March 2016

WK	M	T	W	T	F	S	S
10		1	2	3	4	5	6
11	7	8	9	10	11	12	13
12	14	15	16	17	18	19	20
13	21	22	23	24	25	26	27
14	28	29	30	31			

2016
Saturday
January
16
016-350 • WK 03

Video 1

Big data processing pipelines : A dataflow Approach

→ Summarize what dataflow means and its role in data science

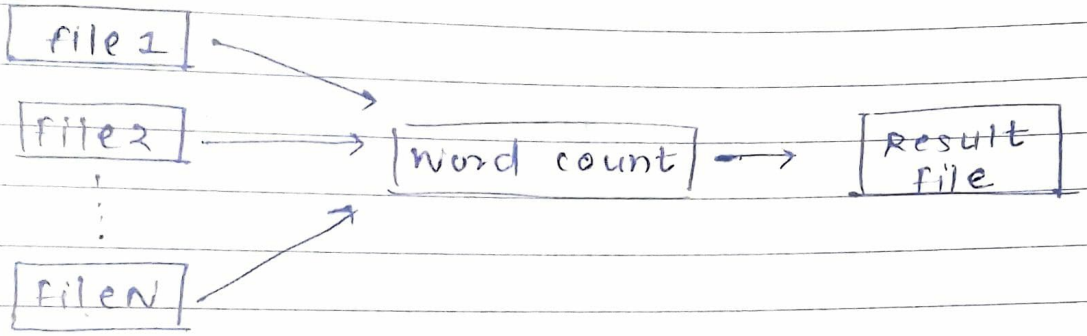
10 00 → Explain 'split → do → merge' big data pipeline with eg,

→ Define the terms data parallel.

11 00

Example MapReduce App :- Wordcount

12 00



1 00

2 00

3 00

step 1 : split

4 00 step 2 : Map

step 3 : shuffle and sort

5 00 step 4 : Reduce

In general

6 00 Split → Do something → Merge

Represents a large no. of applications

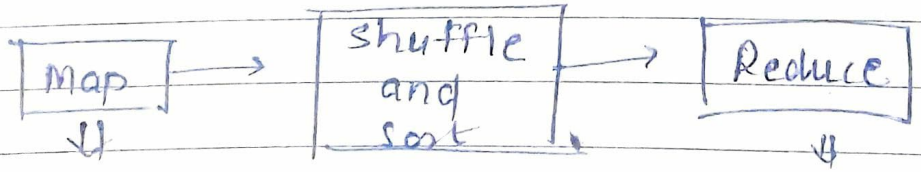
7 00 Big data pipelines :

pipe = Unix operation

Word count Example :-

Sunday 17

NOTES



parallelization over the I/P

parallelization data sorting

parallelization over data groups

Video 2 ⇒

* Some high-level processing operations in Big data pipelines

Data Transformation

Storage → H-
→ concan
→ HDFS
→ Many oth

→ List common data transformations within big data pipelines

→ Design a conceptual data processing pipeline using the basic data transformations.

(1) Map :- Apply some operation to each member of a collection. → color each member of set
→ Discount each products price by

(2) Reduce :- 'collecting' things that have same key → collect bloc as per color

(3) cross / cartesian :- Multiplication

(4) match / join :- selective Multiplication
('Do some process to each pair from two sets - which have same key')

(5) Co-Group :- Group common items
→ collect similar things.
→ Apply a process to each collection.

Do some process to each pair from two sets.

(6) Filter :- Select Elements that match a criteria
Filter even no.
X % 2 == 0

NOTES

Aggregation in Big data Pipelines:

→ Compare and select the Aggregation operation that you require to solve your problem.

→ Explain how you can use Aggregations to compact your dataset and reduce vol^n (in many cases)

→ Design complex operations in your pipelines using a series of Aggregations.

What is Aggregation :- → f (all elements)

Symbol for any transformation

Aggregation → f (all elements)

Σ → symbol for summation

① GroupBy ② Average ③ MAX ④ MIN
⑤ standard deviation.

