PURDUE UNIVERSITY | Department of Computer Science

CS 44800: Introduction To Relational Database Systems

Cloud: NoSQL Databases

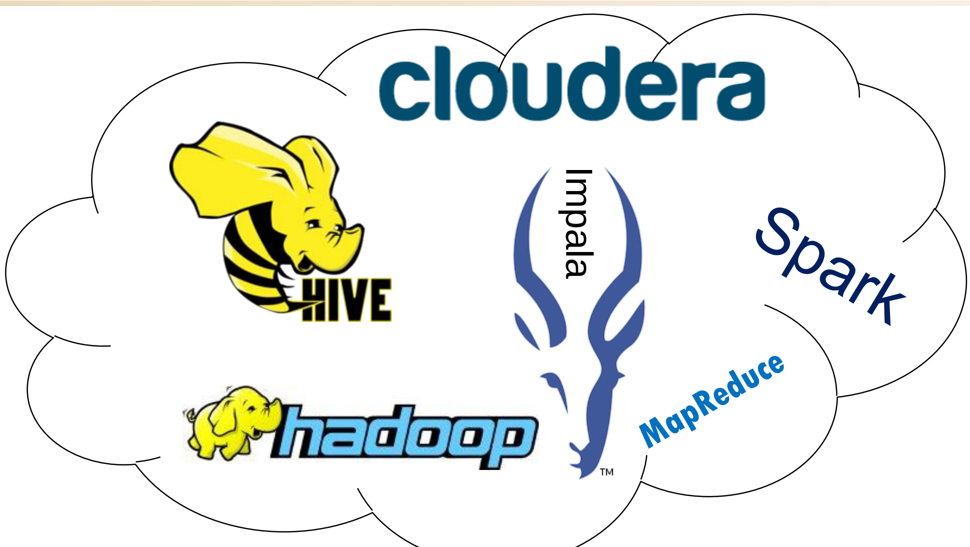
Prof. Chris Clifton

7 December 2021

Indiana Center for Database Systems ™

PURDUE UNIVERSITY Department of Computer Science

The Cloud: What's it all About?



Beyond RDBMS

The Relational Model is too limiting!

- Simple data model – doesn't capture semantics
  - Object-Oriented DBMS ('80s)
- Fixed schema – not flexible enough
  - XML databases ('90s)
- Too heavyweight/slow
  - NoSQL databases ('00s)

The Latest: Cloud Databases

- PERFORMANCE!
  - More speed, bigger data
- But this doesn't come for free
  - *Eventual* consistency (eventually all the updates will occur)
  - No isolation guarantees
  - Limited reliability guarantees

Cloud Databases: Why?

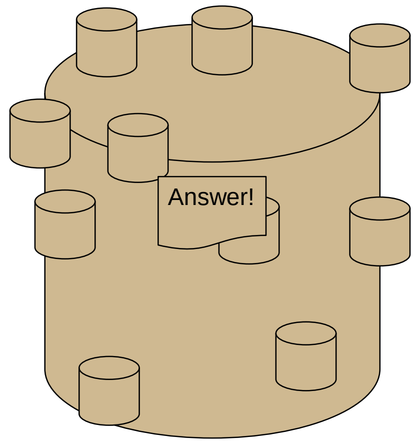
- Scaling
  - 1000's of nodes working simultaneously to analyze data
- Answer challenging queries on big data
  - If you can express the query in a limited query language
- Several examples
  - Hadoop, Spark, …

Are we Post-Relational?

- Object-oriented database → object-relational database
  - Today: Commercial RDBMS includes type extensibility and OO features
- XML database
  - XML storage tools for RDBMS
- Cloud Database
  - See Hive – will we see Map-Reduce engines as part of traditional RDBMS?

Cloud Data Processing 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

Map/Reduce

- Map/Reduce is a programming model for efficient distributed computing
- 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

CCA – Oct 2008

YAHOO!

