

Session 1

# BigData Storage and Processing Systems

Big Data Analytics Technology, MSc in Data Science,  
Coventry University UK

Miyuru Dayarathna

# Presentation Outline

- Introduction
- Modes of BigData Processing
- Big Data Storage Approaches
- Conclusion



## The problem of Big Data

- Increasingly organizations have been collecting and storing data from their daily operations.
- Data generation have been facilitated by recent hardware trends.

# The problem of Big Data

- Especially web scale organizations are collecting massive data sets with various structures.
- In 2020,
  - Internet users generate about 2.5 quintillion bytes (2.5 million terabytes) every day
  - There were 4.72 billion users on the Internet every day with more than 900,000 new users daily
- IDC expects 175 zettabytes (175 billion terabytes) of data worldwide by 2025

<https://www.forbes.com/sites/forbestechcouncil/2021/08/02/understanding-generation-data/?sh=5823436b36b7>

<https://www.networkworld.com/article/3325397/idc-expect-175-zettabytes-of-data-worldwide-by-2025.html>

# Technology Trends and sources of Big Data

- Smart Phones
  - ▷ Approx. 6.9 billion Smartphones by 2023
- Road Cameras
- IoT Devices
- Satellites
- Industrial Machinery

<https://www.technologyreview.com/technology/big-data-from-cheap-phones/>  
<https://www.computerworld.com/article/2473730/smartphones--big-data--storage-and-you.html>

# The 5V's of Big Data



<https://www.virtasant.com/blog/data-lake-architecture>

# The 5V's of Big Data

- **Volume:** the size and amounts of big data that companies manage and analyze
- **Value:** the most important “V” from the perspective of the business, the value of big data
- **Variety:** the diversity and range of different data types, including unstructured data, semi-structured data and raw data
- **Velocity:** the speed at which companies receive, store and manage data – e.g., the specific number of social media posts or search queries received within a day, hour or other unit of time
- **Veracity:** the “truth” or accuracy of data and information assets, which often determines executive-level confidence

# Batch Processing

- Periodically complete high-volume, repetitive data jobs
  - ▷ backups
  - ▷ filtering
  - ▷ sorting
- An ecommerce system that receives orders throughout the day. Instead of processing every order as it occurs, the system might collect all orders at the end of each day and share them in one batch with the order fulfillment team.



## Batch Processing - Pros

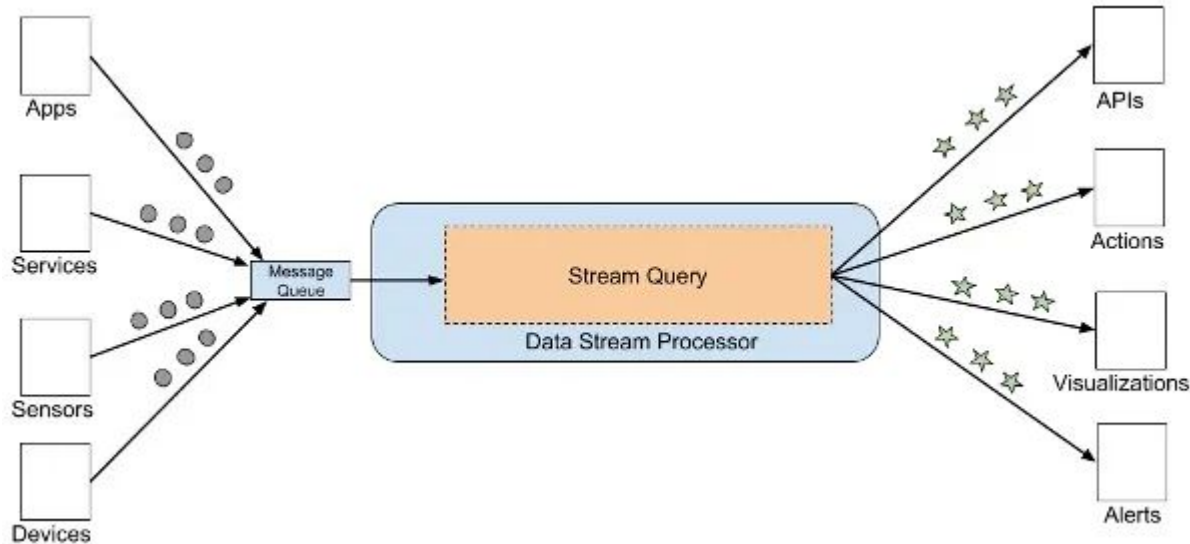
- Avoid data errors
- Economical approach which does not need special hardware or system support. Also can leverage secondary storage.
- Batch systems can work offline so it makes less stress on processors
- Can assign specific time for batch jobs so that when the computer is idling state it can start processing the batch jobs.
- Processor takes significant time to process the batch hence we can do accurate estimations of the time taken.

