ISA 414 – Managing Big Data

Lecture 22 – Introduction to Hadoop

(Part II)

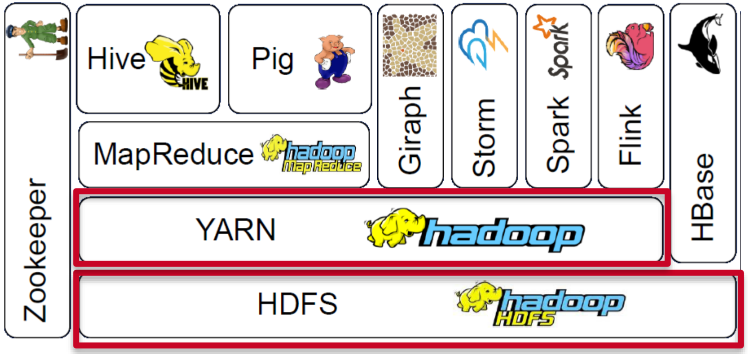
Dr. Arthur Carvalho

arthur.carvalho@miamioh.edu

MIAMI UNIVERSITY

Lecture Objectives

- Understand the inner workings of Hadoop
  - HDFS



Lecture Instructions

- There are several new concepts today
    - Suggestion: actively take notes
    - Important keywords are highlighted in the slides

Hadoop

- Summary of previous lecture
  - Handling massive amounts of data is one of the biggest challenges brought by big data
    - How to <u>store</u> massive amounts of data?
      - Cheap solution: **commodity clusters** and **distributed storage**
        - Individual files are stored in different computers in a cluster
        - Massive files are broken down into chucks of data, which are stored in different computers (nodes) in a cluster
        - A **distributed file system** tracks where each piece of data is located

Hadoop

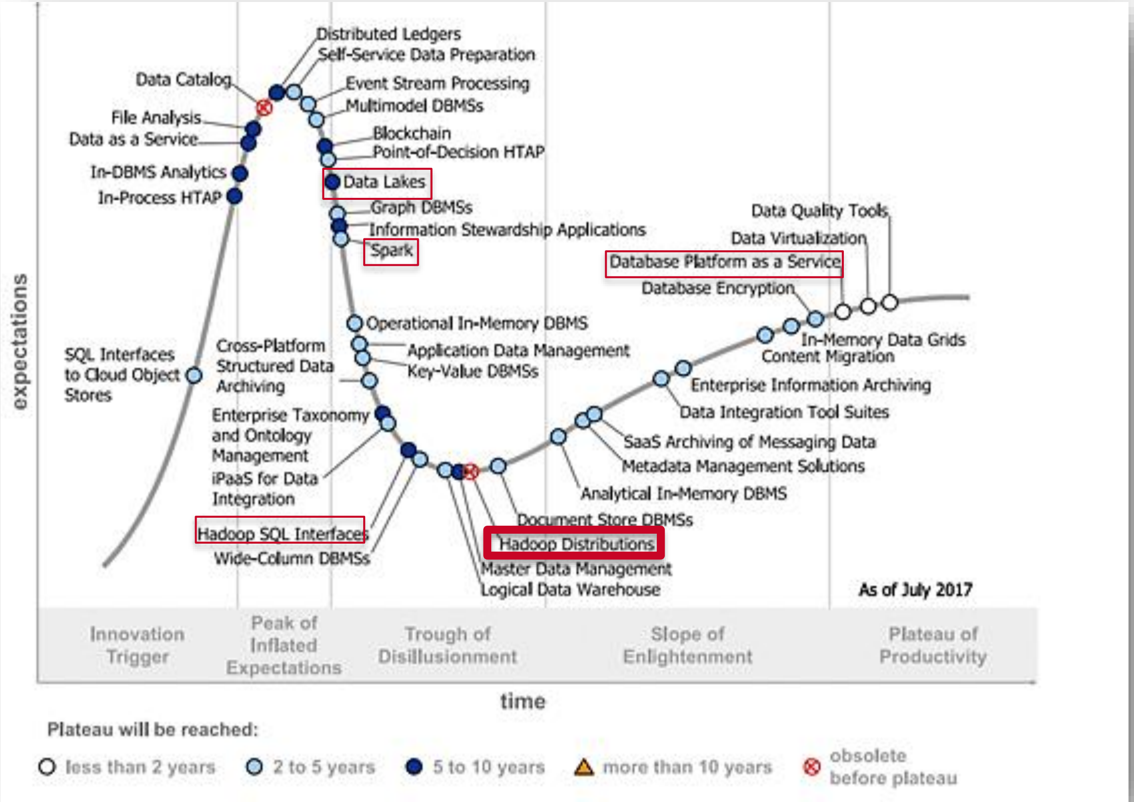
- Summary of previous lecture
  - Handling massive amounts of data is one of the biggest challenges brought by big data
    - How to analyze (perform computations using) big data?
      - Cheap solution: **commodity clusters** and **distributed computation**
        - Paradigm: “**move computation to data**”

MIAMI UNIVERSITY

Hadoop

- Summary of previous lecture
  - **Hadoop**: framework used for distributed storage and computing
    - *I.e.*, a tool that manages commodity clusters
  - A collection of technologies
    - From data storage to data collection and analysis, these technologies might have drastically different roles
  - Many of the Hadoop technologies are still part of Gartner's Hype Cycle
    - https://blogs.gartner.com/andrew_white/2020/08/07/data-and-analytics-hype-cycles-for-2020-just-published/

