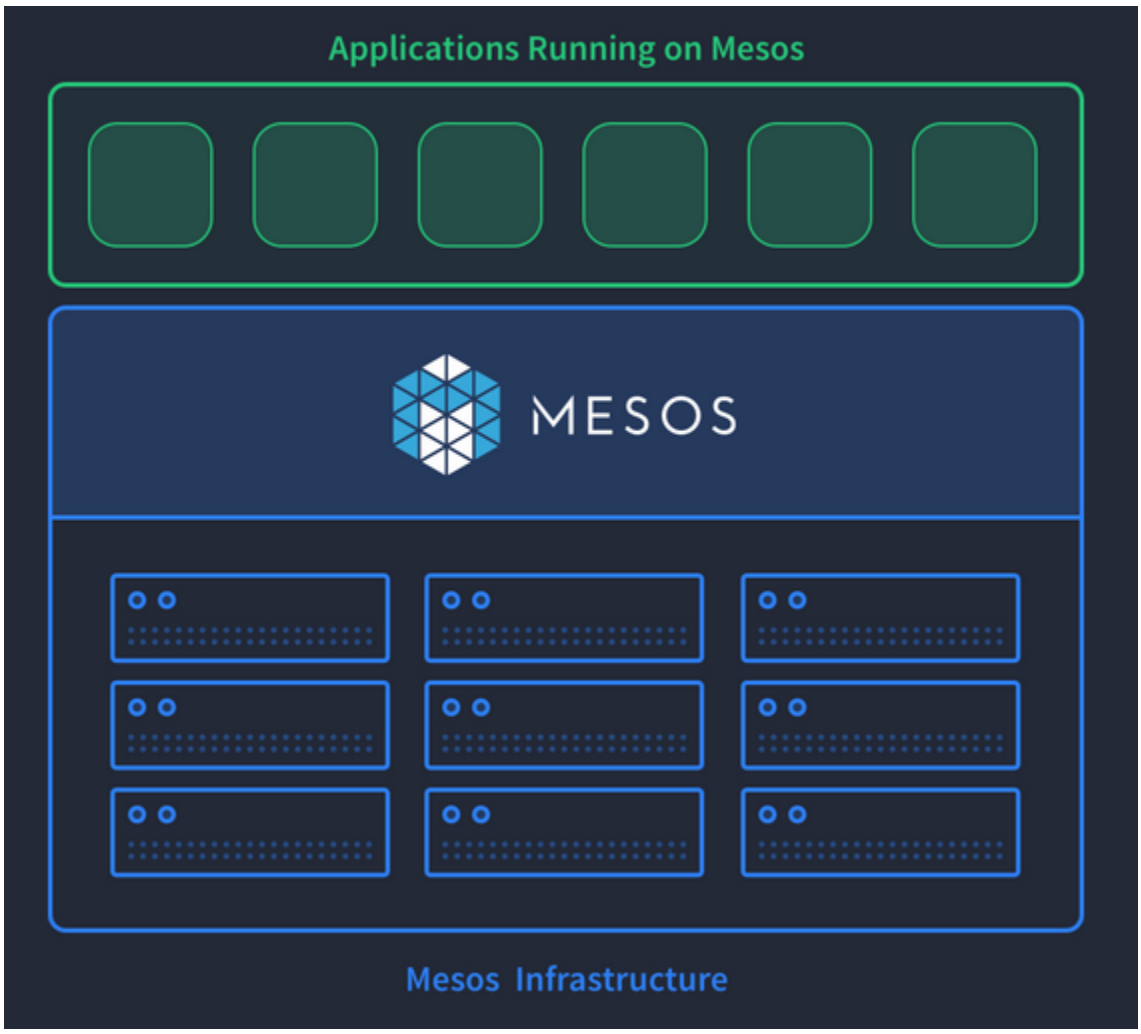


9. Mesos

Apache Mesos is the open-source distributed systems kernel at the heart of the Mesosphere DC/OS. It abstracts the entire datacenter into a single pool of computing resources, simplifying running distributed systems at scale.
A key design criteria of Apache Mesos is its two-level scheduler architecture, making it easier to operate, scale and extend.

Traditional monolithic schedulers maintain the complete state of the application and infrastructure underneath, while also performing workload placement logic. This architecture makes it very challenging to scale and even harder to introduce new features and capabilities.