Connecting Aggregations

sum → Max ⇒ MAX (SUM)

sum → min ⇒ MIN(SUM)

Boolean Aggregation :- ① AND
⑤ OR

Sets :- Union, intersection, difference

NOTES strings :- concatenation

Aggregations → Organized & compact data

variety & vol^n ⇒ Actionable insights

Analytical operations

→ List common analytic operations within big data pipeline.

→ Describe sample applications for these analytical operations.

Analytical operations :-

Patterns → Insights → Decisions

- Purpose :- ① Discover meaningful trends and patterns in data.

② Gain insights into patterns

③ make data-driven decisions

Sample Analytical Operations :- ① classification ③ Path Analysis
② clustering ④ Connecitivity Analysis.

K-means in Spark

- Spark python code for performing k-means on data

```
data = sc.textFile("data/mllib/kmeans_data.txt")
parsedData = data.map(lambda line:
    array([float(x) for x in line.split(' ')]))
# cluster the data
clusters = KMeans.train(parsedData, 2, maxIterations=10,
    runs=10, initializationMode="random")
```

path Analysis :-

- Path analysis using Cypher on neo4j

// Finding shortest path bth specific nodes:

```
match p = shortestPath ( (a) -[:TO*]- (c) )
where a.Name = 'A' and c.Name = 'P'
return P, length (P) limit 1
```

// Find all shortest paths :

```
match p = allShortestPaths ( (source) -[r:TO*]-
                              (destination) )
where source.Name = 'A' and destination.Name = 'P
return extract (n in nodes (p) | n.Name) as Paths
```

Connectivity analysis using Cypher on neo4j

// Find the degree of all nodes

```
match (n:myNode) -[r]- ()
with n as nodes , count (distinct r) as degree
return degree, count (nodes) order by degree
```

Graph Analytics Techniques

1. Path analytics
2. Connectivity Analytics
3. community -11-
4. centrality -11-

Videos

Overview of Big data processing system

→ Handoop Ecosystem

→ Layer Diagram with three layers for data storage, data processing and workflow management.

