PURDUE UNIVERSITY | Department of Computer Science

CS47300: Web Information Search and Management

Scaling

Prof. Chris Clifton

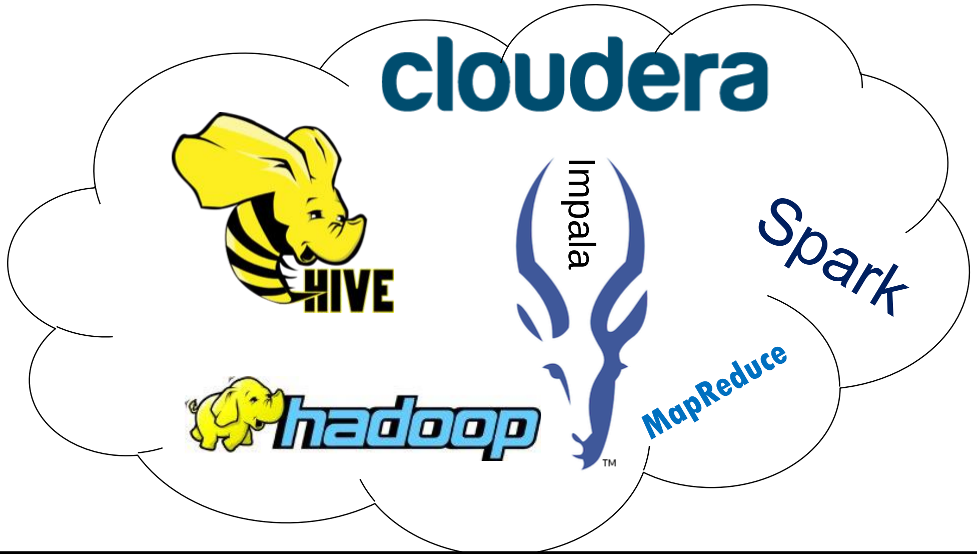
18 November 2020

(Some material from Yahoo!, Croft et al.)

Indiana Center for Database Systems ™

PURDUE UNIVERSITY Department of Computer Science

The Cloud: What's it all About?

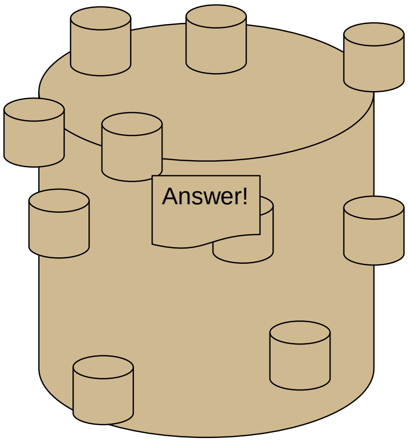


Cloud Databases: Why?

- Scaling
  - 1000’s of nodes working simultaneously to analyze data
- Answer challenging queries on big data
  - If you can express the query in a limited query language
- Several examples
  - Hadoop, Spark, …

Basic Idea: Divide and Conquer

- Divide data into units
- Compute on those units
- Combine results
- *Need algorithms where this works!*