Hadoop



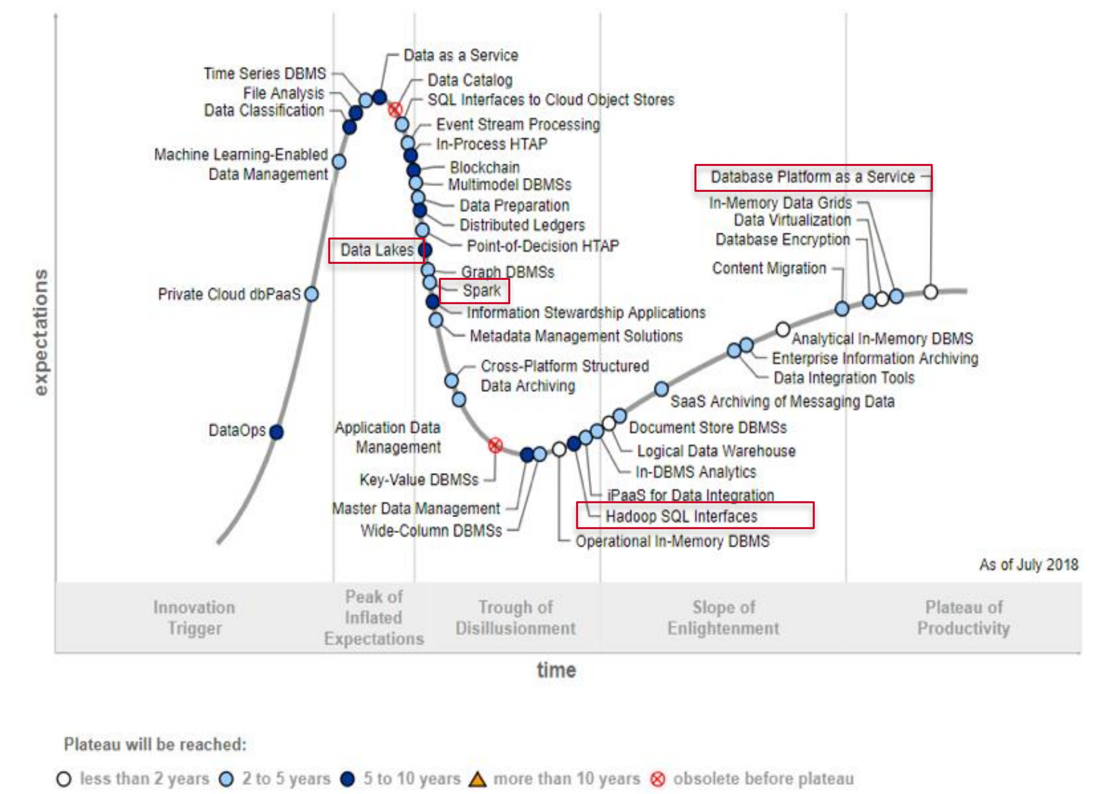
Gartner 2017:

Hype Cycle for **Data Management**

(source: https://www.gartner.com/newsroom/id/3809163)

MIAMI UNIVERSITY

Hadoop

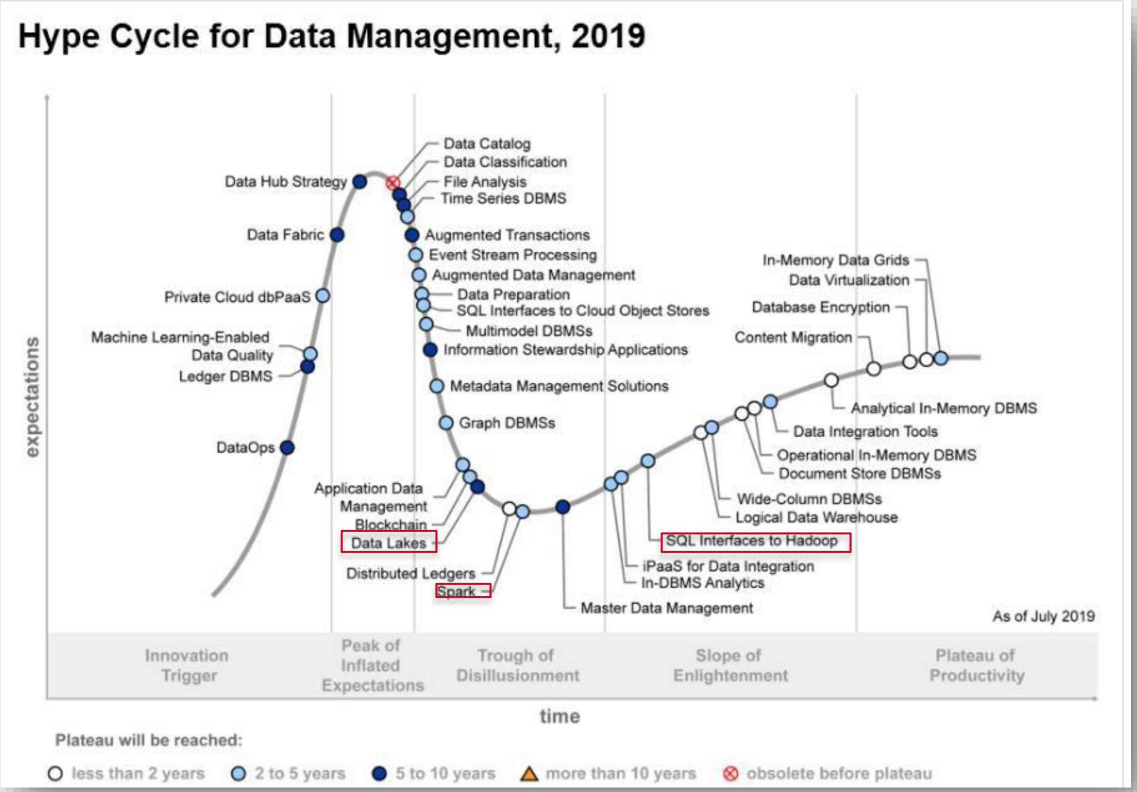


Gartner 2018:

Hype Cycle for **Data Management**

(source: https://www.gartner.com/en/newsroom/press-releases/2018-09-11-gartner-hype-cycle-for-data-management-positions-three-technologies-in-the-innovation-trigger-phase-in-2018)

Hadoop

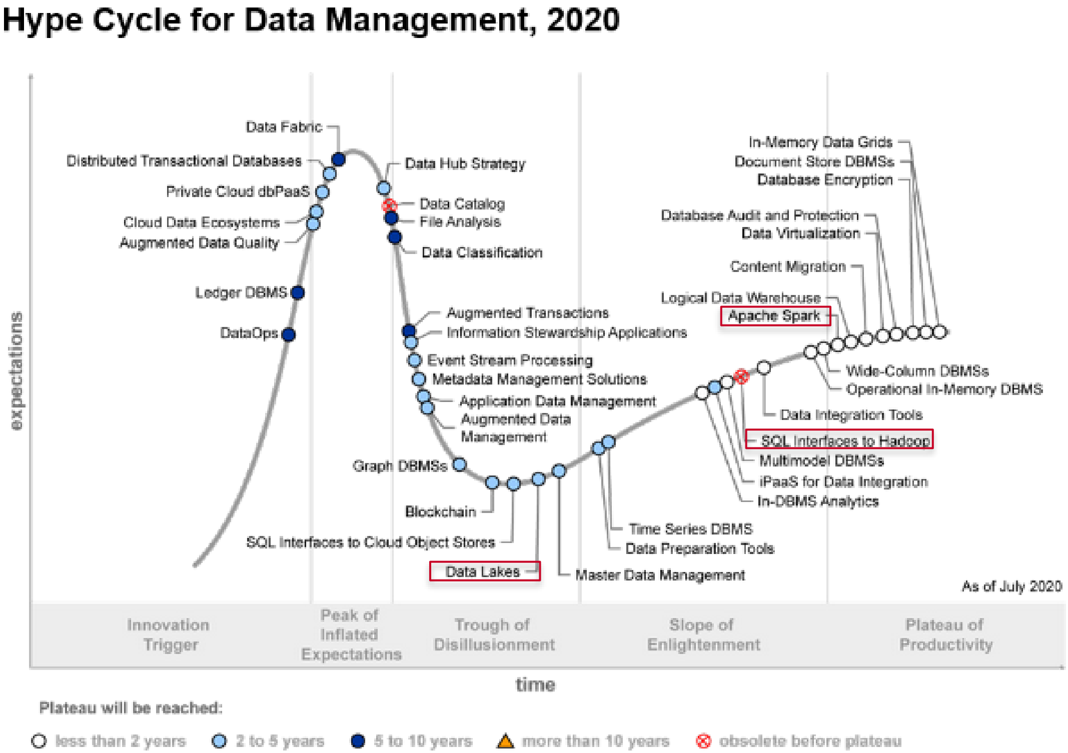


Gartner 2019:

Hype Cycle for **Data Management**

(source: https://www.gartner.com/en/documents/3955768/hype-cycle-for-data-management-2019)

Hadoop



Gartner 2020:

Hype Cycle for **Data Management**

(source: https://www.denodo.com/en/document/analyst-report/gartner-hype-cycle-2020)

MIAMI UNIVERSITY

HDFS

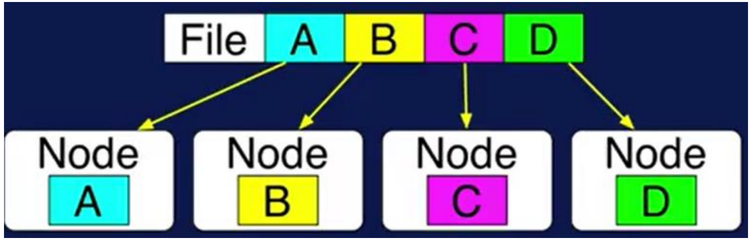
- **Hadoop Distributed File System** (HDFS)
  - Remember: **distributed storage** requires a **distributed file system**
    - That is what **HDFS** is
  - Main idea: HDFS breaks down large files into file blocks and spread them across multiple computers in a cluster
  - Storage layer
    - Foundation for most tools in the Hadoop ecosystem
    - Scalable: one can easily add more nodes to increase total storage space
    - Reliable: fault tolerant (more on this soon)

HDFS

- Think about a very large data file (gigabytes or petabytes)
  - Example: Kaggle’s Data Science Bowl 2017
    - Goal: improve lung cancer detection
    - Prize: $1,000,000
    - Data size: 67+ GB
- HDFS breaks such files into chunks (blocks) and spread them over a computer cluster

HDFS

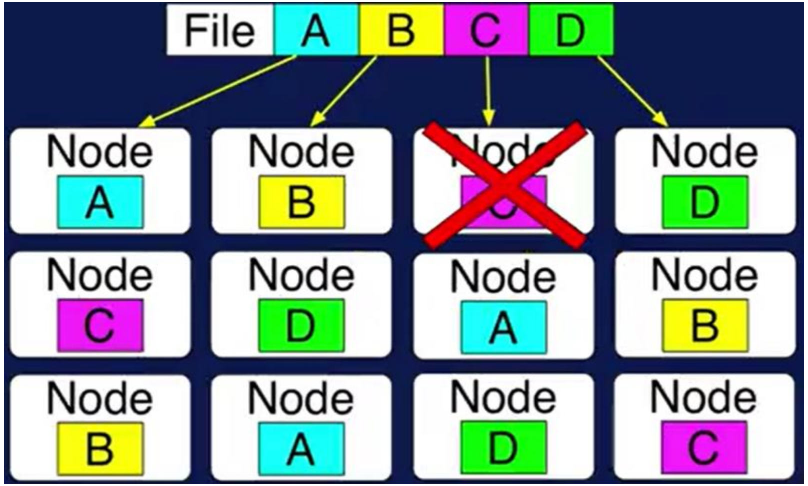
- Example: suppose a person uploads a 256-MB file to HDFS
  - Default block (piece) size: 64 MB
    - Configurable
  - That file is split into four parts (256/64 = 4) and the resulting blocks are spread across the cluster