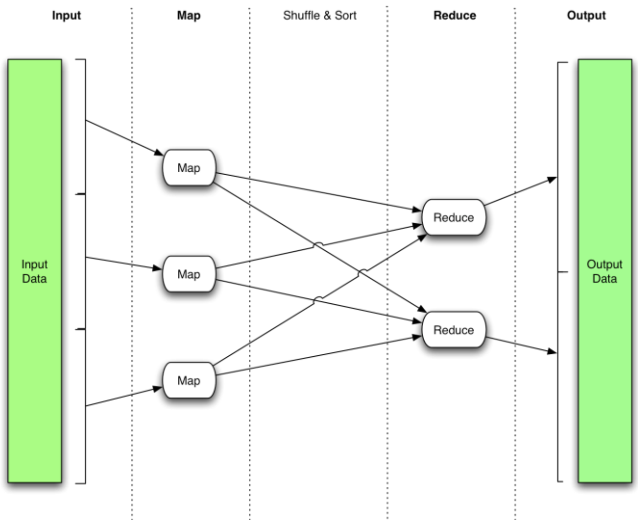
PURDUE UNIVERSITY

Department of Computer Science

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

Map/Reduce Dataflow



CCA – Oct 2008

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

Select word, count(*) from doc group by word;

```
public class WordCount {
public static class Map extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable> {
private final static IntWritable one = new IntWritable(1);
private Text word = new Text();
public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {
    String line = value.toString();
    StringTokenizer tokenizer = new StringTokenizer(line);
    while (tokenizer.hasMoreTokens()) {
        word.set(tokenizer.nextToken());
        output.collect(word, one);
        }
    }
}
```

```
public static class Reduce extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable> {
public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {
    int sum = 0;
    while (values.hasNext()) {
        sum += values.next().get();
    }
    output.collect(key, new IntWritable(sum));
}
}
```

Select word, count(*) from doc group by word;

```
public static void main(String[] args) throws Exception {
    JobConf conf = new JobConf(WordCount.class);
    conf.setJobName("wordcount");
    conf.setOutputKeyClass(Text.class);
    conf.setOutputValueClass(IntWritable.class);
    conf.setMapperClass(Map.class);
    conf.setCombinerClass(Reduce.class);
    conf.setReducerClass(Reduce.class);
    conf.setInputFormat(TextInputFormat.class);
    conf.setOutputFormat(TextOutputFormat.class);
    FileInputFormat.setInputPaths(conf, new Path(args[0]));
    FileOutputFormat.setOutputPath(conf, new Path(args[1]));
    JobClient.runJob(conf);
}
}
```

Introduction to Hadoop

Owen O'Malley
Yahoo!, Grid Team
owen@yahoo-inc.com

Problem

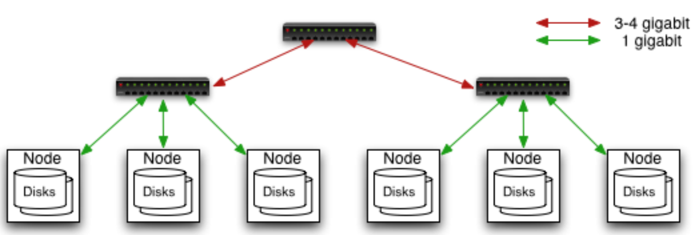
- How do you scale up applications?
  - Run jobs processing 100's of terabytes of data
  - Takes 11 days to read on 1 computer
- Need lots of cheap computers
  - Fixes speed problem (15 minutes on 1000 computers), but…
  - Reliability problems
    - In large clusters, computers fail every day
    - Cluster size is not fixed
- Need common infrastructure
  - Must be efficient and reliable

Solution

- Open Source Apache Project
- Hadoop Core includes:
  - Distributed File System - distributes data
  - Map/Reduce - distributes application
- Written in Java
- Runs on
  - Linux, Mac OS/X, Windows, and Solaris
  - Commodity hardware

CCA – Oct 2008 YAHOO!

Commodity Hardware Cluster



- Typically in 2 level architecture
  - Nodes are commodity PCs
  - 40 nodes/rack
  - Uplink from rack is 8 gigabit
  - Rack-internal is 1 gigabit

CCA – Oct 2008 YAHOO!

Distributed File System

- Single namespace for entire cluster
  - Managed by a single *namenode*.
  - Files are single-writer and append-only.
  - Optimized for streaming reads of large files.
- Files are broken in to large blocks.
  - Typically 128 MB
  - Replicated to several *datanodes*, for reliability
- Client talks to both namenode and datanodes
  - Data is not sent through the namenode.
  - Throughput of file system scales nearly linearly with the number of nodes.
- Access from Java, C, or command line.

Data Correctness

- Data is checked with CRC32
- File Creation
  - Client computes checksum per 512 byte
  - DataNode stores the checksum
- File access
  - Client retrieves the data and checksum from DataNode
  - If Validation fails, Client tries other replicas
- Periodic Validation

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

How is Yahoo using Hadoop?

