Foundations for Big Data Systems and Programming

Pejman Rasti

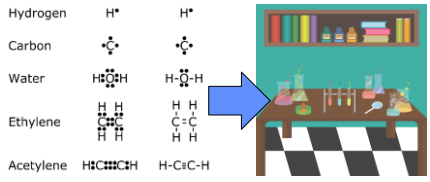
Email: prasti@esaip.org
pejman.rasti@univ-angers.fr

Course Website: Access from your "Moodle" portal

1

Why do we worry about foundations?

Chemistry before the tubes!



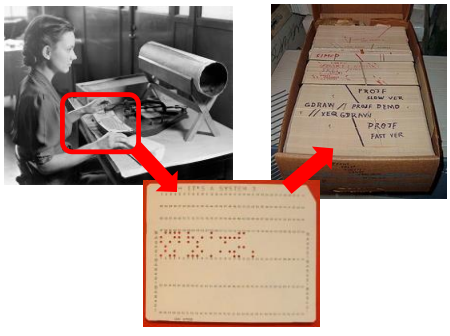
2

"It is not the beauty of a building you should look at; its the construction of the foundation that will stand the test of time."
-- David Allan Coe

3

What is a Distributed File System?:

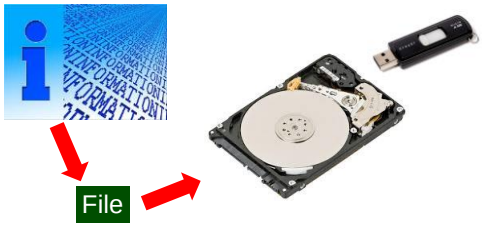
4



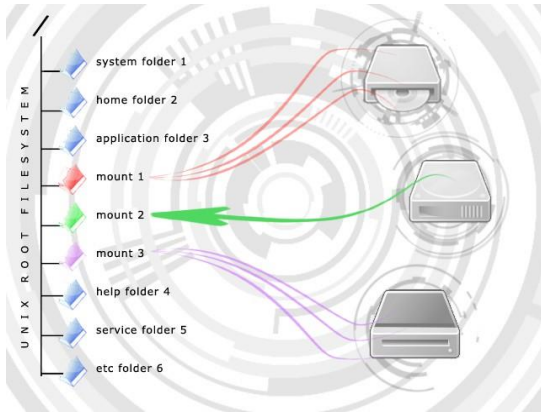
5

- Long-term information storage
- Access result of a process later
- Store large amounts of information
- Enable access of multiple processes

6



7



8

Accessing files

9

10

? What if you have more data?

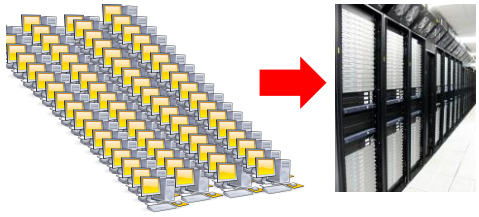
11

? Buy a bigger disk?

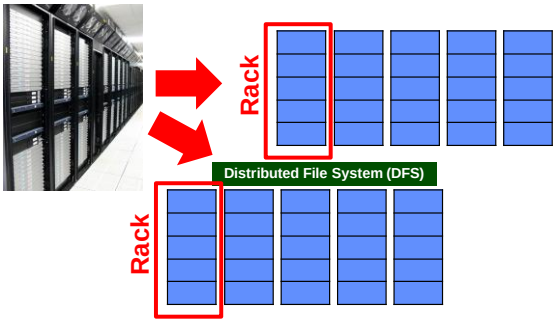
? Copy data to an external hard drive?

