



CLOUD COMPUTING APPLICATIONS

Hortonworks

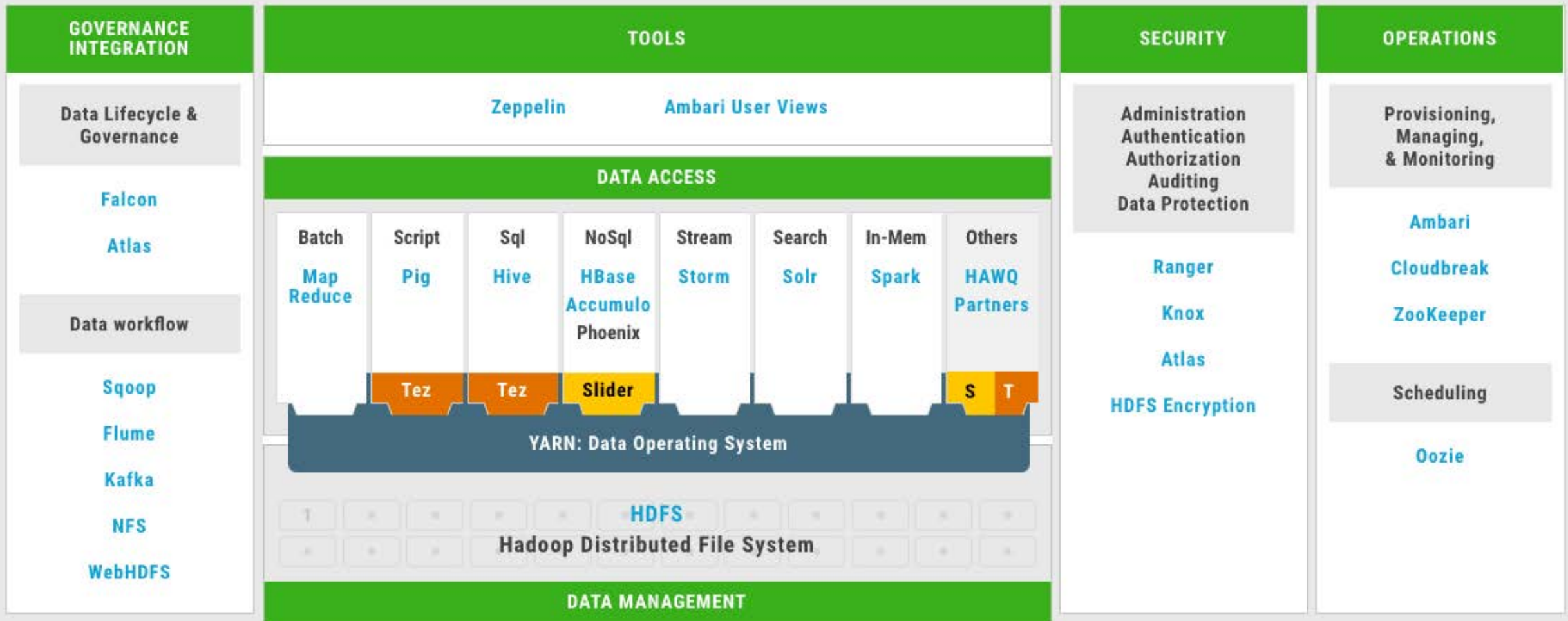
Roy Campbell & Reza Farivar

Pieces of the Puzzle

Connected Data Strategy

- HDP: Apache Hadoop® is an open source framework for distributed storage and processing of large sets of data on commodity hardware. Hadoop enables businesses to quickly gain insight from massive amounts of structured and unstructured data.
- HDF enables real-time data collection, curation, analysis and delivery of data to and from any device, source or system, either on-premise and in the cloud

Hortonworks Data Platform (HDP)



HDP Tools

- Apache Zeppelin: open web-based notebook that enables interactive data analytics
- Apache Ambari: source management platform for provisioning, managing, monitoring and securing Apache Hadoop clusters.

Apache Zeppelin (Python, Spark, Scala...)