- We started with building better applications
  - Scale up web scale batch applications (search, ads, …)
  - Factor out common code from existing systems, so new applications will be easier to write
  - Manage the many clusters we have more easily
- The mission now includes research support
  - Build a **huge** data warehouse with many Yahoo! data sets
  - Couple it with a huge compute cluster and programming models to make using the data easy
  - Provide this as a service to our researchers
  - We are seeing great results!
    - Experiments can be run much more quickly in this environment

Running Production WebMap

- Search needs a graph of the “known” web
  - Invert edges, compute link text, whole graph heuristics
- Periodic batch job using Map/Reduce
  - Uses a chain of ~100 map/reduce jobs
- Scale
  - 1 trillion edges in graph
  - Largest shuffle is 450 TB
  - Final output is 300 TB compressed
  - Runs on 10,000 cores
  - Raw disk used 5 PB
- Written mostly using Hadoop’s C++ interface

CCA – Oct 2008

Hadoop Community

- Apache is focused on project communities
  - Users
  - Contributors
    - write patches
  - Committers
    - can commit patches **too**
  - Project Management Committee
    - vote on new committers and releases **too**
- Apache is a meritocracy
- Use, contribution, and diversity is growing
  - But we need and want more!

CCA – Oct 2008

A ⋈ B

- A and B are Hadoop files
  - Produce new Hadoop file that is join of A and B
- Reduce-side join
  - Send different keys to different reducer
- Map-side join
  - Broadcast join

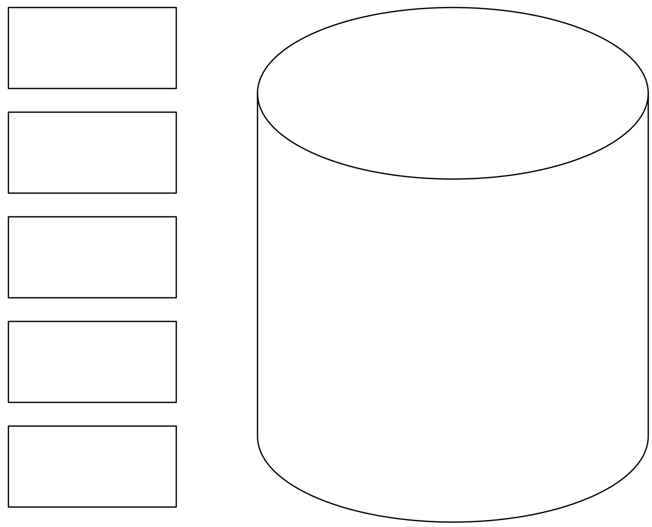
How to join efficiently?

- Sort-Merge join
  - Sort tables
  - Read both, outputting tuples that join
- Hash join
  - Hash function divides into groups
    - All keys that can match go into same group
  - Groups small enough to fit in memory
- *We'll use both ideas*

Hash Join Revisited

PURDUE UNIVERSITY
Department of Computer Science

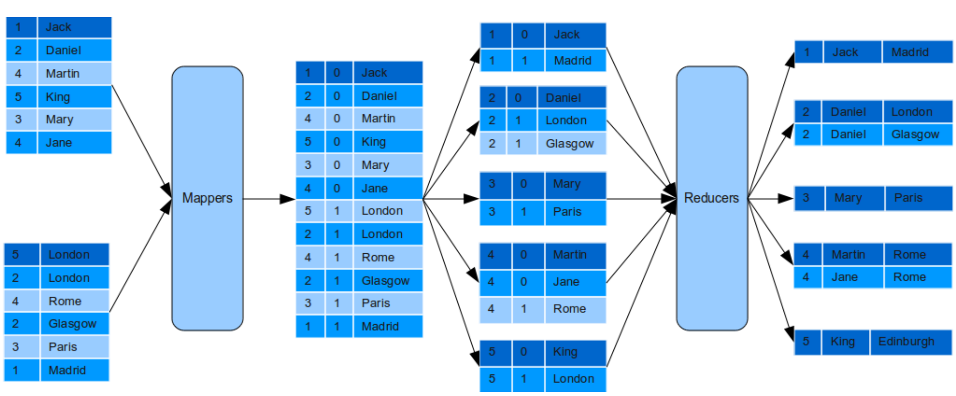
Aardvark
Caiman
Eagle
Deer
Alpaca
Alligator
Butterfly
Ferret
Bison
Bobcat
Bear
Bird
Bat



Aardvark
Caiman
Eagle
Deer
Albatross
Avocet
Butterfly
Ferret
Badger
Bobcat
Bear
Bird
Bat

