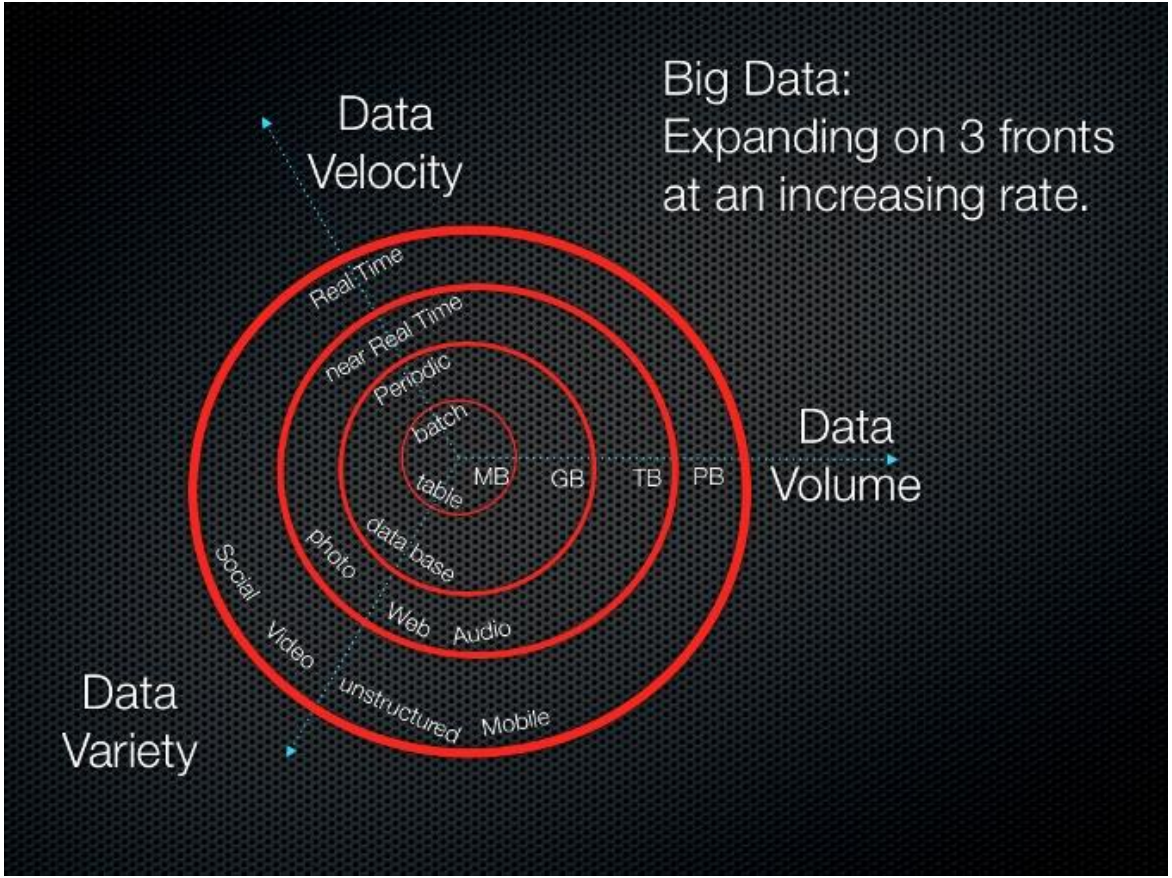
PROFINIT

B0M33BDT – BigData Hadoop

Josef Vonášek

25. října 2023

Big Data


Big Data:
Expanding on 3 fronts
at an increasing rate.
Data Velocity
Real Time
near Real Time
Periodic
batch
Data Volume
MB
GB
TB
PB
table
data base
photo
Web
Audio
Social
Video
unstructured
Mobile
Data Variety

BIG DATA – Chat GPT

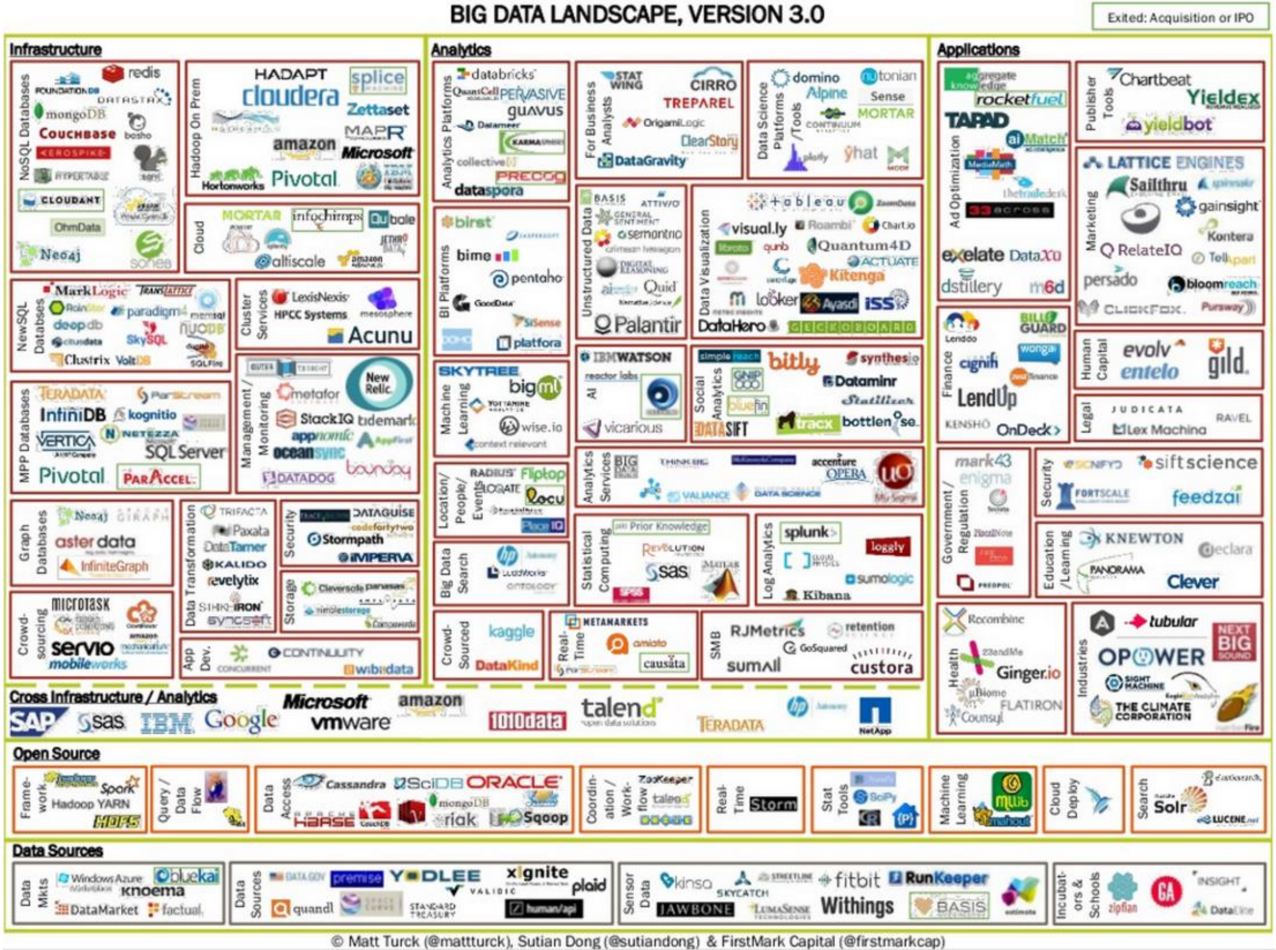
PROFINIT

- Big data se liší od běžných dat v několika ohledech [1]:
- **Velikost**: Běžná data jsou obvykle menší a mohou být uložena na jednom serveru, zatímco big data jsou obvykle mnohem větší a mohou být měřena v petabytech, zettabajtech nebo exabytech [1][2].
- **Struktura**: Běžná data jsou obvykle strukturovaná a relační, zatímco big data mohou být strukturovaná, nestrukturovaná nebo polostrukturovaná [1][2].
- **Zdroj**: Běžná data pocházejí obvykle z interních systémů organizace, zatímco big data mohou pocházet z různých zdrojů, jako jsou sociální sítě, senzory IoT, mobilní zařízení a další [1][2].
- **Rychlost**: Big data se často generují rychle a mohou být zpracovávána v reálném čase, zatímco běžná data se obvykle generují pomaleji a mohou být zpracovávána v dávkách [1][2].