12

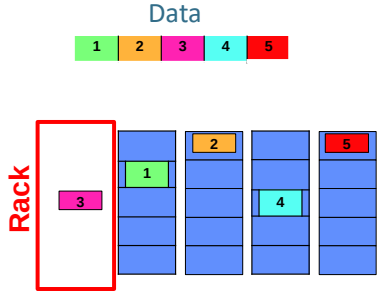
13



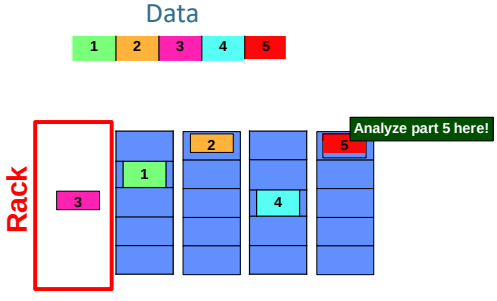
14



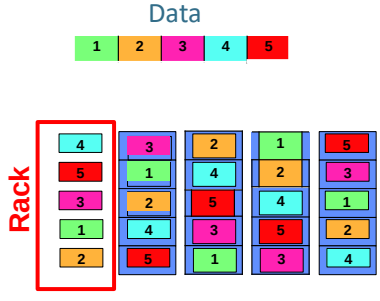
15



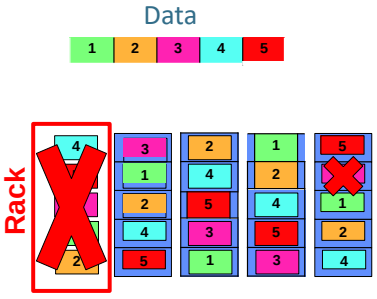
16



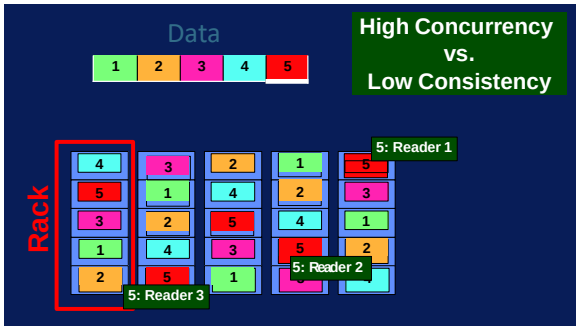
17



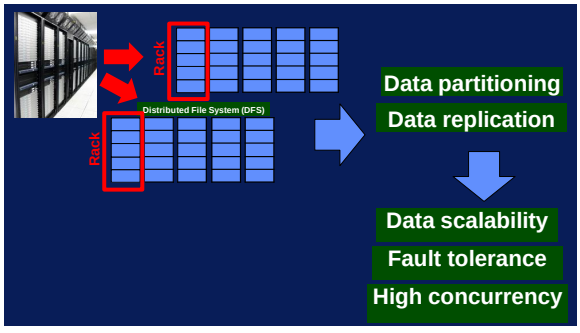
18



19

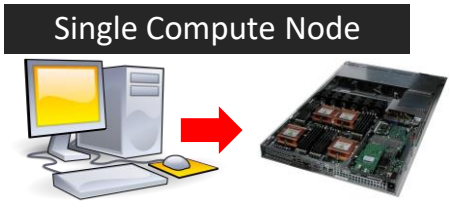


20

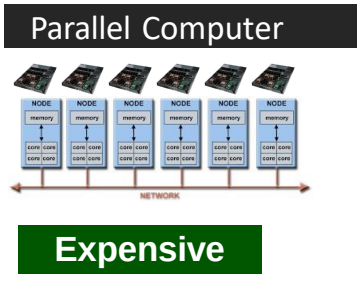


21

Scalable Computing over the Internet



22



23

24

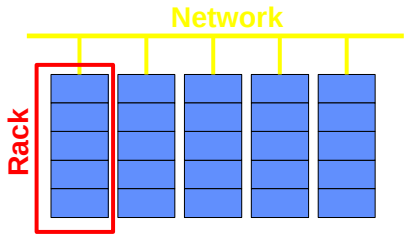
Commodity Cluster

Reduced computing cost

"Distributed computing" over the Internet

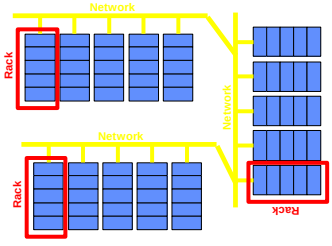
25

Architecture of a Commodity Cluster



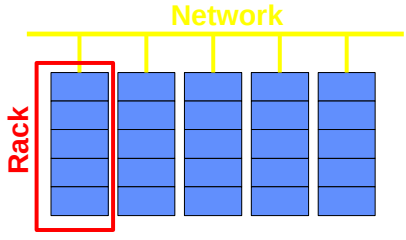
26

Distributed Computing



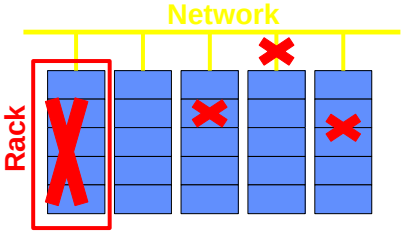
27

Enables data-parallelism



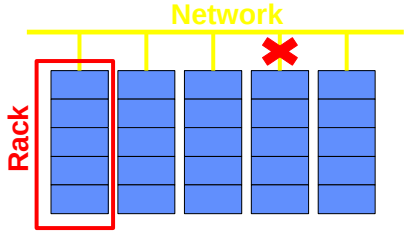
28

Common failures in commodity clusters



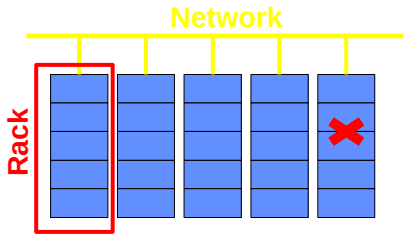
29

Common failures in commodity clusters



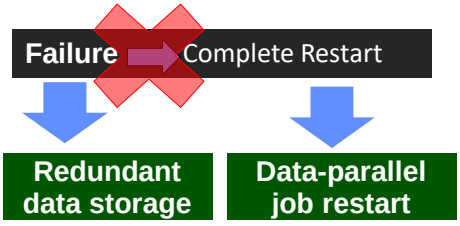
30

Common failures in commodity clusters