Reduce-Side Join *(Chandar'10)*

PURDUE UNIVERSITY
Department of Computer Science



PURDUE UNIVERSITY
Department of Computer Science

Map function

- Read tuples
  - Write tuples with “tag”

```
void map(Text key , Text values ,
        OutputCollector <TextPair, TextPair> output ,
        Reporter reporter) throws IOException
{
        output.collect(new TextPair(key.toString (), tag),
                new TextPair(values.toString (), tag));
}
```

PURDUE UNIVERSITY
Department of Computer Science

Partition the Data

- Special partition function
  - Partition only on key (join attribute)

```
int getPartition (TextPair key ,
        TextPair value , int numPartitions ) {
        return (key.getFirst ().hashCode () &
                        Integer.MAX_VALUE)
                % numPartitions ;
}
```

Reduce

Department of Computer Science

- Read file
  - If first dataset, save
  - If second dataset, output matches
- Assumes data sorted
  - But Hadoop takes care of this

Reduce

Department of Computer Science


```
void reduce(TextPair key , Iterator <TextPair> values , OutputCollector <Text , Text > output , Reporter reporter)
        throws IOException {

        ArrayList <Text > T1 = new ArrayList <Text >();
        Text tag = key.getSecond ();
        TextPair value = null;
        while(values.hasNext ()) {
                value = values.next ();
                if(value.getSecond ().compareTo(tag)==0) {
                        T1.add(value.getFirst ());
                } else {
                        for(Text val : T1) {
                                output.collect(key.getFirst (),
                                new Text(val.toString () + "\t" +
                                        value.getFirst ().toString ()));
}       }       }       }
```

Broadcast Join *(Blanas et al. SIGMOD'10)*

- Map-only algorithm
  - Everything done in the "first phase"
  - Saves move/sort of data
- Limitation: One dataset must fit in memory
  - Copy kept at every mapper
  - Mapper(s) then run on large dataset "in place"
  - Outputs join

But what about…

- Schema
  - Need to know what the data is about
- Queries
  - Do you really want to write map-reduce programs?
  - Optimization?

PURDUE UNIVERSITY
Department of Computer Science

HIVE: RDBMS on Hadoop

- Limited schema
  - Tables
  - Primitive types
- Subset of SQL
  - Select-Project
  - (equi)join
  - Group by
- Operations implemented using Map-Reduce

PURDUE UNIVERSITY
Department of Computer Science

What is Hive?

- A system for managing and querying structured data built on top of Hadoop
- Three main components:
  - MapReduce for execution
  - Hadoop Distributed File System for storage
  - Metadata in an RDBMS
- Hive QL based on SQL
  - Easy for users familiar with SQL

PURDUE UNIVERSITY
Department of Computer Science

Hive Architecture

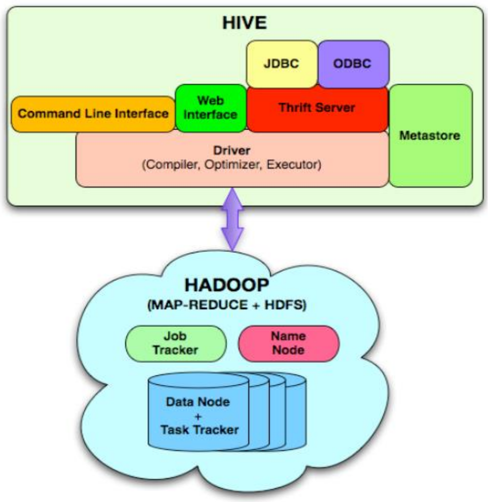
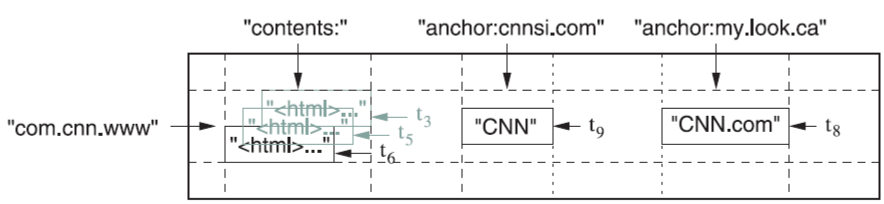


Fig. 1: Hive System Architecture

PURDUE UNIVERSITY
Department of Computer Science

Google BigTable

- Simple data model



- Tables distributed
  - "row keys"
- Transactional consistency only on a per-row basis

Google Bigtable

- Multi-version
  - Each row timestamped
- Three-level location hierarchy
  - Claim: “B-tree like”