Data Mining & MapReduce

Irene Finocchi

Data mining

What is “data mining”

- Discovery of useful, possibly unexpected, patterns in data (models)
- Subsidiary issues:
  - Data cleaning: detection of bogus data (e.g., age=150)
  - Visualization: better than MB files of output
    - A picture is worth 10 thousand words

Data mining approaches (1)

- Machine-learning: small data used as a training set to predict different phenomena at large
  - E.g., success of a movie (Netflix challenge)
  - Good when we have little idea of what we are looking for in the data

Data mining approaches (2)

- Statistics: inference of statistical models
  - Result = parameters of the model
- Databases/algorithms: concentrate on large-scale data, typically stored in external memory
  - Analytic processing: query the data, result = query answer
  - E.g., number of papers in a catalog written between 2010 and 2014

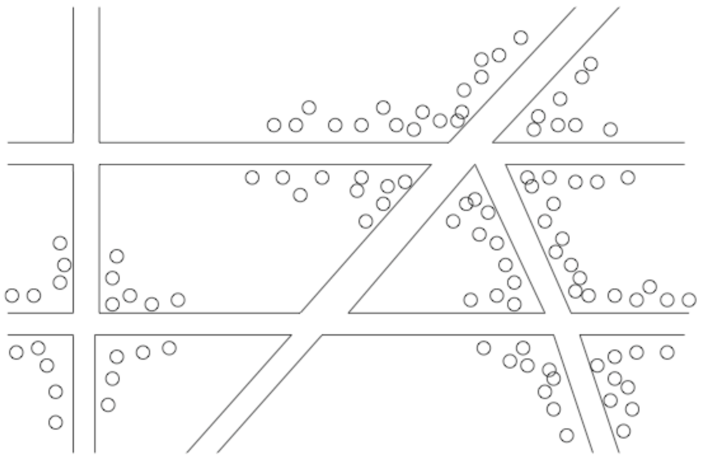
(Way too simple) example

- DB/algorithm person: given a billion numbers, compute their average and standard deviation
- Statistician: fit the points to the best Gaussian distribution and report the mean and standard deviation of that distribution

Computational mining (1)

- Summarization: summarizing the data succinctly and approximately
  - Pagerank: a number reflecting the importance of a page
  - Clustering: data viewed as points in a multidimensional space, points close in this space assigned to the same cluster

Clustering cholera cases on a map of London



Computational mining (2)

- Feature extraction
  - Frequent itemsets: “market-basket” problem
    - Given “baskets” of small sets of items, find small sets of items that appear together in many baskets
    - E.g., hamburger and ketchup
  - Similar items
    - At the base of recommendation systems
    - What do you buy on Amazon? Find “similar” customers and recommend something many of these customers have bought
    - Clustering customers does not work here: each of us has interests in many different things (e.g., popular science books and historical biographies)

Beware of false positives

Are answers meaningful?

- Big data-mining risk: "discover" meaningless patterns

- Statisticians call it Bonferroni's principle:
  - (Roughly) if you look for interesting patterns in more places than your amount of data supports, you are bound to find crap
  - Carlo Emilio Bonferroni: Italian mathematician, 1892-1960

Examples of Bonferroni's principle

- November 2002, TIA - Total Information Awareness

  As reported by the New York Times, the Defense Advanced Research Projects Agency (DARPA) was developing a tracking system called Total Information Awareness, which was intended to detect terrorists through analyzing troves of information

- The Rhine Paradox: a great example of how not to conduct scientific research

The TIA story

- Suppose we believe that certain groups of evil-doers are meeting occasionally in hotels to plot doing evil
- We want to find (unrelated) people who at least twice have stayed at the same hotel on the same day

Some details

- 10^9 people being tracked
- 1000 days
- Each person stays in a hotel 1% of the time (10 days out of 1000 in a hotel)
- Hotels hold 100 people (so 10^5 hotels)

- If everyone behaves randomly (i.e., no evil-doers) will the data mining detect anything suspicious?