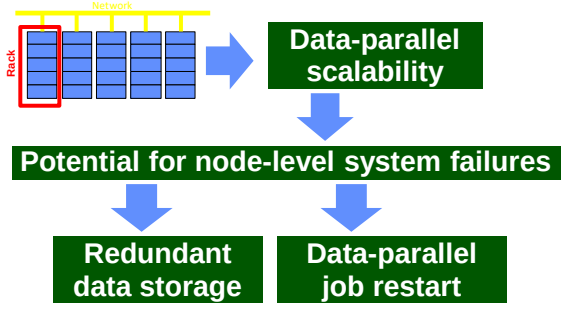


31

32



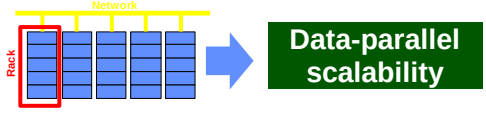
33



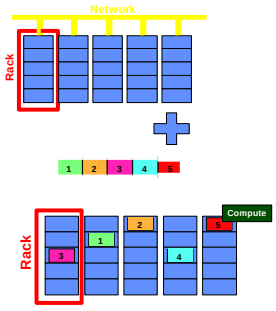
34

Programming Models for Big Data

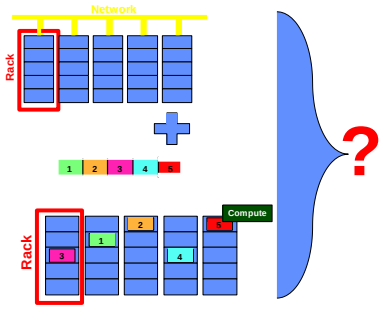
35



36



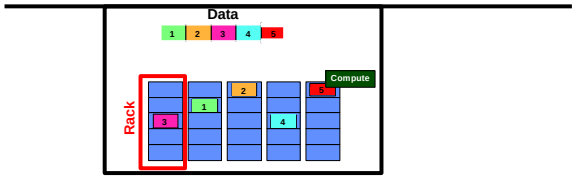
37



38

Programming Model = abstractions

Runtime Libraries + Programming Languages



39

Programming Model for Big Data

Programmability on top of Distributed File Systems

40

Requirements for Big Data Programming Models

41

1. Support Big Data Operations

Split volumes of data

Access data fast

42

1. Support Big Data Operations

Split volumes of data

Access data fast

Distribute computations to nodes

43

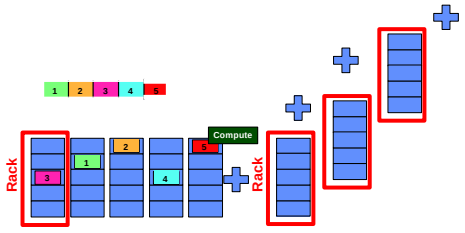
2. Handle Fault Tolerance

Replicate data partitions

Recover files when needed

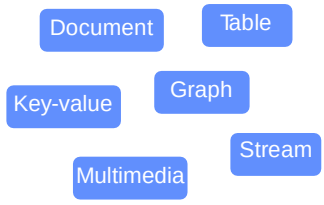
44

3. Enable Adding More Racks



45

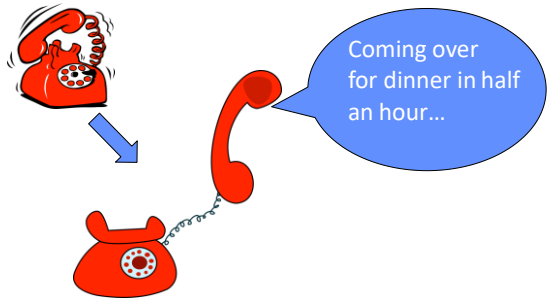
4. Optimized for specific data types



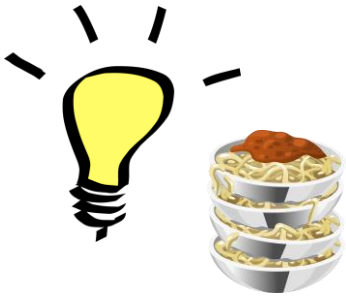
46

Natural model for independent parallel tasks over multiple resources!