Zdroj: Konverzace s Bingem, 1. 10. 2023

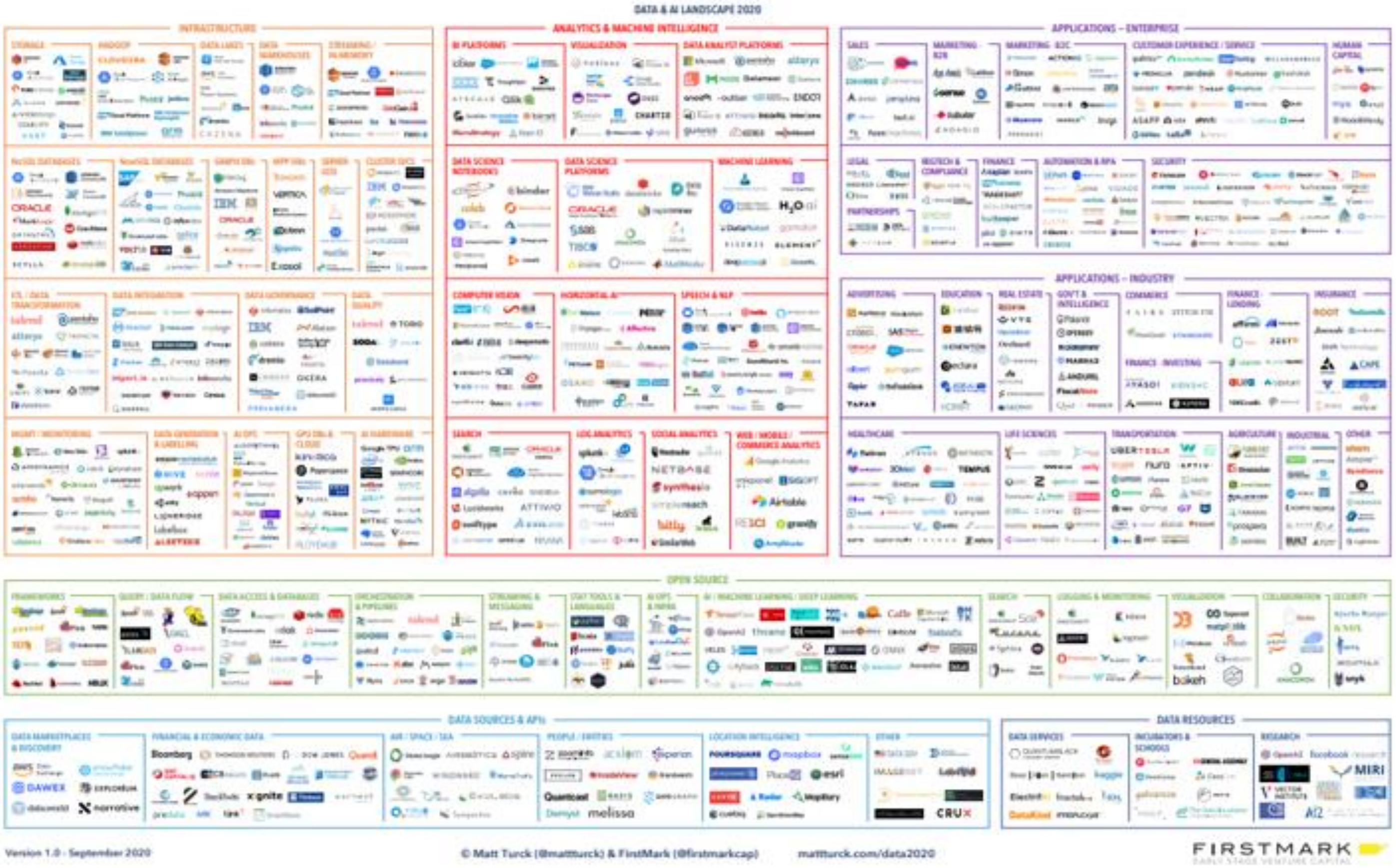
Big data Landscape 3.0

PROFINIT

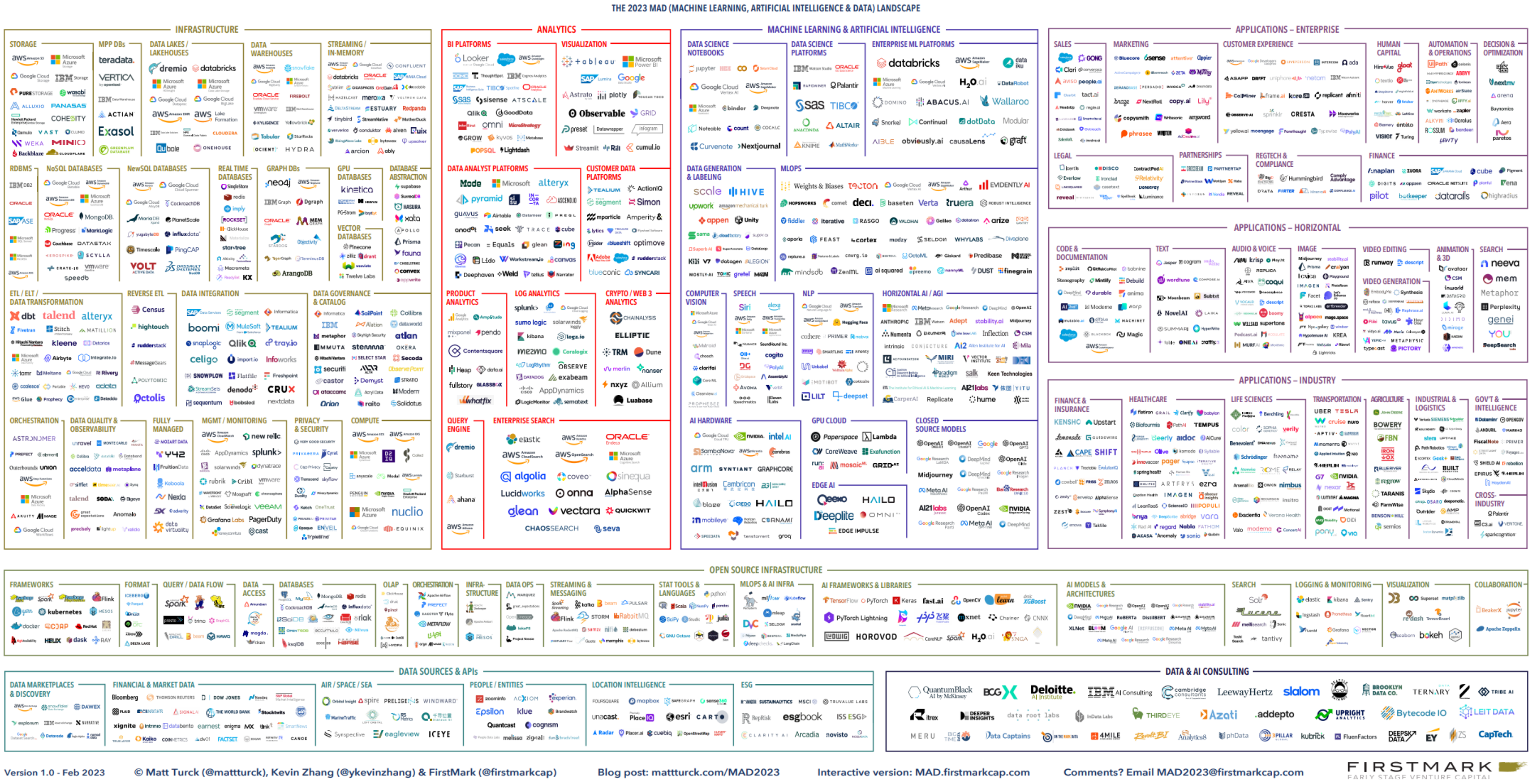


Data Landscape 2020

PROFINIT



Data Lanscape 2023



Apache Hadoop

PROFINIT

- Wikipedia:
  - Apache Hadoop (pronunciation: /həˈduːp/) is an open-source software framework for **distributed storage** and **distributed processing** of **very large data sets** on computer clusters ***built from commodity hardware***. All the modules in Hadoop are designed with a **fundamental assumption that hardware failures are common** and should be automatically handled by the framework.

- Commodity hardware
  - Approx. 10 000 EUR+ (but not 100kEUR)
    - **2**-4 CPU, each CPU 8-20 cores
    - 256-512 GB RAM, min. 128GB
    - 10-20 2-4-8TB HDD
  - But you will probably not buy it in Alza

History

- 2006 – part of Nutch project
  - Doug Cutting
  - Mike Cafarella
- 2007 – Yahoo Hadoop on 1000 node cluster
- 2008 – part of Apache projject
- 2011 – first 1.0 version
- 2012 – 2.0 version with YARN
- 2017 – 3.0 version – two NameNodes
- Jun 2023 – 3.3.6 version – ARM support

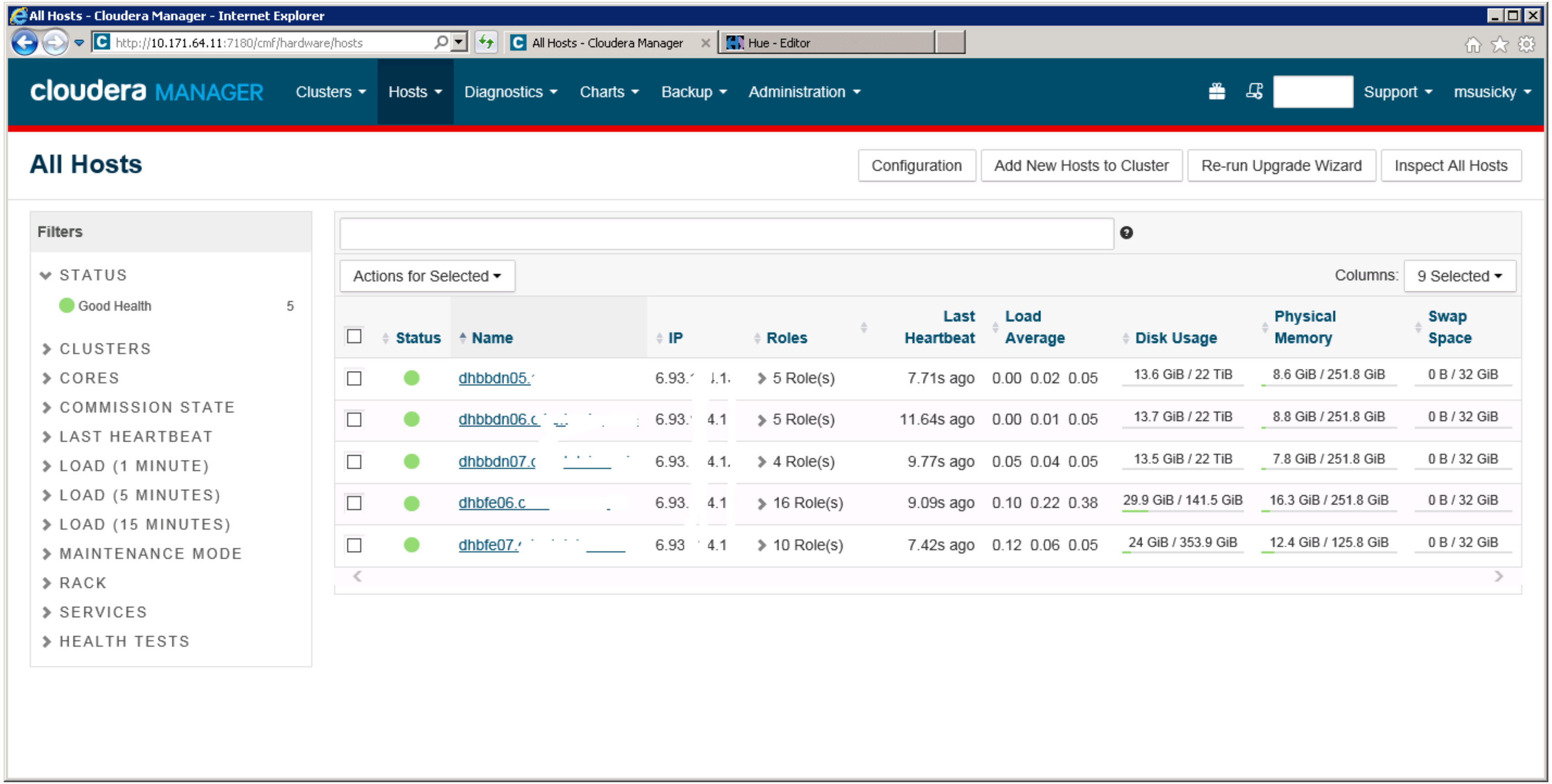
How Hadoop looks like?

- Yahoo

Hadoop - Seznam

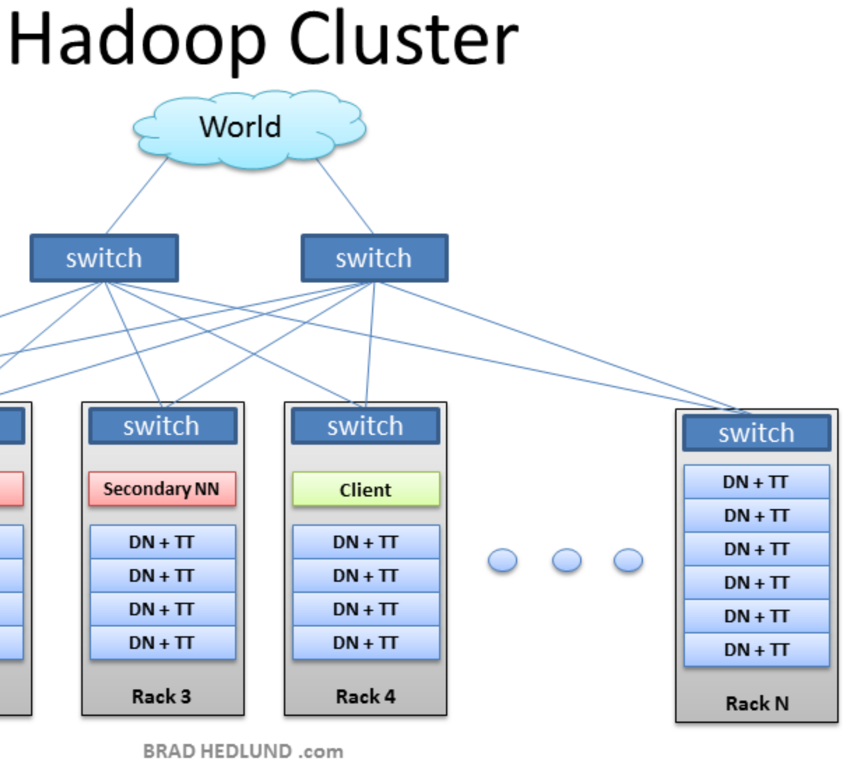
PROFINIT

Cheap HW


All Hosts - Cloudera Manager - Internet Explorer
http://10.171.64.11:7180/cmf/hardware/hosts
All Hosts - Cloudera Manager
Hue - Editor
cloudera MANAGER
Clusters
Hosts
Diagnostics
Charts
Backup
Administration
Support
msusicky
All Hosts
Configuration
Add New Hosts to Cluster
Re-run Upgrade Wizard
Inspect All Hosts
Filters
STATUS
Good Health 5
CLUSTERS
CORES
COMMISSION STATE
LAST HEARTBEAT
LOAD (1 MINUTE)
LOAD (5 MINUTES)
LOAD (15 MINUTES)
MAINTENANCE MODE
RACK
SERVICES
HEALTH TESTS
Actions for Selected
Columns: 9 Selected
Status Name IP Roles Last Heartbeat Load Average Disk Usage Physical Memory Swap Space
dhbbdn05 5 Role(s) 7.71s ago 0.00 0.02 0.05 13.6 GiB / 22 TiB 8.6 GiB / 251.8 GiB 0 B / 32 GiB
dhbbdn06 5 Role(s) 11.64s ago 0.00 0.01 0.05 13.7 GiB / 22 TiB 8.8 GiB / 251.8 GiB 0 B / 32 GiB
dhbbdn07 4 Role(s) 9.77s ago 0.05 0.04 0.05 13.5 GiB / 22 TiB 7.8 GiB / 251.8 GiB 0 B / 32 GiB
dhbfe06 16 Role(s) 9.09s ago 0.10 0.22 0.38 29.9 GiB / 141.5 GiB 16.3 GiB / 251.8 GiB 0 B / 32 GiB
dhbfe07 10 Role(s) 7.42s ago 0.12 0.06 0.05 24 GiB / 353.9 GiB 12.4 GiB / 125.8 GiB 0 B / 32 GiB