With a dual-level architecture, Mesos handles low level infrastructure scheduling operations, while another layer on top (The framework) handles all the application specific operations and logic. This architecture has multiple benefits:



10. Oozie

Apache Oozie is a Java Web application used to schedule Apache Hadoop jobs. Oozie combines multiple jobs sequentially into one logical unit of work. It is integrated with the Hadoop stack and supports Hadoop jobs for Apache MapReduce, Apache Pig, Apache Hive, and Apache Sqoop. It can also be used to schedule jobs specific to a system, like Java programs or shell scripts

There are two basic types of Oozie jobs:

Oozie Workflow jobs are Directed Acyclical Graphs (DAGs), specifying a sequence of actions to execute. The Workflow job has to wait. Oozie Coordinator jobs are recurrent Oozie Workflow jobs that are triggered by time and data availability. Oozie Bundle provides a way to package multiple coordinator and workflow jobs and to manage the lifecycle of those jobs

Features:

a. Oozie is a workflow scheduler system to manage Apache Hadoop jobs.
b. Oozie Workflow jobs are Directed Acyclical Graphs (DAGs) of actions.

c. Oozie Coordinator jobs are recurrent Oozie Workflow jobs triggered by time (frequency) and data availabilty.
d. Oozie is integrated with the rest of the Hadoop stack supporting several types of Hadoop jobs out of the box (such as Java map-reduce, Streaming map-reduce, Pig, Hive, Sqoop and Distcp) as well as system specific jobs (such as Java programs and shell scripts).
e. Oozie is a scalable, reliable and extensible system.

11. Zookeeper

ZooKeeper is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services. All of these kinds of services are used in some form or another by distributed applications. Each time they are implemented there is a lot of work that goes into fixing the bugs and race conditions that are inevitable. Because of the difficulty of implementing these kinds of services, applications initially usually skimp on them ,which make them brittle in the presence of change and difficult to manage. Even when done correctly, different implementations of these services lead to management complexity when the applications are deployed

Features:

a. Fast. ZooKeeper is especially fast with workloads where reads to the data are more common than writes. The ideal read/write ratio is about 10:1.
b. Reliable. ZooKeeper is replicated over a set of hosts (called an ensemble) and the servers are aware of each other. As long as a critical mass of servers is available, the ZooKeeper service will also be available. There is no single point of failure.
c. Simple. ZooKeeper maintain a standard hierarchical name space, similar to files and directories.
d. Ordered. The service maintains a record of all transactions, which can be used for higher-level abstractions, like synchronization primitives.

12. Ambari

The Apache Ambari project is aimed at making Hadoop management simpler by developing software for provisioning, managing, and monitoring Apache Hadoop clusters. Ambari provides an intuitive, easy-to-use Hadoop management web UI backed by its RESTful APIs.

Features:

a. Ambari provides a dashboard for monitoring health and status of the Hadoop cluster.
b. Ambari leverages Ganglia for metrics collection.
c. Ambari leverages Nagios for system alerting and will send emails when your attention is needed (e.g., a node goes down, remaining disk space is low, etc)

13. Sqoop

Sqoop is a tool designed to transfer data between Hadoop and relational databases. You can use Sqoop to import data from a relational database management system (RDBMS) such as MySQL or Oracle into the Hadoop Distributed File System (HDFS), transform the data in Hadoop MapReduce, and then export the data back into an RDBMS.

Features:

a. Connecting to database server
b. Controlling parallelism
c. Controlling the import process
d. Import data to hive
e. Import data to Hbase

14. Flume

Flume is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of log data. It has a simple and flexible architecture based on streaming data flows. It is robust and fault tolerant with tunable reliability mechanisms and many failover and recovery mechanisms. It uses a simple extensible data model that allows for online analytic application.

Features:

a. New in-memory channel that can spill to disk
b. A new dataset sink that use Kite API to write data to HDFS and HBase
c. Support for Elastic Search HTTP API in Elastic Search Sink
d. Much faster replay in the File Channel.

15. Kafka

A fast, scalable, fault-tolerant messaging system

Kafka is a fast, scalable, durable, and fault-tolerant publish-subscribe messaging system. Kafka is often used in place of traditional message brokers like JMS and AMQP because of its higher throughput, reliability and replication.