Calculations (1)

Q at some hotel

P at some hotel

Same hotel

- Probability that given persons P and Q will be at the same hotel on given day d :
  - $\boxed{1/100} \times \boxed{1/100} \times \boxed{10^{-5}} = 10^{-9}$
- Probability that P and Q will be at the same hotel on given days d_1 and d_2:
  - $10^{-9} \times 10^{-9} = 10^{-18}$
- Pairs of days: 5×10^5

Calculations (2)

- Probability that P and Q will be at the same hotel on some two days:
  - $5 \times 10^{5} \times 10^{-18} = 5\times10^{-13}$
- Pairs of people:
  - 5×10^{17}
- Expected number of "suspicious" pairs of people:
  - $5 \times 10^{17} \times 5 \times 10^{-13} = 250{,}000.$

Conclusion

- Suppose there are (say) 10 pairs of evil-doers who definitely stayed at the same hotel twice
- Analysts have to sift through 250,000 candidates to find the 10 real cases
  - Not gonna happen

When looking for a property (e.g., "two people stayed at the same hotel twice"), make sure that the property does not allow so many possibilities that random data will surely produce facts "of interest"

Another story: Rhine paradox

- Joseph Rhine was a parapsychologist in the 1950's
- He hypothesized that some people had Extra-Sensory Perception
- He devised (something like) an experiment where subjects were asked to guess 10 hidden cards – **red** or **blue**.
- He discovered that almost 1 in 1000 had ESP – they were able to get all 10 right!

The second Rhine test

- He told these people they had ESP and called them in for another test of the same type.
- Alas, he discovered that almost all of them had lost their ESP
- What did he conclude?
  - Answer on next slide.

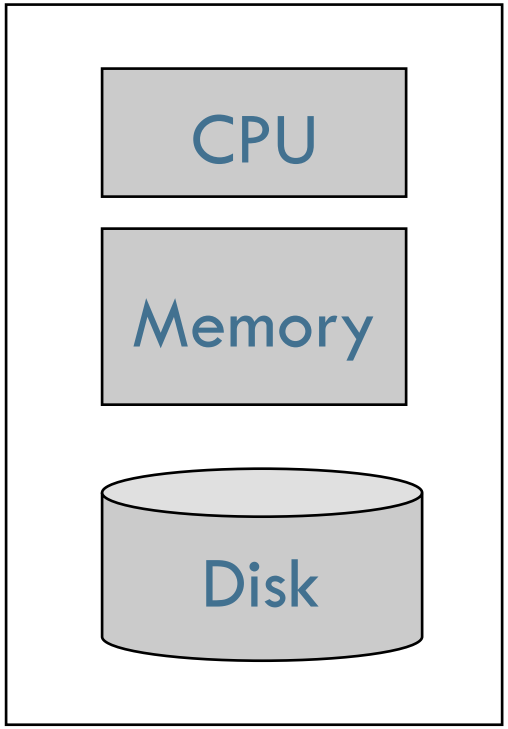
Rhine conclusion

- He concluded that you shouldn't tell people they have ESP: it causes them to lose it!

Understanding Bonferroni's Principle will help you look a little less stupid than a parapsychologist