→ Summarize an evaluation criteria for big data processing systems

→ Hadoop, Spark, Flink, Beam and Storm

Hadoop Ecosystem :- ① Coordination and Workflow management

② Data integration and processing

③ Data management and storage

↓

hadoop, mangodB, redis, Verti ca

gephi

Hive, Pig, MapReduce

Acquire ⇒ Prepare ⇒ Analyze ⇒ Report ⇒ Act

Categorize of Big Data processing

→ Execution model → Batch
→ streaming

→ Latency

→ Scalability

→ programming language

→ Fault Tolerance

Big data processing system → Apache storm
→ spark
→ hadoop
→ flink
→ beam

NOTES

Hadoop map Reduce :-

→ EM - Batch processing using disk storage

→ Latency :- High

→ programming lang :- java

→ Fault tolerance :- Replication

Spark :- EM :- Batch and stream processing using disk or memory storage

Latency → low-latency for small micro batch size

program language :- scala, python, java, R

Flink :- EM - Batch and stream processing using disk or memory storage

Latency → low

P.L :- java and scala

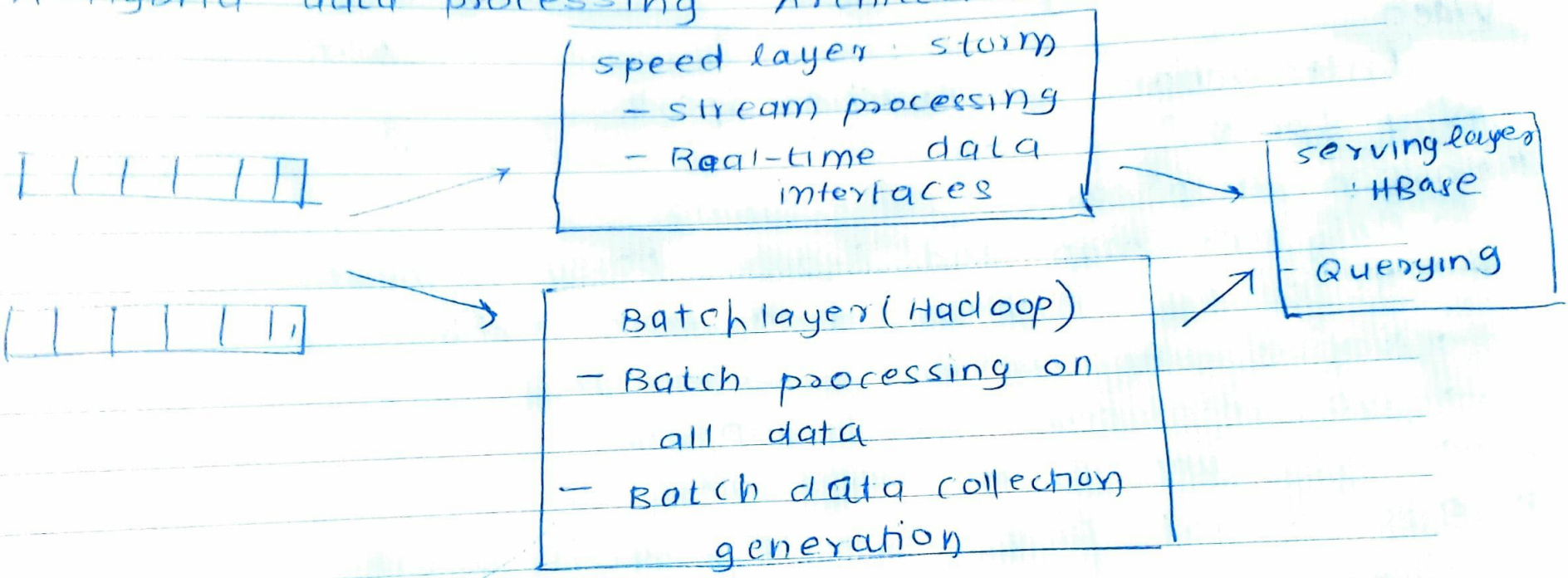
Beam :- Batch and stream processing
low-latency
java and scala

Storm :- stream processing, very low-latency, many programming langus.