Kafka works in combination with Apache Storm, Apache HBase and Apache Spark for real-time analysis and rendering of streaming data. Kafka can message geospatial data from a fleet of long-haul trucks or sensor data from heating and cooling equipment in office buildings. Whatever the industry or use case, Kafka brokers massive message streams for low-latency analysis in Enterprise Apache Hadoop.
What Kafka Does

Apache Kafka supports a wide range of use cases as a general-purpose messaging system for scenarios where high throughput, reliable delivery, and horizontal scalability are important. Apache Storm and Apache HBase both work very well in combination with Kafka. Common use cases include:

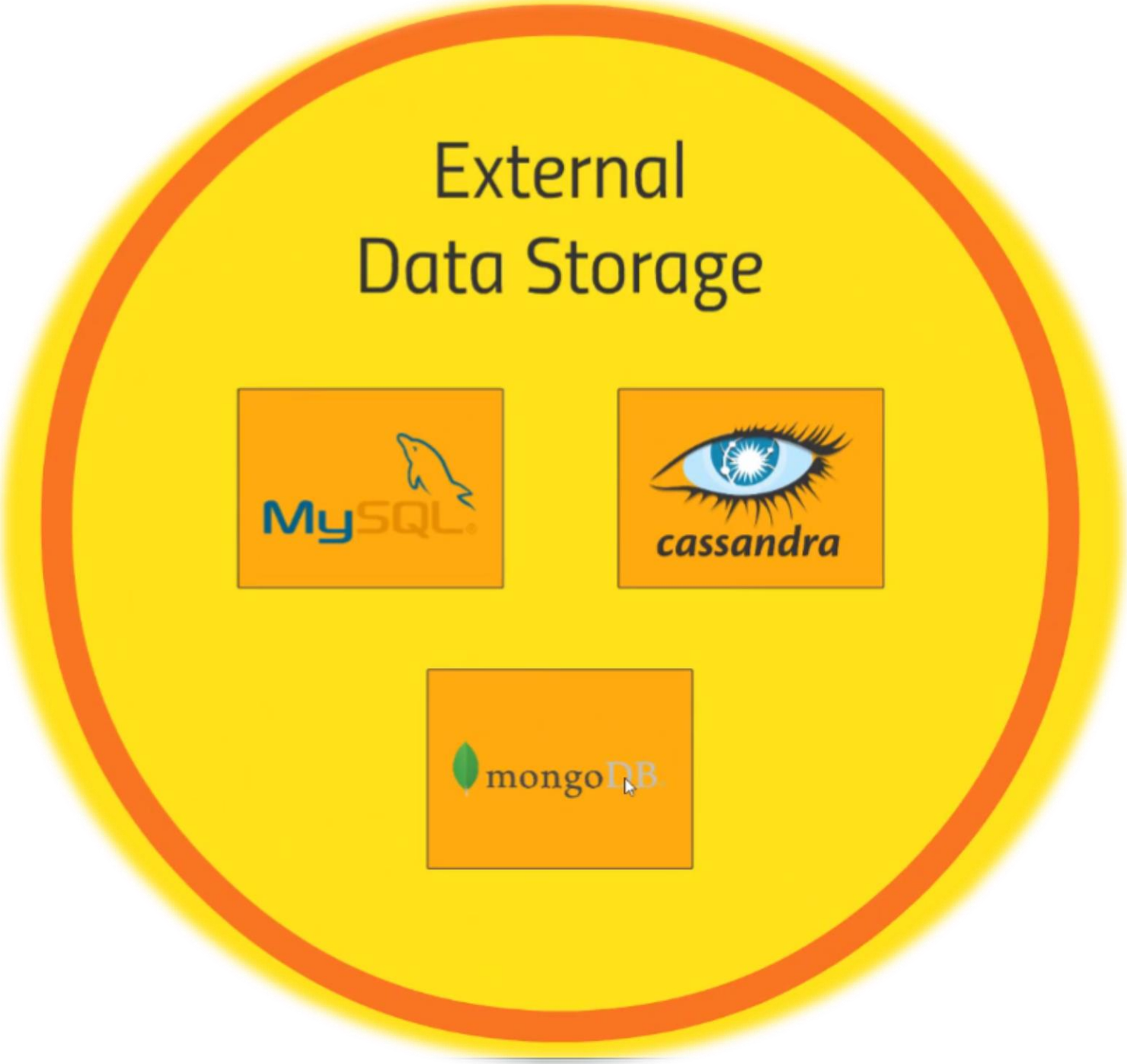
- Stream Processing
- Website Activity Tracking
- Metrics Collection and Monitoring
- Log Aggregation

Some of the important characteristics that make Kafka such an attractive option for these use cases include the following:

Features:

a. Scalability - Distributed system scales easily with no downtime
b. Durability - Persists messages on disk, and provides intra-cluster replication
c. Reliability - Replicates data, supports multiple subscribers, and automatically balances consumers in case of failure
d. Performance - High throughput for both publishing and subscribing, with disk structures that provide constant performance even with many terabytes of stored messages