Mining big data with MapReduce

Single-node architecture



Machine learning, statistics

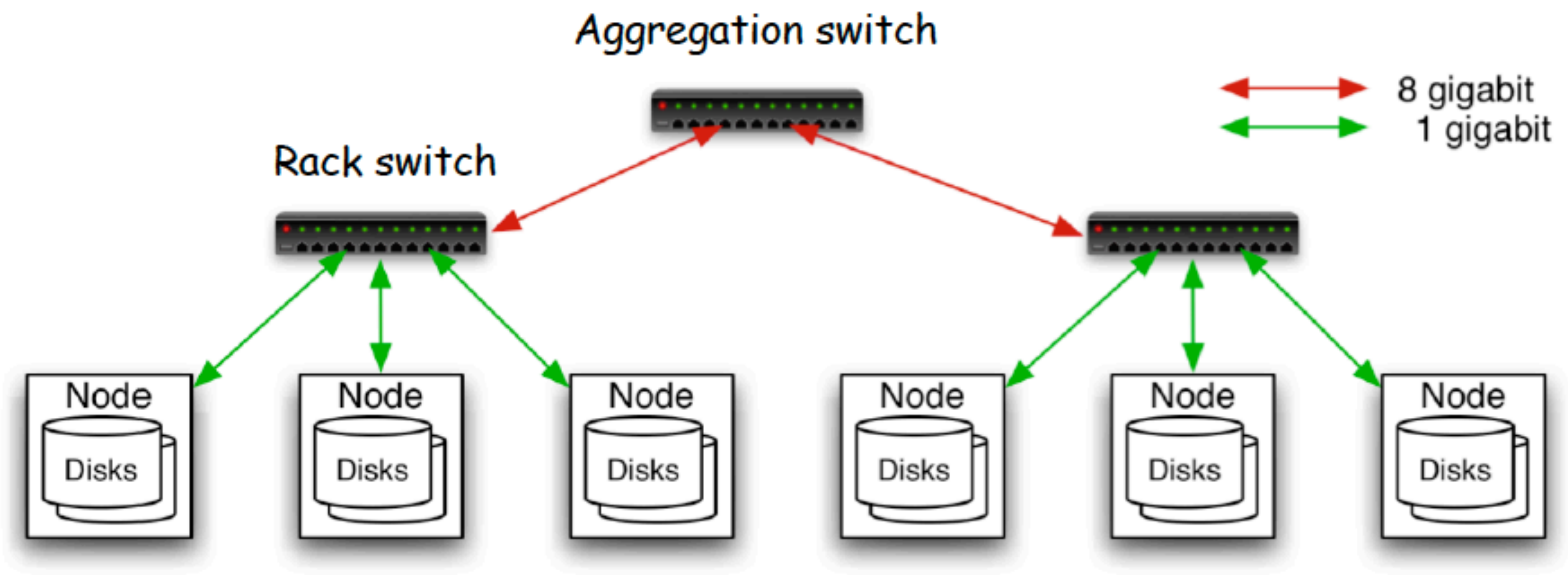
"Classical" data mining

Commodity clusters

- Cannot mine tens to hundreds of Terabytes of data on a single server
- Standard architecture emerging:
  - Cluster of commodity Linux nodes
  - Gigabit ethernet interconnections
- How to organize computations on these architectures?
- How to program these architectures?
- How to mask issues such as hardware failures in these architectures?

Real cluster architecture

Cluster architecture



- Each rack contains 10/64 nodes
- Sample node configuration: 8 x 2GHz cores, 8 GB RAM, 4 disks (4 TB)

Stable storage

- First order problem: if nodes can fail, how can we store data persistently?
- Answer: Distributed File System
  - Provides global file namespace
  - Google GFS; Hadoop HDFS; Kosmix KFS
- Typical usage pattern
  - Huge files (100s of GB to TB)
  - Data is rarely updated in place
  - Reads and appends are common

Distributed file system

- Chunk servers
  - File is split into contiguous chunks
  - Typically each chunk is 16-64MB
  - Each chunk replicated (usually 2x or 3x)
  - Try to keep replicas in different racks
- Master node
  - Stores metadata
  - Might be replicated
  - (a.k.a. Name Node in HDFS)
- Client library for file access
  - Talk to master to find chunk servers
  - Connect directly to chunk servers to access data

Warm up: word count

- We have a large file of words, one word per line
- Count the number of times each distinct word appears in the file

- Sample application: analyze web server logs to find popular URLs

Different scenarios

- Case 1: Entire file fits in memory
  - Load file into main memory
  - Keep also a hash table with <word, count> pairs
- Case 2: File too large for mem, but all <word, count> pairs fit in mem
  - Scan file on disk
  - Keep <word, count> hash table in main memory
- Case 3: Too many distinct words to fit in memory
  - External sort, then scan file (all occurrences of the same word are consecutive: one running counter suffices)
  - **`sort datafile | uniq -c`**

Making things a little bit harder

- Now suppose we have a large corpus of documents
- Count the number of times each distinct word occurs in the corpus
  - `words(docs/*) | sort | uniq -c`
  - where `words` takes a file and outputs the words in it, one to a line