  - In the above example, what happens if Node C fails?
    - *E.g.*, power outage

HDFS

- HDFS is designed for fault tolerance
  - Replication: HDFS makes copies of blocks on different nodes to prevent data loss
    - Default: 3 copies (configurable)



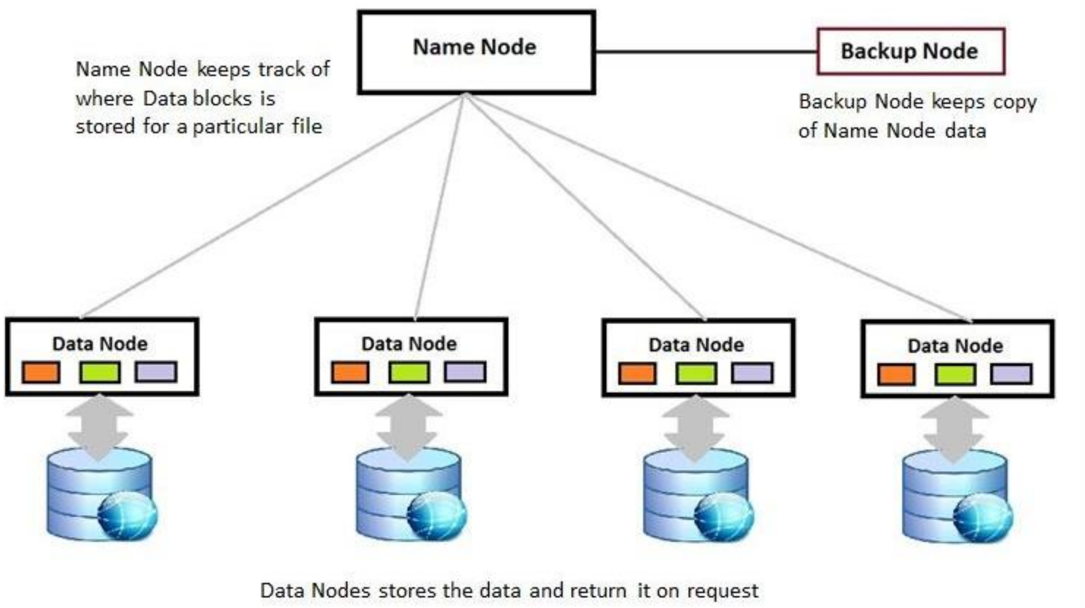
Example: if a node containing block C fails, two other nodes can still provide block C for a requesting application

HDFS

- HDFS vs Relational Databases Vs MongoDB
  - HDFS stores files (virtually any type)
    - Structured (*e.g.*, CSV) and unstructured (*e.g.*, images) data are *inside files*
  - Relational databases store data *inside tables*
    - Highly structured data
    - One must design a database model first
      - Not very flexible
  - MongoDB stores data using the *JSON* format
    - Very flexible when handling textual data
      - No need for a predefined model

HDFS

- Technical aspects: bird's-eye view



HDFS

- Technical aspects: two key components
  - **Name Node** for metadata
    - Handles metadata (data about the cluster)
    - Manager of the HDFS cluster
      - Keeps track of file names, location of blocks in directories/computers, *etc.*
    - Usually one per cluster, but there might exist a secondary, backup node
  - **Data Node** for block storage
    - Nodes store data (file blocks)
    - Usually, one Data Node refers to one machine
      - So, there are often many Data Nodes per cluster
    - Listens to commands from the Name Node
      - Block creation, deletion, replication

HDFS

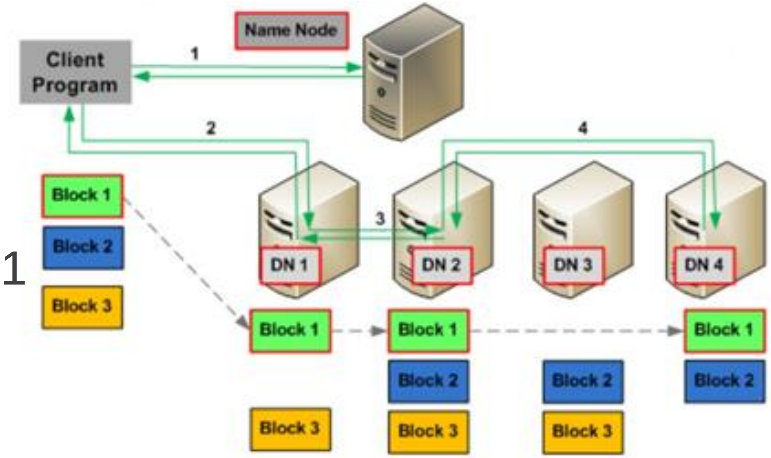
- Simplified example: suppose a task (client) wants to write a big file to a Hadoop cluster

1. Client makes a request to store Block 1;

   Name node informs the client the locations (*e.g.*, IP addresses) where Block 1 must be stored

   - First location: Data Node 1 (DN 1)

2. Client sends Block 1 and a list of locations to DN 1
3. DN 1 stores Block 1 and sends Block 1 plus a list of locations to DN 2
4. DN 2 stores Block 1 and sends Block 1 plus a list of locations to DN 4