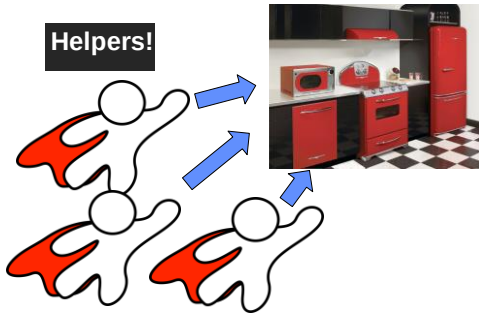
47



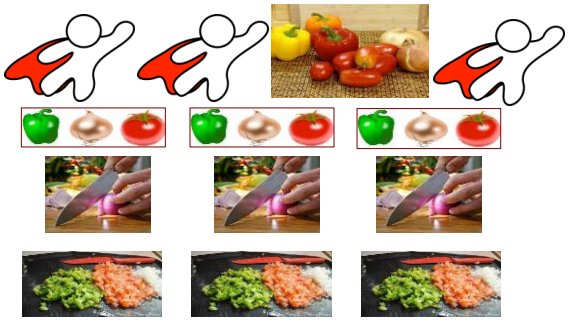
48



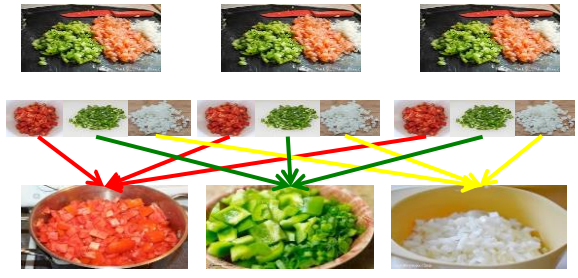
49



50



51



52

MapReduce

A programming model for Big Data

Many implementations

53

Programming Model = abstractions

Runtime Libraries + Programming Languages

Support large data volumes

Provide fault tolerance

Enable scale out

MapReduce

54

Why Hadoop?

55

"The Hadoop Ecosystem is great for Big Data"

The 4 W's (and H):

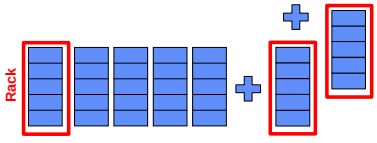
- ***What's** in the ecosystem?*
- ***Why** is it beneficial?*
- ***Where** is it used?*
- ***Who** uses it?*
- ***How** do these tools work?*

56

Major Goals

1. Enable Scalability

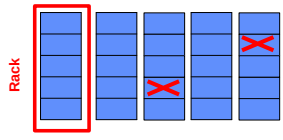
Commodity hardware is cheap



57

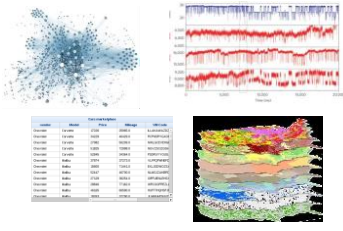
2. Handle Fault Tolerance

Be ready: crashes happen



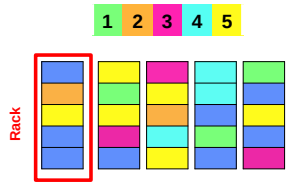
58

3. Optimized for a Variety Data Types



59

4. Facilitate a Shared Environment



60

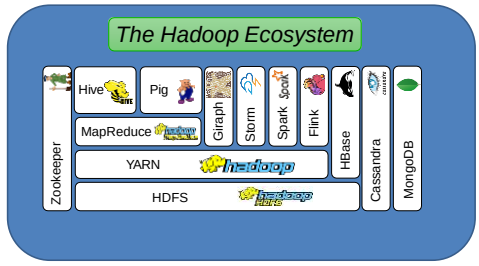
5. Provide Value

Community-supported

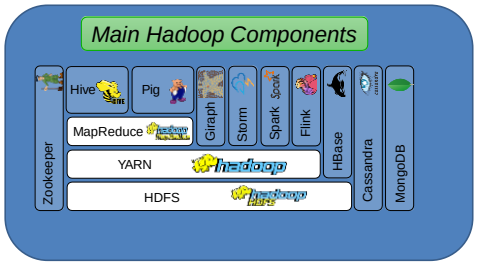
Wide range of applications

The Apache Software Foundation
http://www.apache.org/

61



62



63

65

The Hadoop Ecosystem:

So much free stuff!

Yahoo created Hadoop in 2005

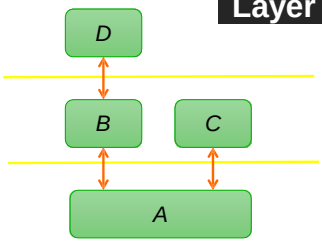
66

More Big Data frameworks released

67

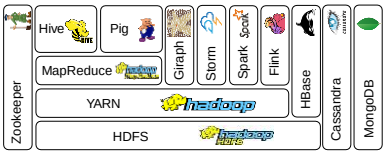
68

Layer Diagram



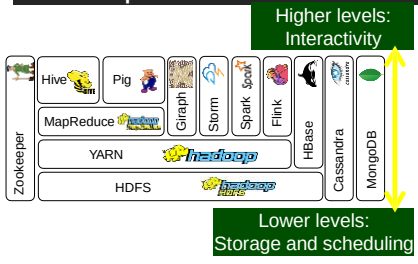
69

One possible layer diagram for Hadoop



70

One possible layer diagram for Hadoop

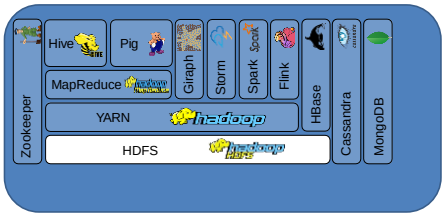


71

Distributed file system as foundation

Scalable storage

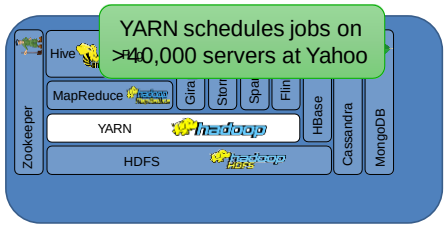
Fault tolerance



72

Flexible scheduling and resource management

YARN schedules jobs on >40,000 servers at Yahoo



73

Simplified programming model

Map → apply()