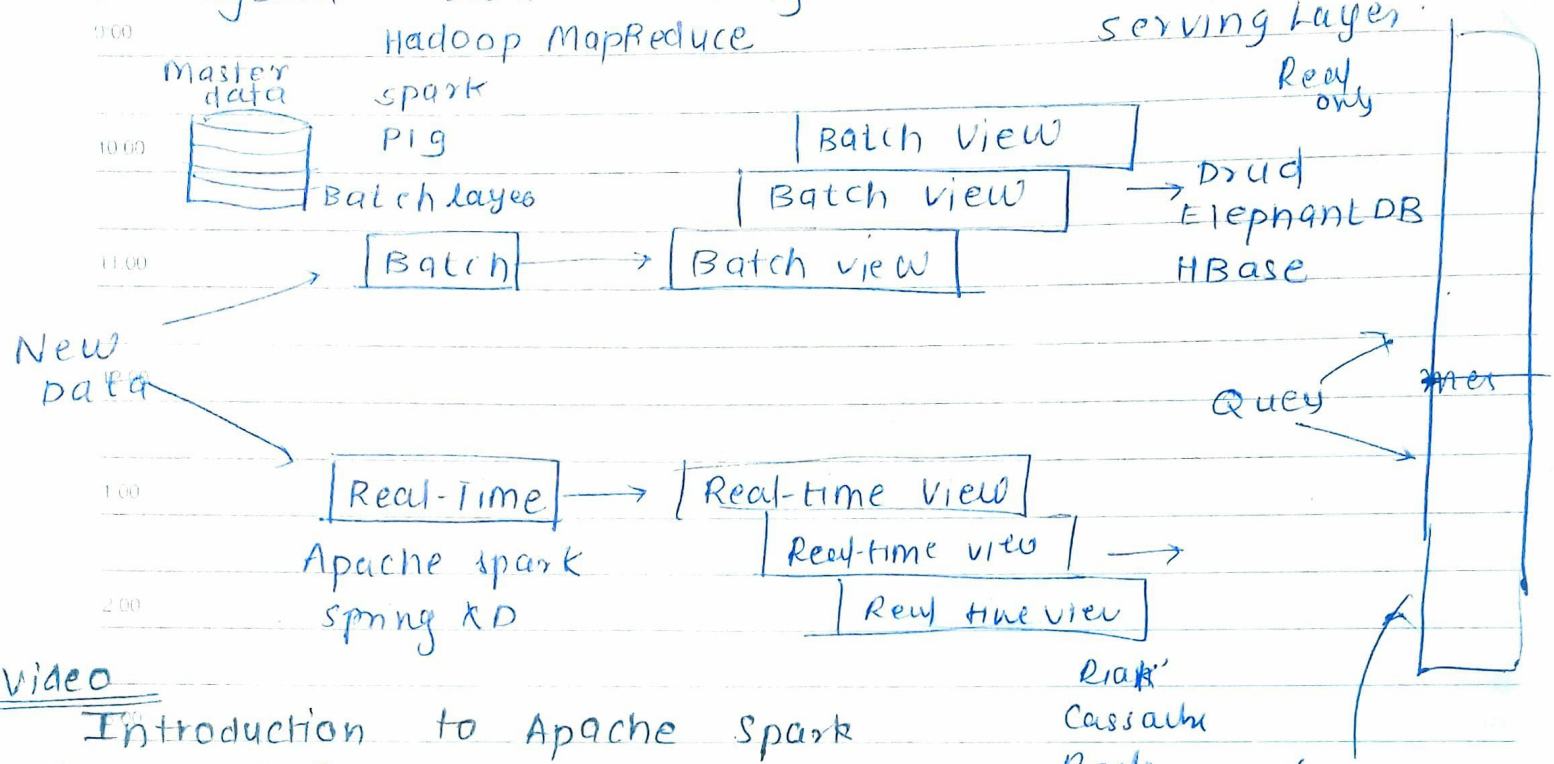
NOTES

Lamba Architecture :-

A Hybrid data processing Architecture.



A Hybrid Data Processing Architecture



Video

Introduction to Apache Spark

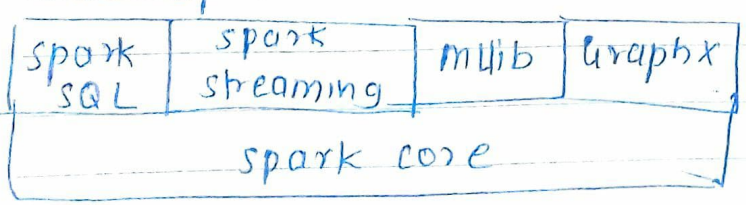
Why spark ?

* Hadoop MapReduce Shortcomings :-
  - → Only for map and Reduce based computations.
  - → Relies on reading data from HDFS.
  - → Native support for Java only.
  - → No interactive shell support
  - → No support for streaming.
* Basics of Data Analysis with spark :-
  - → Expressive programming model
  - → In-memory processing
  - → Support for diverse workloads

NOTES → Interactive shell

Explore → Build → scale

The spark stack



ting started with Spark :-

The Architecture and Basic concepts

W / T / F / S / SU /

Date

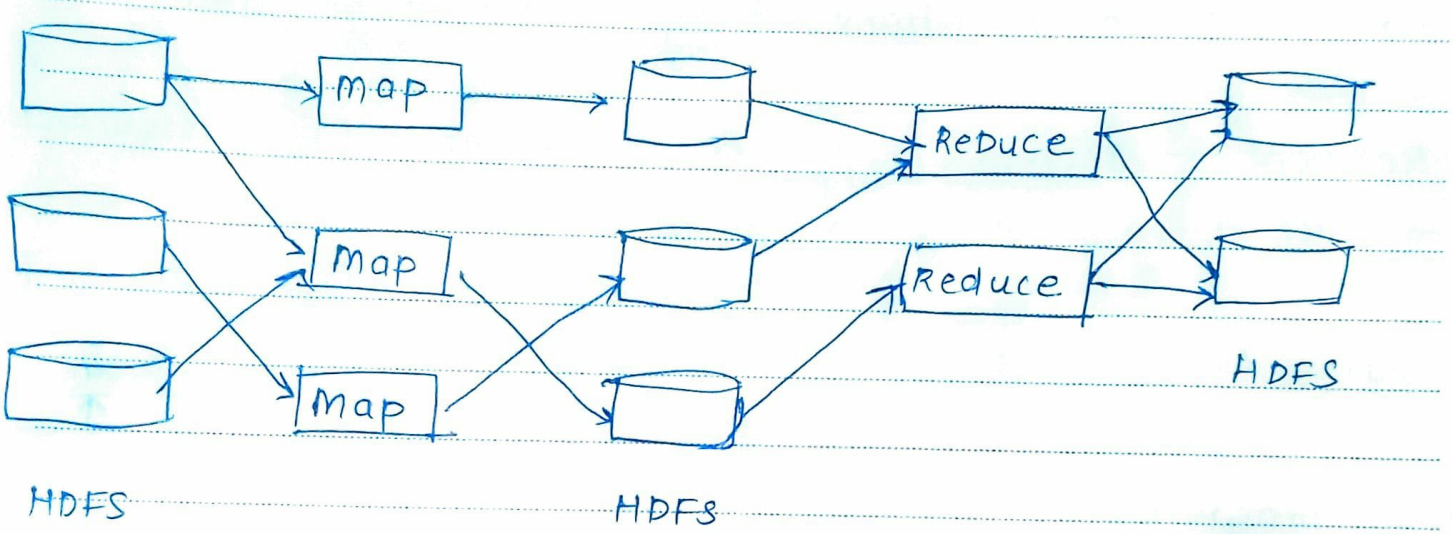
Describe how Spark does in-memory processing using RDD abstraction.