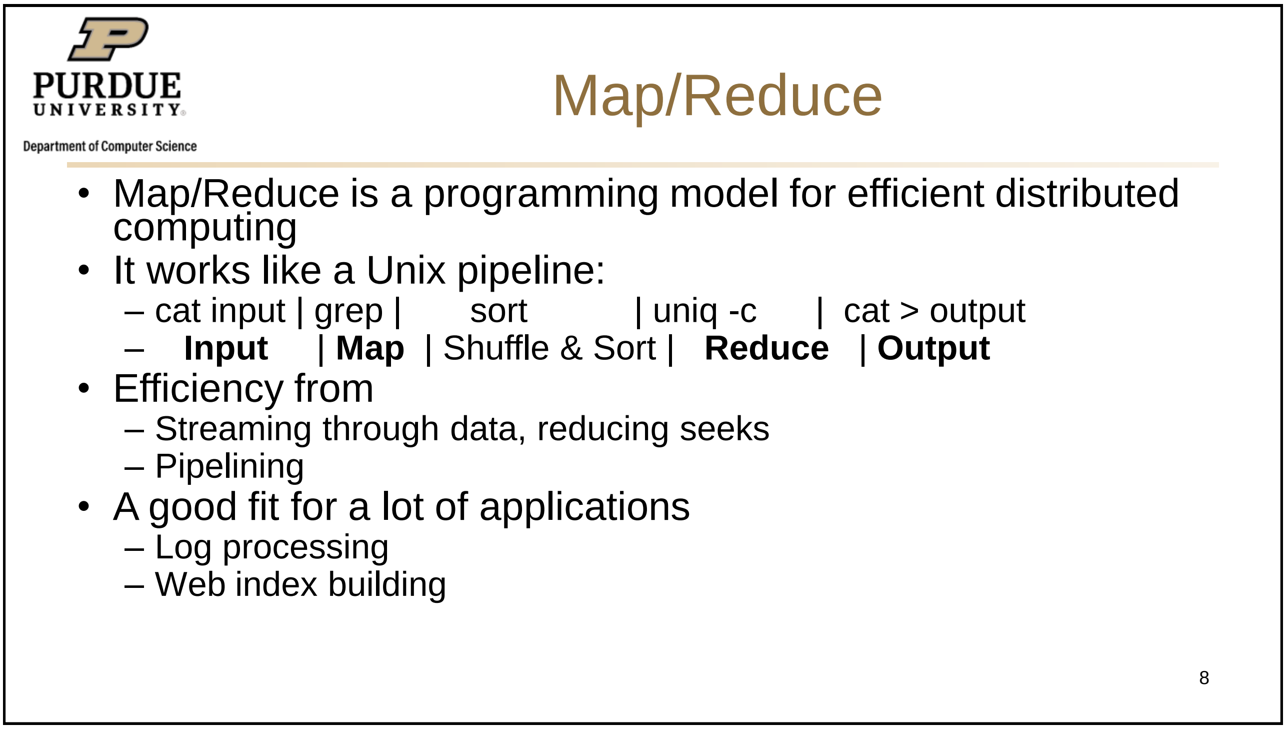


Distributed Indexing

- Distributed processing driven by need to index and analyze huge amounts of data (i.e., the Web)
- Large numbers of inexpensive servers used rather than larger, more expensive machines
- *MapReduce* is a distributed programming tool designed for indexing and analysis tasks