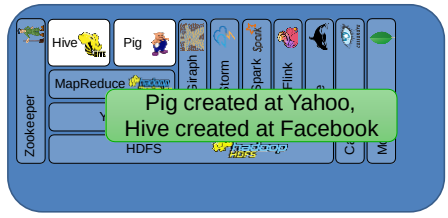
Reduce → summarize()

74

Higher-level programming models

Pig = dataflow scripting

Hive = SQL-like queries



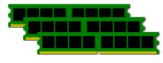
75

Specialized models for graph processing

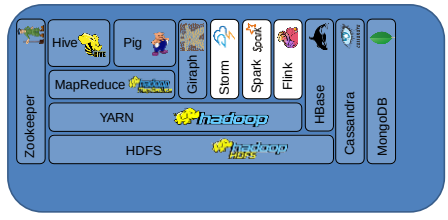
Giraph used by Facebook to analyze social graphs

76

Real-time and in-memory processing



In-memory → 100x faster for some tasks

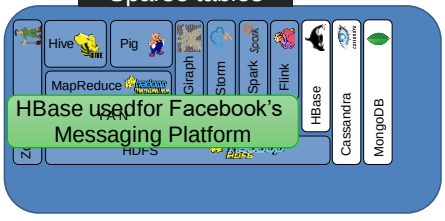


77

NoSQL for non-files

Key-values

Sparse tables



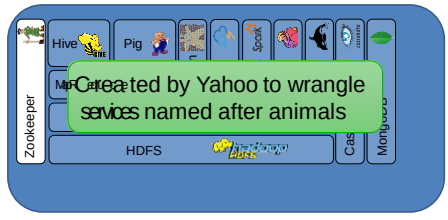
78

Zookeeper for management

Synchronization

Configuration

High-availability



79

All these tools are open-source

80

All these tools are open-source

Large community for support

81

All these tools are open-source

Large community for support

Download separately or part of pre-built image

82

All these tools are open-source

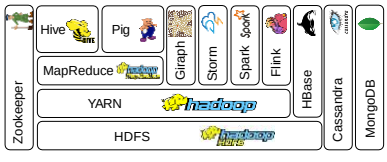
Large community for support

Download separately or part of pre-built image

cloudera

MAPR Hortonworks

83



Growing number of open-source tools

84

The Hadoop Distributed File System (HDFS):

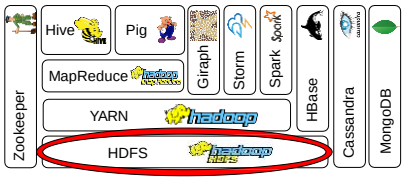
A Storage System for Big Data

85

HDFS = foundation for Hadoop ecosystem

Scalability

Reliability



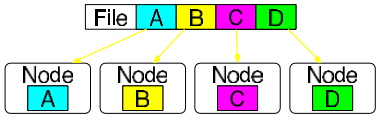
86

Store massively large data sets

up to 200 Petabytes,
4500 servers,
1 billion files and blocks!

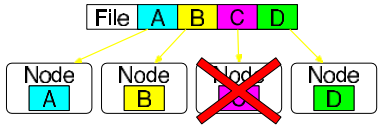
87

HDFS splits files across nodes for parallel access



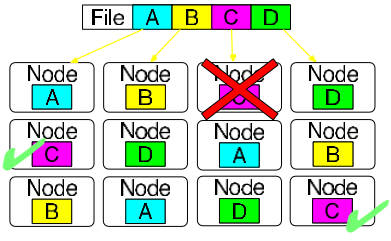
88

What happens if node fails?



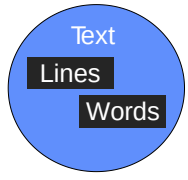
89

Replication for fault tolerance



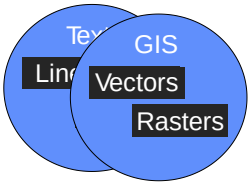
90

Customized reading to handle *variety* of file types



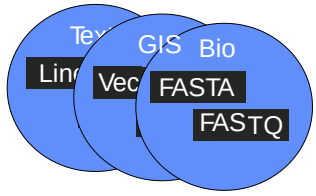
91

Customized reading to handle *variety* of file types



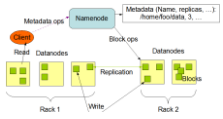
92

Customized reading to handle *variety* of file types



93

Two key components of HDFS



1. NameNode for metadata
   Usually one per machine
2. DataNode for block storage
   Usually one per machine

94

The NameNode coordinates operations

Keeps track of file name, location in directory, etc.

Mapping of contents on DataNode.