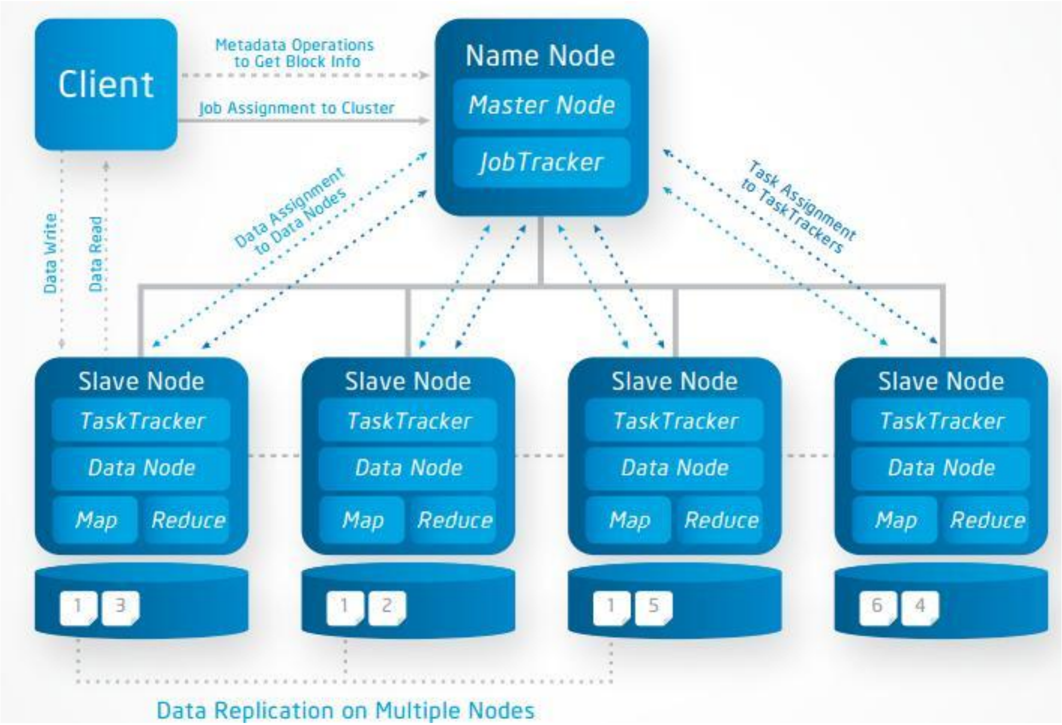
Hadoop – architecture I

PROFINIT



Hadoop – architecture II

PROFINIT



Read speed

- RAM
  - DDR4 approx. 15 GB/s
- Network 10 Gbit
  - 1.25 GB/s
- SSD disk
  - 200-700 MB/s
  - There are „Enterprise level“ - guarantee 5 years
  - Small capacity (~ TB) and pretty expensive
- HDD 7.2k
  - latency approximately 4ms
  - Sequential read 50-100 MB/s
  - Large capacities (4TB-8TB+) and relatively cheap
  - ➔ Hadoop works typically with storage

Read speed – HDD

- **Sequential read** – circa 100 MB/s on one disk
- Random access
  - Block size of ext4 is 4kB
  - latency, to find the block circa 4ms
  - max. speed of the pure random read 1/0.004*4096 = 1 MB/s

Bottlenecks

- Example: 10 nodes, each node 12 * 2 TB HDD
  - Read speed within the node: 12*100 MB/s = 1.2 GB/s
  - Read speed within the cluster: 12 GB/s

- Limits:
  - RAM speed – 10x faster ✔
  - CPU – read is not consuming CPU ✔
  - Network – for one node limited! ✖
  - Bus – beware – number of disks and watch cache and throughput !

Principles

- Storage capacity
  - Many servers = nodes [4 s -> 1000s]
  - Every node many disks [10-20]
- High availability
  - Data replication (typically three copies of every file)
  - 2 replicase in same rack, the third one different rack
- Reading
  - data is spreaded across cluster – single file can be spreaded as well!
  - data is replicated – Parallel reading on several nodes
  - Big files – **sequential reading**
- Distributed computing
  - Many nodes

Sizing

PROFINIT

- How to build Hadoop
  - Why do I need Hadoop ?
    - What data
    - What task
    - How do I use it
  - HDD parameters
    - Transfer speed
  - RAID
    - 0, 1, 1+0, 5, 6, (2,3,4,7)
  - Network speed
  - Memory
  - CPU cores

HDFS

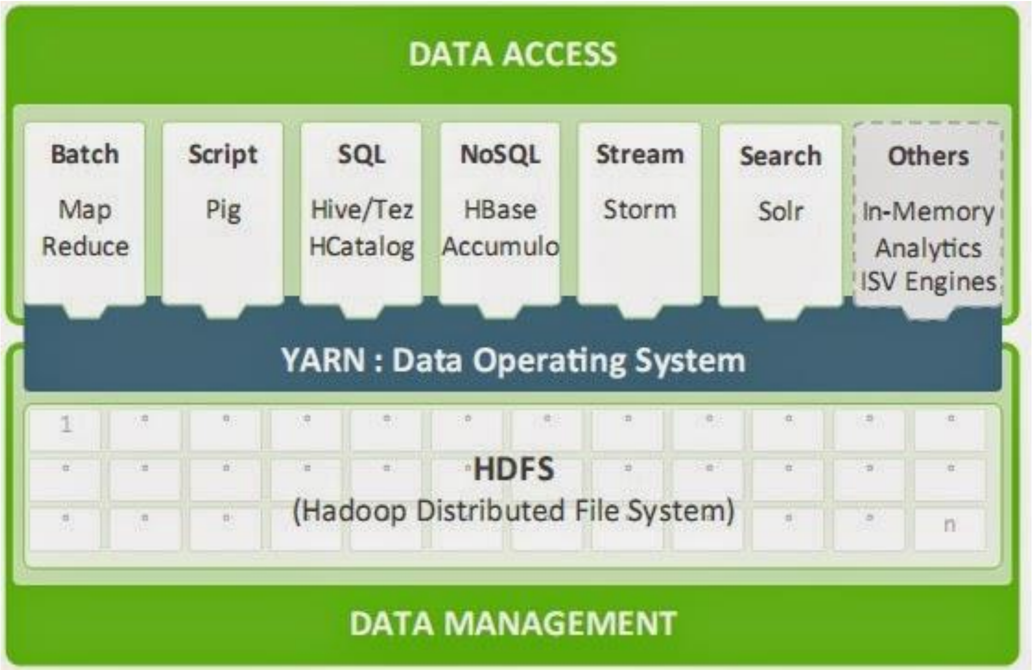
HDFS

PROFINIT

- HDFS
  - NameNode, DataNode
  - Replication
  - File system operations
  - Blocks, block size

HDFS- architecture

PROFINIT

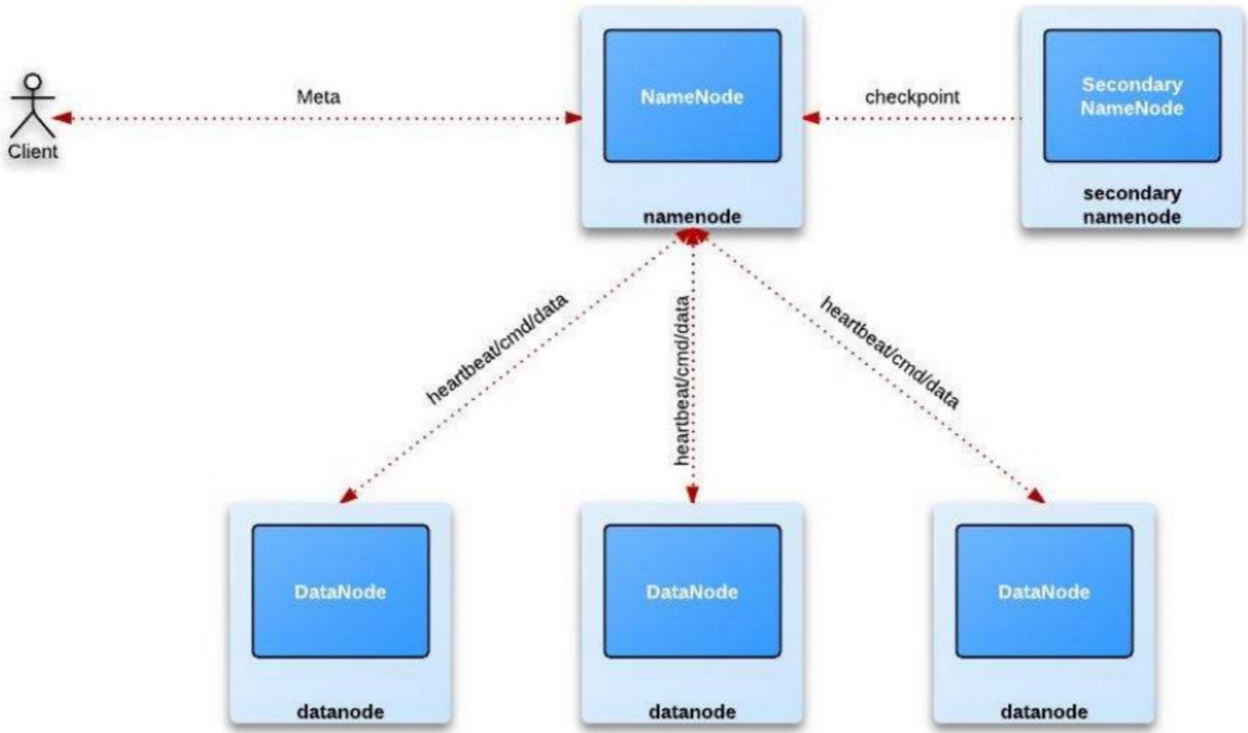


HDFS

- Hadoop Distributed Filesystem
- Good for
  - Large files
  - Stream access
- Bad for
  - Small files
  - Random access
  - Low latency access
- Master-slave design
  - Master – NameNode (Secondary NameNode)
  - Slave – DataNode

HDFS

PROFINIT



HDFS

PROFINIT

- HDFS files are splitted to blocks
  - Default 64MB/128MB - > can be changed
  - Very good for big files
  - Very bad for small files
- Replication
  - Every block is (can be) replicated between nodes
  - Fault tolerant
  - Default replication factor -> 3

DataNode

- Store data block
- Get data block from clients
- Get data block from other DataNodes
  - Replication
- Get delete request from NameNode

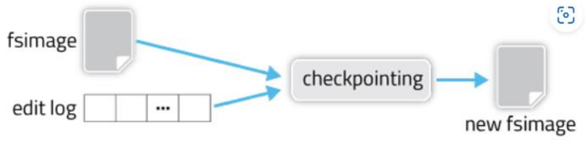
NameNode

PROFINIT

- Metadata
  - Where are data
  - Stored in Memory !
  - Cca 1GB per 1M objects

Object	size estimate (bytes)	typical size (bytes)
File	224 + 2 * fileName.length	250
Directory	264 + 2 * fileName.length	290
Block	152 + 72 * replication	368

- Conected with:
  - Clients
  - DataNodes
  - SecondaryNamenodes
    - Checkpointing
      - Editlogs a fsimage



HDFS filesystem

PROFINIT

- put
- get
- copyFromLocal
- ls
- Rights
  - Chmod
  - Chown
  - Chgrp

  - https://hadoop.apache.org/docs/r2.7.1/hadoop-project-dist/hadoop-hdfs/HDFSCommands.html

MapReduce

MapReduce

PROFINIT

- Paradigm / framework for distributed computation
- Consists of 2 phases/functions:
  - `map(key: object, value: object) -> Tuple[object, object]`
  - `reduce(key: object, values: List[object]) -> Tuple[object, object]`
- Example: Lets count the number of students signed up for a course

student	course
Alice	BDT
Bob	BDT
Cate	ABC

→ map

course	value
BDT	1
BDT	1
ABC	1

→ shuffle

course	values
BDT	[1, 1]
ABC	1

→ reduce

course	value
BDT	2
ABC	1

Inputs

PROFINIT

- Usually an input for a MapReduce job is a file/directory on HDFS
- The input is divided into data blocks of fixed sized called input splits
- For each split a map task is created
- Map tasks can run on the same or different machines allowing us to scale data pipeline as needed
- Hadoop tries to allocate a map task next to a data block of the input if possible (data locality principle)

Mapper

PROFINIT

- The mapper function is called once for the input row and it can generate any number of output key-value pairs (even none)
- The mapper function is stateless
- The intermediate results are sorted by key and written to local disk
- We can apply reduce operation on map-side to reduce amount of data transferred over the network, this operation is called **Combiner**

Shuffle

- There might be more than one map task that processed data with a specific key
- It is responsibility of the shuffle stage to make sure that all map-outputs with a specific key are delivered to a single reducer
- Each result file is partitioned and sorted before it is sent to a reducer