## Batch Processing - Cons

- Immediate update to query responses cannot be made
- Periodically scheduled reports get out-dated between scheduled processing
- Difficulty in dealing with the corrupted data
- Large scale failures are more likely to happen

# Stream Processing

- Stream processing is a data processing paradigm that continuously collects and processes real-time or near-real-time data.



# Stream Processing

- Digital enterprises depend on streaming analytics and real-time data processing to gain a competitive edge in day-to-day business operations.
- For some use cases, the value of insights decline with time. Hence, quickly collecting insights and responding to them would likely result in higher profits.

## Stream Processing - Pros

- Lower latency in arriving at processing results
  - ▷ logic is executed on the data as and when they arrive
  - ▷ Enhances responsiveness to customer needs, market trends, business opportunities, etc.
  - ▷ Responding quickly to customer queries increases customer satisfaction.
  - ▷ Reduces potential losses by providing real-time intelligence on financial downturns, system outages, cyber security issues, etc.

## Stream Processing - Cons

- It is difficult to implement at scale.
  - ▷ Without additional processing capacities the system may get overloaded when a large amount of data flows into the system at a short period of time.
- Need additional engineering to ensure resilience against issues such as out of order data

## BigData Storage Approaches

- BigData applications need to handle enormous datasets which may scale in to multiple peta bytes
- Single storage server is not suitable for storing such large data

# Distributed File Systems

- A file system that runs on multiple servers simultaneously.
- Can contain files larger than any one computer disk
- Not bound by single server's resource limitations



# Distributed File Systems

- Access to data stored at servers using file system interfaces
  - ▷ Open a file, check status of a file, close a file
  - ▷ Read/write data
  - ▷ Lock file or part of file
  - ▷ List files in a directory
  - ▷ Create/delete a directory
  - ▷ Delete/rename/add a symlink to a file
  - ▷ etc.

# Distributed File Systems

Distributed File System	License	Website
HDFS	Apache License 2.0	<a href="https://hadoop.apache.org/">https://hadoop.apache.org/</a>
Gfarm	Open source	<a href="https://github.com/oss-tsukuba/gfarm">https://github.com/oss-tsukuba/gfarm</a>
Google File System	Proprietary	-
Lustre	GPL v2, LGPL	<a href="https://www.lustre.org/">https://www.lustre.org/</a>

# Hadoop Distributed File System (HDFS)

A distributed file system designed for storing very large files with streaming data access pattern , runs on clusters of commodity hardware

- ▷ Very large files - Hundreds of megabytes, gigabytes, or terabytes.
- ▷ Streaming data access
  - ▷ Most efficient data processing pattern is a write-once, read-many-times pattern
- ▷ Commodity hardware - Doesn't require expensive, highly reliable hardware to run on.
  - ▷ File replication to handle hardware failures
  - ▷ Failure detection and recovery
- ▷ Optimized for batch processing
  - ▷ Data locations are exposed enabling computations to move to where data resides
  - ▷ Provide very high aggregate bandwidth