95

DataNode stores file blocks

Listens to NameNode for block creation, deletion, replication

96

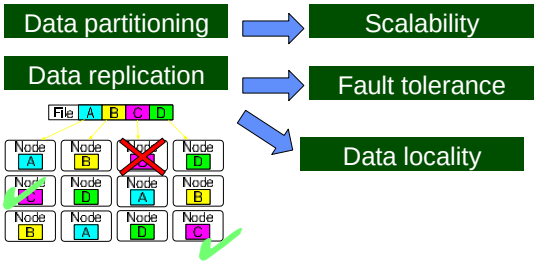
DataNode stores file blocks

Listens to NameNode for block creation, deletion, replication

Fault Tolerance

Data locality

97

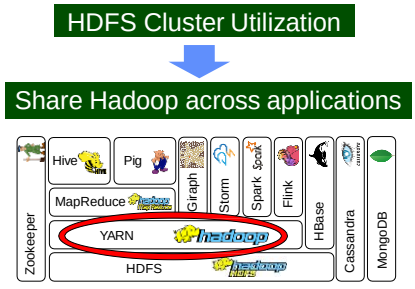


98

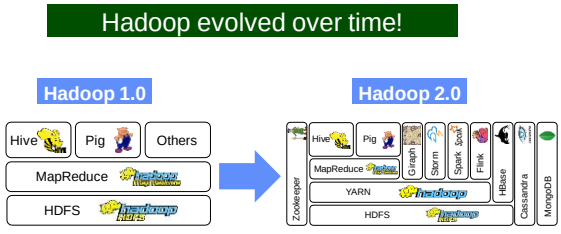
YARN:

The Resource Manager for Hadoop

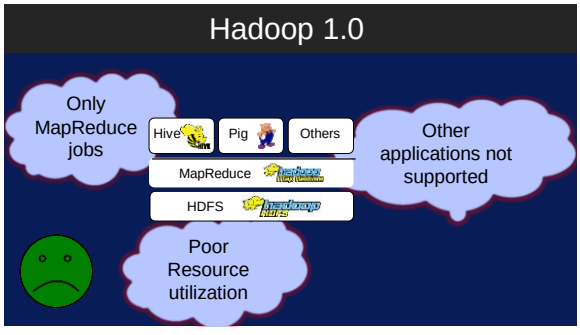
99



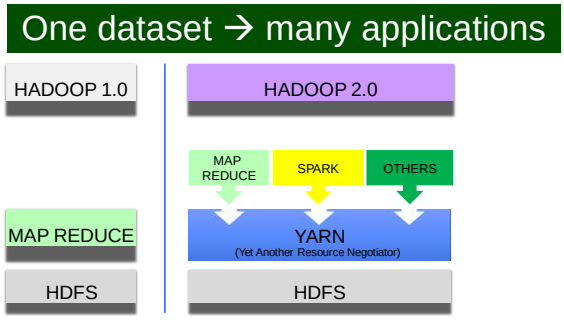
100



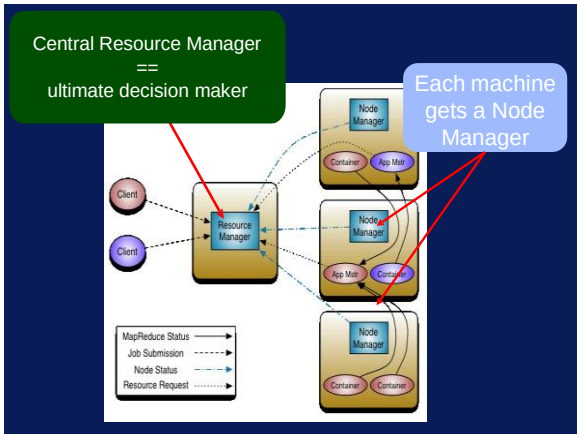
101



102



103



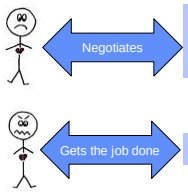
104

Resource Manager + Node Manager

= Data Computation Framework

105

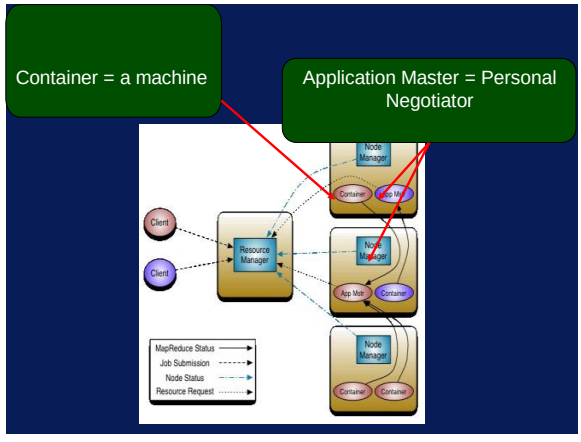
Application Master = personal negotiator



Resource Manager

Node Manager

106



107

Essential gears in YARN engine

- Resource Manager
- Applications Master
- Node Manager
- Container

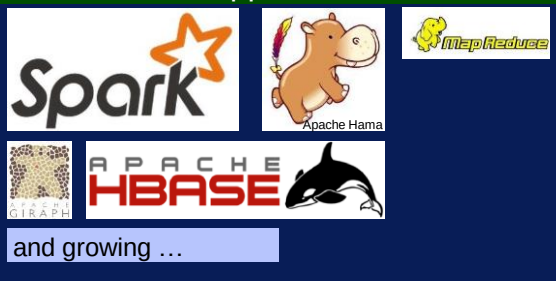
108

YAHOO!

