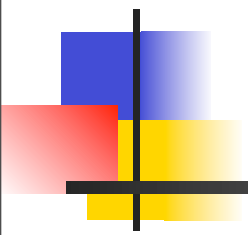
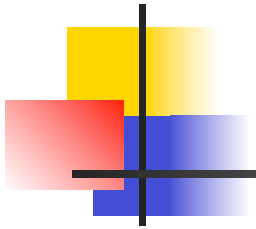


An introduction to MapReduce and MapReduce algorithms for the Web

Mauro Sozio



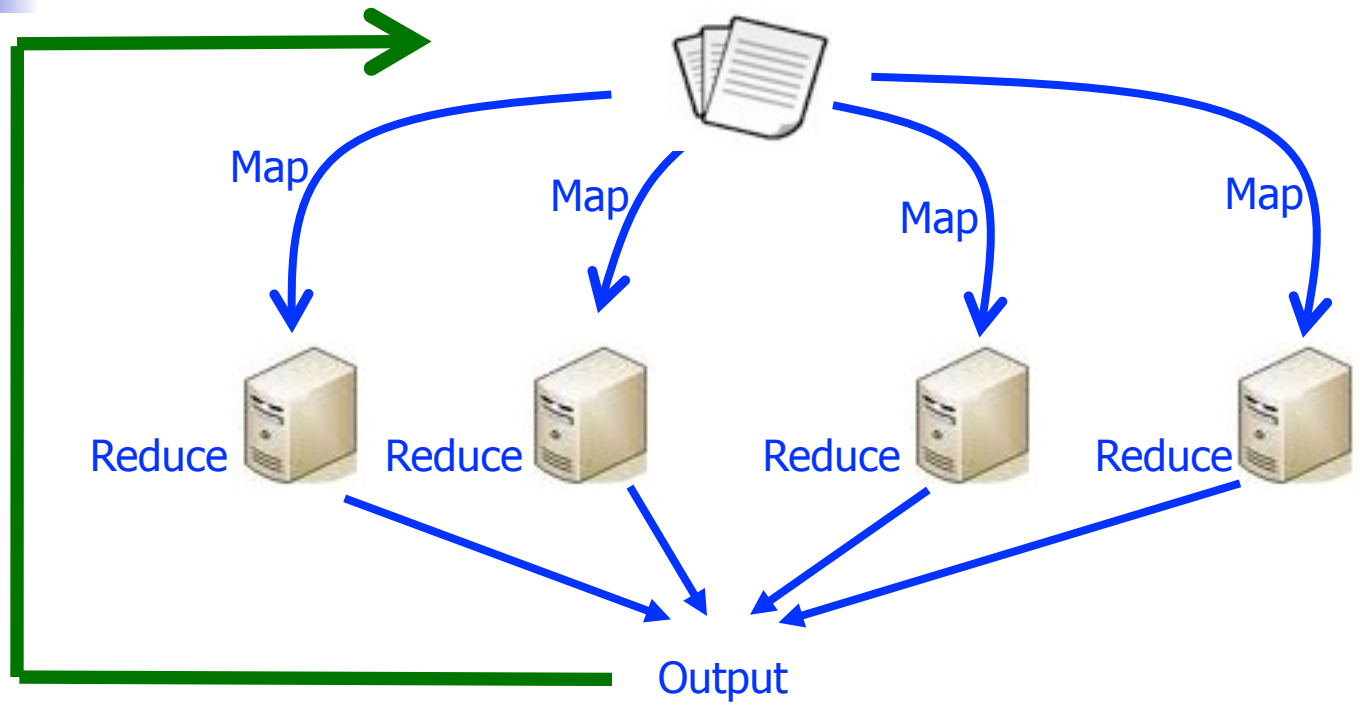
Massive amount of data generated daily

Some facts:

- Facebook >800M users, 900M objects (pages, groups, events).
- Flickr >50 million users, 6 billion images!
- Twitter > 300M users, 1.6 billion search queries per day
- Google > 3 billion search queries per day

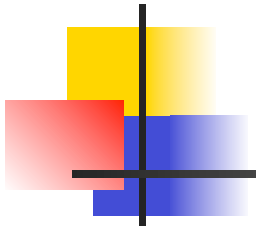
How to make sense of all these data?

MapReduce



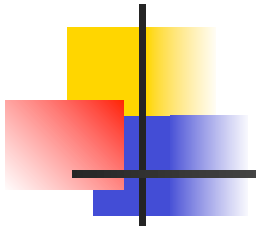
Initially developed by Google, it is nowadays used by several companies (Yahoo!, IBM) and universities (Cornell, CMU...).

Many sequential algorithms have been adapted to MapReduce.



