GMD 2A: MapReduce

Sabeur Aridhi

TELECOM Nancy, University of Lorraine

Contents

Growth of data size

Data sets

We need to analyze bigger and bigger datasets:

- **Web datasets:** web topology (Google: est. 46 billion pages)
- **Network datasets:** routing efficiency analysis, tcp-ip logs
- **Biological datasets:** genome sequencing data, protein patterns

Big Data

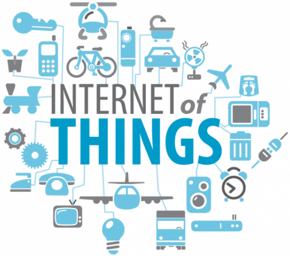
Three main characteristics:

- **Volume:** Huge size of datasets (i.e., distributed storage).
- **Variety:** Several types of data such as text, image and video.
- **Velocity:** Data is generated at high speed (i.e., CERN experiments: one petabyte /sec).

Key sources of Big Data

Internet of Things (IoT)

- Networking of hardware equipment (e.g., household appliances, sensors, mobile pohones)
- A significant source of Big Data



Key sources of Big Data

Smart cities

- Use of information technology to enhance quality, performance and interactivity of urban services in a city.
- Data is collected from sensors installed on:
  - utility poles
  - water lines
  - buses
  - trains
  - traffic lights, ...

Key sources of Big Data - Smart cities

Big Data in everyday life

Big Data in everyday life

Does one minute fit into RAM? Five Minutes?

Distributed Computing (DC)

- Requires a **Programming Model** (PM) that is inherently parallelizable.
- Requires a **framework** that runs the program and provides fault-tolerance and scalability.

Distributed Computing (DC)

A **programming model** for DC should:

- abstract all low level details such as networking, scheduling, ...
- allow for wide range of functionality
- be easy to use

A **framework** for DC should provide:

- fault tolerance
- scalability
- data integrity

Distributed File System

- Big Data requires storage over many machines
- Each computing node needs to access the data.
- A distributed file system (DFS) abstracts the distributed storage
- A widely used file system (FS) is Google DFS

Distributed File System

- Big Data requires storage over many machines
- Each computing node needs to access the data.
- A distributed file system (DFS) abstracts the distributed storage
- A widely used file system (FS) is Google DFS

Google's DFS

- data is replicated across the network. (Failure resistent, faster read operations).
- "namenode" stores file metadata e.g. name, location of chunks

Distributed File System

- Big Data requires storage over many machines
- Each computing node needs to access the data.
- A distributed file system (DFS) abstracts the distributed storage
- A widely used file system (FS) is Google DFS

Google's DFS

- data is replicated across the network. (Failure resistent, faster read operations).
- "namenode" stores file metadata e.g. name, location of chunks
- "datanode" stores chunks of that file

Contents

Definition

Google, 2004:

> *MapReduce refers to a programming model and the corresponding implementation for processing and generating large data sets.*

Created to help Google developers to analyze huge datasets

Map and Reduce in MapReduce

- Data format: **key-value** pairs
- The user (you) must define two functions:

Mapper

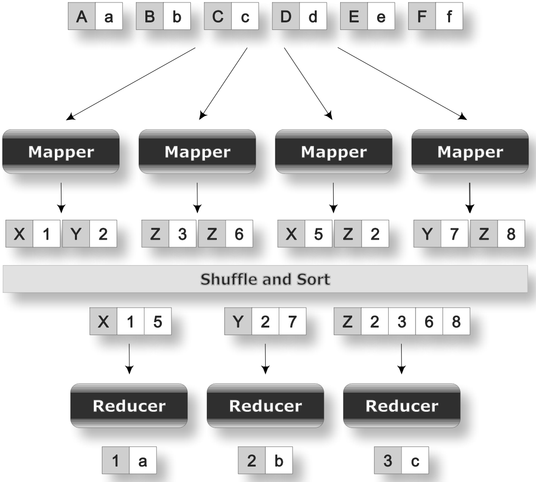
Map function: $(k_1, v_1) \rightarrow \text{list}(k_2, v_2)$

Reducer

Reduce function: $(k_2, \text{list}(v_2)) \rightarrow \text{list}(v_3)$

- The Map functions are executed in parallel.
- k_2, v_2 are *intermediate* pairs.
- Mapper results are grouped using k_2.
- The Reduce functions are executed in parallel.

MapReduce data flow (simplified)



Basic steps of a MapReduce program:

- *Input is read from file system (fs)*
- The Map function is executed in parallel.
- *These results are written to fs*
- *Input is read from fs*
- The Reduce function is executed in parallel.
- *These results are written to fs*

Example: Word Count (MapReduce's Hello World)

Input

A set of documents stored in a DFS

Output

The number of occurrences of each word in the set of documents.