- The above captures the essence of MapReduce
  - Great thing is it is naturally parallelizable

MapReduce

- A novel programming model
- Everything built on top of <key,value> pairs
  - Keys and values are user-defined: can be anything
- Only two user-defined functions:
  - Map
    - map(k_1,v_1) → list(k_2,v_2)
    - given input data $<k_1,v_1>$, produce intermediate data v_2 labeled with key k_2
  - Reduce
    - reduce(k_2, list(v_2)) → list(v_3)   preserves key
    - given a list of values list(v_2) associated with a key k_2, return a list of values list(v_3) associated with the same key

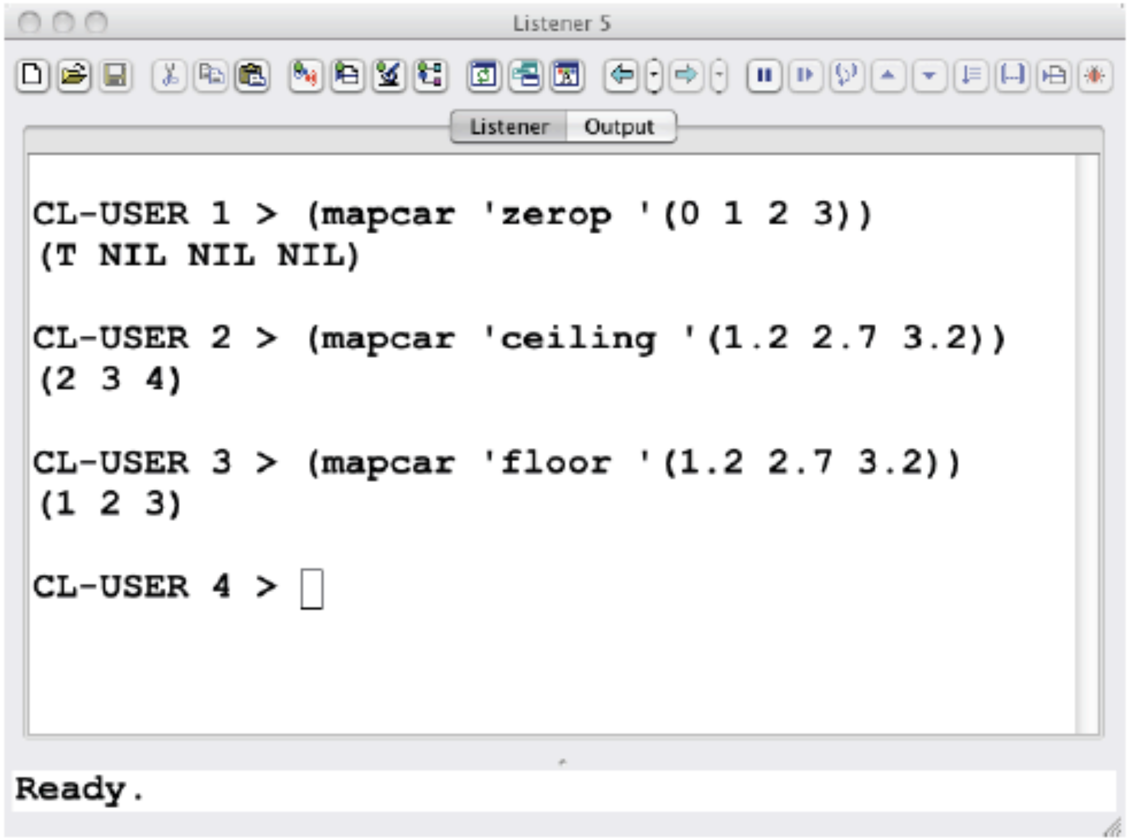
The origins (2004)

"Our abstraction is inspired by the map and reduce primitives present in Lisp and many other functional languages. We realized that most of our computations involved applying a map operation to each logical "record" in our input in order to compute a set of intermediate key/value pairs, and then applying a reduce operation to all the values that shared the same key, in order to combine the derived data appropriately."