MIAMI UNIVERSITY

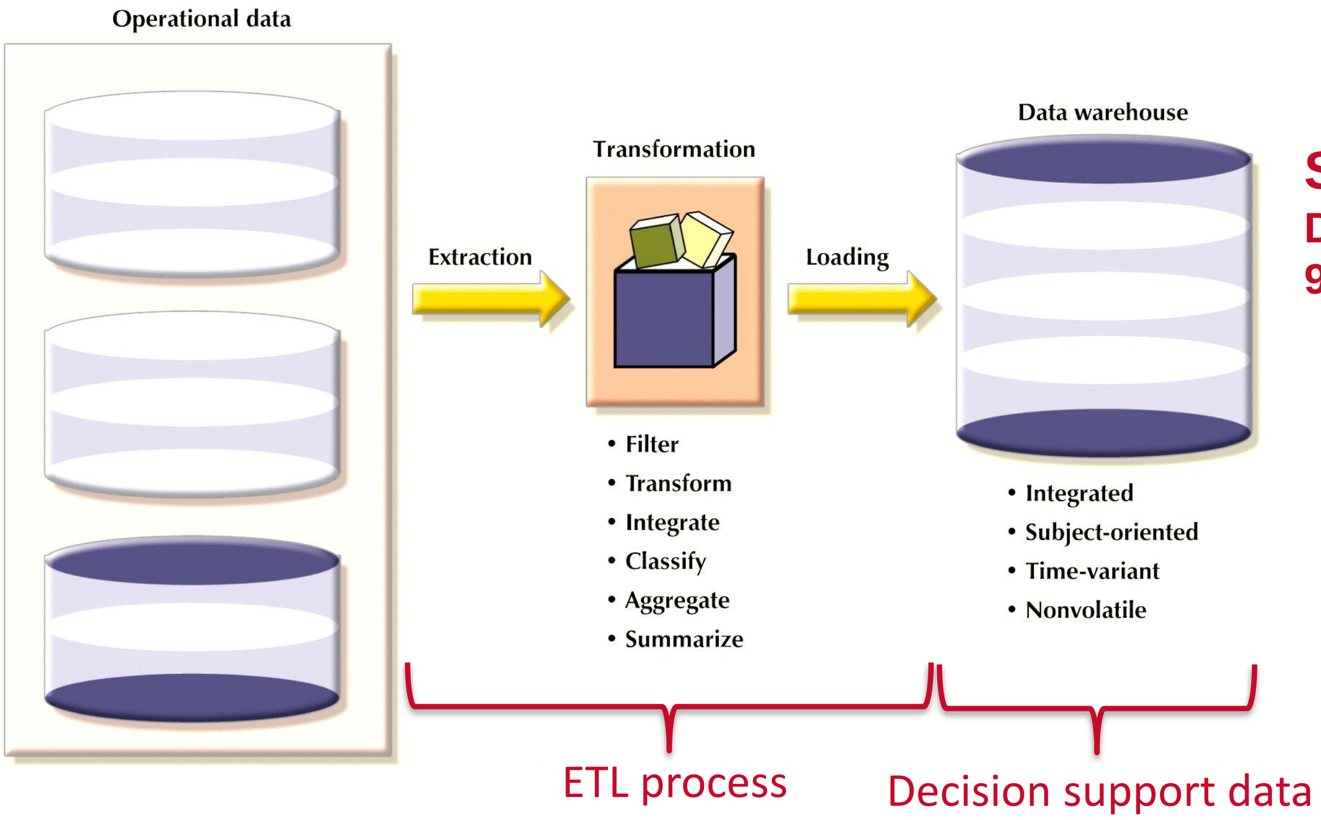
HDFS

- HDFS is all about storing files in a distributed manner
  - We will soon observe how HDFS works in practice
  - Before doing do, let’s review some relevant concepts
    - **Data warehouse**
    - **Data lake**

Data Warehouse

- Data Warehouse
  - Business intelligence (BI) technique to integrate and analyze data
  - Oftentimes, it is about an enormous collection of data
    - *Subject oriented* – designed around key entities (concepts)
    - *Integrated* – consistency in naming convention, encoding, translation, …
    - *Time-variant* – data are organized by time periods
    - *Non-volatile* – data are updated in batches, rather than as transactions occur