Example: Word Count (MapReduce's Hello World)

Input

A set of documents stored in a DFS

Output

The number of occurrences of each word in the set of documents.

Idea

Work on each document in parallel and then "reduce" the results for each word.

Example: Word Count (MapReduce's Hello World)

Algorithm: Map function

Input: String filename, String content

foreach word w **in** content **do**

  EmitIntermediate(w,*1*);

Algorithm: Reduce function

Input: String key, Iterator values

result $\leftarrow 0$;

foreach v **in** values **do**

  result $\leftarrow$ result $+ v$;

Emit (key,result);

Possible optimization

Creating a pair for each occurrence of a particular word is inefficient

Algorithm: Optimized Map function

Input: String filename, String content

$H \leftarrow$ **new** HashTable;

% H is an associative array that maps keys to values

foreach word w **in** content **do**

$\quad H[w] \leftarrow H[w] + 1$

foreach word w **in** H **do**

$\quad$ EmitIntermediate(w,$H[w]$)

Combiner and Partitioner

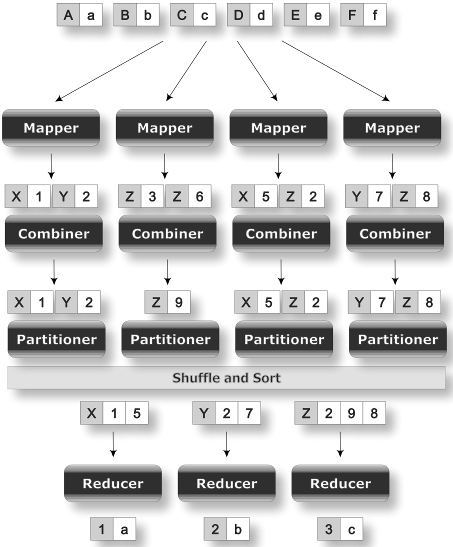
Combiner

- Operates between Mapper and Reducer.
- $Combiner : (k_2, \text{list}(v_2)) \rightarrow (k_2, \text{list}(v_3))$
- The combiner is applied before the global sort.
- Just like in the WordCount example.

Partitioner

Before global sort, decide which reducer receives which key.

Complete MapReduce flow



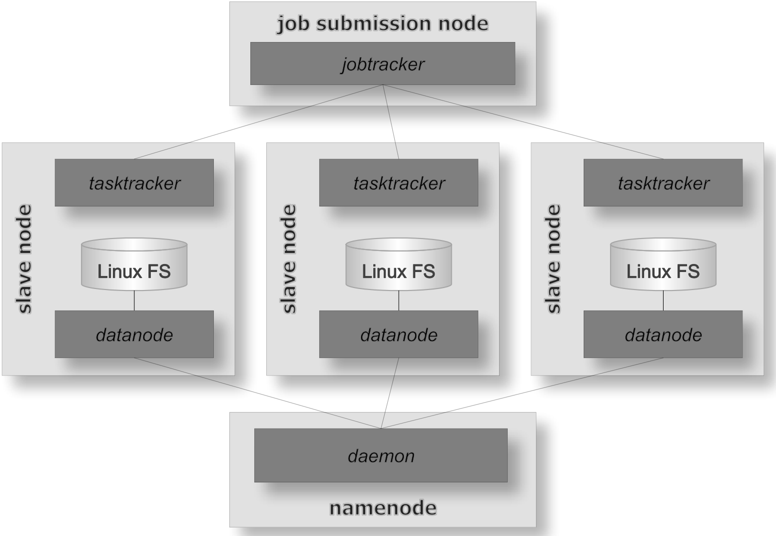
Fault tolerance management

- In case of a worker failure
  - map (and not reduce) tasks completed reset to idle state
  - map and reduce tasks in progress reset to idle
  - Notification to and re-execution by workers executing reduce tasks
- in case of Master failure
  - Version of 2004:
    - Abort the computation and retry
  - In MRv2 (YARN):
    - Periodic check points of the data structures
    - Launching of a new copy from the check point

Framework

- Google's MapReduce framework is not freely available
- Open source alternative: Apache's Hadoop
- Offered as service in Amazon Elastic Computing (via virtual machines)