Jeffrey Dean & Sanjay Ghemawat [OSDI 2004]

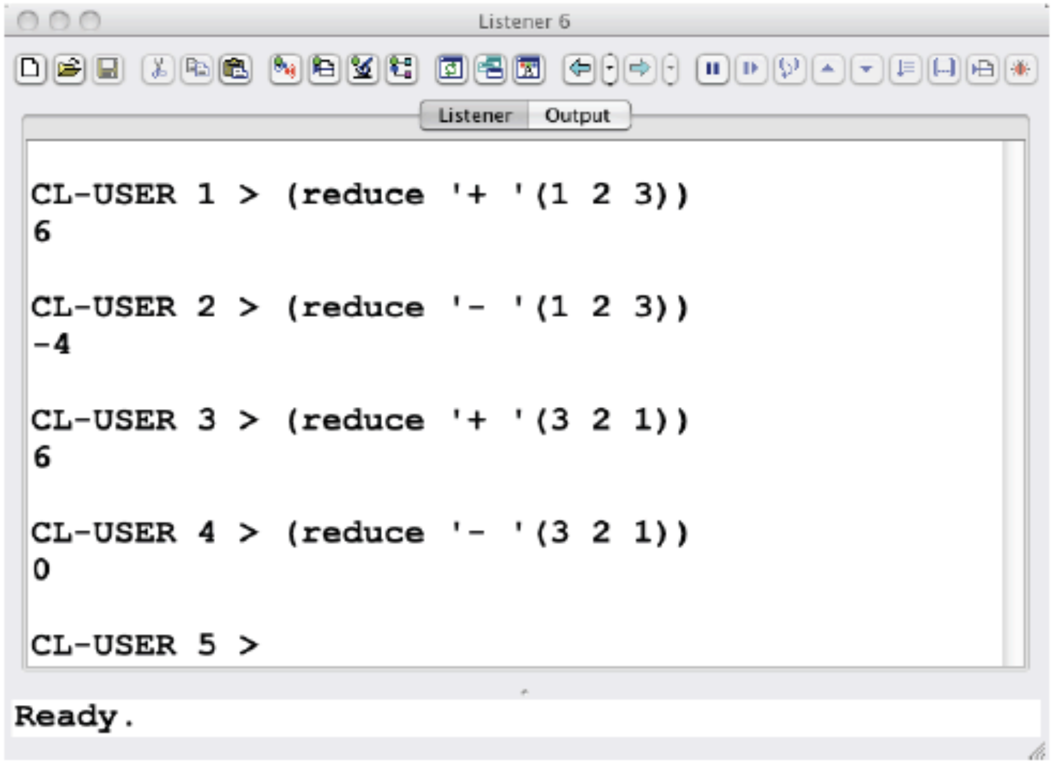
Map in Lisp

The *map(car)* is a function that calls its first argument with each element of its second argument, in turn.