- 2X ↑ Jobs per day
- 2X ↑ CPU utilization
- 2.5X ↑ Number of tasks from all jobs

109

YARN → More Applications



110

Data → Value

Many choices in Hadoop 2.0

One dataset → Many applications

Higher Resource Utilization → Lower Cost

111

MapReduce:

Simple Programming for Big Results

112

MapReduce = Programming Model for Hadoop Ecosystem

113

Parallel Programming = Requires Expertise

114

MapReduce = Only Map and Reduce!

115

Based on Functional Programming

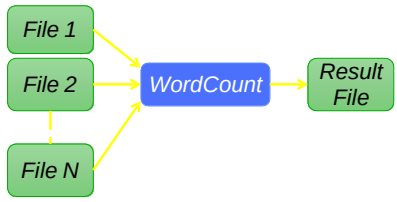
Map = apply operation to all elements

f (x) = y

Reduce = summarize operation on elements

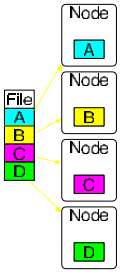
116

Example MapReduce Application: WordCount



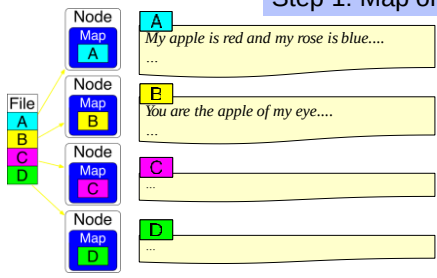
117

Step 0: File is stored in HDFS

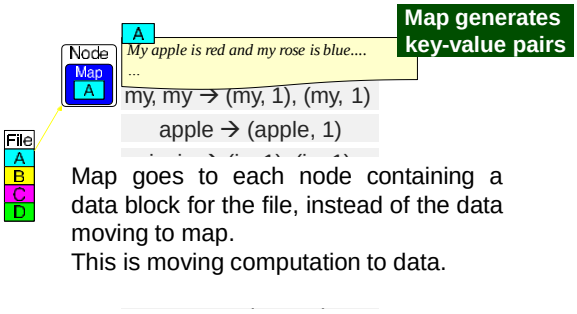


118

Step 1: Map on each node

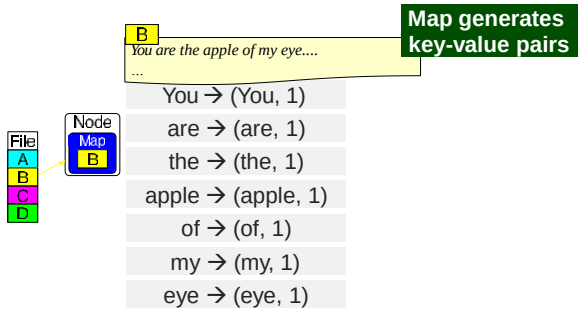


119

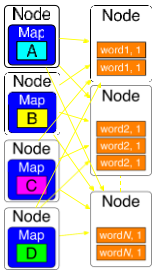


Map goes to each node containing a data block for the file, instead of the data moving to map.
This is moving computation to data.

120



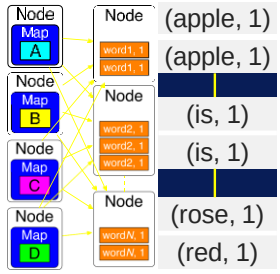
121



Step 2: Sort and Shuffle

Pairs with same key moved to same node

122



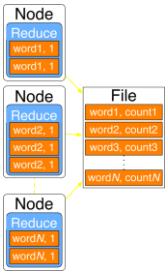
Step 2: Sort and Shuffle

Pairs with same key moved to same node

123

Step 3: Reduce

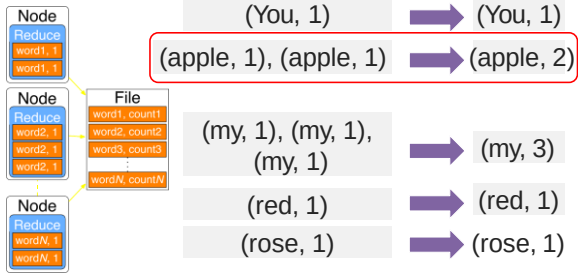
Add values for same keys



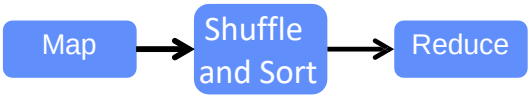
124

Step 3: Reduce

Add values for same keys



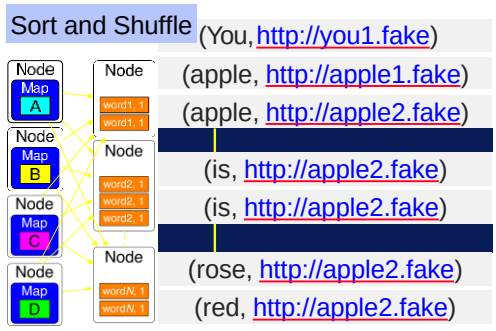
125



Represents a large number of applications.