Data Warehouse

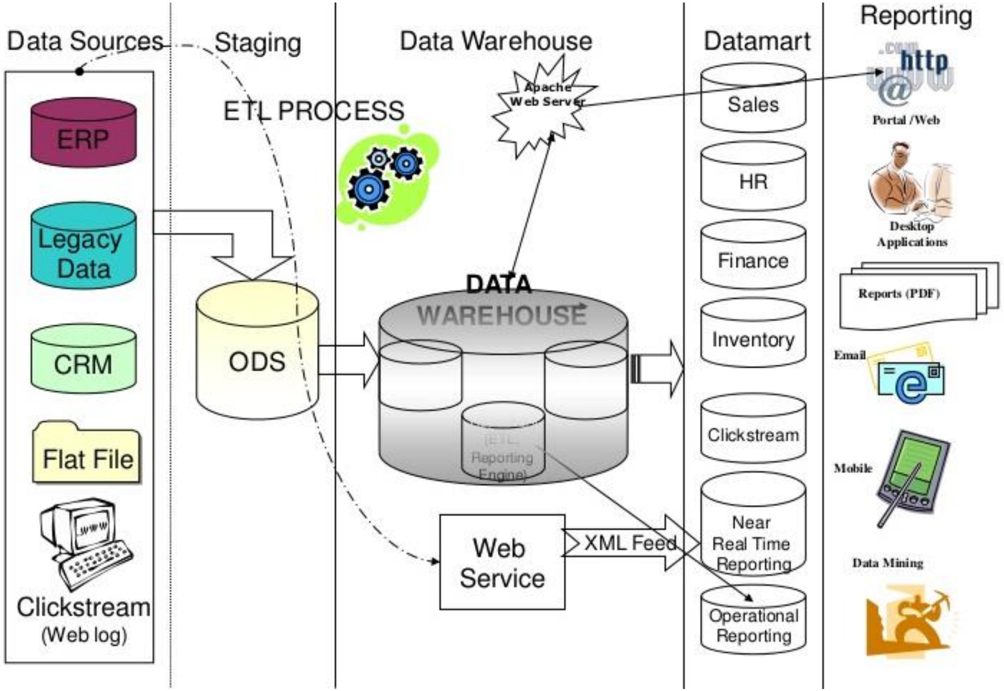


Source:
Database Systems, 9th Edition

MIAMI UNIVERSITY

Data Warehouse

- A more realistic view



Data Warehouse

- **Operational data**
  - Mostly stored in relational databases
  - Optimized to support transactions representing daily operations
- **Decision support data**
  - Derived from operational data
  - Stored in a database optimized for data analysis and query processing

Data Warehouse

- **ETL** - key processes for deriving decision support data from operational data
  - **EXTRACT**
    - The process of extracting operational data, *e.g.*, query a relational database
    - Can be very time-consuming
      - Often happens when the operational databases are not heavily used
    - Data can be stored in distributed databases
    - Potentially different database vendors (*e.g.*, Oracle, MySQL) or representations (*e.g.*, relational model, network model, hierarchical model)

Data Warehouse

- **ETL** - key processes of deriving decision support data from operational data
  - **TRANSFORM**
    - Standardizing different pieces of data
      - Example: convert “male”/“female” to 0/1, inches to centimeters, date format from “dd/mm/yy” to “yyyy/mm/dd”
    - Cleaning data
      - Same record stored in different databases
      - Misuse of data entry fields
        - *E.g.*, age = 200, SSN = 0000000000

Data Warehouse

- **ETL** - key processes of deriving decision support data from operational data
  - **LOAD**
    - The process of loading the data into the warehouse
    - Occur in batches, rather than after each transaction involving operational data

Data Warehouse

- Is HDFS a data warehouse technology?
  - Technically, no!

- Unlike HDFS, a data warehouse:
  1. Follows a pre-defined architecture
     - *E.g.*, a star schema having dimensions, facts, and attributes
  2. Follows a strong data quality assurance process (*i.e.*, ETL)

Data Lake

- So, what is HDFS?
  - Answer: a ***Data Lake***
    - Centralized repository of data
    - Stores all kinds of data in raw format (*i.e.*, files)
      - Images, texts, audio, …
    - Data are not necessarily curated before being dumped into a data lake
      - One of the main criticism of Hadoop HDFS

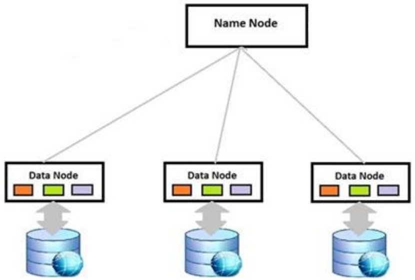
Data Lake

- So, what is Hadoop HDFS?
  - Answer: a *Data Lake*
    - Hadoop was “the next big thing” about 10 years ago
    - Common approach followed by many companies during Hadoop’s hype peak time:

“We see customers creating big data graveyards, dumping everything into HDFS [Hadoop Distributed File System] and hoping to do something with it down the road. But then they just lose track of what’s there. The main challenge is not creating a data lake, but taking advantage of the opportunities it presents” (Sean Martin, CTO of Cambridge Semantics)