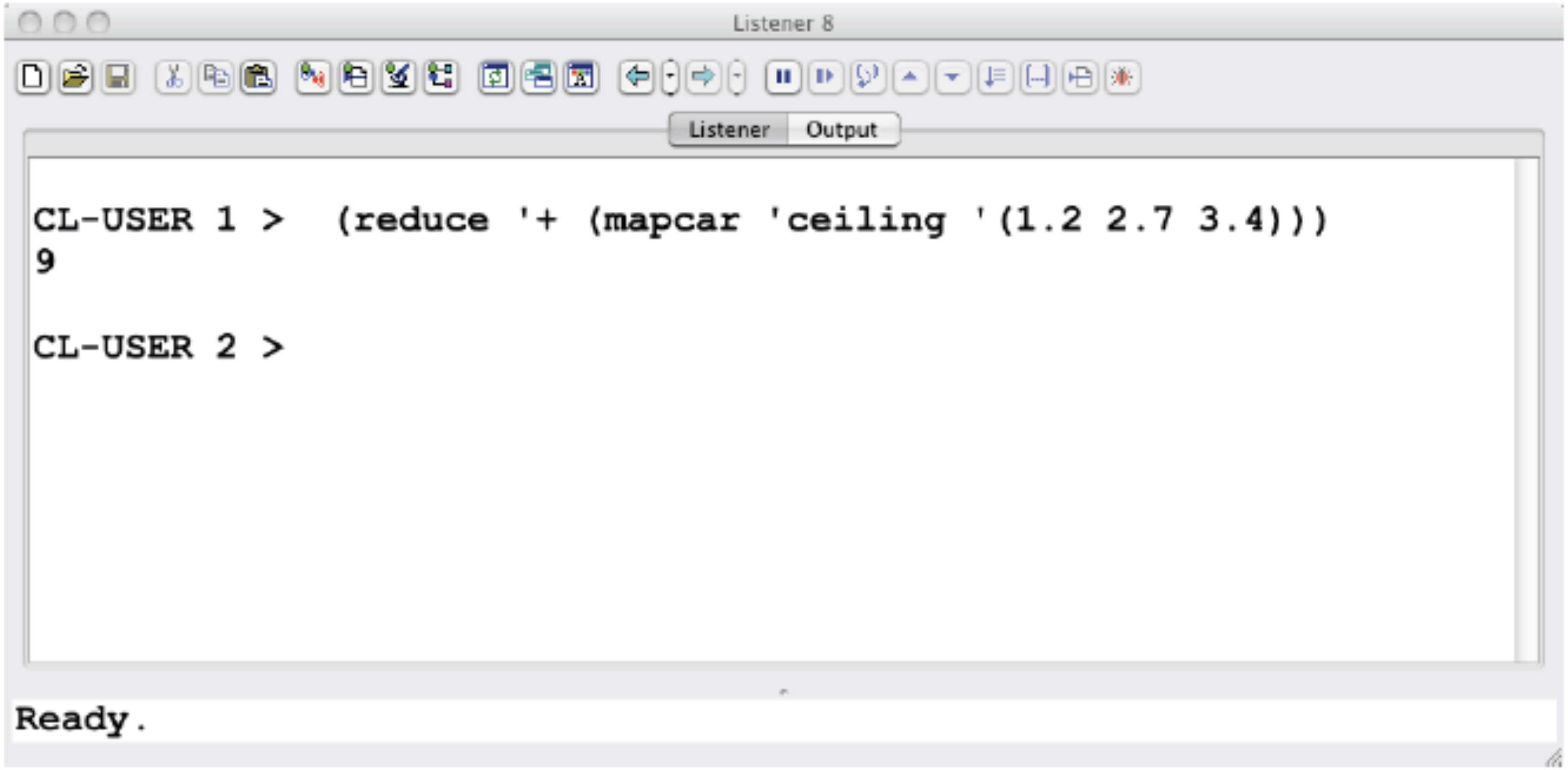
Reduce in Lisp

The *reduce* is a function that returns a single value constructed by calling the first argument (a function) function on the first two items of the second argument (a sequence), then on the result and the next item, and so on .



MapReduce in Lisp

Our first MapReduce program :-)



Parallelism in MapReduce

- All mappers in parallel
- All reducers in parallel
- Different pairs transparently distributed across available machines