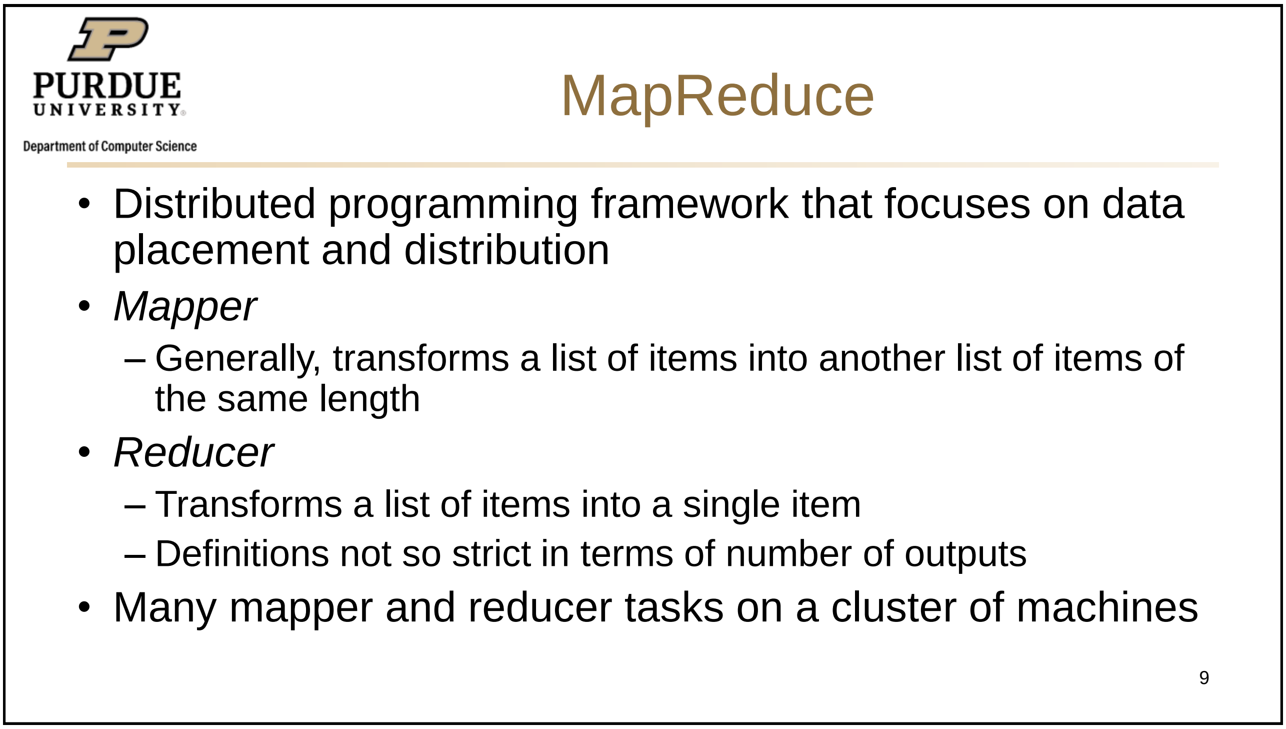
Example

- Given a large text file that contains data about credit card transactions
  - Each line of the file contains a credit card number and an amount of money
  - Determine the number of unique credit card numbers
- Could use hash table – memory problems
  - counting is simple with sorted file
- Similar with distributed approach
  - sorting and placement are crucial



Map/Reduce

- Map/Reduce is a programming model for efficient distributed computing
- It works like a Unix pipeline:
  - cat input | grep | sort | uniq -c | cat > output
  - **Input** | **Map** | Shuffle & Sort | **Reduce** | **Output**
- Efficiency from
  - Streaming through data, reducing seeks
  - Pipelining
- A good fit for a lot of applications
  - Log processing
  - Web index building



MapReduce

- Distributed programming framework that focuses on data placement and distribution
- *Mapper*
  - Generally, transforms a list of items into another list of items of the same length
- *Reducer*
  - Transforms a list of items into a single item
  - Definitions not so strict in terms of number of outputs
- Many mapper and reducer tasks on a cluster of machines

PURDUE UNIVERSITY

Department of Computer Science

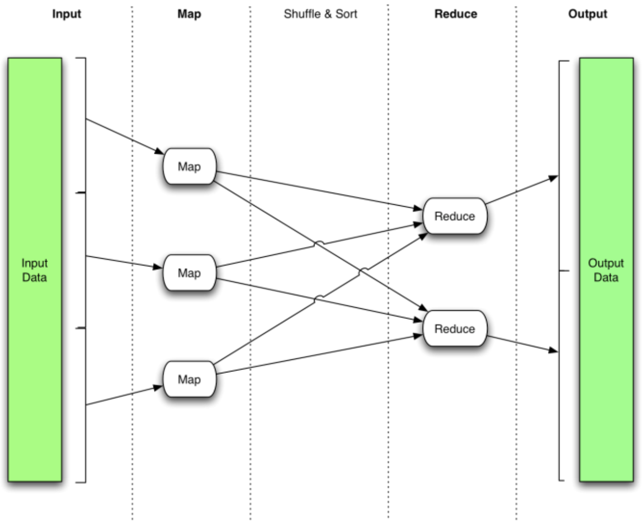
MapReduce

- Basic process
  - *Map* stage which transforms data records into pairs, each with a key and a value
  - *Shuffle* uses a hash function so that all pairs with the same key end up next to each other and on the same machine
  - *Reduce* stage processes records in batches, where all pairs with the same key are processed at the same time
- *Idempotence* of Mapper and Reducer provides fault tolerance
  - multiple operations on same input gives same output