MapReduce Algorithms

- Matrix-vector iterative algorithms efficient in MapReduce:
  - PageRank;
  - Linear and logistic regression, naive Bayes, k-means clust., SVM;
  - Pair-wise document similarity, language modeling.



MapReduce in Brief

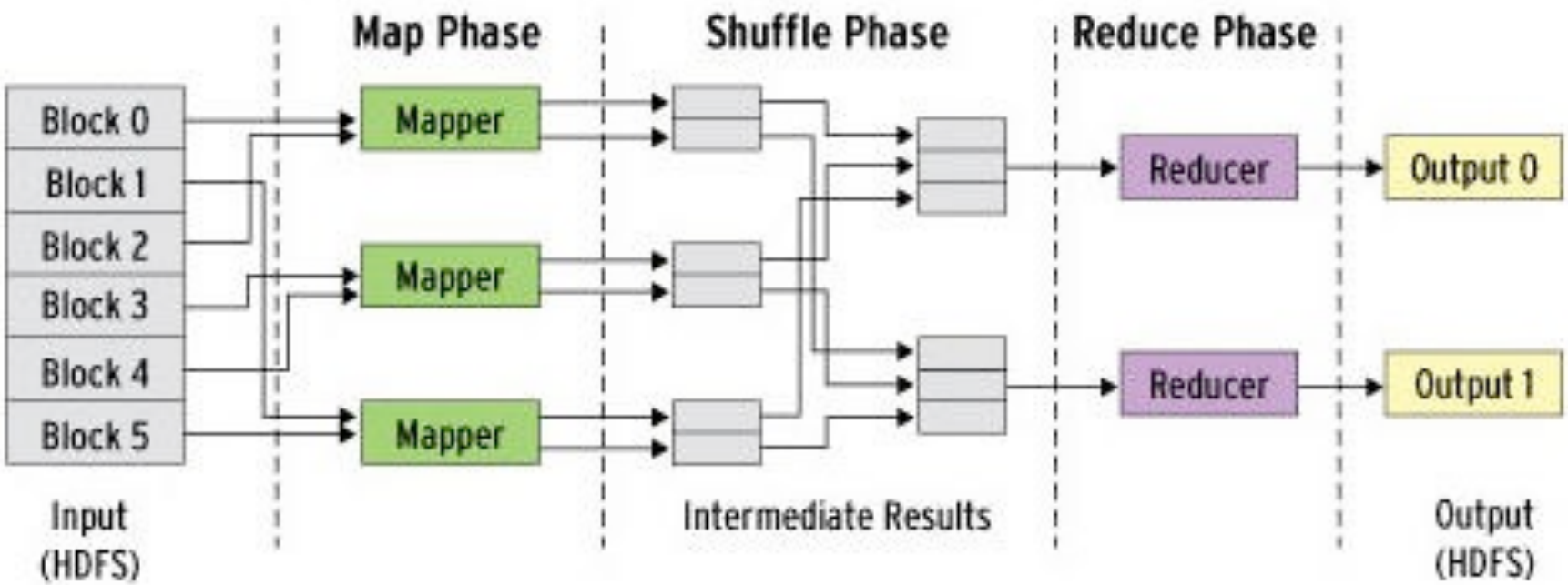
- MapReduce code consists of two functions:
  - Map transforms the input into key-value pairs to process
  - Reduce aggregates the list of values for each key
- The MapReduce environment takes care of distributing the computation
- Non-trivial MapReduce programs consist of a sequence of Map and Reduce tasks.
- Higher-level languages (Pig, Hive, etc.) help with writing distributed applications.

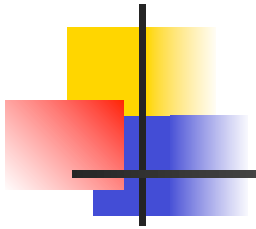
Three operations on key-value pairs

- User-defined: *map* : (key,value) → List(key',value')
  e.g. *map (docId, document)* → *((jaguar,1),(mac,1),(jaguar,1),...)*
- Built-in function: *shuffle*. Group pairs with a same key and assign them to a reducer. Seamless to the user.
- User-defined: *reduce*: (key,List(values)) → List(key',value')
  e.g. reduce: (jaguar,(1,2,1)) → ((jaguar,4))

- Mapping operations are independent from each other!
- The output from a reduce could be the input for another MapReduce iteration.