# Hadoop Distributed File System (HDFS)

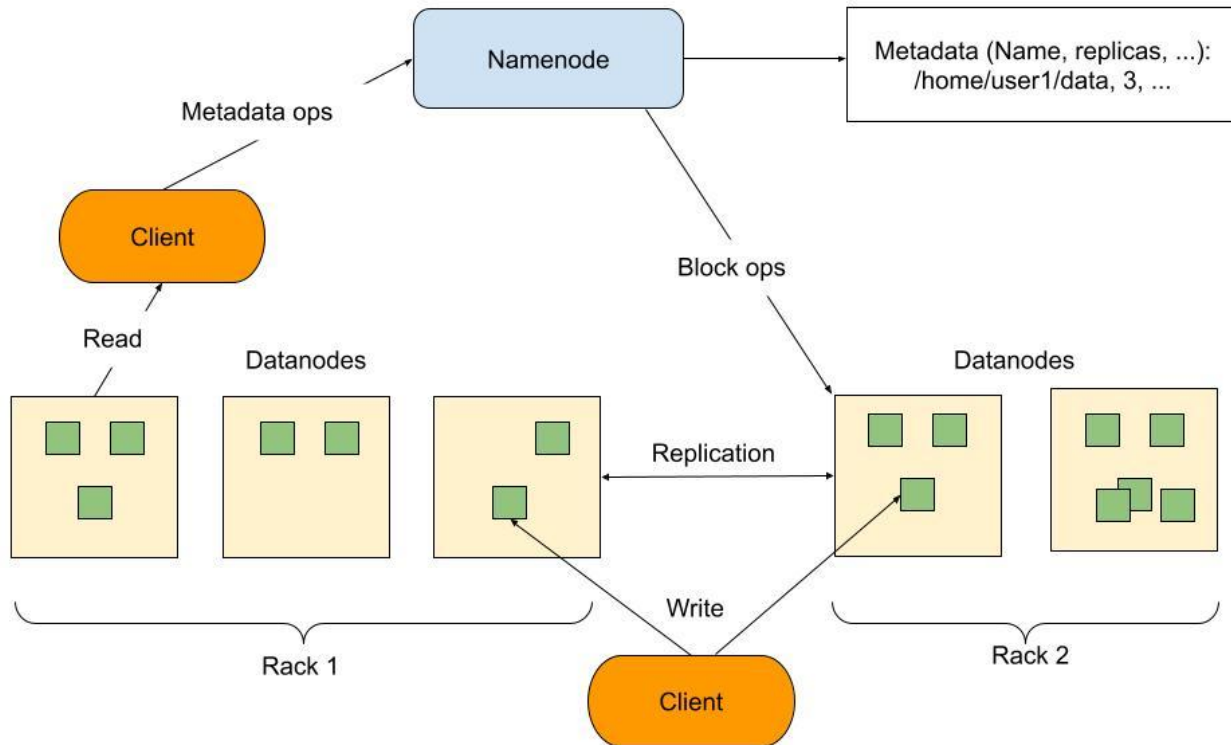
- Single namespace for entire cluster
- Files are broken up into multiple blocks
  - ▷ 64MB block size typically
  - ▷ Each block is replicated on multiple data nodes
- Intelligent Client
  - ▷ Client can find location of blocks
  - ▷ Client accesses data directly from DataNode

## Hadoop Distributed File System (HDFS)

HDFS does not fit well for the following scenarios,

- Low-latency data access - Applications requiring ten's of milliseconds range will not work well with HDFS. It is optimized for delivery of high throughput data.
- Lots of small files - Since the namenode keeps filesystem metadata in memory, the limit to the number of files in the filesystem is determined by the amount of memory in the namenode.
- Multiple writers, arbitrary file modifications

# HDFS Architecture



## HDFS Architecture (Contd.)

- HDFS has master/worker architecture
- An HDFS cluster has
  - ▷ One single **NameNode** (**the master**)
    - ▷ Manages the filesystem namespace
    - ▷ Regulates access to files by clients
    - ▷ Replication engine for blocks
  - ▷ A number of **Datanodes** (**workers**)
    - ▷ Usually one datanode per node in the cluster which manage the storage attached to the nodes that they run

## HDFS Architecture - Client

- Accesses the filesystem on behalf of the user by communicating with the namenode and datanodes.
- Client provides a POSIX-like filesystem interface where the user code does not need to know about the name node and data node to function.



