Edge Hill University

Cloud Computing

Topic 8A, Session 4

Content

Big Data Explosion

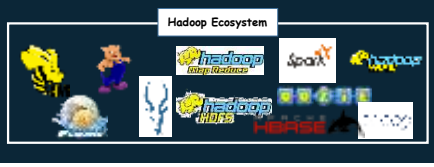
- In 2011 the digital universe became **10 times** the size it was in 2006
- 2015: **7.9 Zettabytes** of data
  - Enough data to fill 135.7 billion iPads (64 GB storage/iPad)
- 2017: Google processes **100 petabytes/day**
  - YouTube: 1,000 PB database; 4 billion views/day

Solution to Big Data Problem

- **Word in parallel**
  - Break 1 TB into smaller blocks of data
  - Read smaller blocks of data in parallel
  - Combine reading jobs to get final result
- **Divide and Conquer** paradigm
  - **Divide:**
    - break large problem into smaller sub-problems
    - Solve smaller sub-problems
  - **Conquer:**
    - Combine results from all sub-problems to get final solution

What is Hadoop?

- Currently: Hadoop is an **Ecosystem**
  - A collection of different software components for:
    - Distributed Storage (files are stored in many computers)
    - Distributed processing (programs are executed by many computers)

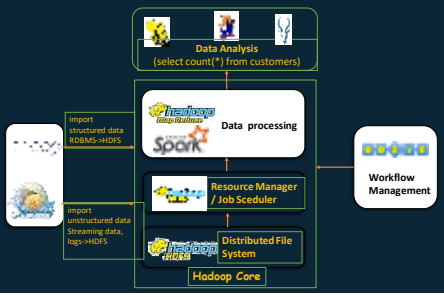


Improved scalability
- Horizontal scaling (adding more computers)
- Vertical Scaling (resizing existing computers – more RAM)

High Fault Tolerant
- Resource Replication
- Stores copies of data on multiple machines

Low Complexity of Distributed Programming
- Various Hadoop Frameworks that are tailored to divide and Conquer programming (e.g., MapReduce, Spark)

Organization of the Hadoop Ecosystem



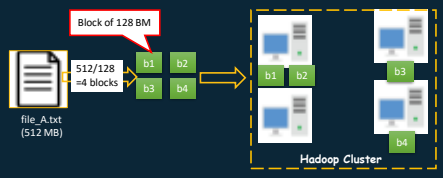
Hadoop Distributed File System (HDFS)

- File Systems: Manages how data is **stored** and **retrieved**
  - New Technology File System (NTFS) - Microsoft Windows
  - Distributed File System (DFS) - Microsoft Windows servers
  - Virtual Machine File System (VMFS) - File System in Virtual Machines
  - Fourth Extended File System (ext4) - Linux
- **H**adoop **D**istributed **F**ile **S**ystem (**HDFS**)
  - Manages storage and retrieval of very large files across a network of machines

HDFS vs. Conventional File Systems

- How is HDFS **different** than other typical file systems (e.g., NTFS) ?
  - **File blocks** are larger in HDFS than in conventional file systems
  - Any File System breaks a file into blocks, which are stored as independent units
  - Typical File System: size of single block is **512 bytes**
  - HDFS: size of single block is **128 MB**

HDFS - Blocks

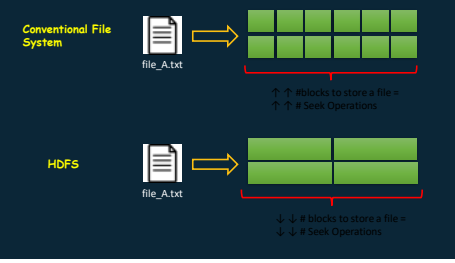


Why is a Block in HDFS so Large?

- Time to retrieve a file = Time to locate it + Time to read it
- Locating a file: Seek operation
  - Locating the beginning of a block
  - Many blocks = Many Seek Operations = Long time to locate a file
- In HDFS, we use a large block size of 128 MB to:
  - Minimise the number of blocks that we need to store a file
  - Minimise the seek operations to locate a file
  - Minimise the total time to retrieve a file

Worthy to note that a too large block size is also discouraged, since putting all or most of a file data in a single block or very few blocks increases the probability of data corruption (a single block corruption would corrupt the data for a whole file)

Why is a Block in HDFS so Large?