External Data Storage



1. MongoDB

A MongoDB deployment hosts a number of databases. A database holds a set of collections. A collection holds a set of documents. A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data

Features:

a. Flexibility: MongoDB stores data in JSON documents (which we serialize to BSON). JSON provides a rich data model that seamlessly maps to native programming language types, and the dynamic schema makes it easier to evolve your data model than with a system with enforced schemas such as a RDBMS.
b. Power: MongoDB provides a lot of the features of a traditional RDBMS such as secondary indexes, dynamic queries, sorting, rich updates, upserts (update if document exists, insert if it doesn't), and easy aggregation. This gives you the breadth of functionality that you are used to from an RDBMS, with the flexibility and scaling capability that the non-relational model allows.
c. Speed/Scaling: By keeping related data together in documents, queries can be much faster than in a relational database where related data is separated into multiple tables and then needs to be joined later. MongoDB also makes it easy to scale out your database. Autosharding allows you to scale your cluster linearly by adding more machines. It is possible to increase capacity without any downtime, which is very important on the web when load can increase suddenly and bringing down the website for extended maintenance can cost your business large amounts of revenue.

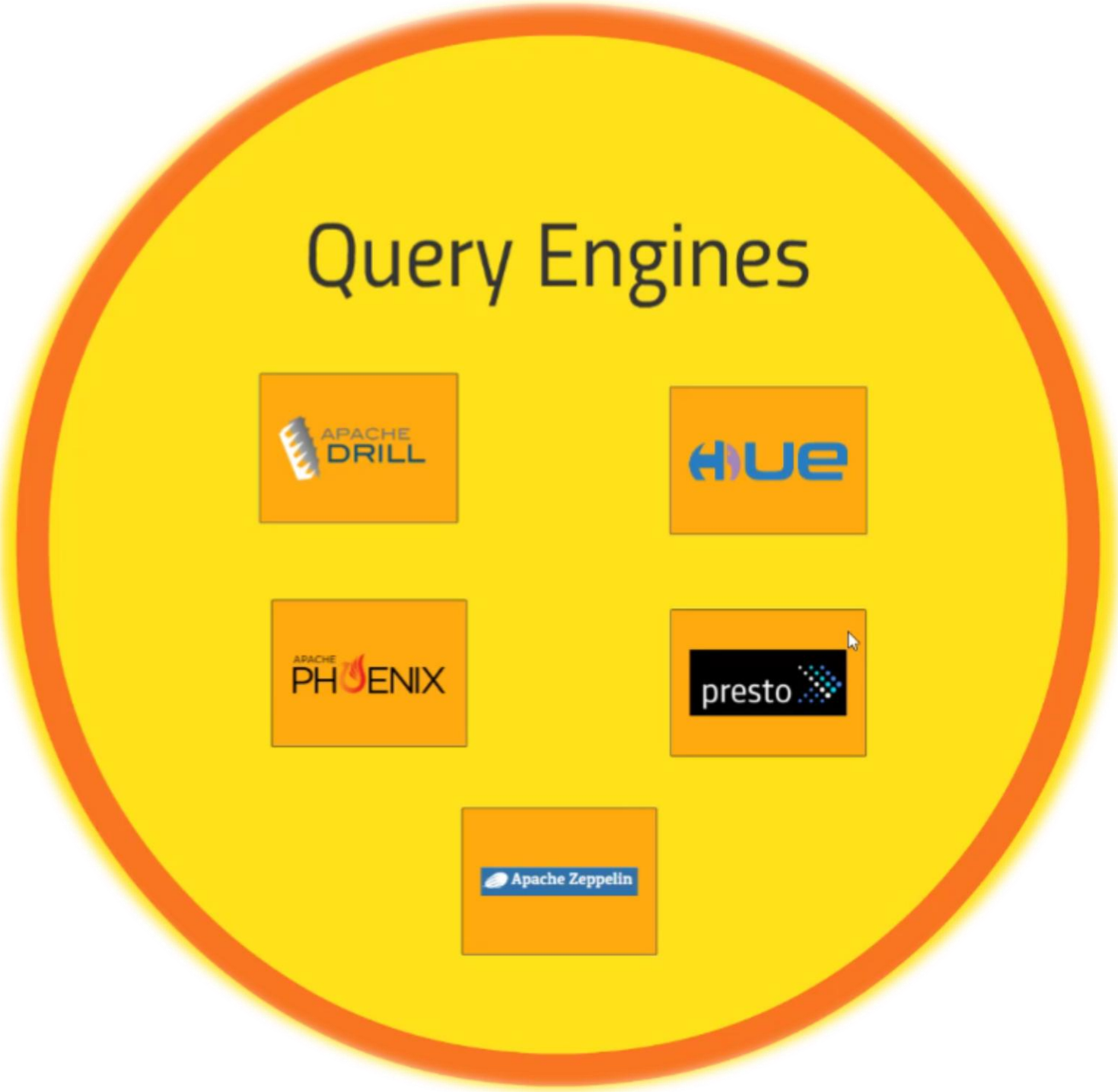
2. Cassandra

Cassandra is a NoSQL Column family implementation supporting the Big Table data model using the architectural aspects introduced by Amazon Dynamo. Some of the strong points of Cassandra are:

- Highly scalable and highly available with no single point of failure
- NoSQL column family implementation
- Very high write throughput and good read throughput
- SQL-like query language (since 0.8) and support search through secondary indexes
- Tunable consistency and support for replication
- Flexible schema

These positive points make it easy to recommend Cassandra, but it is crucial for a developer to delve into the details and tricky points of Cassandra to grasp the intricacies of this program.

Query Engines

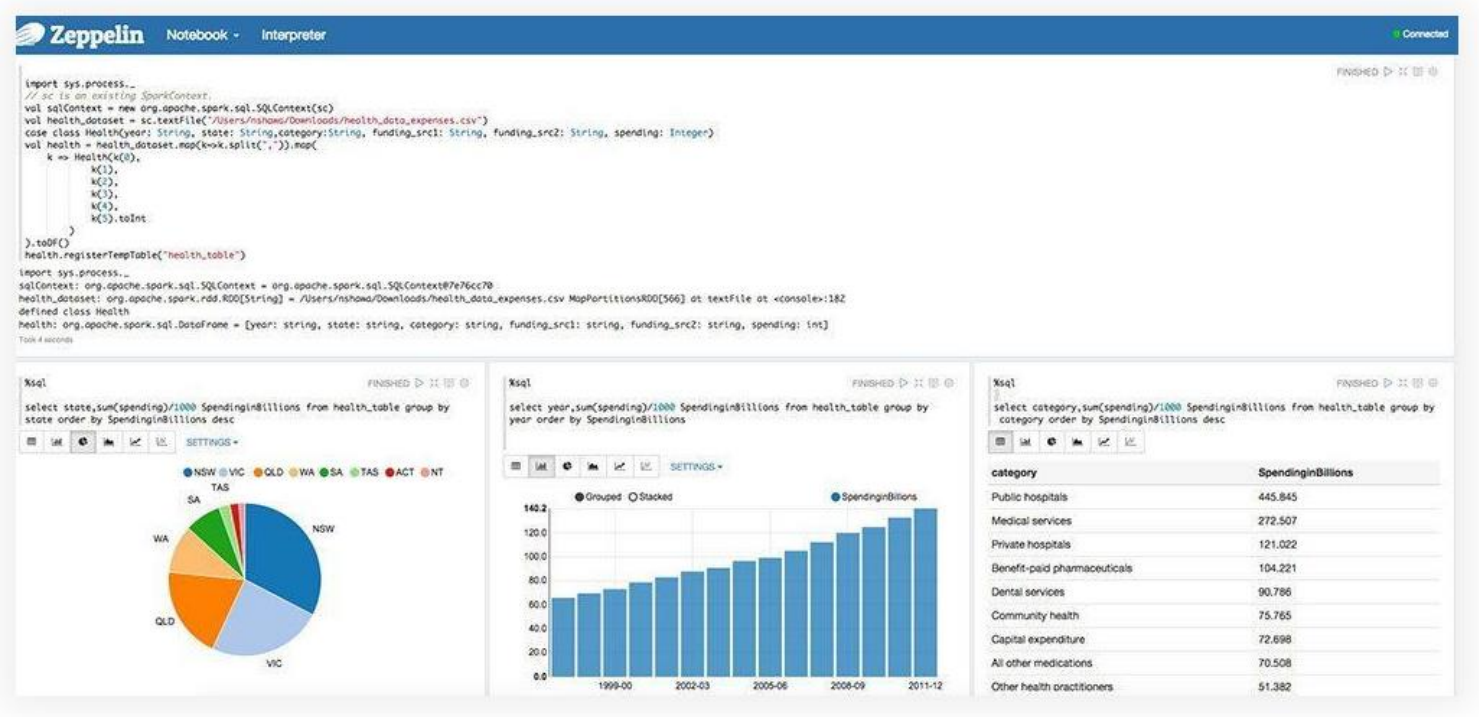


1. Zeppelin

Apache Zeppelin is a new and incubating multi-purposed web-based notebook which brings data ingestion, data exploration, visualization, sharing and collaboration features to Hadoop and Spark.