## HDFS Architecture - Namenode

- Manages the filesystem namespace
- Manages the filesystem tree
- Manages the metadata for all the files and directories in the tree
- Namenode keeps track of the datanodes on which all the blocks for a given file are located.
- Does not store block locations persistently, because that information is recreated from datanodes when the system starts

## HDFS Architecture - Namenode (Contd.)

- Types of metadata
  - ▷ List of files
  - ▷ List of Blocks for each file
  - ▷ List of DataNodes for each block
  - ▷ File attributes (e.g., replication factor, creation time, etc.)
- A transaction log
  - ▷ Keeps track of file creations, deletions, etc.

## HDFS Architecture - Datanode

- Datanodes carry out the actual workload of the filesystem
- A Block Server
  - ▷ Datanodes store and retrieve blocks as per the instructions from clients or the namenode
  - ▷ Stores metadata of a block (e.g., CRC) and also serves the metadata to clients
- Support Data pipelining
  - ▷ Forewards data to other specified DataNodes
- Block Report
  - ▷ Datanode periodically reports back to the namenode with lists of blocks that they are storing

## HDFS Architecture - Fault resilience - NameNode

A single point of failure

- If the machine running the namenode was crashed all the files in the filesystem would be lost.
  - ▷ There is no way to know how to reconstruct the files from the blocks on the datanodes.
- Approaches for resiliency
  - ▷ Approach 1 - Namenode can write its persistent state (e.g., transaction log) to multiple filesystems synchronously and atomically. (e.g., write to local disk as well as to a remote file system (NFS/CIFS))

## HDFS Architecture - Fault resilience - NameNode (Contd.)

- Approaches for resiliency
  - Approach 2 - Run a secondary namenode which does not actually act as a namenode.
  - The secondary namenode generally runs on a separate physical machine because it requires abundant amount of CPU and lot of memory as the namenode does to perform merge.

## HDFS Architecture - Fault resilience - NameNode (Contd.)

- Approaches for resiliency
  - ▶ The secondary namenode periodically merges the namespace image with the edit log to prevent the edit log becoming too large.
    - ▶ Copy FSImage and Transaction log from NameNode to a temporary directory
    - ▶ Merge these two types of data into a new FSImage inside the temporary directory
    - ▶ Upload the new FSImage to the NameNode
    - ▶ Transaction Log on NameNode is purged

## HDFS Architecture - Fault resilience - DataNode

- DataNode sends heartbeat to the NameNode
  - ▷ Once every 3 seconds
  - ▷ Heart beats are used by NameNode to identify DataNode failure
- If DataNode failure detected
  - ▷ Chooses new DataNode for new replicas
  - ▷ Balances disk usage and communication traffic to DataNodes

## HDFS Architecture - Fault resilience - DataNode (Contd.)

- DataNode uses a CRC 32 checksum to validate data
- During file creation the client computes checksum per 512 bytes and this checksum is stored in DataNode
- During the file access the the client obtains the data and the checksum from DataNode.
- If there is a failure in validation the client tries with other replicas



## HDFS Architecture - Rebalancer

- Rebalancer maintains the percentage of disk usage on the DataNodes at similar levels
- Rebalancer does not obstruct the execution of the HDFS cluster. Rebalancer is throttled to avoid network congestion.