Hadoop Architecture



Hadoop Details

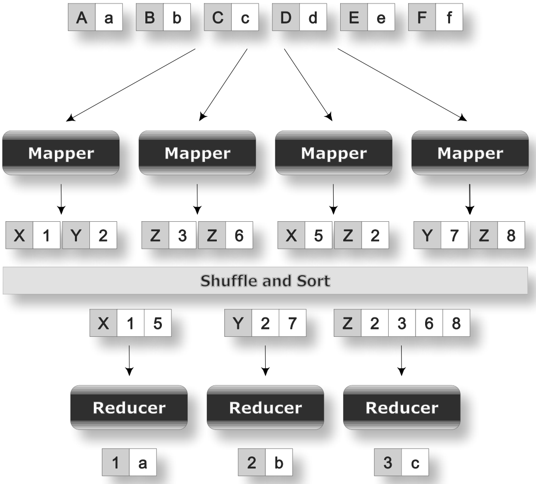
- Library in Java
- Started by Yahoo in 2006
- By default, Map and Reduce functions must be written in Java
- Possibility to use external programs as mapper and reducers

Inverted Index (Job interview question ;)

Problem

We are given a set of webpages $P = \{P_1, \ldots, P_n\}$.
We want to know for each word in the dataset, in which documents it appears (and how many times)

Reminder of MapReduce flow



Baseline solution

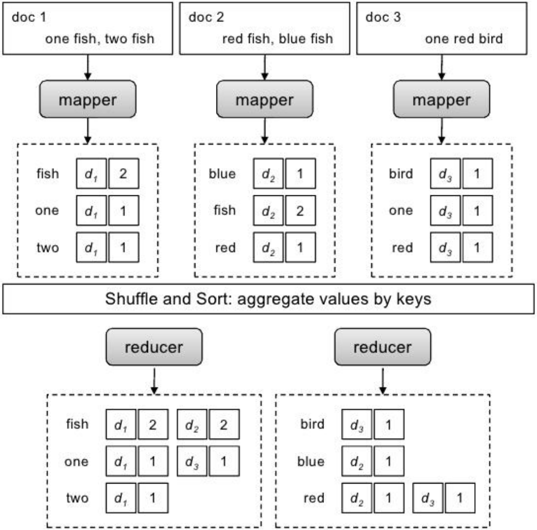
Algorithm: Map function(String filename, String content)

$H \leftarrow$ **new** HashTable;
foreach word w **in** content **do**
  $H[w] \leftarrow H[w] + 1$
foreach word w **in** H **do**
  EmitIntermediate(w,($filename, H[w]$))

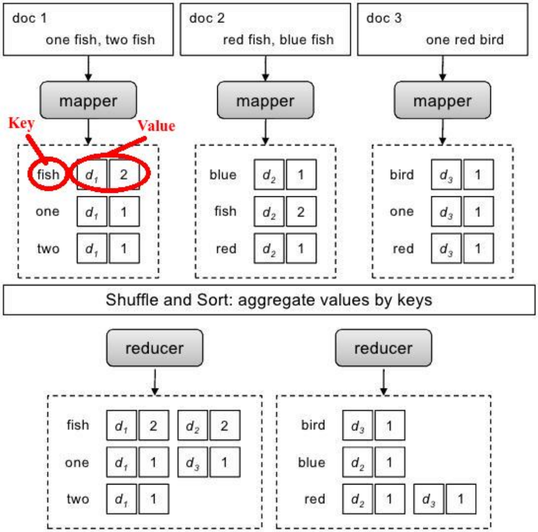
Algorithm: Reduce function(String key, Iterator values)

result $\leftarrow []$;
foreach v **in** values **do**
  result.$add(v)$;
result.$sortbyfreq()$;
Emit (key,result);

Simple run


doc 1
one fish, two fish
doc 2
red fish, blue fish
doc 3
one red bird
mapper
mapper
mapper
fish d_1 2
one d_1 1
two d_1 1
blue d_2 1
fish d_2 2
red d_2 1
bird d_3 1
one d_3 1
red d_3 1
Shuffle and Sort: aggregate values by keys
reducer
reducer
fish d_1 2 d_2 2
one d_1 1 d_3 1
two d_1 1
bird d_3 1
blue d_2 1
red d_2 1 d_3 1

Simple run


doc 1
one fish, two fish
doc 2
red fish, blue fish
doc 3
one red bird
mapper
mapper
mapper
Key
Value
fish d_1 2
one d_1 1
two d_1 1
blue d_2 1
fish d_2 2
red d_2 1
bird d_3 1
one d_3 1
red d_3 1
Shuffle and Sort: aggregate values by keys
reducer
reducer
fish d_1 2 d_2 2
one d_1 1 d_3 1
two d_1 1
bird d_3 1
blue d_2 1
red d_2 1 d_3 1

Contents

That's it. Why was this important?

Questions?
sabeur.aridhi@loria.fr