Reduce

- Transforms the output of shuffle stage to final result
- Reduce task download partial results files from mappers and merge-sort them before processing
- Number of reduce tasks is determined by the job author
- We can set the number of reduce tasks to 0

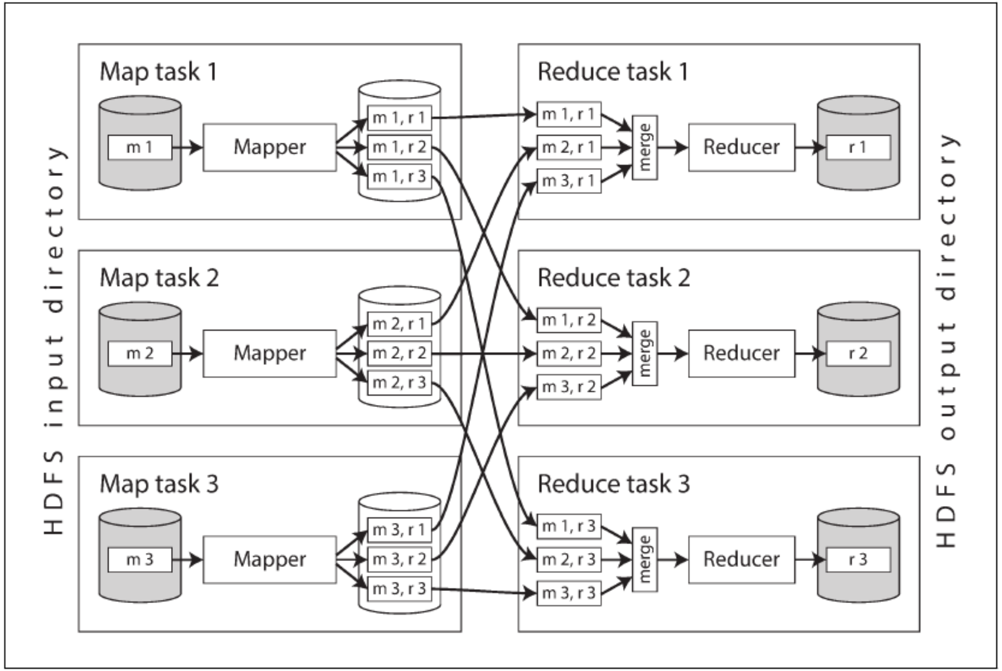
Parallelization

PROFINIT

- Reduce operations should be
  - Associative (A x B) x C = A x (B x C) (since values for the same key aren't sorted)
  - Neutral "zero" element should exist
  - Optionally: Commutative (A x B) = (B x A) (allows combine operations)

- Typical operations include
  - min, max
  - count, sum, multiplication
  - string concatenation
  - set union and intersection

MapReduce task anatomy



Storage

Data access patterns

PROFINIT

- OLTP
  - Online transaction processing is the heart of any IT application
  - Touches only a single (a very few) rows at the time
  - Typical operations: insert, update, delete
  - “As a user I want to place a new order with X, Y, Z items”

- OLAP
  - Online analytical processing
  - Allows business to answer a question “what is going on”
  - Touches large fraction of rows in a table (all or a specific subset/segment)
  - Typical operations: read
  - “What is the average sum a customer pays us monthly”

Data exchange formats

PROFINIT

- Plain text
  - XML
  - json
  - separated (CSV, TSV) etc.

- Binary
  - Various proprietary formats (Excel, pdf, protobuf)
  - Multimedia files (image, video, sound)

Data exchange formats considerations

PROFINIT

- Plain text
  - Self-contained and human-readable, no need for special software to understand/change the content ✓
  - Takes more space on disk compared to binary formats ✕

- Binary
  - Efficient use of disk space (data types and compression) ✓
  - Use of data types allows data validation and data integrity ✓
  - Is a must when performance is key ✓
  - Requires specialized tool to read/change ✕

Tabular data formats

PROFINIT

- Row-oriented
  - Faster writes
  - Works best for OLTP mode
- Columnar
  - Optimal reads
  - More efficient compression
  - OLAP

Traditional Row Based Storage

ID	Continent	City	Dept	Value
1	Europe	Paris	Sales	£500
2	USA	New York	Sales	£300
3	Europe	Paris	Sales	£700
4	Europe	London	Sales	£500
5	USA	New York	Sales	£200
6	Europe	London	Web	£100

Column Based Storage

ID	1	2	3	4	5	6
Continent	Europe	USA	Europe	Europe	USA	Europe
City	Paris	New York	Paris	London	New York	London
Team	Sales	Sales	Sales	Sales	Sales	Web
Value	£500	£300	£700	£500	£200	£100

Hadoop data formats

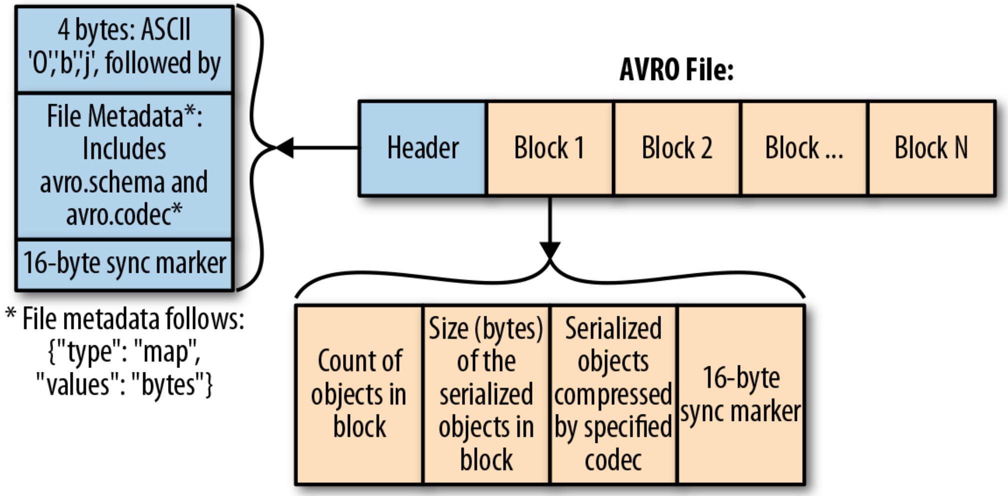
PROFINIT

- AVRO
  - Binary row format (+ json schema definition)
  - Has schema and schema evolution support
  - Kafka
- ORC
  - Binary columnar format
  - Well integrated into Hive (optimizations, data types, compression)
- **Parquet**
  - Binary columnar format
  - Schema is a part of file, nested objects are also supported
  - Has wide adoption and good performance on different workloads

Hadoop data formats

PROFINIT

- AVRO
  - Binary row format (+ json schema definition)
  - Has schema and schema evolution support
  - Supported in Kafka
  - Logo from British aircraft manufacturer
  - See Avro F.C. ☺



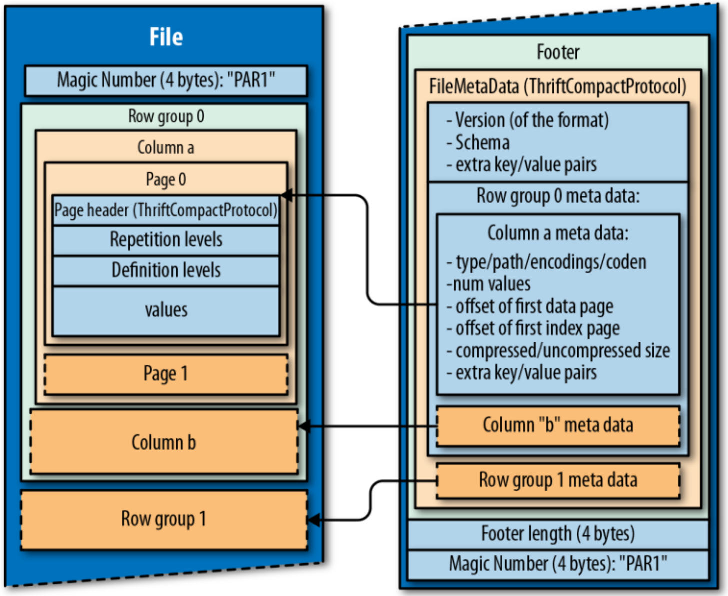
Hadoop data formats

PROFINIT

Parquet

- Binary columnar format
- Schema is a part of file, nested objects are also supported
- Has wide adoption and good performance on different workloads
- “Twitter/Cloudera” format
- Row group size 512-1024 MB

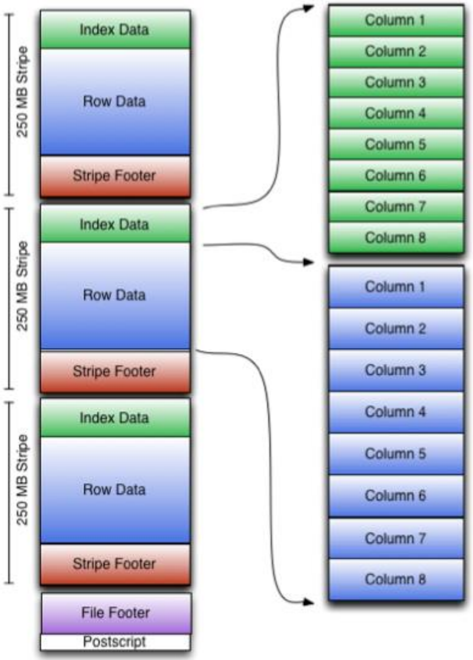
Parquet



Hadoop data formats

❮ PROFINIT ❯

- ORC
  - Binary columnar format
  - Well integrated into Hive (optimizations, data types, compression)
  - “Hortonworks” format
  - Group of row data - stripe
  - Stripe – 250MB (orc.stripe.size)
  - File footer
    - list of stripes, number or rows per stripe
    - Column level aggregates
  - Compression – snappy, zlib, none
  - You cannot use ORC in Impala



Small files trouble

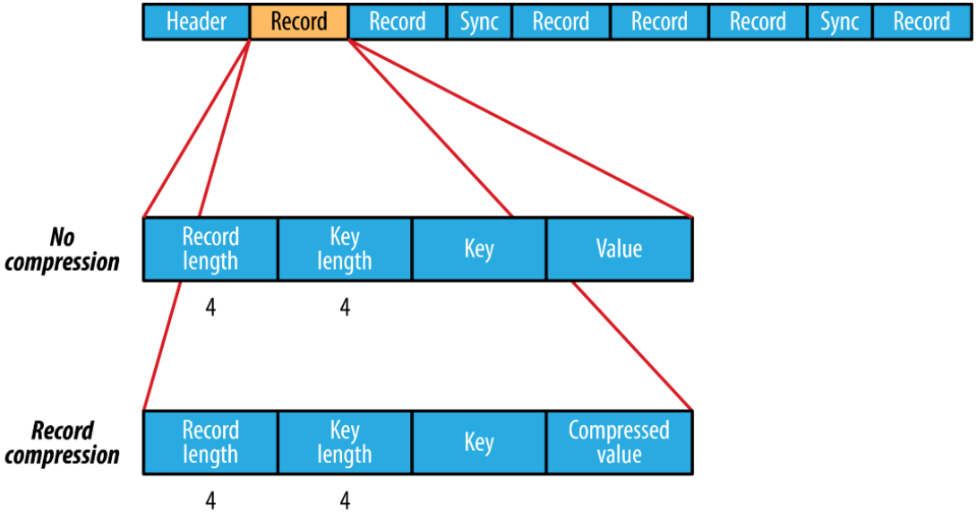
PROFINIT

- Many small files is a problem
  - small file typically hundreds of bytes to hundreds of kilobytes
- Typical situation - if I have to process small files, then I can have a lot of them (energetic company example)
- How many ? 10M +
- 1 block record on HDFS about 200 bytes in RAM on NameNode
  - Example 1 10kB file
  - 10E6 files 100GB of data approx 2GB of RAM on NameNode
  - 1E9 files of 10TB data approx 200GB RAM
- Typical solution – sequence file

Hadoop data formats

PROFINIT

- **RCFile**
  - Older, not used
  - ORC is a successor
- SequenceFile
  - Good when you have many increments
  - Supports append
  - Block compression
  - Good for fullscan
  - Supports key/value
    - Key – filename
    - Value – file itself



Compression

PROFINIT

- Motivating example:
  - How long does it take to read 100GB table in 10 nodes cluster (each node has 4 disk with peak read speed 100MB/s)?
- Helps us to trade I/O time for CPU time
- Practical considerations is tradeoff between compression coefficient and compression/decompression speed
  - Gzip – very efficient in terms of compression, but is relatively slow
  - Snappy – good balance between compression efficiency and speed

Comparison

PROFINIT

Algorithm	Speed	Effectivity	„Splittable“
GZIP/ZLib		✔	
BZip2		✔	✔
LZO	✔		✔
Snappy	✔		