## HDFS User interface Commands

- HDFS User
  - ▷ `/opt/hadoop-3.2.1/bin/hdfs dfs -ls /`
  - ▷ `/opt/hadoop-3.2.1/bin/hdfs dfs -mkdir /sample`
  - ▷ `/opt/hadoop-3.2.1/bin/hdfs dfs -rm -r /sample`
- HDFS Admin
  - ▷ `/opt/hadoop-3.2.1/bin/hdfs dfsadmin -report`
- Web interface

# Setup HDFS

- Prerequisites
  - ▷ Have Ubuntu setup on your computer (e.g., VM)
  - ▷ Install docker and git on your Ubuntu system
- clone <https://github.com/big-data-europe/docker-hadoop>
- run *docker-compose up*

```
bde2020/hadoop-base          latest          852f71d04ec2    3 years ago    1.37GB
bde2020/hadoop-base          master         852f71d04ec2    3 years ago    1.37GB
bde2020/hadoop-nodemanager    2.0.0-hadoop3.2.1-java8 4e47dabd148f    3 years ago    1.37GB
bde2020/hadoop-resourcemanager 2.0.0-hadoop3.2.1-java8 3deba4a1885f    3 years ago    1.37GB
bde2020/hadoop-namenode       2.0.0-hadoop3.2.1-java8 839ec11d95f8    3 years ago    1.37GB
bde2020/hadoop-historyserver  2.0.0-hadoop3.2.1-java8 173c52d1f624    3 years ago    1.37GB
bde2020/hadoop-datanode       2.0.0-hadoop3.2.1-java8 df288ee0a7f9    3 years ago    1.37GB
miyurud@miyurud-XPS-15-9510: ~/git/iasminagraph-prerequisites$
```

# Setup HDFS

```
miyurud@miyurud-XPS-15-9510: ~/git/jasminegraph-prerequisites
bde2020/hadoop-historyserver      2.0.0-hadoop3.2.1-java8  173c52d1f624  3 years ago  1.37GB
bde2020/hadoop-datanode           2.0.0-hadoop3.2.1-java8  df288ee0a7f9  3 years ago  1.37GB
miyurud@miyurud-XPS-15-9510: ~/git/jasminegraph-prerequisites$
miyurud@miyurud-XPS-15-9510: ~/git/jasminegraph-prerequisites$ docker ps
```

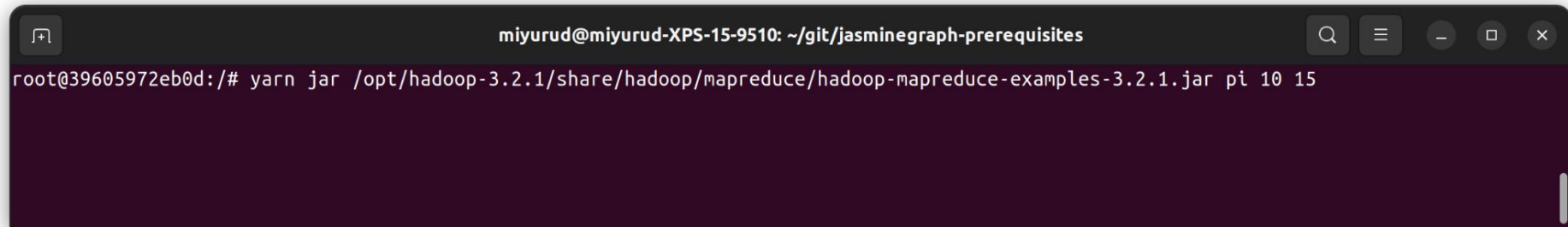
CONTAINER ID	IMAGE	NAMES	COMMAND	CREATED	STATUS	PORTS
e64d505df9f6	bde2020/hadoop-historyserver:2.0.0-hadoop3.2.1-java8	historyserver	"/entrypoint.sh /run..."	2 days ago	Up 2 days (healthy)	8188/tcp
51d2d6dd7e96	bde2020/hadoop-namenode:2.0.0-hadoop3.2.1-java8	namenode	"/entrypoint.sh /run..."	2 days ago	Up 2 days (healthy)	0.0.0.0:9000->9000/tcp, :::9000->9000/tcp, 0.0.0.0:9870->9870/tcp, :::9870->9870/tcp
0e822b4ae56f	bde2020/hadoop-resourcemanager:2.0.0-hadoop3.2.1-java8	resourcemanager	"/entrypoint.sh /run..."	2 days ago	Up 2 days (healthy)	8088/tcp
39605972eb0d	bde2020/hadoop-nodemanager:2.0.0-hadoop3.2.1-java8	nodemanager	"/entrypoint.sh /run..."	2 days ago	Up 2 days (healthy)	8042/tcp
6c6eee3b6328	bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8	datanode	"/entrypoint.sh /run..."	2 days ago	Up 2 days (healthy)	9864/tcp

```
miyurud@miyurud-XPS-15-9510: ~/git/jasminegraph-prerequisites$
```