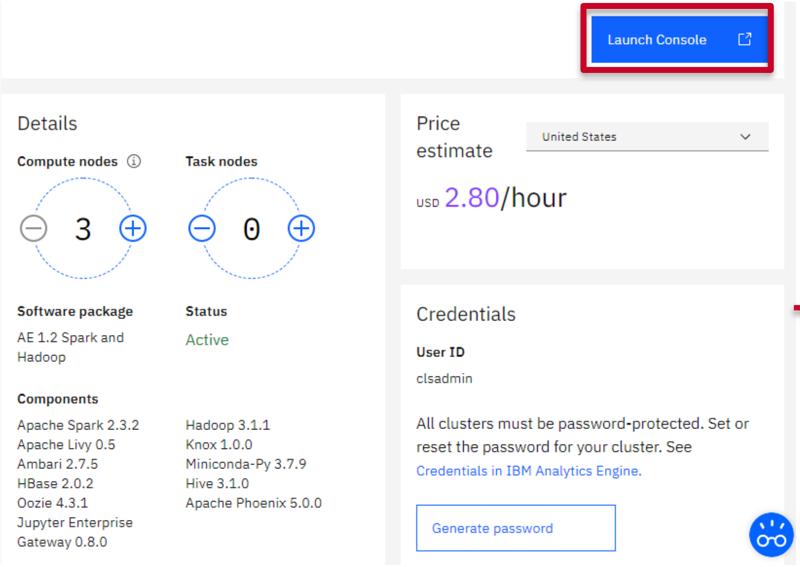
HDFS

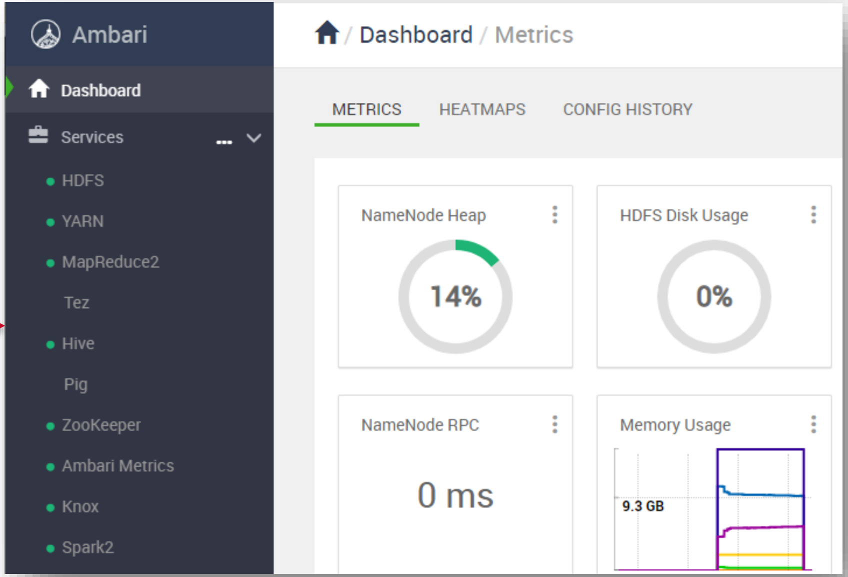
- Let's visualize some of the previous concepts
  - Recall that HDFS is a back-end technology
    - "Boring," not visually appealing
  - Cluster created on IBM Cloud
    - PaaS called *Analytics Engine*
      - See our previous class for instructions on how to set up this service
    - 1 Name Node (4 cores), 3 Data Nodes (12 cores)
      - $2.8 per hour
    - Topology



HDFS

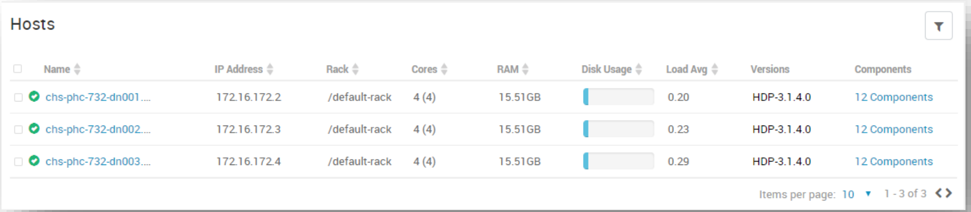
- Cluster administrator's perspective
  - Ambari software





HDFS

- Cluster administrator's perspective
  - Ambari software
    - Services: Hadoop services running in the cluster
    - *Services -> HDFS -> DATANODES*: Info about data nodes

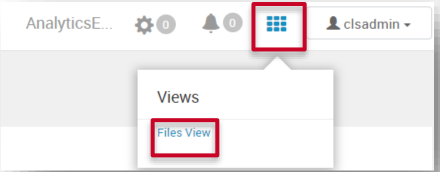


Hosts

	Name	IP Address	Rack	Cores	RAM	Disk Usage	Load Avg	Versions	Components
	chs-phc-732-dn001....	172.16.172.2	/default-rack	4 (4)	15.51GB		0.20	HDP-3.1.4.0	12 Components
	chs-phc-732-dn002....	172.16.172.3	/default-rack	4 (4)	15.51GB		0.23	HDP-3.1.4.0	12 Components
	chs-phc-732-dn003....	172.16.172.4	/default-rack	4 (4)	15.51GB		0.29	HDP-3.1.4.0	12 Components

Items per page: 10 1 - 3 of 3

    - *Views -> "Files Views"*: Upload data to HDFS



MIAMI UNIVERSITY

HDFS

- There are 2 files inside my cluster
  - */user/clsadmin/Data.csv* (1.2 MB)
  - */user/clsadmin/ Docklands.jmp* (450 MB)

- What happened behind the scenes when I uploaded these files?
  - Tip: IBM uses blocks of size 128 MB; replication factor = 3

- Hacking time
  - I will connect to the Name Node using SSH
  - Issue two commands to get info about the above files

```
hdfs  fsck /user/clsadmin/Data.csv -files  -locations  -blocks
hdfs  fsck /user/clsadmin/Docklands.jmp -files  -locations  -blocks
```

HDFS

File #1: Data.csv (1.2 MB)

```
FSCK started by clsadmin (auth:SIMPLE) from /172.16.122.135 for path /user/clsadmin/Data.csv at Tue Mar 09 15:48:52 UTC 2021
/user/clsadmin/Data.csv 1230849 bytes, replicated: replication=3, 1 block(s):  OK
0. BP-2089008039-172.16.122.134-1615302408196:blk_1073743602_2778 len=1230849 Live_repl=3  [DatanodeInfoWithStorage[172.16.122.131:50010,DS-f4d957e2
-1c71-42e9-8310-c5d0aec12cb7,DISK], DatanodeInfoWithStorage[172.16.122.132:50010,DS-df4aac5a-b829-4b47-adbe-b5d79cbc75de,DISK], DatanodeInfoWithStor
age[172.16.122.130:50010,DS-126c24f2-174c-46ca-b9bb-7ba343d48a2c,DISK]]


Status: HEALTHY
 Number of data-nodes:  3
 Number of racks:               1
 Total dirs:                    0
 Total symlinks:                0

Replicated Blocks:
 Total size:    1230849 B
 Total files:   1
 Total blocks (validated):      1 (avg. block size 1230849 B)
 Minimally replicated blocks:   1 (100.0 %)
 Over-replicated blocks:        0 (0.0 %)
 Under-replicated blocks:       0 (0.0 %)
 Mis-replicated blocks:         0 (0.0 %)
 Default replication factor:    3
 Average block replication:     3.0
 Missing blocks:                0
 Corrupt blocks:                0
 Missing replicas:              0 (0.0 %)
```