- „Splitable“
  - Create blocks that can be decompressed independently
- Compatibility
  - **Not every tool can read/write everything!**

YARN

YARN

PROFINIT

- **Y**et **A**nother **R**esource **N**egotiator
- e.g. Resource Manager
  - **RAM**
  - **CPU**
  - Number of threads
  - Network...
- Not all application used YARN, e.g.. Impala has it‘s own
  - Every resource manager should have dedicated resources

YARN

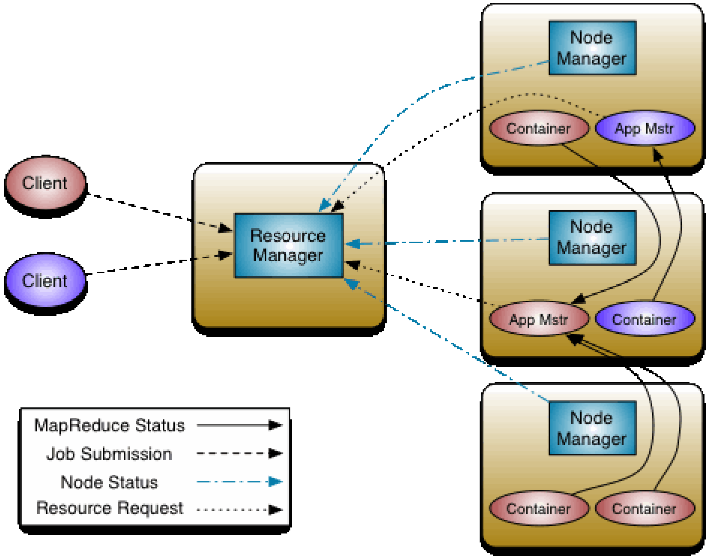
PROFINIT

- Application – client application
- Container – resources allocated to application on a defined node
- Resource Manager – global resource manager for a whole cluster
  - Scheduler - responsible for allocating resources
  - ApplicationsManager -responsible for accepting ApplicationMaster
- Node Manager – resource manager (launching and managing container) for a defined node
- Application master - negotiating resources from the ResourceManager and NodeManager

YARN

PROFINIT

- Flow
  - Job Submission
  - Resource request for application master
  - Start AM container
  - Resource request for „working“ containers
  - Start „working“ containers



YARN

PROFINIT

Apps | Copyright in sports | Jobs | Python List (With Ex | enable_acid1 | Solved: Re: How to | Python Pandas Tuto | Python Pandas Tuto | Why Should You Lea

trics for dr.who

ps itted	Apps Pending	Apps Running	Apps Completed	Containers Running	Containers Pending	Containers Reserved	Memory Used	Memory Pending	Memory Reserved	VCores Used	VCores Pending	VCores Reserved
	0	0	0	0	0	0	0 B	0 B	0 B	0	0	0

entries | Search:

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	Reserved CPU VCores	Reserved Memory MB	Progress	Tracking UI
on_1528804284572_0005	Admin	Hive on Spark	SPARK	root.users.Admin	Tue Jun 12 17:38:23 +0550 2018	N/A	ACCEPTED	UNDEFINED	0	0	0	1	2048		UNASSIGNED
on_1528804284572_0003	Admin	Hive on Spark	SPARK	root.users.Admin	Tue Jun 12 17:36:35 +0550 2018	N/A	RUNNING	UNDEFINED	1	1	2048	4	5632		ApplicationMaster
on_1528804284572_0004	Admin	Hive on Spark	SPARK	root.users.Admin	Tue Jun 12 17:37:30 +0550 2018	N/A	RUNNING	UNDEFINED	1	1	2048	4	5632		ApplicationMaster
on_1528804284572_0001	Admin	Hive on Spark	SPARK	root.users.Admin	Tue Jun 12 17:24:50 +0550 2018	N/A	RUNNING	UNDEFINED	3	9	13312	0	0		ApplicationMaster
on_1528804284572_0002	Admin	Hive on Spark	SPARK	root.users.Admin	Tue Jun 12 17:25:33 +0550 2018	N/A	RUNNING	UNDEFINED	3	9	13312	0	0		ApplicationMaster

1 to 5 of 5 entries

First Previous 1 Next Last

YARN

PROFINIT

- `yarn application`
- `yarn container`
- `yarn logs`
- `yarn node`

https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YarnCommands.html

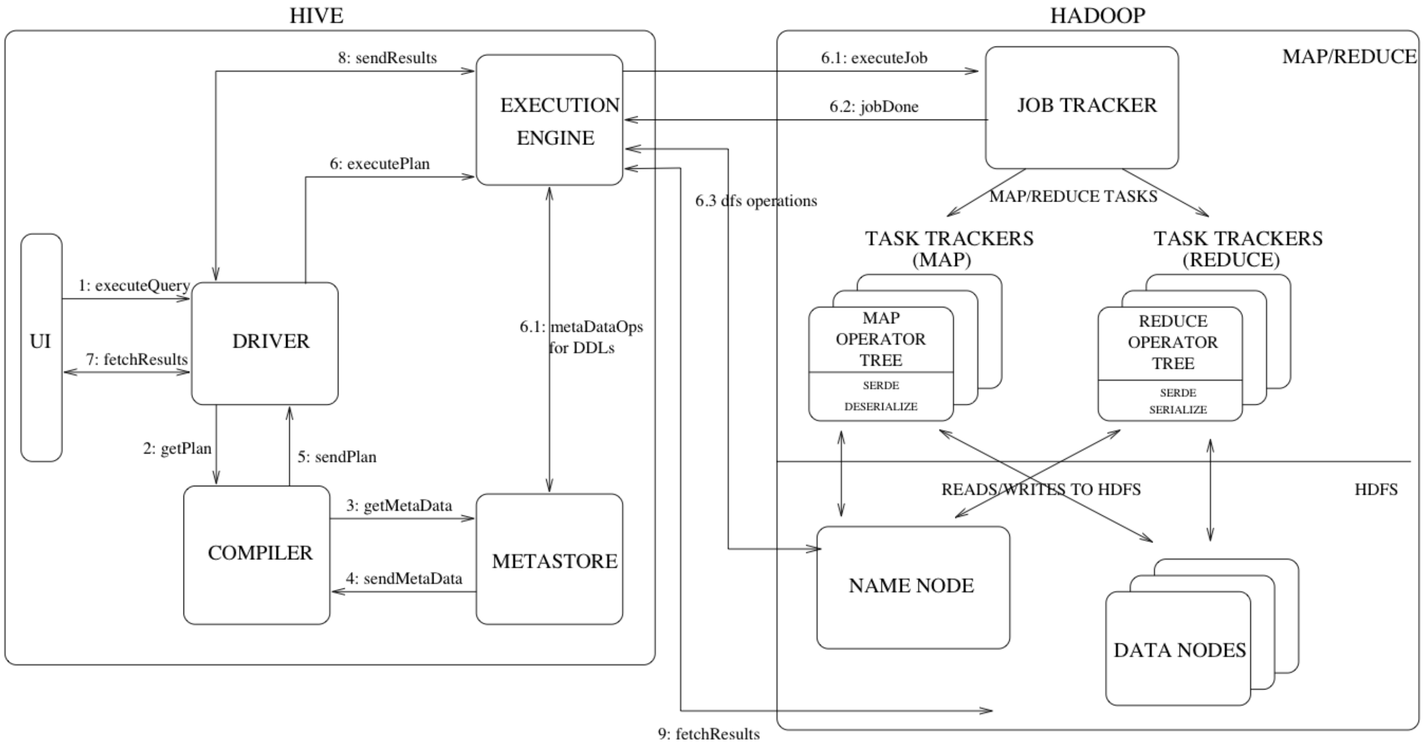
Hive

Hive

- MapReduce is a big step towards easier distributed computation, but requires a lot of coding in Java even for simple counting
- SQL is *lingua franca* for data analytics
- Apache Hive is a SQL-engine built on top of MapReduce

High-level architecture

PROFINIT



Hive - data

PROFINIT

- Data is organized into tables stored on HDFS
  - Table's data files are stored in a HDFS directory
  - Schema on read – schema is checked during the query
- A table is metadata stored in the metastore. Metastore contains:
  - Table schema
  - Table data location and format
  - Custom attributes
  - Table statistics

Hive compared to relational DBs

- Schema-on-read
- Indexing is not supported*
- Limited support for transactions and isolation
- Materialized views are not supported

**initial design had flaws, usage was discouraged and indexing was removed in Hive 3.0*

Hive - HQL

PROFINIT

- DDL (Data Definition Language)
  - `CREATE [EXTERNAL] TABLE`
  - `DROP TABLE, TRUNCATE TABLE, ALTER TABLE`
- DML (Data Manipulation Language)
  - `LOAD DATA, INSERT INTO TABLE, INSERT OVERWRITE TABLE`
- Query
  - `SELECT`
- Limited support*
  - `UPDATE`
  - `DELETE`

**supported only in transactional tables*

Hive – loading data

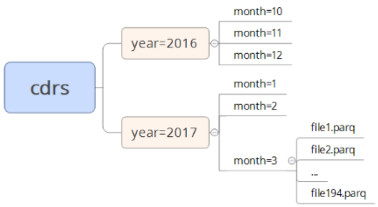
PROFINIT

- Hive defines multiple ways to load data
- Load data from HDFS using LOAD DATA statement
  - HDFS cp/mv operations, schema is not checked during load
- Insert query results into a table using INSERT INTO table select * from tbl
- Inserting literal values using INSERT INTO table values (1, 2, 3)
  - Least efficient way to insert values into a Hive, use this only for testing
  - Every insert statement will create a single (small) file

Partitioning & Bucketing

PROFINIT

- Partitioning – a way to organize data into smaller chunks
  - Logical and physical separation
  - Can speed-up some queries
  - Simplify governance



- Design partitioning schema with the data volume in mind, we do not want to have too many small files
  - If we don’t have enough data, daily partitioning might not be very efficient
- Bucketing – additional layer of organization data into files by using hash function applied on bucketed column.
  - We can make sure that rows with the same bucketing key will be in the same file

Execution engines

PROFINIT

- MapReduce
- Hive on Tez
  - Optimized query execution that avoid some limitations of MapReduce
  - Eliminate unnecessary stages and HDFS writes
- Hive on Spark
  - Another approach onto speeding up MapReduce jobs by translating it into Spark jobs
- LLAP
  - Live Long And Process daemons for small/short queries

Hive example workflow

PROFINIT

1. Raw data is delivered to front-end server
2. Raw data is copied to HDFS folder for raw data
3. An external table is registered in Hive
4. An internal table is created in Hive (optimized file format)
5. Data is transformed and inserted into internal table

External table

PROFINIT

```
CREATE EXTERNAL TABLE IF NOT EXISTS ap_temp (
   ACC_KEY BIGINT,
   BUS_PROD_TP_ID VARCHAR(255),
   START_DATE TIMESTAMP,
   BUS_PROD_TP_DESCR VARCHAR(255)
)
ROW FORMAT
   DELIMITED FIELDS TERMINATED BY '~'
   LINES TERMINATED BY '\n'
STORED AS TEXTFILE
LOCATION '/data/input/acc';
```

Internal table

PROFINIT

```
CREATE TABLE IF NOT EXISTS ap (
   ACC_KEY BIGINT,
   BUS_PROD_TP_ID VARCHAR(255),
   START_DATE TIMESTAMP)
PARTITIONED BY (BUS_PROD_TP_DESCR VARCHAR(255))
CLUSTERED BY (ACC_KEY) INTO 32 BUCKETS
STORED AS ORC tblproperties ("orc.compress"="ZLIB");
```

Insert data

PROFINIT

```
INSERT OVERWRITE TABLE ap
PARTITION (BUS_PROD_TP_DESCR)
SELECT
 ACC_KEY,
 BUS_PROD_TP_ID,
 START_DATE,
 BUS_PROD_TP_DESCR
FROM ap_temp;


DROP TABLE ap_temp;
```

Hive catalog

PROFINIT

- Hive defines statements that return information about databases and tables
  - show databases
  - show tables <db_name>
  - show create table <table>
  - show partitions <table>
  - describe <table>
  - show columns from <table>