PURDUE UNIVERSITY

Department of Computer Science

Map/Reduce Dataflow



Map/Reduce features

- Java and C++ APIs
  - In Java use Objects, while in C++ bytes
- Each task can process data sets larger than RAM
- Automatic re-execution on failure
  - In a large cluster, some nodes are always slow or flaky
  - Framework re-executes failed tasks
- Locality optimizations
  - Map-Reduce queries HDFS for locations of input data
  - Map tasks are scheduled close to the inputs when possible

Example: MapReduce to count word frequency

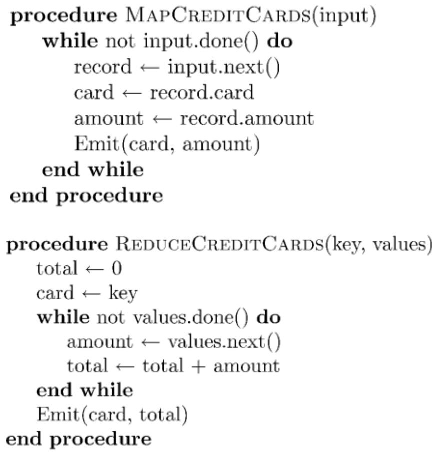
- SQL:
  select word, count(*) from documents group by word
- MapReduce:

```
– function map (String name, String document):
     for each word w in document:  emit (w, 1)
– function reduce (String word, Iterator partCounts):
     sum = 0
     for each pc in PartCounts:
       sum += pc
     emit (word, sum)
```

PURDUE UNIVERSITY

Department of Computer Science

Example



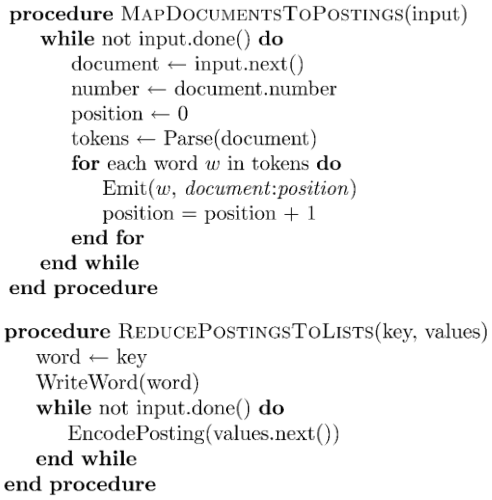
```
procedure MAPCREDITCARDS(input)
    while not input.done() do
        record ← input.next()
        card ← record.card
        amount ← record.amount
        Emit(card, amount)
    end while
end procedure

procedure REDUCECREDITCARDS(key, values)
    total ← 0
    card ← key
    while not values.done() do
        amount ← values.next()
        total ← total + amount
    end while
    Emit(card, total)
end procedure
```

PURDUE UNIVERSITY

Department of Computer Science

Indexing Example



```
procedure MAPDOCUMENTSTOPOSTINGS(input)
    while not input.done() do
        document ← input.next()
        number ← document.number
        position ← 0
        tokens ← Parse(document)
        for each word $w$ in tokens do
            Emit($w$, $document{:}position$)
            position = position + 1
        end for
    end while
end procedure

procedure REDUCEPOSTINGSTOLISTS(key, values)
    word ← key
    WriteWord(word)
    while not input.done() do
        EncodePosting(values.next())
    end while
end procedure
```