TDDD43

Theme NOSQL 4.2: DFS, Map-Reduce

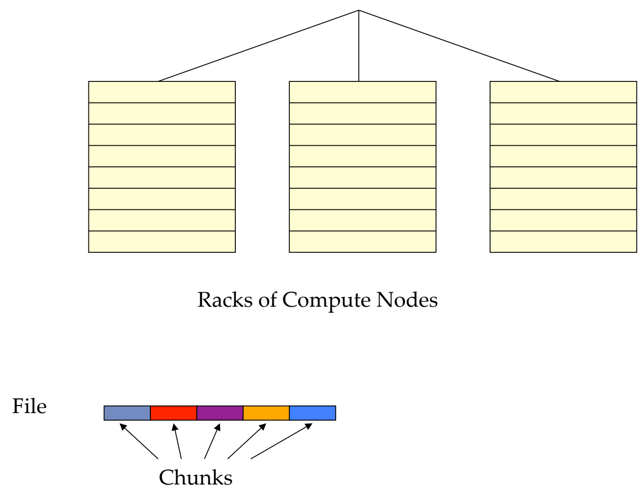
Material from Chapter 2 in
Mining of Massive Datasets
Anand Rajaraman, Jeffrey David Ullman
http://infolab.stanford.edu/~ullman/mmds.html (download the book)

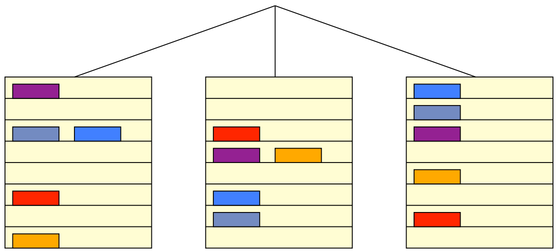
Distributed File System

- Files are very large, read/append.
- They are divided into *chunks*.
  - Typically 64MB to a chunk.
- Chunks are replicated at several *compute-nodes*.
- A *master* (possibly replicated) keeps track of all locations of all chunks.

Compute Nodes

- Organized into racks.
- Intra-rack connection typically gigabit speed.
- Inter-rack connection faster by a small factor.



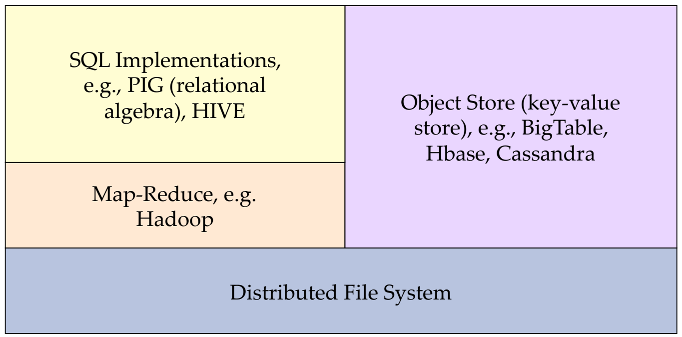


3-way replication of files, with copies on different racks.

Implementations

- *GFS* (Google File System – proprietary).
- *HDFS* (Hadoop Distributed File System – open source).
- *CloudStore* (Kosmix File System, open source).

The New Stack



Map-Reduce Systems

- Map-reduce (Google) and open-source (Apache) equivalent Hadoop.
- Important specialized parallel computing tool.
- Cope with compute-node failures.
  - Avoid restart of the entire job.

Key-Value Stores

- *BigTable* (Google), *Hbase, Cassandra* (Apache), *Dynamo* (Amazon).
  - Each row is a key plus values over a flexible set of columns.
  - Each column component can be a set of values.

SQL-Like Systems

- *PIG* – Yahoo! implementation of relational algebra.
  - Translates to a sequence of map-reduce operations, using Hadoop.
- *Hive* – open-source (Apache) implementation of a restricted SQL, called QL, over Hadoop.
- *Sawzall* – Google implementation of parallel select + aggregation.
- *Scope* – Microsoft implementation of restricted SQL.

Map-Reduce

- You write two functions, Map and Reduce.
  - They each have a special form to be explained.
- System (e.g., Hadoop) creates a large number of tasks for each function.
  - Work is divided among tasks in a precise way.

Word Count

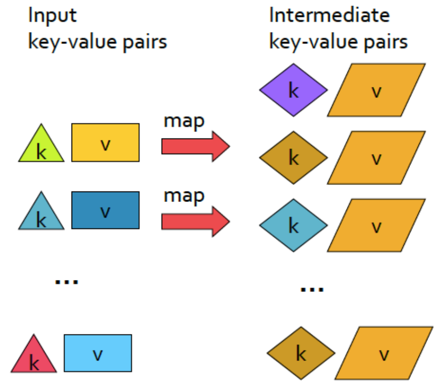
- We have a huge text document
- Count the number of times each distinct word appears in the file
- **Sample application:** Analyze web server logs to find popular URLs