Hadoop Distribution Components

Hadoop distribution

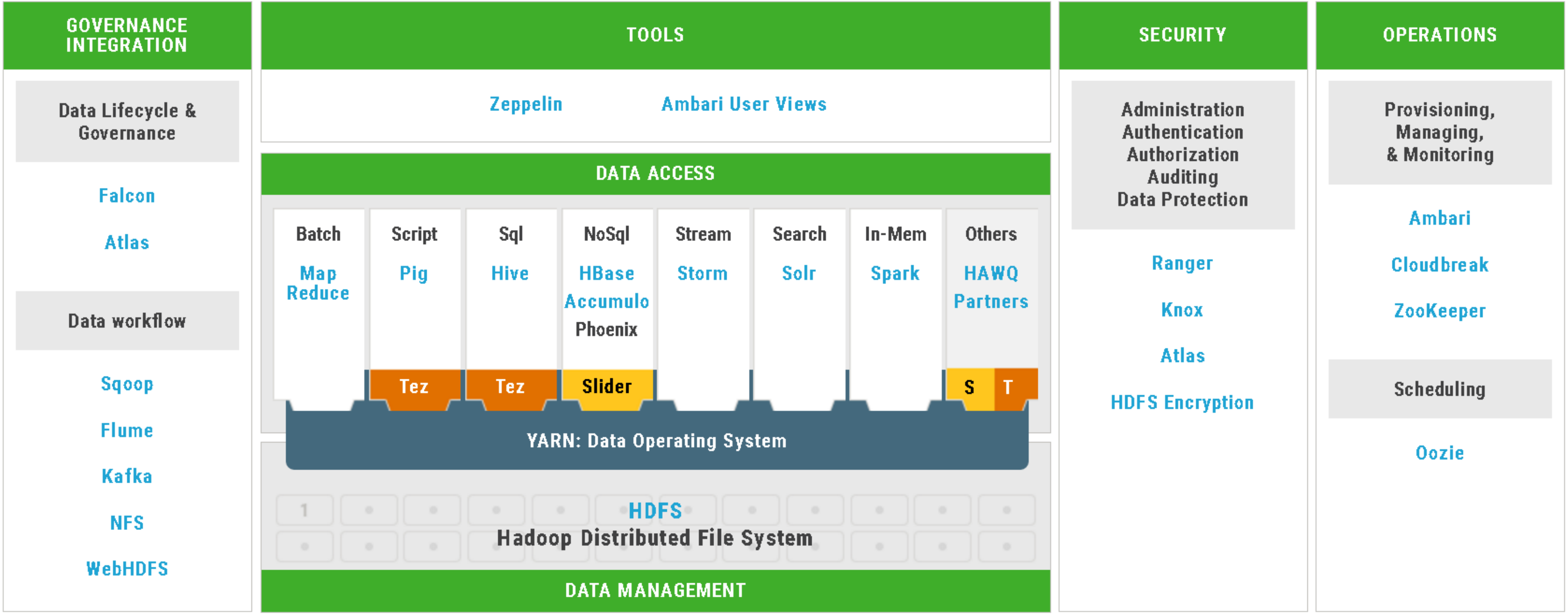
cloudera

~~HORTONWORKS~~®

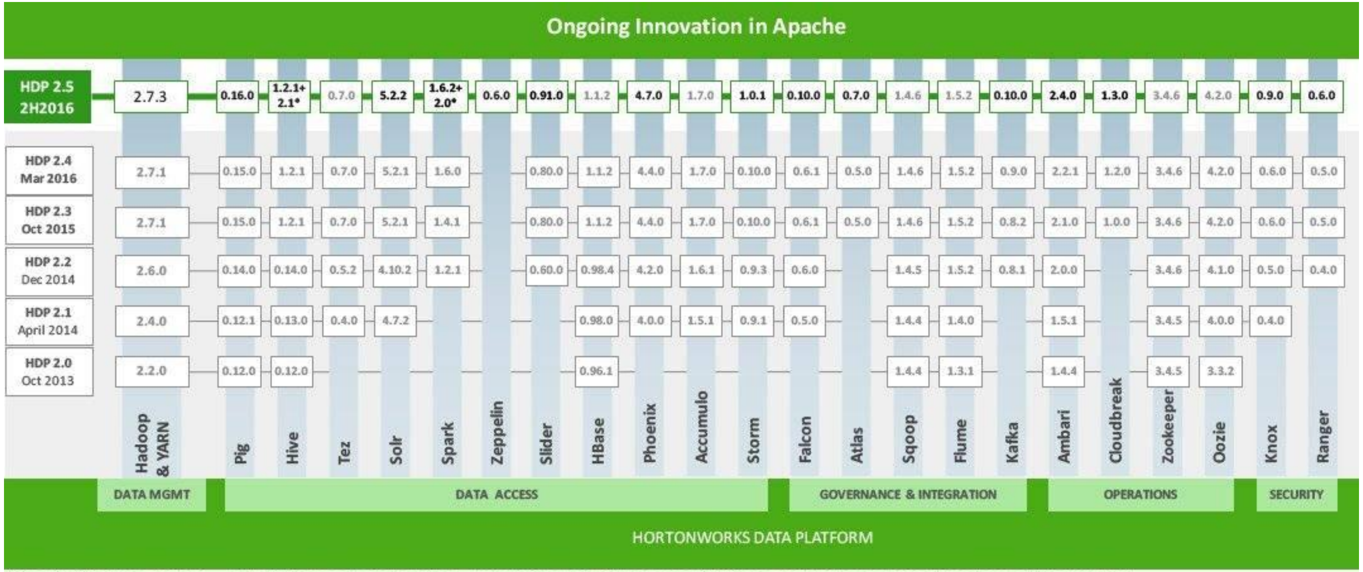
MAPR

Hadoop ZOO

PROFINIT



Component ZOO versions



Ongoing Innovation in Apache

	Hadoop & YARN	Pig	Hive	Tez	Solr	Spark	Zeppelin	Slider	HBase	Phoenix	Accumulo	Storm
HDP 2.5 2H2016	2.7.3	0.16.0	1.2.1+ 2.1*	0.7.0	5.2.2	1.6.2+ 2.0*	0.6.0	0.91.0	1.1.2	4.7.0	1.7.0	1.0.1
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 Oct 2015	2.7.1	0.15.0	1.2.1	0.7.0	5.2.1	1.4.1		0.80.0	1.1.2	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 April 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
HDP 2.0 Oct 2013	2.2.0	0.12.0	0.12.0						0.96.1			

	Falcon	Atlas	Sqoop	Flume	Kafka	Ambari	Cloudbreak	Zookeeper	Oozie	Knox	Ranger
HDP 2.5 2H2016	0.10.0	0.7.0	1.4.6	1.5.2	0.10.0	2.4.0	1.3.0	3.4.6	4.2.0	0.9.0	0.6.0
HDP 2.4 Mar 2016	0.6.1	0.5.0	1.4.6	1.5.2	0.9.0	2.2.1	1.2.0	3.4.6	4.2.0	0.6.0	0.5.0
HDP 2.3 Oct 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 April 2014	0.5.0		1.4.4	1.4.0		1.5.1		3.4.5	4.0.0	0.4.0	
HDP 2.0 Oct 2013			1.4.4	1.3.1		1.4.4		3.4.5	3.3.2		

DATA MGMT | DATA ACCESS | GOVERNANCE & INTEGRATION | OPERATIONS | SECURITY

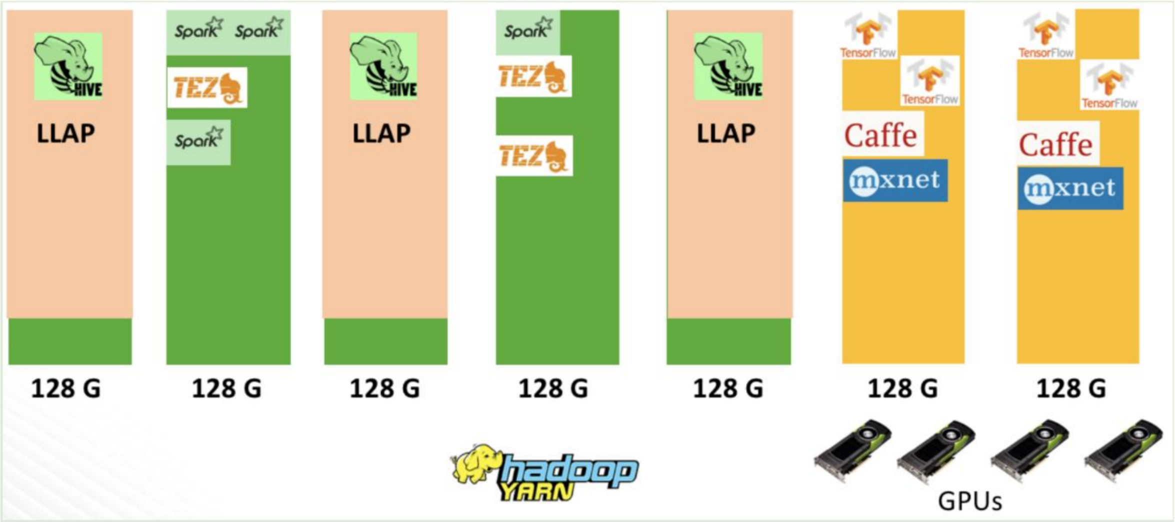
HORTONWORKS DATA PLATFORM

** Spark 1.6.2+ Spark 2.0 – HDP 2.5 support installation of both Spark 1.6.2 and Spark 2.0. Spark 2.0 is Technical Preview within HDP 2.5.*
Hive 1.2.1+ Hive 2.1 – Hive 2.1 is Technical Preview within HDP 2.5.

Hadoop Platform Enhancement

GPUs

PROFINIT



With GPU scheduling support, YARN can allocate non-GPU applications (like LLAP / Spark / Tez, etc.) to machines without GPU (left side), and allocate GPU applications (like Tensorflow / Caffe / MXNet) to machines with GPU. (right side)

Apache Ozone

- Distributed key-value store
- Can manage both small and large files alike
- Separates the namespace management from block and node management layer
- Possible deployment with HDFS
- Multi-protocol support (S3 API, HDFS API, …)

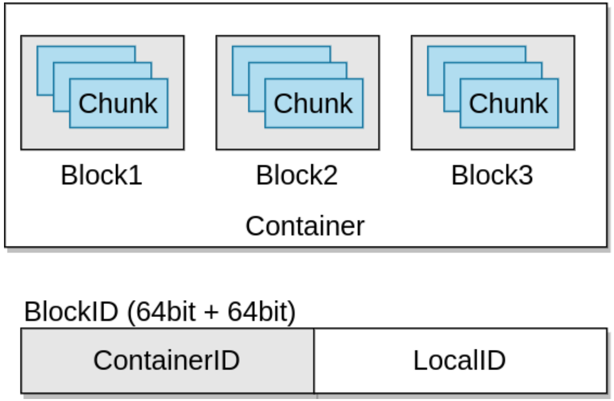
Apache Ozone

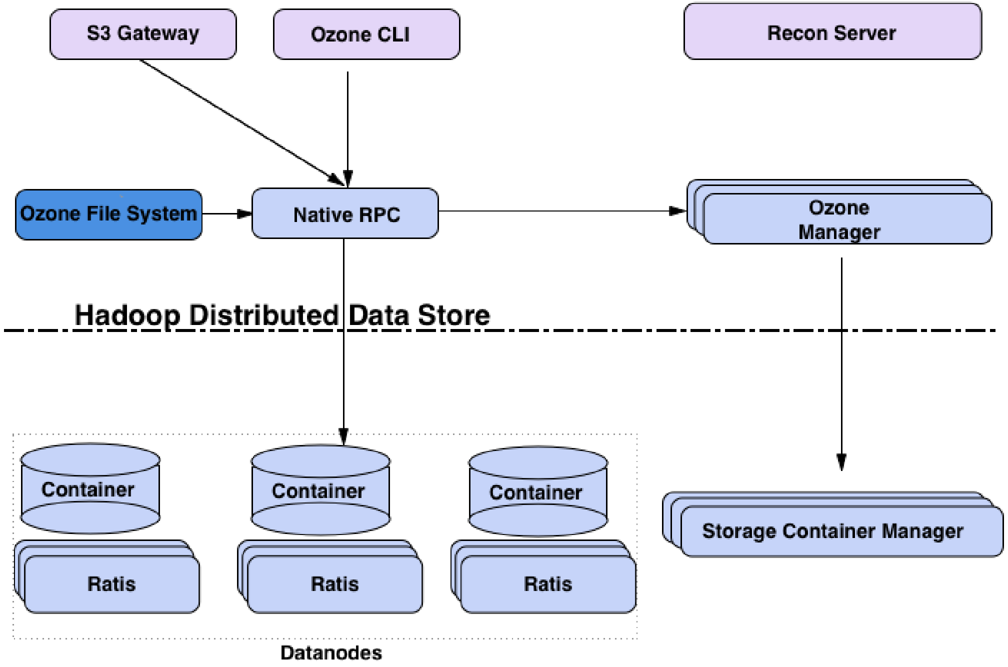
PROFINIT

- Ozone Manager - is the namespace manager (relation between file and container)
- Storage Container Manager - is the leader node of the *block space management* (create and manage containers)
- Containers - big binary units (5Gb by default) which can contain multiple blocks