→ Explain the inner workings of the Spark architecture

→ Summarize how Spark manages and executes code on clus

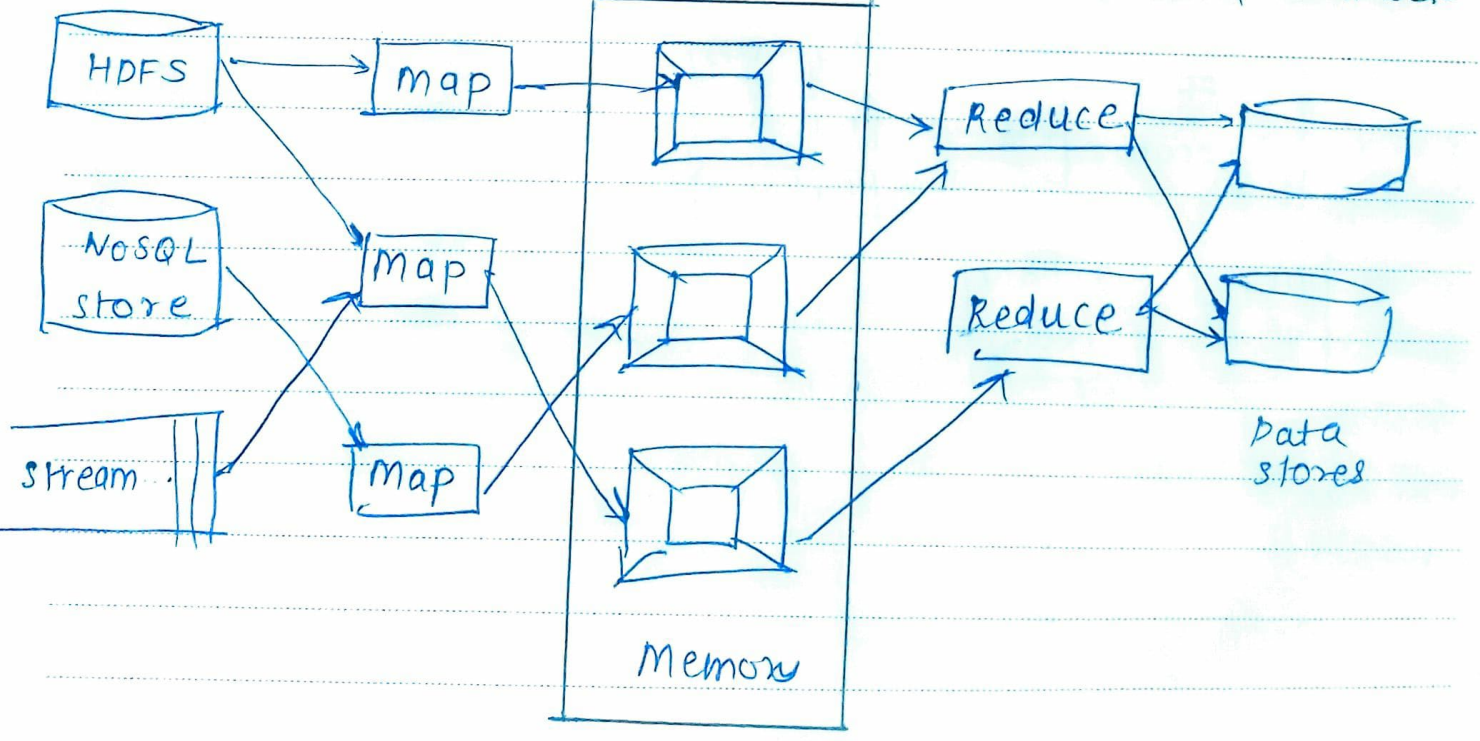
Q. What does in memory processing mean ?