Map,Reduce,Shuffle





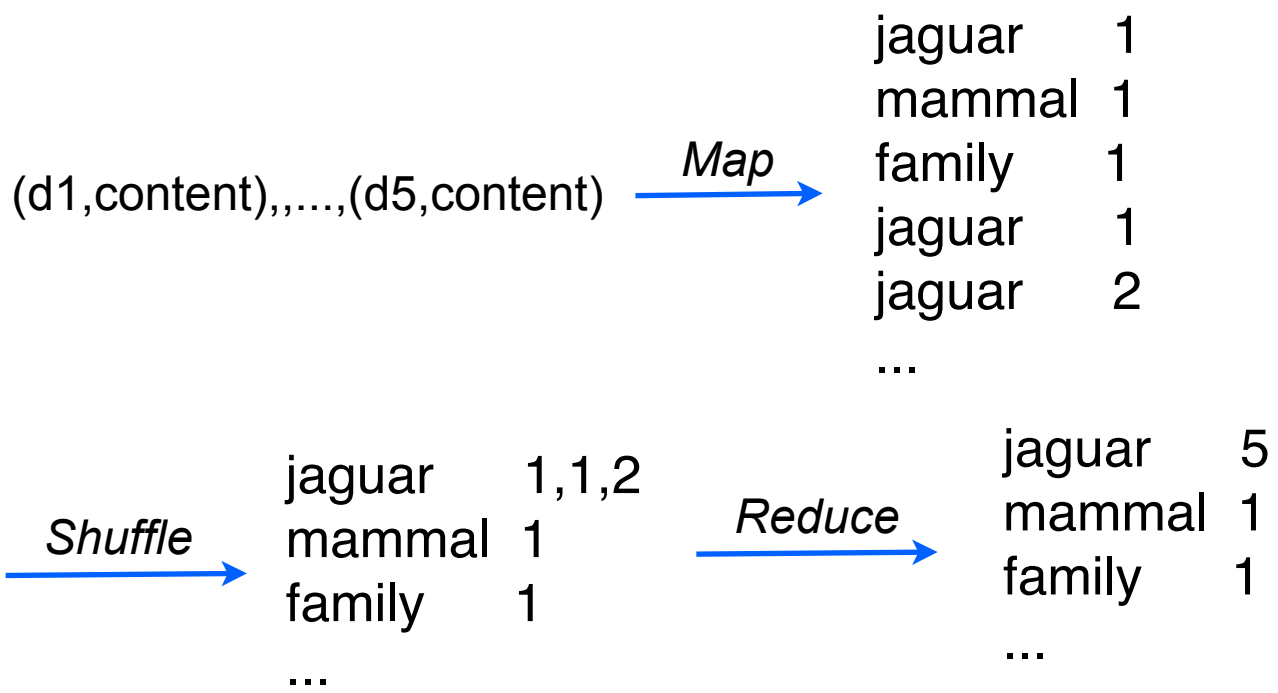
MR: Counting Words

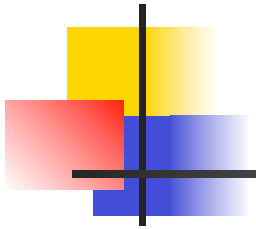
```
void map(String docId, String document):
  // docId: document name
  // document: document content
  for each word w in document:
    Output(w, "1");

void reduce(String word, Iterator partialSums):
  // word: a word
  // partialCounts: a list of partial sums
  int sum = 0;
  for each pc in partialCounts:
    sum += ParseInt(pc);
  Output(word, AsString(sum));
```

Example

DocID	Content
d1	the jaguar is a new world mammal of the felidae family.
d2	for jaguar, atari was keen to use a 68k family device.
d3	mac os x jaguar is available at a price of us $199 for apple's
d4	one such ruling family to incorporate the jaguar into their
d5	It is a big cat.

Example

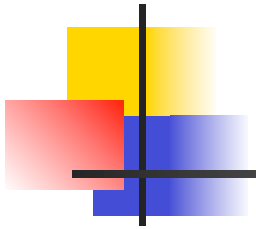

(d1,content),,...,(d5,content)
Map
jaguar 1
mammal 1
family 1
jaguar 1
jaguar 2
...
Shuffle
jaguar 1,1,2
mammal 1
family 1
...
Reduce
jaguar 5
mammal 1
family 1
...



A MapReduce cluster

- Nodes inside a MapReduce cluster are decomposed as follows:
  - A jobtracker acts as a master node; MapReduce jobs are submitted to it.
  - Several tasktrackers run the computation itself, i.e., map and reduce tasks
  - A given tasktracker may run several tasks in parallel
  - Tasktrackers usually also act as data nodes of a distributed filesystem (e.g., GFS, HDFS)

+ a client node where the application is launched.



PageRank

- Find r s.t. r = Mr
- Suppose there are N web pages
- Initialize: $r_0 = [1/N,....,1/N]^T$
- Iterate:
  - $r_{k+1} = Mr_k$
  - Stop when $|r_{k+1} - r_k|_1 < \varepsilon$
- $|x|_1 = \sum_{1 \le i \le N} |x_i|$ is the L_1 norm