Interactive 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. There are a number of notebooks available with Spark. iPython remains a mature choice and great example of a data science notebook. The Hortonworks Gallery provides an Ambari stack definition to help our customers quickly set up iPython on their Hadoop clusters.

Apache Zeppelin is a new and upcoming web-based notebook which brings data exploration, visualization, sharing and collaboration features to Spark. It support Python, but also a growing list of programming languages such as Scala, Hive, SparkSQL, shell and markdown.

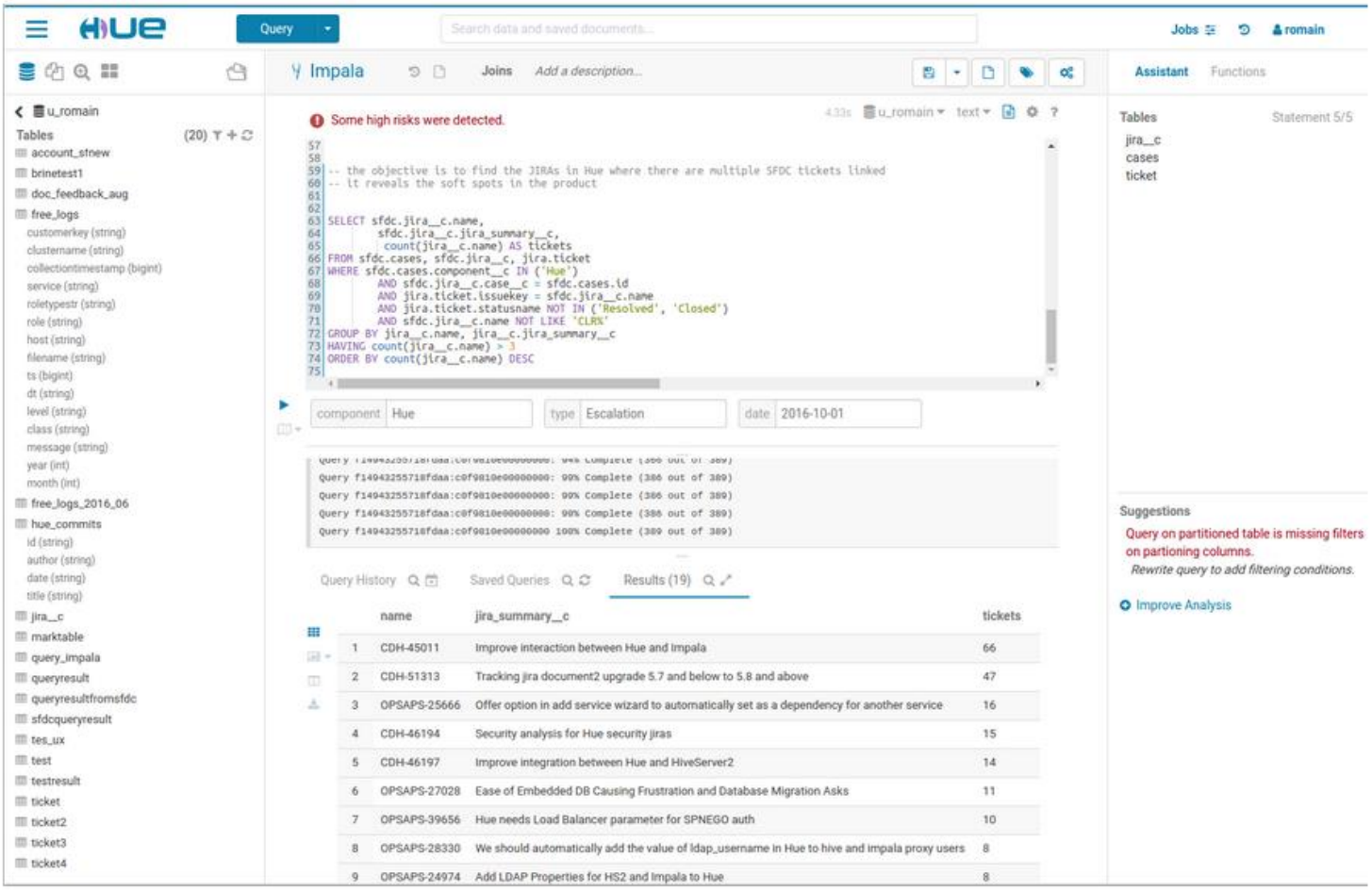


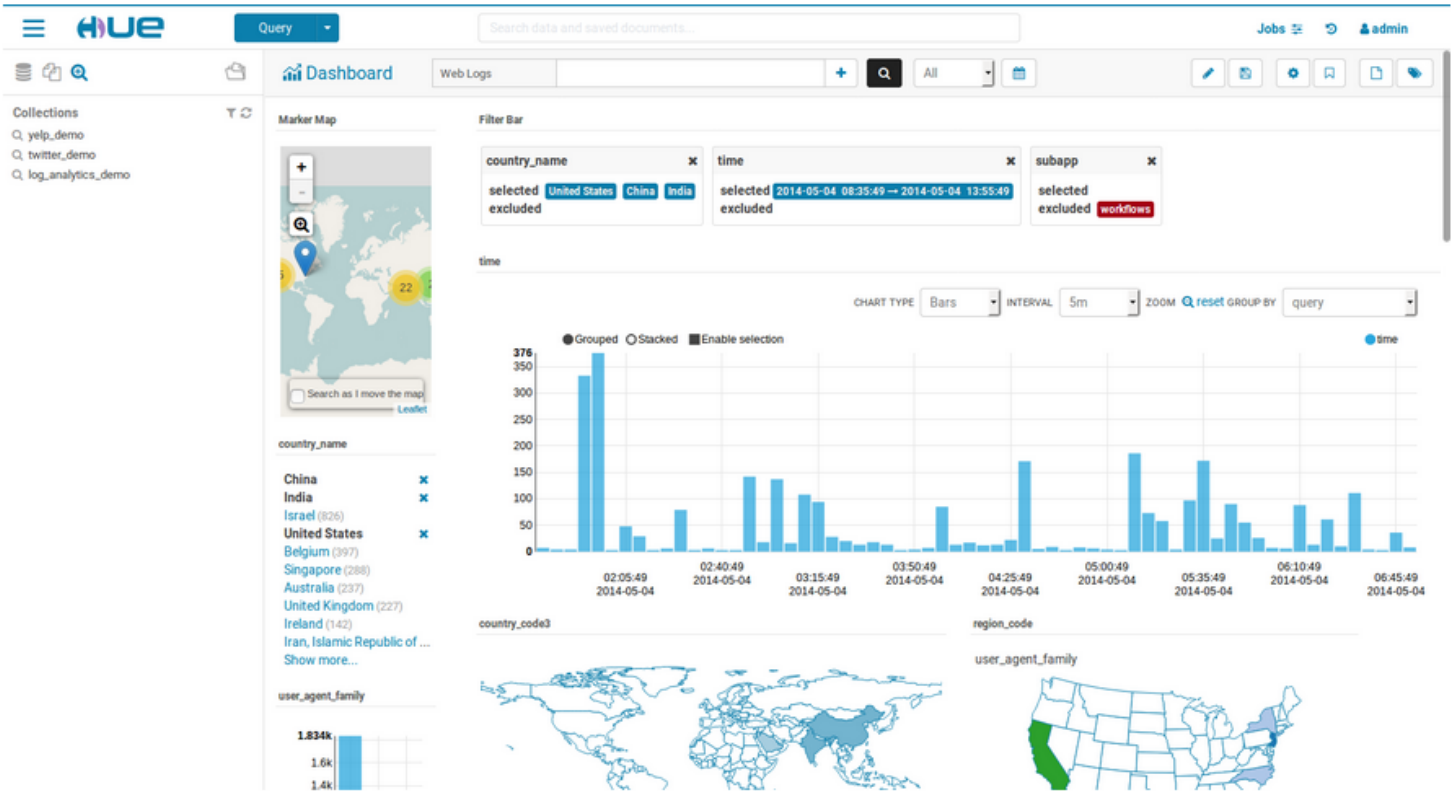
2. Hadoop User Experience (Hue)

Hue (Hadoop User Experience) is an open-source Web interface that supports Apache Hadoop and its ecosystem, licensed under the Apache v2 license.