Zeppelin Notebook - Interpreter Connected

```
import sys, process...
// sc is an existing SparkContext
val sqlContext = new org.apache.spark.sql.SQLContext(sc)
val health_dataset = sc.textFile("/Users/nishant/Downloads/health_data_expenses.csv")
case class Health(year: String, state: String, category: String, funding_src1: String, funding_src2: String, spending: Integer)
val health = health_dataset.map(k => k.split(",")).map{
  k => Health(k(0),
    k(1),
    k(2),
    k(3),
    k(4),
    k(5).toInt)
}.toDF()
health.registerTempTable("health_table")

import sys, process...
sqlContext = org.apache.spark.sql.SQLContext = org.apache.spark.sql.SQLContext@7e76cc70
health_dataset: org.apache.spark.rdd.RDD[String] = /Users/nishant/Downloads/health_data_expenses.csv MapPartitionsRDD[566] at textFile at <console>:182
defined class Health
health: org.apache.spark.sql.DataFrame = [year: string, state: string, category: string, funding_src1: string, funding_src2: string, spending: int]
```

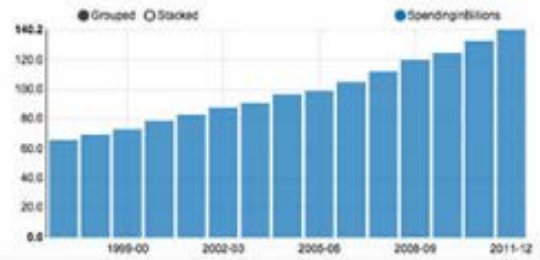
Sql FINISHED

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



Sql FINISHED

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



Sql FINISHED

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	90.766
Community health	75.765
Capital expenditure	72.698
All other medications	70.506
Other health practitioners	51.382

Apache Zeppelin (Python, Spark, Scala...)

User View	Description
Tez	The Tez View helps you understand and optimize your cluster resource usage. Using the view, you can optimize and accelerate individual SQL queries or Pig jobs to get the best performance in a multi-tenant Hadoop environment.
Hive	Hive View allows the user to write & execute SQL queries on the cluster. It shows the history of all Hive queries executed on the cluster whether run from Hive view or another source such as JDBC/ODBC or CLI. It also provides graphical view of the query execution plan. This helps the user debug the query for correctness and for tuning the performance. It integrates Tez View that allows the user to debug any Tez job, including monitoring the progress of a job (whether from Hive or Pig) while it is running. This view contribution can be found here .
Pig	Pig View is similar to the Hive View. It allows writing and running a Pig script. It has support for saving scripts, and loading and using existing UDFs in scripts. This view contribution can be found here .
Capacity Scheduler	Capacity Scheduler View helps a Hadoop operator setup YARN workload management easily to enable multi-tenant and multi-workload processing. This view provisions cluster resources by creating and managing YARN queues. This view contribution can be found here .
Files	Files View allows the user to manage, browse and upload files and folders in HDFS. This view contribution can be found here .

Data Access

- **YARN:** Data Operating System
 - **MapReduce.** Batch application framework for structured and unstructured data
 - **Pig.** Script Extract-transform-load (ETL) data pipelines, Research on raw data, and Iterative data processing.
 - SQL. **Hive** interactive SQL queries over petabytes of data in Hadoop
 - NoSql (**Hbase Accumulo**) non-relational (NoSQL) database on top of HDFS
 - **Storm** (Stream) distributed real-time --large volumes of high-velocity data.
 - **Solr** (Search) -- full-text search and near real-time indexing.
 - **Spark** (In-Mem)
- Data Management
 - Hadoop Distributed File System (**HDFS**)

MapReduce

Benefit	Description
Simplicity	Developers can write applications in their language of choice, such as Java, C++ or Python, and MapReduce jobs are easy to run
Scalability	MapReduce can process petabytes of data, stored in HDFS on one cluster
Speed	Parallel processing means that MapReduce can take problems that used to take days to solve and solve them in hours or minutes
Recovery	MapReduce takes care of failures. If a machine with one copy of the data is unavailable, another machine has a copy of the same key/value pair, which can be used to solve the same sub-task. The JobTracker keeps track of it all.
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 contributes to Hadoop's processing speed.