MapReduce Overview

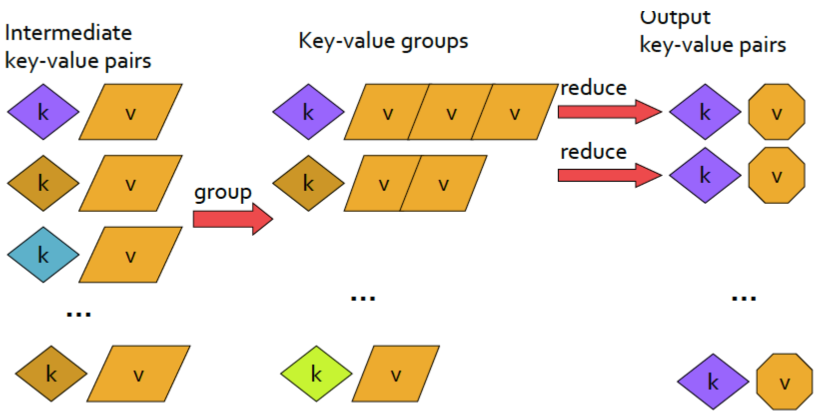
- Sequentially read a lot of data
- **Map:** Extract something you care about
- **Group by key:** Sort and Shuffle
- **Reduce:** Aggregate, summarize, filter or transform
- Write the result

✓ Outline stays the same, **map** and **reduce** change to fit the problem

Map step



Reduce step



More specifically

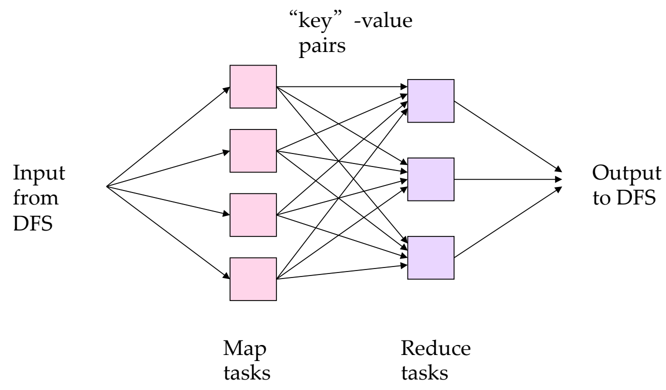
- **Input:** a set of key/value pairs
- Programmer specifies two methods:
  - **Map(k, v) →** <k',v'>*
    - Takes a key value pair and outputs a set of key value pairs (E.g., key is the filename, value is a single line in the file)
    - There is one Map call for every (k,v) pair
  - **Reduce(k', <v'>*) →** <k', v''>*
    - All values v' with same key k' are reduced together and processed in v' order
    - There is one Reduce function call per unique key k'

Word Count using MR

map(key, value):
// key: document name; value: text of the document
  for each word w in value:
    emit(w, 1)

reduce(key, values):
// key: a word; value: an iterator over counts
  result = 0
  for each count v in values:
    result += v
  emit(key, result)

Map-Reduce Pattern



MR environment

Map-Reduce environment takes care of:

- Partitioning the input data
- Scheduling the program's execution across a set of machines
- Handling machine failures
- Managing required inter-machine communication

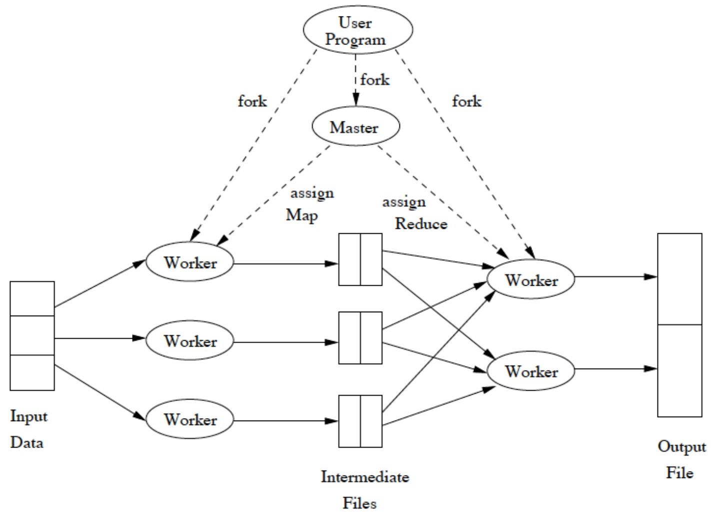


Figure 2.3: Overview of the execution of a map-reduce program

MapReduce Implementation Details

- The user program forks a Master controller process and some number of Worker processes at different compute nodes.
  - Normally, a Worker handles either Map tasks (a Map worker) or Reduce tasks (a Reduce worker), but not both.
- The Master creates some number of Map tasks and some number of Reduce tasks
  - These numbers being selected by the user program.
  - These tasks will be assigned to Worker processes by the Master.

MapReduce Implementation Details

- The Master keeps track of the status of each Map and Reduce task (idle, executing at a particular Worker, or completed).
- A Worker process reports to the Master when it finishes a task, and a new task is scheduled by the Master for that Worker process.