$\text{map}(k_1,v_1) \rightarrow \text{list}(k_2,v_2)$

Shuffle: group values with the same key to be passed to a single reducer

$\text{reduce}(k_2, \text{list}(v_2)) \rightarrow \text{list}(v_3)$

THE MapReduce example: WordCount

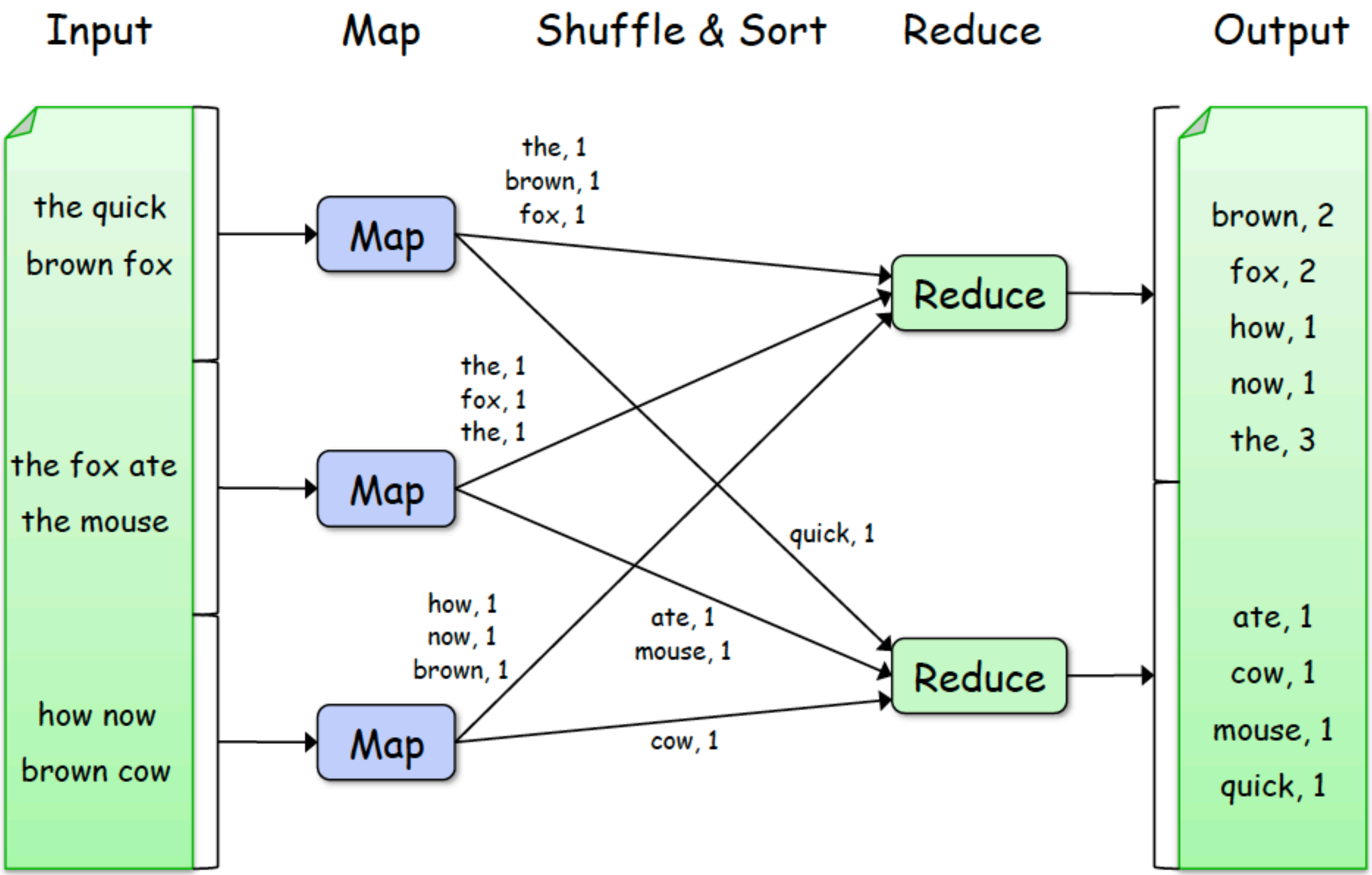
```
map(key, value):
// key: document name; value: text of document
    for each word w in value:
        emit(w, 1)

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

A programmer's perspective

> *The beauty of MapReduce is that any programmer can understand it, and its power comes from being able to harness thousands of computers behind that simple interface.*
>
> David Patterson

WordCount data flow



Readings

- J. Leskovec, A. Rajaraman & J. Ullman

  Mining of massive data sets

  Chapters 1 and 2 (Sections 2.1 & 2.2)

  http://i.stanford.edu/~ullman/mmds.html

- Jeffrey Dean and Sanjay Ghemawat,

  MapReduce: Simplified Data Processing on Large Clusters

  http://labs.google.com/papers/mapreduce.html

Acknowledgments

Part of these slides are based on material from the "Data Mining" Stanford course CS345A by Rajaraman &Ullman and from the book "Mining of Massive Data Sets" by Leskovec, Rajaraman & Ullman