Pig

Characteristic	Benefit
Extensible	Pig users can create custom functions to meet their particular processing requirements
Easily programmed	Complex tasks involving interrelated data transformations can be simplified and encoded as data flow sequences. Pig programs accomplish huge tasks, but they are easy to write and maintain.
Self-optimizing	Because the system automatically optimizes execution of Pig jobs, the user can focus on semantics.

Hive

Feature	Description
Familiar	Query data with a SQL-based language
Fast	Interactive response times, even over huge datasets
Scalable and Extensible	As data variety and volume grows, more commodity machines can be added, without a corresponding reduction in performance
Compatible	Works with traditional data integration and data analytics tools.

Nosql HBase

Characteristic	Benefit
Fault tolerant	<ul style="list-style-type: none">• Replication across the data center• Atomic and strongly consistent row-level operations• High availability through automatic failover• Automatic sharding and load balancings of tables
Fast	<ul style="list-style-type: none">• Near real time lookups• In-memory caching via block cache and bloom filters• Server side processing via filters and co-processors
Usable	<ul style="list-style-type: none">• Data model accommodates wide range of use cases• Metrics exports via File and Ganglia plugins• Easy Java API as well as Thrift and REST gateway APIs

Nosql Accumulo

Feature	Benefit
Table design and configuration	<ul style="list-style-type: none">• Includes cell tables for cell-level access control• Large rows need not fit into memory
Integrity and availability	<ul style="list-style-type: none">• Master fail-over with ZooKeeper locks• Write-ahead logs for recovery• Scalable master metadata store• Fault tolerant executor (FATE)
Performance	<ul style="list-style-type: none">• Relative encoding to compress similar consecutive keys• Speed long scans with parallel server threads• Cache recently scanned data
Data Management	<ul style="list-style-type: none">• Group columns within a single file• Automatic tablet splitting and rebalancing• Merge tablets and clone tables

Storm

Feature	Description
Fast	– benchmarked as processing one million 100 byte messages per second per node
Scalable	– with parallel calculations that run across a cluster of machines
Fault-tolerant	– when workers die, Storm will automatically restart them. If a node dies, the worker will be restarted on another node.
Reliable	– Storm guarantees that each unit of data (tuple) will be processed at least once or exactly once. Messages are only replayed when there are failures.
Easy to operate	– standard configurations are suitable for production on day one. Once deployed, Storm is easy to operate.

Solr

Feature	Description
Advanced full-text search	
Near real-time indexing	
Standards-based open interfaces like XML, JSON and HTTP	
Comprehensive HTML administration interfaces	
Server statistics exposed over JMX for monitoring	
Linearly scalable, auto index replication, auto failover and recovery	
Flexible and adaptable, with XML configuration	