MapReduce Implementation Details

- Each Map task is assigned one or more chunks of the input file(s) and executes on it the code written by the user.
- The Map task creates a file for each Reduce task on the local disk of the Worker that executes the Map task.
- The Master is informed of the location and sizes of each of these files, and the Reduce task for which each is destined.
- When a Reduce task is assigned by the Master to a Worker process, that task is given all the files that form its input.
- The Reduce task executes code written by the user and writes its output to a file that is part of the surrounding distributed file system.

Coping With Failures

- **Map worker failure**
  - Map tasks completed or in-progress at worker are reset to idle
  - Reduce workers are notified when task is rescheduled on another worker
- **Reduce worker failure**
  - Only in-progress tasks are reset to idle
- **Master failure**
  - MapReduce task is aborted and client is notified

Things Map-Reduce is Good At

1. Matrix-Matrix and Matrix-vector multiplication.
   - One step of the PageRank iteration was the original application.
2. Relational algebra operations.
3. Many other parallel operations.

Matrix-Vector Multiplication

- Suppose we have an $n \times n$ matrix M, whose element in row i and column j will be denoted m_{ij}.
- Suppose we also have a vector **v** of length n, whose jth element is v_j.
- Then the matrix-vector product is the vector x of length n, whose ith element x_i is given by

$$x_i = \sum_{j=1}^{n} m_{ij} v_j$$

Matrix-Vector Multiplication

- The matrix M and the vector **v** each will be stored in a file of the DFS. We assume that the row-column coordinates of each matrix element will be discoverable, either from its position in the file, or because it is stored with explicit coordinates, as a triple (i, j, m_{ij}).
- We also assume the position of element v_j in the vector **v** will be discoverable in the analogous way.

Matrix-Vector Multiplication

- The Map Function:
  - Each Map task will take the entire vector **v** and a chunk of the matrix M.
  - From each matrix element m_{ij} it produces the key-value pair $(i, m_{ij}v_j)$. Thus, all terms of the sum that make up the component x_i of the matrix-vector product will get the same key.
-
- The Reduce Function:
  - A Reduce task has simply to sum all the values associated with a given key i. The result will be a pair (i, x_i).

Relational Algebra

- Selection
- Projection
- Union, Intersection, Difference
- Natural join
- Grouping and Aggregation

✓ A relation can be stored as a file in a distributed file system. The elements of this file are the tuples of the relation.

Union

- Suppose relations R and S have the same schema.
- Map tasks will be assigned chunks from either R or S .
- The Map tasks don't really do anything except pass their input tuples as key-value pairs to the Reduce tasks.
  - The latter need only eliminate duplicates as for projection.

- The Map Function :
  - Turn each input tuple t into a key-value pair (t, t).
- The Reduce Function :
  - Associated with each key t there will be either one or two values. Produce output (t, t) in either case.

Intersection

- Suppose relations R and S have the same schema.
- Map tasks will be assigned chunks from either R or S .

Intersection

- Suppose relations R and S have the same schema.
- Map tasks will be assigned chunks from either R or S .

- The Map Function :
  - Turn each input tuple t into a key-value pair (t, t).
- The Reduce Function :
  - If key t has value list [t, t], then produce (t, t). Otherwise, produce (t, NULL).

Difference?

- Suppose relations R and S have the same schema.
- Map tasks will be assigned chunks from either R or S .

Difference

- Suppose relations R and S have the same schema.
- Map tasks will be assigned chunks from either R or S .

- The Map Function :
  - For a tuple t in R , produce key-value pair (t, R), and for a tuple t in S , produce key-value pair (t, S). Note that the intent is that the value is the name of R or S , not the entire relation.
- The Reduce Function :
  - For each key **t** , do the following.
    - If the associated value list is [**R**], then produce (**t, t**).
    - If the associated value list is anything else, which could only be [**R, S**], [**S,R**], or [**S**], produce (**t,** NULL).

Natural join

- Joining **R** (**A,B**) with **S** (**B,C**).
- We must find tuples that agree on their B components.

Natural join

- Joining **R** (**A,B**) with **S** (**B,C**).
- We must find tuples that agree on their B components.

- The Map Function:
  - For each tuple (a, b) of R, produce the key-value pair (b, (R, a)).
  - For each tuple (b, c) of S, produce the key-value pair (b, (S, c)).

- The Reduce Function:
  - Each key value b will be associated with a list of pairs that are either of the form (R, a) or (S, c).
  - Construct all pairs consisting of one with first component R and the other with first component S, say (R, a) and (S, c). The output for key b is (b, [(a**1**, b, c**1**), (a**2**, b, c**2**), . . .]),
  - that is, b associated with the list of tuples that can be formed from an R-tuple and an S-tuple with a common b value.

Grouping and Aggregation

- R(A,B,C)
Select SUM(B)
From R
Group by A

- The Map Function:
  - For each tuple (a, b, c) produce the key-value pair (a, b).

- The Reduce Function:
  - Each key a represents a group. Apply SUM to the list [b_1, b_2, . . . , b_n] of B - values associated with key a . The output is the pair (a, x), where x is b_1 + b_2 + . . . + b_n.