OPEN	CLOSED
mutable	immutable
replicated with RAFT (Ratis)	Replicated with async container copy
Raft leader is used to READ / WRITE	All the nodes can be used to READ

Apache Ozone





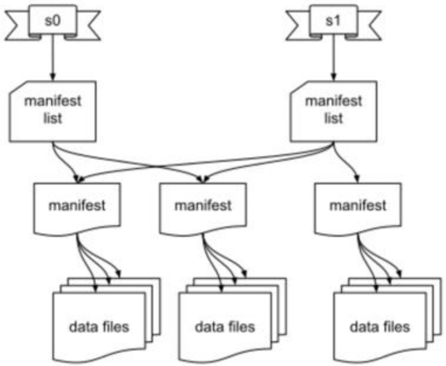
Apache Iceberg

- The open table format for analytic datasets.
- Table format that helps simplify data processing on large datasets stored in data lakes.
  - Full Schema Evolution
  - Expressive SQL
  - Hidden Partitioning
  - Time Travel and Rollback
  - Data Compaction

Apache Iceberg

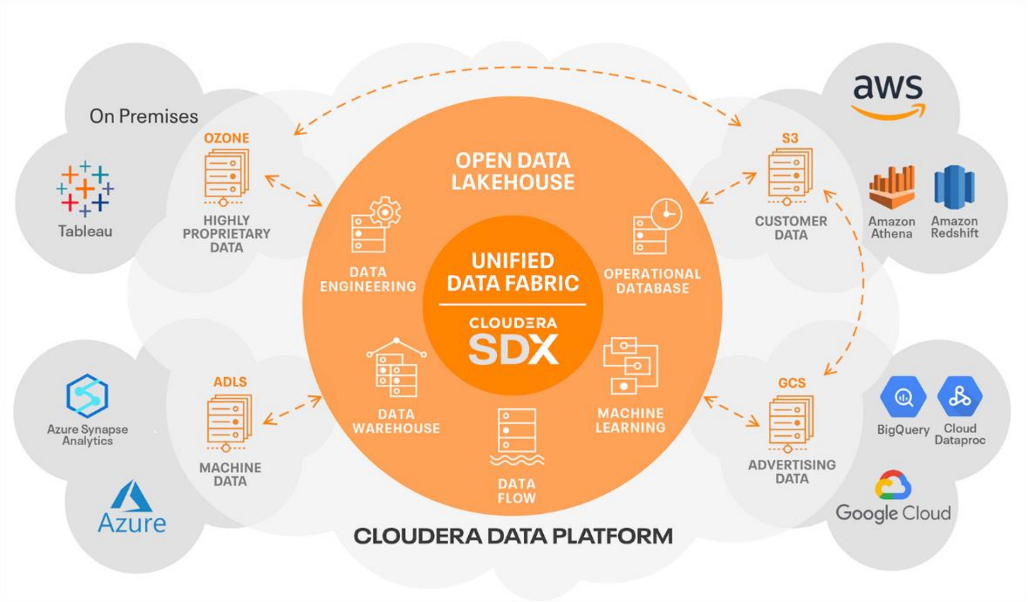
PROFINIT

- Snapshot metadata file - contains metadata about the table (data) at a point in time
- Manifest list - contains an entry for each manifest file associated with the snapshot
- Manifest file - contains a list of paths to related data files. Each entry for a data file includes some metadata about the file
- Data file - the physical data file, written in formats like Parquet, ORC, Avro etc



CDP – Cloudera Data Platform

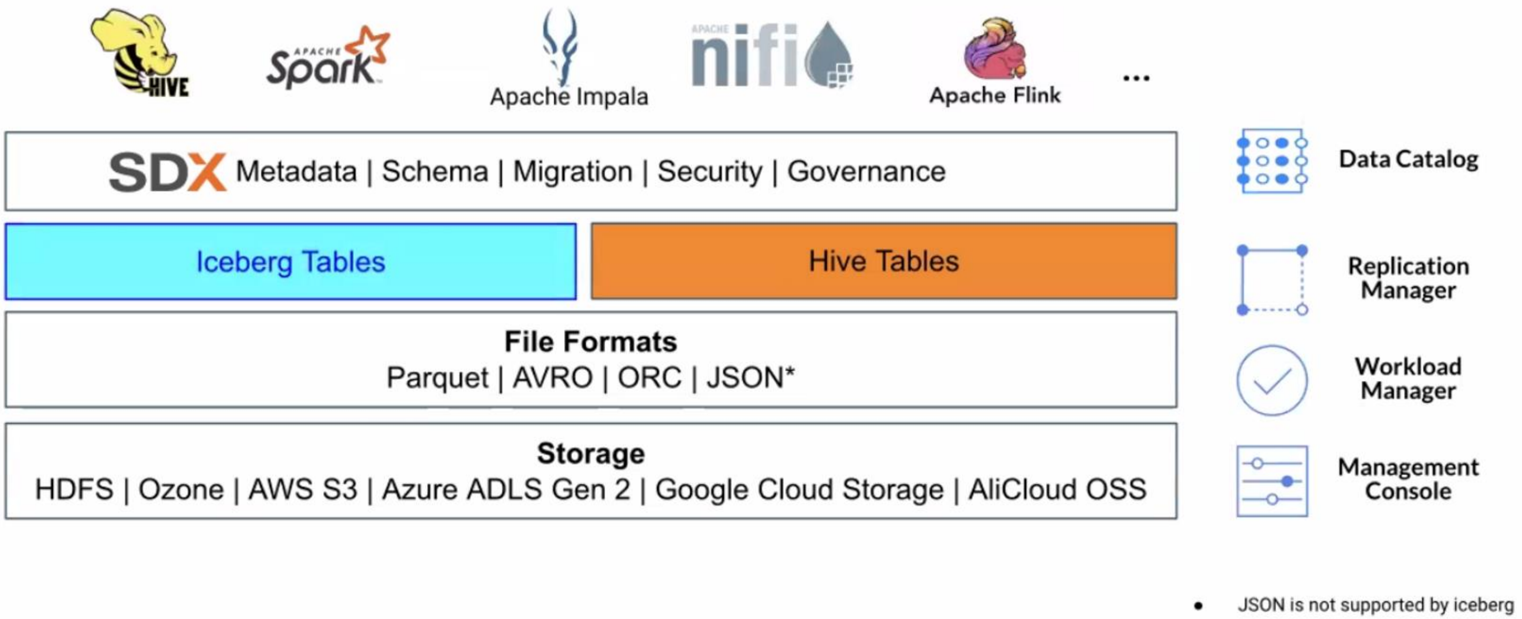
PROFINIT



https://www.cloudera.com/products/open-data-lakehouse.html

Cloudera CDP

PROFINIT



https://www.cloudera.com/products/open-data-lakehouse.html

Cloud?

Cloud and Big Data

PROFINIT

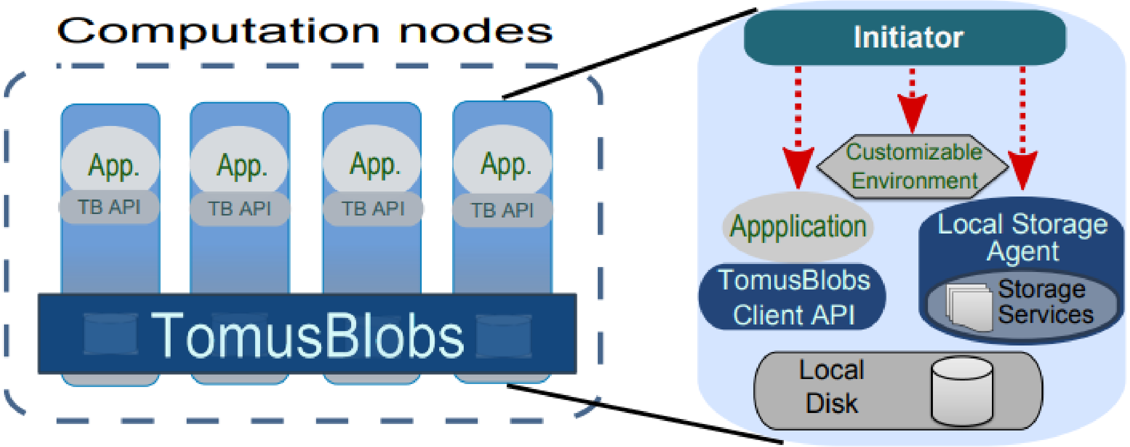
- Advantages
  - Multi-site oriented infrastructure
  - The on-demand scaling
  - Infinitely scalable infrastructure
  - Various tools and libraries

- Issues
  - The computational nodes are separated from the storage nodes
  - High latency between computation units
  - I/O throughput