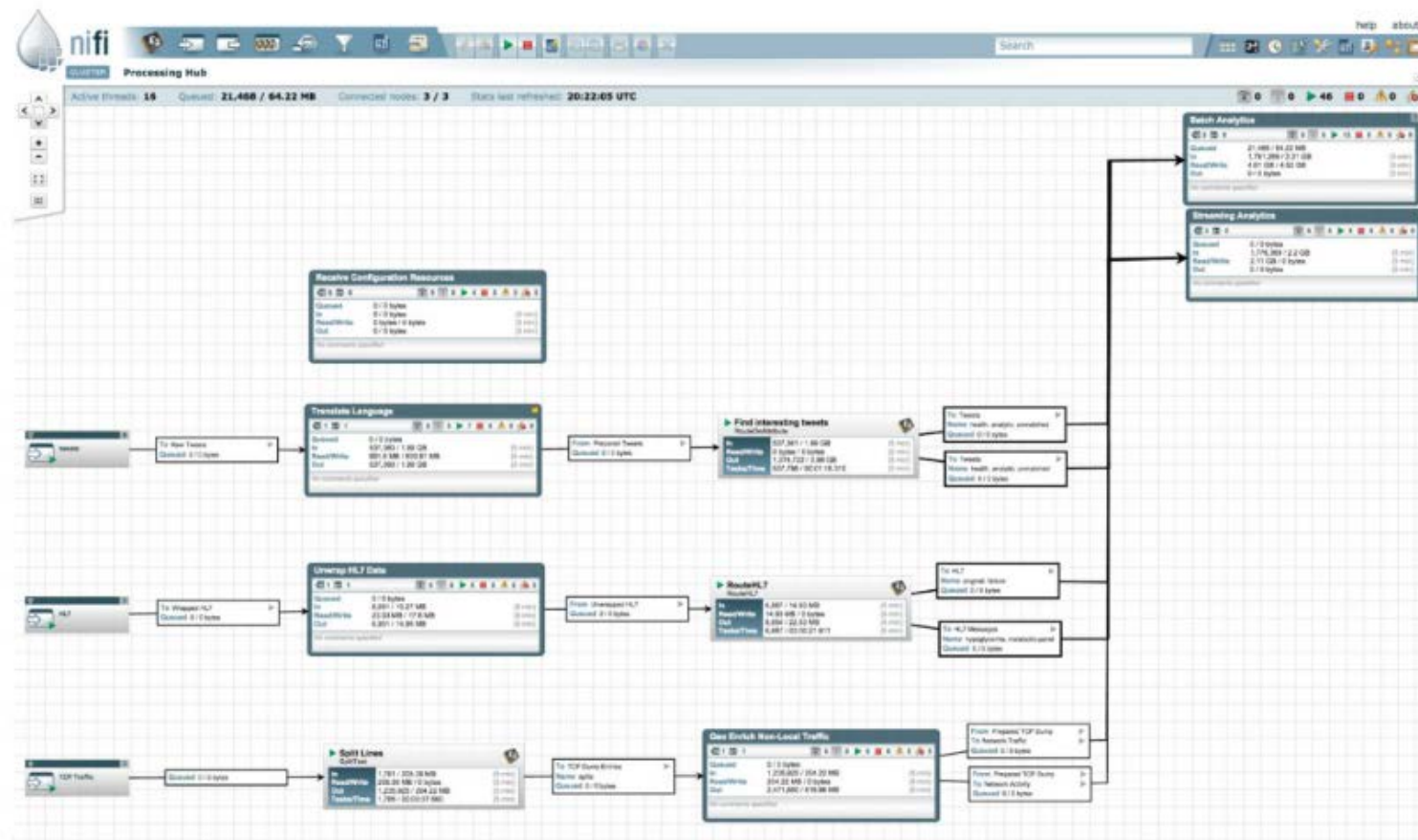
Spark

Feature	Description
• In memory Spark Core	
• Programming Scala, Java, Python, R, APIs	
• Libraries	
• Mlib	
• Spark SQL	
• Spark Streaming	
• Graphx	
• ML Pipeline for data science workflow	

Hortonworks Data Flow (HDF)

- Apache NiFi, Kafka and Storm provide real-time dataflow management and streaming analytics.
- HDF enables real-time data collection, curation, analysis and delivery of data to and from any device, source or system, either on-premise and in the cloud.
- Tools
 - KAFKA: fast, scalable, durable, and fault-tolerant publish-subscribe messaging
 - NIFI, integrated data logistics and simple event processing
 - STORM

NiFi



NiFi

NiFi Term	FBP Term	Description
FlowFile	Information Packet	A FlowFile represents each object moving through the system and for each one, NiFi keeps track of a map of key/value pair attribute strings and its associated content of zero or more bytes.
FlowFile Processor	Black Box	Processors actually perform the work. In [eip] terms a processor is doing some combination of data Routing, Transformation, or Mediation between systems. Processors have access to attributes of a given FlowFile and its content stream. Processors can operate on zero or more FlowFiles in a given unit of work and either commit that work or rollback.
Connection	Bounded Buffer	Connections provide the actual linkage between processors. These act as queues and allow various processes to interact at differing rates. These queues then can be prioritized dynamically and can have upper bounds on load, which enable back pressure.
Flow Controller	Scheduler	The Flow Controller maintains the knowledge of how processes actually connect and manages the threads and allocations thereof which all processes use. The Flow Controller acts as the broker facilitating the exchange of FlowFiles between processors.
Process Group	Subnet	A Process Group is a specific set of processes and their connections, which can receive data via input ports and send data out via output ports. In this manner process groups allow creation of entirely new components simply by composition of other components.