HDFS

File #2: Docklands.jmp (450 MB)

```
FSCK started by clsadmin (auth:SIMPLE) from /172.16.122.135 for path /user/clsadmin/Docklands.jmp at Tue Mar 09 15:52:12 UTC 2021
/user/clsadmin/Docklands.jmp 450200813 bytes, replicated: replication=3, 4 block(s):  OK
0. BP-2089008039-172.16.122.134-1615302408196:blk_1073743598_2774 len=134217728 Live_repl=3  [DatanodeInfoWithStorage[172.16.122.130:50010,DS-126c24
f2-174c-46ca-b9bb-7ba343d48a2c,DISK], DatanodeInfoWithStorage[172.16.122.132:50010,DS-df4aac5a-b829-4b47-adbe-b5d79cbc75de,DISK], DatanodeInfoWithSt
orage[172.16.122.131:50010,DS-f4d957e2-1c71-42e9-8310-c5d0aec12cb7,DISK]]
1. BP-2089008039-172.16.122.134-1615302408196:blk_1073743599_2775 len=134217728 Live_repl=3  [DatanodeInfoWithStorage[172.16.122.132:50010,DS-4ef651
36-5875-4096-a8df-b7d5644a133e,DISK], DatanodeInfoWithStorage[172.16.122.130:50010,DS-7440a047-980e-405f-aa45-d69fc6aa5396,DISK], DatanodeInfoWithSt
orage[172.16.122.131:50010,DS-d4df355f-be8b-4ca6-95b6-75b1a3549cb7,DISK]]
2. BP-2089008039-172.16.122.134-1615302408196:blk_1073743600_2776 len=134217728 Live_repl=3  [DatanodeInfoWithStorage[172.16.122.130:50010,DS-126c24
f2-174c-46ca-b9bb-7ba343d48a2c,DISK], DatanodeInfoWithStorage[172.16.122.131:50010,DS-f4d957e2-1c71-42e9-8310-c5d0aec12cb7,DISK], DatanodeInfoWithSt
orage[172.16.122.132:50010,DS-df4aac5a-b829-4b47-adbe-b5d79cbc75de,DISK]]
3. BP-2089008039-172.16.122.134-1615302408196:blk_1073743601_2777 len=47547629 Live_repl=3  [DatanodeInfoWithStorage[172.16.122.130:50010,DS-7440a04
7-980e-405f-aa45-d69fc6aa5396,DISK], DatanodeInfoWithStorage[172.16.122.131:50010,DS-d4df355f-be8b-4ca6-95b6-75b1a3549cb7,DISK], DatanodeInfoWithSto
rage[172.16.122.132:50010,DS-4ef65136-5875-4096-a8df-b7d5644a133e,DISK]]


Status: HEALTHY
 Number of data-nodes:  3
 Number of racks:               1
 Total dirs:                    0
 Total symlinks:                0

Replicated Blocks:
 Total size:    450200813 B
 Total files:   1
 Total blocks (validated):      4 (avg. block size 112550203 B)
 Minimally replicated blocks:   4 (100.0 %)
 Over-replicated blocks:        0 (0.0 %)
 Under-replicated blocks:       0 (0.0 %)
 Mis-replicated blocks:         0 (0.0 %)
 Default replication factor:    3
 Average block replication:     3.0
 Missing blocks:                0
 Corrupt blocks:                0
 Missing replicas:              0 (0.0 %)
```

134,217,728 * 3 + 47,547,629 = 450,200,813 bytes (450 MB)

HDFS

- All the technicalities related to HDFS (configuration, load balancing, installation, *etc*.) are often handled by IT professionals
  - Data scientists use HDFS to request/send files (data) to/from Hadoop

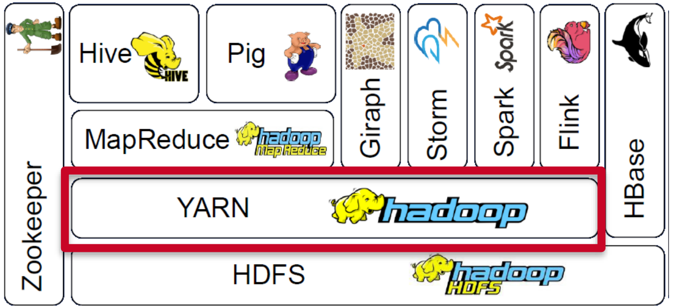
- Analogy: accessing data/storing data from/in a relational database
  - Database administrator = IT professional who takes care of the technical aspects of running/maintaining the database

HDFS

- The Name Node can tell an application/task where each piece of data is stored
  - The relevant computations are then executed in the Data Node
    - “Moving computation to data”
    - Moving data around is costly (bandwidth wise)

- Unanswered questions
  - If a task wants to access the resources from a node (CPU, memory, disk, …), how do we know whether that node is busy
    - Busy = dealing with another task
    - We need some sort of resource manager/negotiator!

YARN



- **Y**et **A**nother **R**esource **N**egotiator
  - Layer on top of HDFS
  - Schedule resources to be used by different applications
    - Enables running multiple applications over HDFS in parallel

- In practice, it is rare to deal with YARN directly
  - In this course, we will use Python to connect to high-level services (such as Spark), which in turn may use YARN behind the scenes

Hadoop

- When to use Hadoop HDFS
  - High data volume
  - High data variety
  - Multiple or parallel computations using the same data
- When not to use Hadoop
  - “Small data” processing
  - To store very well-structured data
    - Relational databases will likely do a better job
  - To store textual data only
    - MongoDB will likely do a better job

Summary

- We learned about a basic service in Hadoop
  - HDFS
    - Distributed file system
- Homework 10 is available on Canvas
- Next lecture
  - Spark