126

Sort and Shuffle



127

Reduce Results for “apple”

(apple -> http://apple1.fake,
http://apple2.fake)

128

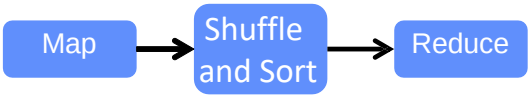
Reduce Results for “apple”

Key Value

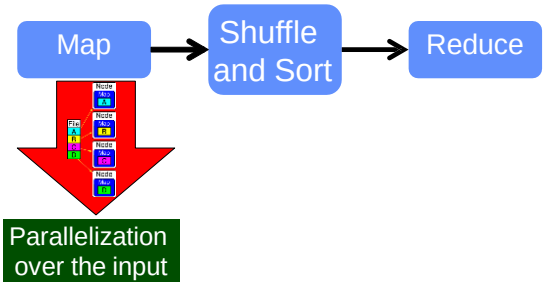
• apple -> http://apple1.fake,
http://apple2.fake)

• apple

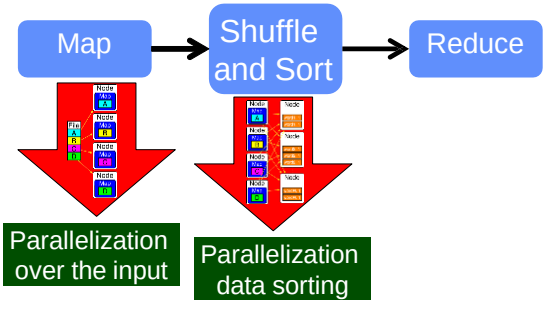
129



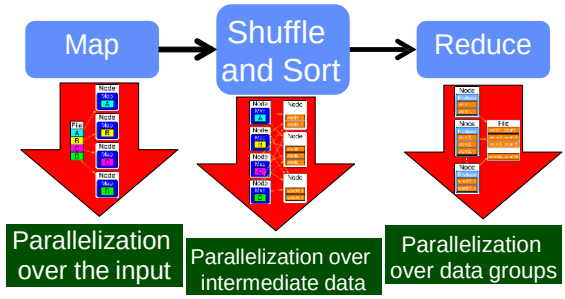
130



131



132



133

MapReduce is bad for:

134

MapReduce is bad for:

Frequently **changing** data

135

MapReduce is bad for:

Frequently **changing** data

Dependent tasks

136

MapReduce is bad for:

Frequently **changing** data

Dependent tasks

Interactive analysis

137

MapReduce

Simplified parallel programming

Applications with independent data-parallel tasks

138

When to reconsider Hadoop ?

139

Future anticipated data growth

Long term availability of data

Hadoop

140

Many platforms over single data store

High Volume

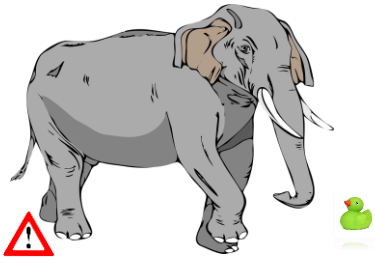
High Variety

Hadoop

141

Small Datasets

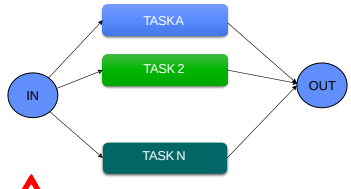
CAUTION!



142

Task Level Parallelism

CAUTION!



143

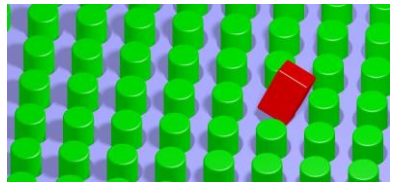
Advanced Algorithms

CAUTION!

144

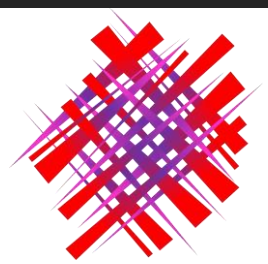
Replacement to your infrastructure

CAUTION!



145

Random Data Access



146

Advanced Analytical Queries

https://tez.apache.org/

Latency sensitive tasks

https://drill.apache.org/
http://storm.apache.org/

Security of Sensitive Data

http://ranger.apache.org

147

Small Datasets

Advanced Algorithms

Infrastructure Replacement

Task Level Parallelism

Random Data Access

148

Value from Hadoop and Pre-built Hadoop Images

149

Download, Configure, Install

150

Pre-built Software images

151

Virtualization software

152

Pre-built Images for Hadoop

153

Hortonworks Sandbox

154

cloudera

155