# Setup HDFS

```
docker exec -it nodemanager /bin/bash
```

```
yarn jar /opt/hadoop-3.2.1/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.1.jar pi 10 15
```

A terminal window with a dark background. The title bar shows the user 'miyurud' on host 'miyurud-XPS-15-9510' in the directory '~/git/jasminegraph-prerequisites'. The terminal prompt is 'root@39605972eb0d:/#'. The command entered is 'yarn jar /opt/hadoop-3.2.1/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.1.jar pi 10 15'.

```
miyurud@miyurud-XPS-15-9510: ~/git/jasminegraph-prerequisites
root@39605972eb0d:/# yarn jar /opt/hadoop-3.2.1/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.1.jar pi 10 15
```

# Setup HDFS

```
miyurud@miyurud-XPS-15-9510: ~/jgit/jasminegraph-prerequisites
root@39605972eb0d: /# yarn jar /opt/hadoop-3.2.1/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.1.jar pt 10 15
Number of Maps = 10
Samples per Map = 15
2023-09-06 18:12:26,589 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localhostTrusted = false, remoteHostTrusted = false
Wrote input for Map #0
2023-09-06 18:12:26,639 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localhostTrusted = false, remoteHostTrusted = false
Wrote input for Map #1
2023-09-06 18:12:26,645 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localhostTrusted = false, remoteHostTrusted = false
Wrote input for Map #2
2023-09-06 18:12:26,651 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localhostTrusted = false, remoteHostTrusted = false
Wrote input for Map #3
2023-09-06 18:12:26,657 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localhostTrusted = false, remoteHostTrusted = false
Wrote input for Map #4
2023-09-06 18:12:26,662 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localhostTrusted = false, remoteHostTrusted = false
Wrote input for Map #5
2023-09-06 18:12:26,668 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localhostTrusted = false, remoteHostTrusted = false
Wrote input for Map #6
2023-09-06 18:12:26,688 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localhostTrusted = false, remoteHostTrusted = false
Wrote input for Map #7
2023-09-06 18:12:26,693 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localhostTrusted = false, remoteHostTrusted = false
Wrote input for Map #8
2023-09-06 18:12:26,699 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localhostTrusted = false, remoteHostTrusted = false
Wrote input for Map #9
Starting Job
2023-09-06 18:12:26,807 INFO client.RMProxy: Connecting to ResourceManager at resourcemanager/172.18.0.5:8032
2023-09-06 18:12:26,877 INFO client.AHSProxy: Connecting to Application History server at historyserver/172.18.0.6:10200
2023-09-06 18:12:26,952 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/root/.staging/job_1693840266628_0004
2023-09-06 18:12:26,966 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localhostTrusted = false, remoteHostTrusted = false
2023-09-06 18:12:26,996 INFO input.FileInputFormat: Total input files to process : 10
2023-09-06 18:12:27,003 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localhostTrusted = false, remoteHostTrusted = false
2023-09-06 18:12:27,009 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localhostTrusted = false, remoteHostTrusted = false
2023-09-06 18:12:27,013 INFO mapreduce.JobSubmitter: number of splits:10
2023-09-06 18:12:27,066 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localhostTrusted = false, remoteHostTrusted = false
2023-09-06 18:12:27,074 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1693840266628_0004
2023-09-06 18:12:27,075 INFO mapreduce.JobSubmitter: Executing with tokens: []
2023-09-06 18:12:27,168 INFO conf.Configuration: resource-types.xml not found
2023-09-06 18:12:27,168 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'
2023-09-06 18:12:27,405 INFO impl.YarnClientImpl: Submitted application application_1693840266628_0004
2023-09-06 18:12:27,434 INFO mapreduce.Job: The url to track the job: http://resourcemanager:8088/proxy/application_1693840266628_0004/
2023-09-06 18:12:27,435 INFO mapreduce.Job: Running job: job_1693840266628_0004
2023-09-06 18:12:30,472 INFO mapreduce.Job: Job job_1693840266628_0004 running in uber mode : false
2023-09-06 18:12:30,472 INFO mapreduce.Job: map 0% reduce 0%
2023-09-06 18:12:35,545 INFO mapreduce.Job: map 28% reduce 0%
2023-09-06 18:12:36,548 INFO mapreduce.Job: map 38% reduce 0%
2023-09-06 18:12:37,552 INFO mapreduce.Job: map 40% reduce 0%
```