Features:

a. Editors for Hive, Impala, Pig, MapReduce, Spark and any SQL like MySQL, Oracle, SparkSQL, Solr SQL, Phoenix and more.
b. Dashboards to dynamically interact and visualize data with Solr or SQL.
Scheduler of jobs and workflows.
c. Browsers for Jobs, HDFS, S3 files, SQL Tables, Indexes, Git files, Sentry permissions, Sqoop and more.


Query
Search data and saved documents...
Jobs
romain
Impala
Joins
Add a description...
Assistant
Functions
u_romain
Tables (20)
account_stnew
brinetest1
doc_feedback_aug
free_logs
customerkey (string)
clustername (string)
collectiontimestamp (bigint)
service (string)
roletypestr (string)
role (string)
host (string)
filename (string)
ts (bigint)
dt (string)
level (string)
class (string)
message (string)
year (int)
month (int)
free_logs_2016_06
hue_commits
id (string)
author (string)
date (string)
title (string)
jira__c
marktable
query_impala
queryresult
queryresultfromsfdc
sfdcqueryresult
tes_ux
test
testresult
ticket
ticket2
ticket3
ticket4
Some high risks were detected.
-- the objective is to find the JIRAs in Hue where there are multiple SFDC tickets linked
-- it reveals the soft spots in the product
SELECT sfdc.jira__c.name,
sfdc.jira__c.jira_summary__c,
count(jira__c.name) AS tickets
FROM sfdc.cases, sfdc.jira__c, jira.ticket
WHERE sfdc.cases.component__c IN ('Hue')
AND sfdc.jira__c.case__c = sfdc.cases.id
AND jira.ticket.issuekey = sfdc.jira__c.name
AND jira.ticket.statusname NOT IN ('Resolved', 'Closed')
AND sfdc.jira__c.name NOT LIKE 'CLRK'
GROUP BY jira__c.name, jira__c.jira_summary__c
HAVING count(jira__c.name) > 3
ORDER BY count(jira__c.name) DESC
component Hue
type Escalation
date 2016-10-01
Query f14943255718fdaa:c0f9810e00000000: 99% Complete (386 out of 389)
Query f14943255718fdaa:c0f9810e00000000: 99% Complete (386 out of 389)
Query f14943255718fdaa:c0f9810e00000000: 99% Complete (386 out of 389)
Query f14943255718fdaa:c0f9810e00000000 100% Complete (389 out of 389)
Query History
Saved Queries
Results (19)
name
jira_summary__c
tickets
1 CDH-45011 Improve interaction between Hue and Impala 66
2 CDH-51313 Tracking jira document2 upgrade 5.7 and below to 5.8 and above 47
3 OPSAPS-25666 Offer option in add service wizard to automatically set as a dependency for another service 16
4 CDH-46194 Security analysis for Hue security jiras 15
5 CDH-46197 Improve integration between Hue and HiveServer2 14
6 OPSAPS-27028 Ease of Embedded DB Causing Frustration and Database Migration Asks 11
7 OPSAPS-39656 Hue needs Load Balancer parameter for SPNEGO auth 10
8 OPSAPS-28330 We should automatically add the value of ldap_username in Hue to hive and impala proxy users 8
9 OPSAPS-24974 Add LDAP Properties for HS2 and Impala to Hue 8
Tables
Statement 5/5
jira__c
cases
ticket
Suggestions
Query on partitioned table is missing filters on partioning columns.
Rewrite query to add filtering conditions.
Improve Analysis


Query
Search data and saved documents...
Jobs
admin
Dashboard
Web Logs
All
Collections
yelp_demo
twitter_demo
log_analytics_demo
Marker Map
Search as I move the map
Leaflet
country_name
China
India
Israel (826)
United States
Belgium (397)
Singapore (288)
Australia (237)
United Kingdom (227)
Ireland (142)
Iran, Islamic Republic of ...
Show more...
user_agent_family
1.834k
1.6k
1.4k
Filter Bar
country_name
selected United States China India
excluded
time
selected 2014-05-04 08:35:49 → 2014-05-04 13:55:49
excluded
subapp
selected
excluded workflows
time
CHART TYPE Bars
INTERVAL 5m
ZOOM reset
GROUP BY query
Grouped
Stacked
Enable selection
time
376
350
300
250
200
150
100
50
0
02:05:49 2014-05-04
02:40:49 2014-05-04
03:15:49 2014-05-04
03:50:49 2014-05-04
04:25:49 2014-05-04
05:00:49 2014-05-04
05:35:49 2014-05-04
06:10:49 2014-05-04
06:45:49 2014-05-04
country_code3
region_code
user_agent_family