TomusBlobs

PROFINIT

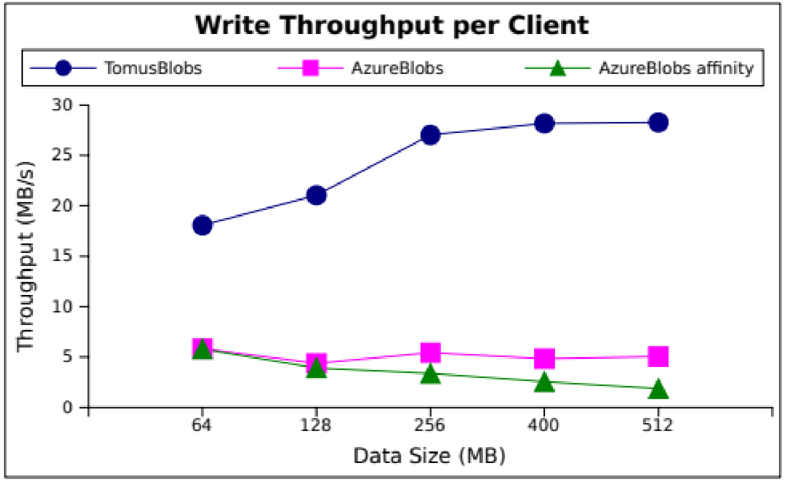
- Federating Virtual Disks

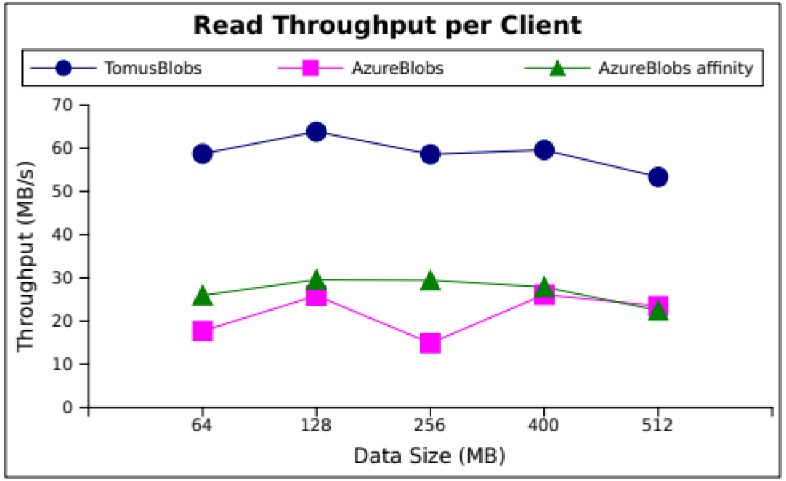


TomusBlobs

PROFINIT

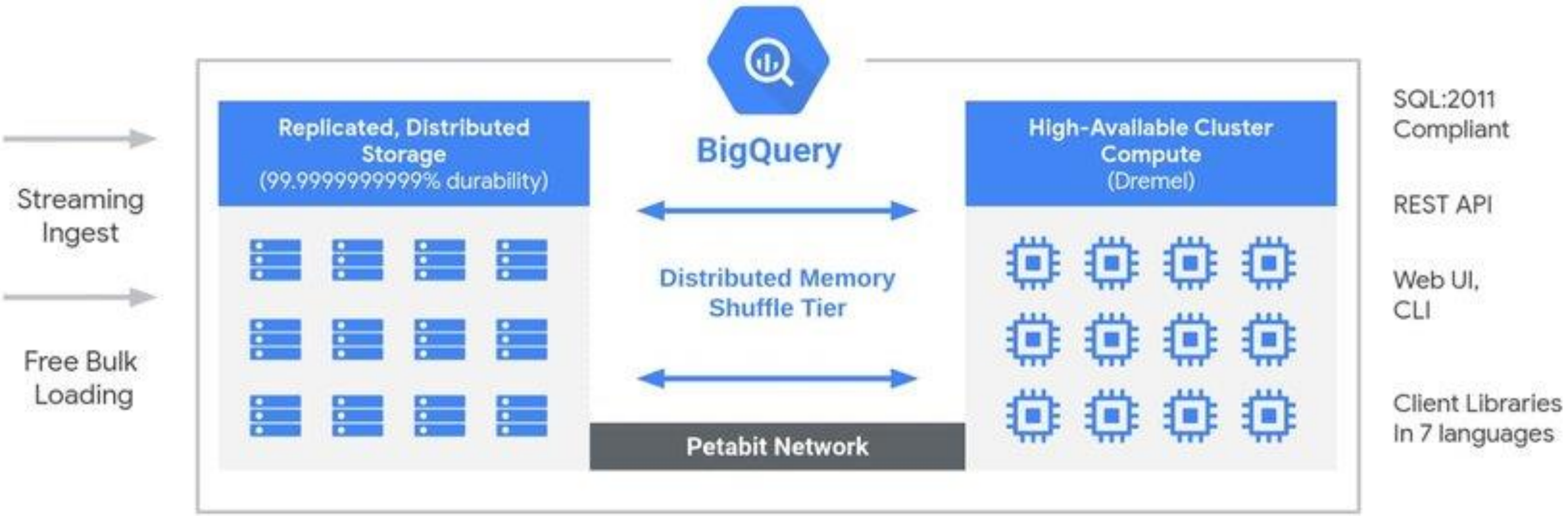
- Performance





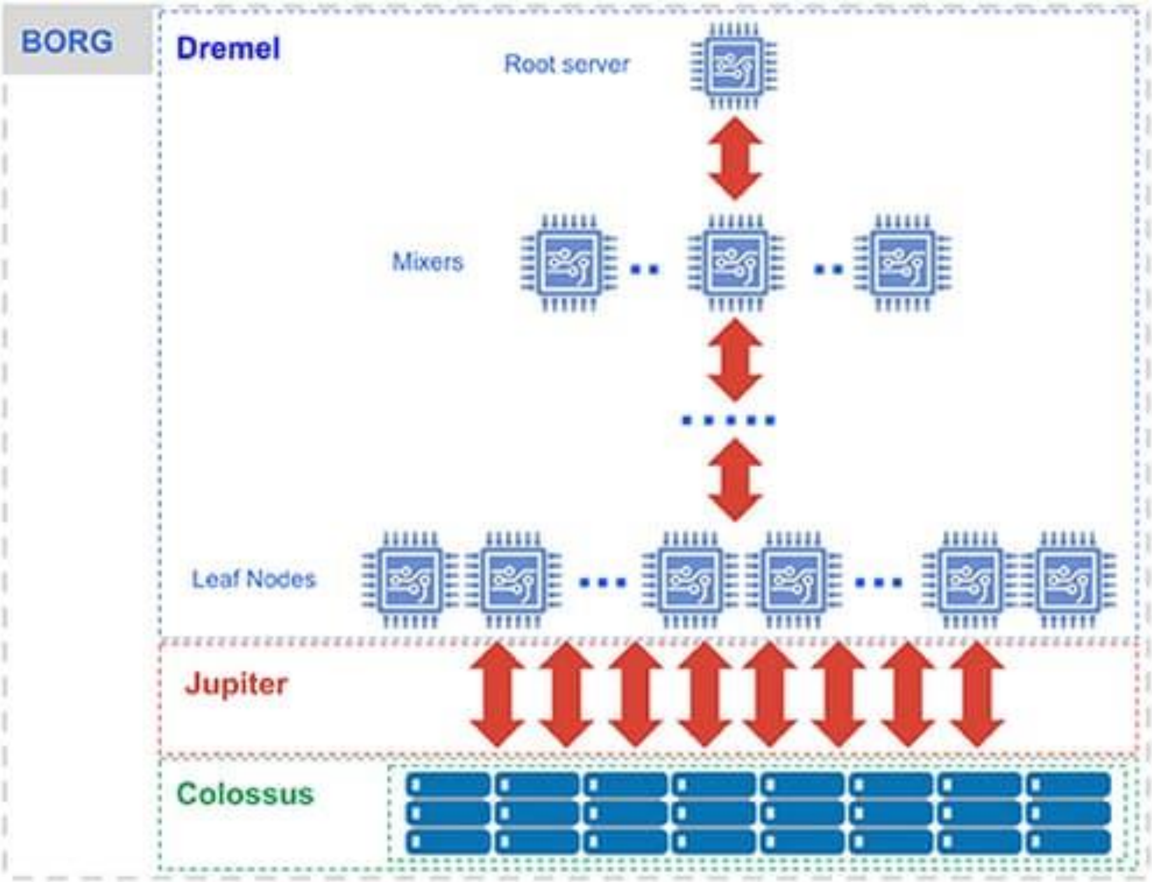
Radu Tudoran. High-Performance Big Data Management Across Cloud Data Centers. Computer science. ENS Rennes, 2014. English. ffNNT : ff. fftel-01093767v1

Google Big Query



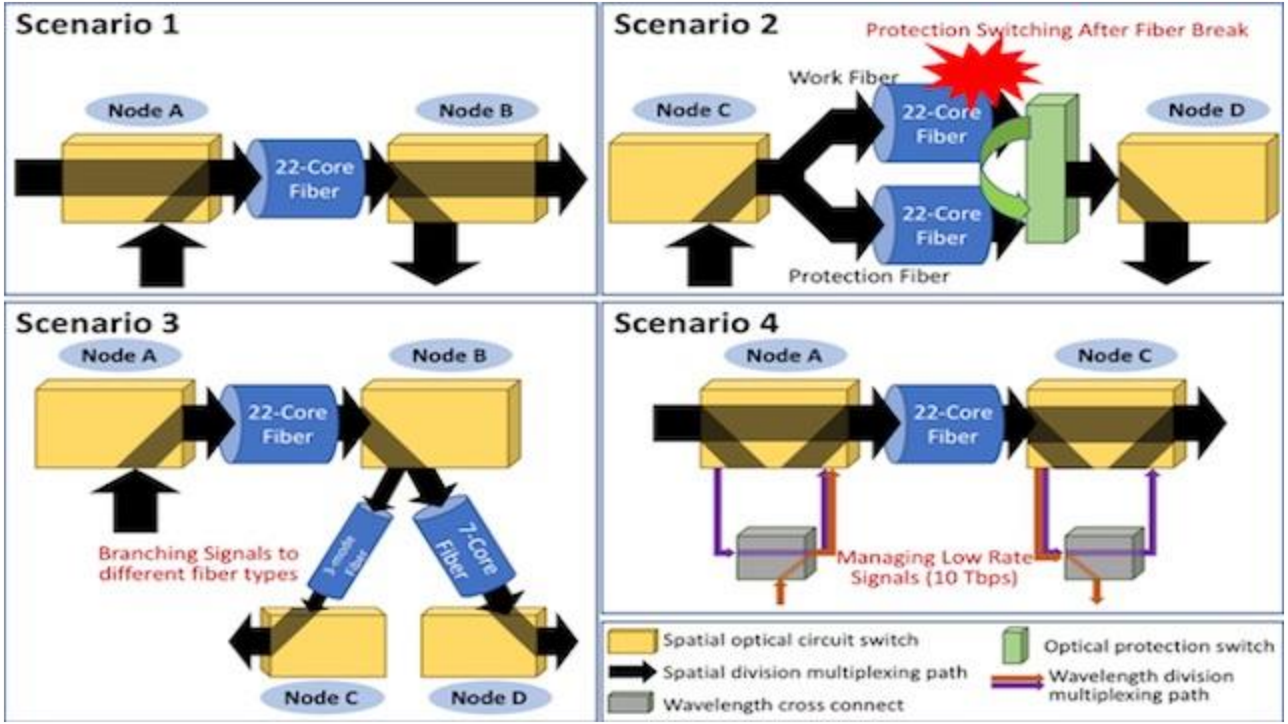
Google Big Query

PROFINIT



Petabit Network

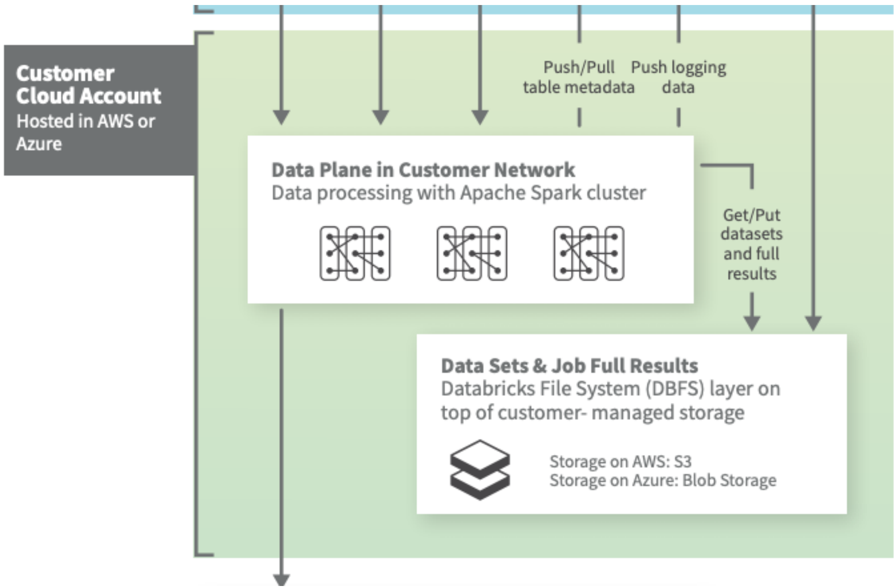
PROFINIT

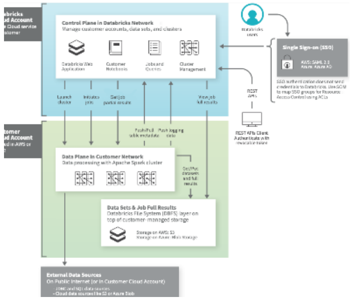


https://incompliancemag.com/first-successful-demonstration-of-a-1-petabit-per-second-network-node/

Databricks

- Cloud object storage (DataLake Parquets)
- Data are cached to local disk during processing

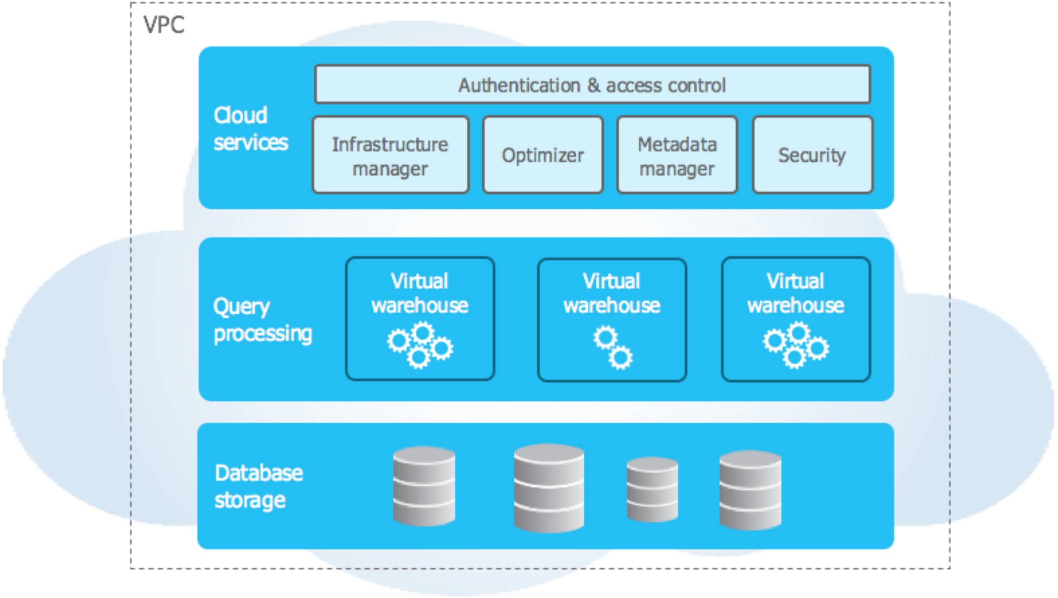




Snowflake

PROFINIT

- Cloud object storage
  - Accessible only using Snowflake
  - Virtual warehouses
    - Dedicated Nodes
      - CPU
      - RAM
      - Storage



Díky za pozornost

Profinit EU, s.r.o.
Tychonova 2, 160 00 Praha 6

Tel.: + 420 224 316 016, web: www.profinit.eu

LINKEDIN
linkedin.com/company/profinit

TWITTER
@profinit_EU

FACEBOOK
facebook.com/Profinit.EU

YOUTUBE
Profinit EU, s.r.o.