Hadoop MapReduce

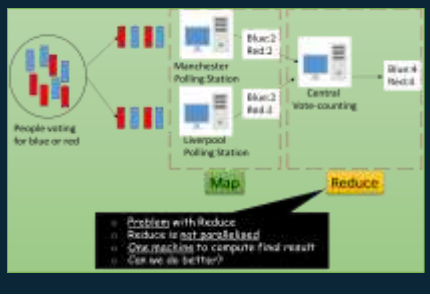
- MapReduce is a programming framework for developing parallel, distributed algorithms based on the *divide and conquer paradigm*

1. Parallel: different processes of the MapReduce algorithm run in parallel
1. Distributed: different processes of the MapReduce algorithm run on multiple computers

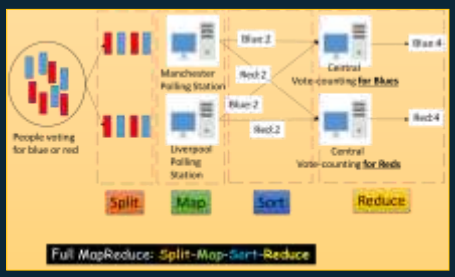
Hadoop MapReduce – Analogies & Examples

- Vote counting in national elections
  - **Map:** Count votes in each polling station
    - Counting runs in parallel
    - The more polling stations we have, the faster we will finish
  - **Reduce:** Collect vote counts from each polling station and compute final total count
- Count number of books in a library
  - **Map:** Count number of books in each shelf
    - Counting runs in parallel
    - The more people we have counting, the faster we will finish
  - **Reduce:** Collect number of books for each shelf and compute final number

Hadoop MapReduce – Voting System



Full Version of MapReduce



Full Version of MapReduce

Split
- Splits input into segments
- Problem-> sub-problem (e.g., voters into polling stations)

Map
- Takes as an input a segment and works to produce a local solution
- Mappers can run in parallel

Sort
- Groups output of Map functions by a common key (e.g., red/blue vote)
- Sends each group to a different reducer

Reduce
- Receives group of Map output
- Aggregates solution for that group
- Reducers can run in parallel

Practical Example of MapReduce

Develop a MapReduce program that computes the frequency of individual words in a given file

Practical Example of MapReduce

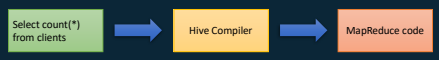
Develop a MapReduce program that computes the maximum temperature for each month across 2015, 2016 and 2017

Hive

- Developed by Facebook in 2007
- Currently an open source project maintained by different companies (e.g., Facebook, Netflix)
- Hive: **Data warehouse** software that uses an **SQL-like interface** to
  - Write, store and retrieve very large datasets from HDFS
  - Analyse very large datasets

Hive

- Solution of Hive
  - Write in SQL-like syntax -> automatically translate into MapReduce
  - Essentially a **MapRecuce wrapper** for data warehousing
  - **HiveQL queries**: Similar to SQL queries but some differences



Hive vs. RDBMS

- How is Hive different than an RDBMS?
  - RDBMS: Full SQL support
  - Hive: Limited SQL support

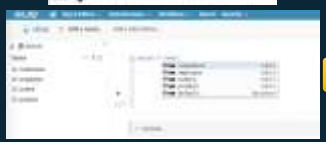
	RDBMS	Hive
Update Individual Records	✓	✗
Delete Individual Records	✓	✗
Data validation	✓	Limited (no primary/foreign keys)
Fast when processing Small databases	✓	✗
Scales to large databases (Petabyte)	✗	✓

Accessing Hive

There are two different ways that you can use to interact with Hive:

1. Command line interface (CLI)
2. Hive web-based interface

Hive Command Line Interface



Hive web-based Interface

Summary - Hive

- Datawarehouse software: translates SQL-like commands into MapReduce
  - Abstracts MapReduce programming complexity
- Partial support of SQL (unlike RDBMS)
  - No delete, update commands for individual instances
  - No primary/foreign keys
- Slower than RDBMS for small, medium-sized datasets
- Scales to large databases (e.g., Petabytes)
  - RDBMS cannot scale to Petabytes of data


Thank
You