MapReduce



Spark

Resilient Distributed Datasets.



Priority

Date

M / T / W / T / F / S / SU

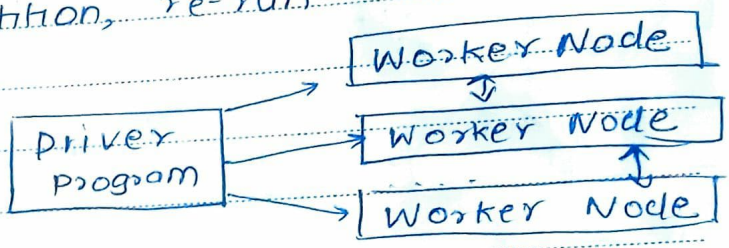
Resilient Distributed Datasets :-

① Dataset → Data storage created from :
HDFS, S3, HBase, JSON, text, Local Hierarchy
→ Or created transforming another RDD.

② Distributed → Distributed across the cluster of machines.
→ Divided in partitions, atomic chunks of data

③ Resilient → Recover from errors eg. node failure, slow processes
→ Track history of each partition, re-run

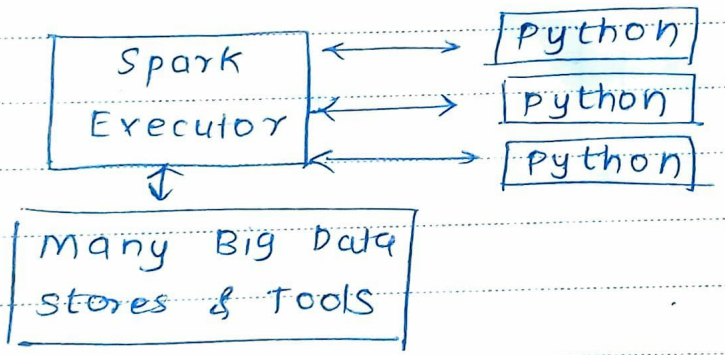
Spark Architecture :-



Driver program :-

```
lines = sc.textFile("          .txt")
```

Worker Node



Q. What are the layers of spark?

→ Graphx

→ Spark core

→ MLib

→ spark streaming

→ Spark SQL

Q. Why is Hadoop is not good platform for machine learning?

→ Bottleneck using HDFS

→ Map and Reduce Based computation.

→ No iterative shell and streaming

→ Java support ovs.

Q. What are three layers of Hadoop Ecosystem?

→ coordination and Workflow Managemeut

→ Data Mangemeut and storage

→ Data integration and processing

Q. What is Data-parallelism as defined

→ Running the same function simultaneously for the partitions of a data set on multiple cores.

Q. Which procedure best generalizes big data procedures, such as the map reduce process?

→ split → do → merge

NOTES

53 26 27 28 29 30 31 05 23 24 25 26 27 28 29

Q. 5 keypoints in order to categorize big data system

→ Execution model, Latency, scalability, Programming language, fault Tolerance.

Q. What is Lambda Architecture?

→ A type of hybrid data processing architecture.

Q. Which of the following scenarios not an aggregation operation?

→ Removing undefined values.

Q. What happens to data when it aggregated

→ Data becomes smaller.

Q. K-means clustering

→ Group samples into k clusters.

Q. Writing data to memory bt^h pipeline steps

↑↑

In-memory processing.

Sunday 27

NOTES