# Setup HDFS




```
miyurud@miyurud-XPS-15-9510: ~/jgit/jasminegraph-prerequisites
HDFS: Number of large read operations=0
HDFS: Number of write operations=3
HDFS: Number of bytes read erasure-coded=0
Job Counters
  Launched map tasks=10
  Launched reduce tasks=1
  Rack-local map tasks=10
  Total time spent by all maps in occupied slots (ms)=41044
  Total time spent by all reduces in occupied slots (ms)=8416
  Total time spent by all map tasks (ms)=10261
  Total time spent by all reduce tasks (ms)=1052
  Total vcore-milliseconds taken by all map tasks=10261
  Total vcore-milliseconds taken by all reduce tasks=1052
  Total megabyte-milliseconds taken by all map tasks=42029056
  Total megabyte-milliseconds taken by all reduce tasks=8617984
Map-Reduce Framework
  Map input records=10
  Map output records=20
  Map output bytes=180
  Map output materialized bytes=250
  Input split bytes=1450
  Combine input records=0
  Combine output records=0
  Reduce input groups=2
  Reduce shuffle bytes=250
  Reduce input records=20
  Reduce output records=0
  Spilled Records=40
  Shuffled Maps =10
  Failed Shuffles=0
  Merged Map outputs=10
  GC time elapsed (ms)=222
  CPU time spent (ms)=2050
  Physical memory (bytes) snapshot=3578118144
  Virtual memory (bytes) snapshot=59863085056
  Total committed heap usage (bytes)=8386600000
  Peak Map Physical memory (bytes)=333901924
  Peak Map Virtual memory (bytes)=5140832256
  Peak Reduce Physical memory (bytes)=311427072
  Peak Reduce Virtual memory (bytes)=8484388864
Shuffle Errors
  BAD_ID=0
  CONNECTION=0
  IO_ERROR=0
  WRONG_LENGTH=0
  WRONG_MAP=0
  WRONG_REDUCE=0
File Input Format Counters
  Bytes Read=1180
File Output Format Counters
  Bytes Written=97
Job Finished in 19.937 seconds
2023-09-06 10:12:46,662 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localhostTrusted = false, remoteHostTrusted = false
Estimated value of pi is 3.173333333333333333333333
root@39605972eb0d:/#
```

# HDFS GUI

← → ↻ ⓘ localhost:9870/explorer.html#/foodir/sample

Hadoop Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities ▾

## Browse Directory

/foodir/sample    

Show  entries Search:

<input type="checkbox"/>	↕ Permission	↕ Owner	↕ Group	↕ Size	↕ Last Modified	↕ Replication	↕ Block Size	↕ Name	↕
--------------------------	--------------	---------	---------	--------	-----------------	---------------	--------------	--------	---

No data available in table

Showing 0 to 0 of 0 entries

Hadoop, 2019.



# Conclusion

- Modes of BigData Processing
  - ▷ Batch processing
  - ▷ Online/Stream processing
- BigData Storage Approaches

**Thank you!**