PURDUE UNIVERSITY

CS34800
Information Systems

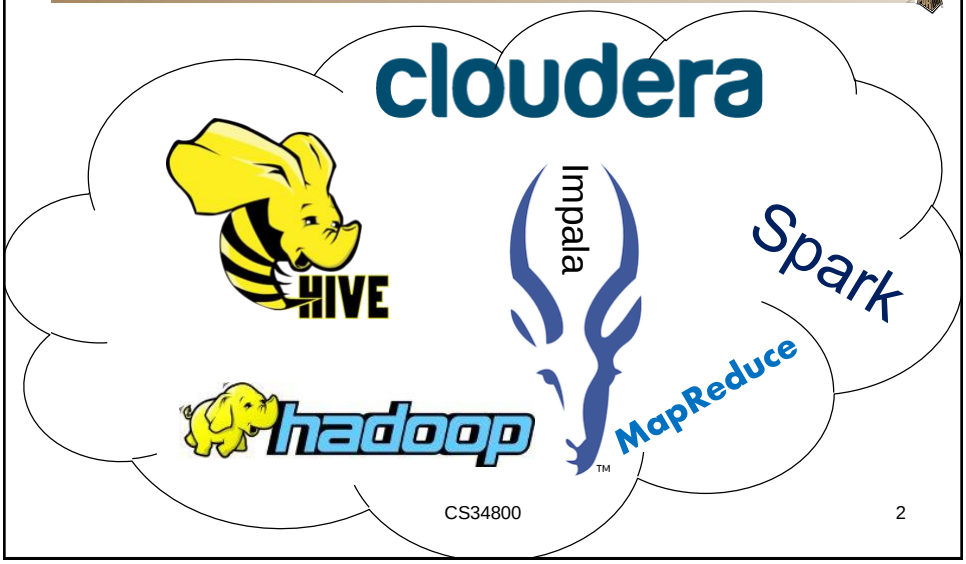
Big Data

Prof. Chris Clifton

2 November 2016

Indiana Center for Database Systems

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

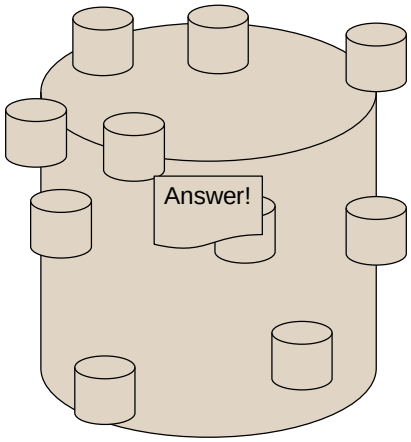
- PERFORMANCE!
  - More speed, bigger data
- But this doesn't come for free
  - *Eventual* consistency (eventually all the updates will occur)
  - No isolation guarantees
  - Limited reliability guarantees

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
  - We will use Spark in this course

Basic Idea: Divide and Conquer

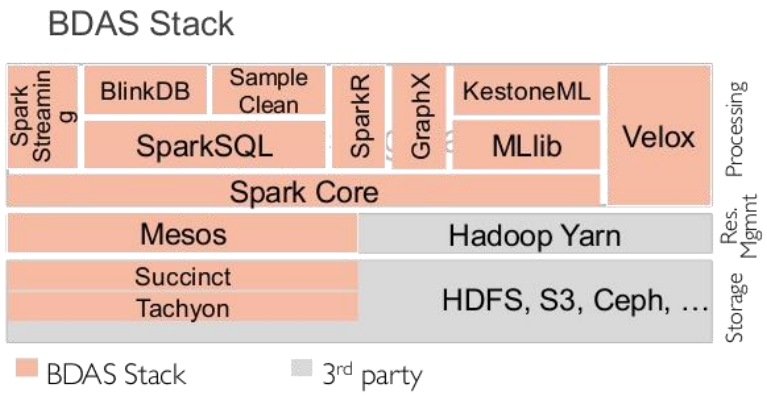
- Divide data into units
- Compute on those units
- Combine results
- *Need algorithms where this works!*



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)

Spark: Implementation of this Programming Model



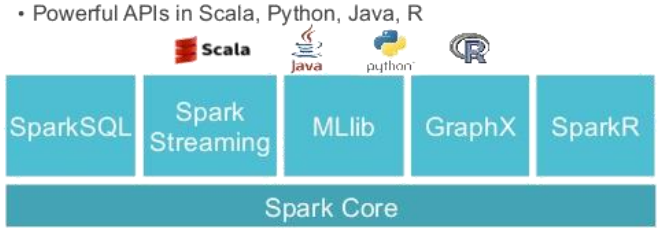
Spark Applications

Spark

Unifies ***batch, interactive, streaming*** computations

Easy to build sophisticated applications

- Support iterative, graph-parallel algorithms
- Powerful APIs in Scala, Python, Java, R



Spark 101

- Apache Spark
  - Open Source
  - Extensive developer community
  - Growing commercial use
- Somewhat heavy to set up
  - *First, you need a cloud…*
  - But we'll handle this for the next project

Creating a Data Object

```
>>> sc
<pyspark.context.SparkContext object at 0x10ea7d4d0>
>>> pagecounts = sc.textFile("data/pagecounts")
>>> pagecounts
MapPartitionsRDD[1] at textFile at
NativeMethodAccessorImpl.java:-2
```

- *Assume data/pagecounts is pageviews of Wikipedia pages*

Viewing Data (first 10 records)

```
>>> pagecounts.take(10)
...
[u'20090505-000000 aa.b ?71G4Bo1cAdWyg 1 14463', u'20090505-
000000 aa.b Special:Statistics 1 840', u'20090505-000000 aa.b
Special:Whatlinkshere/MediaWiki:Returnto 1 1019', u'20090505-
000000 aa.b Wikibooks:About 1 15719', u'20090505-000000 aa
?14mFX1ildVnBc 1 13205', u'20090505-000000 aa
?53A%2FuYP3FfnKM 1 13207', u'20090505-000000 aa
?93HqrnFc%2EiqRU 1 13199', u'20090505-000000 aa
?95iZ%2Fjuimv31g 1 13201', u'20090505-000000 aa File:Wikinews-
logo.svg 1 8357', u'20090505-000000 aa Main_Page 2 9980']
```

Prettier:

```
>>> for x in pagecounts.take(10):
...     print x
...
20090505-000000 aa.b ?71G4Bo1cAdWyg 1 14463
20090505-000000 aa.b Special:Statistics 1 840
20090505-000000 aa.b Special:Whatlinkshere/MediaWiki:Returnto 1 1019
20090505-000000 aa.b Wikibooks:About 1 15719
20090505-000000 aa ?14mFX1ildVnBc 1 13205
20090505-000000 aa ?53A%2FuYP3FfnKM 1 13207
20090505-000000 aa ?93HqrnFc%2EiqRU 1 13199
20090505-000000 aa ?95iZ%2Fjuimv31g 1 13201
20090505-000000 aa File:Wikinews-logo.svg 1 8357
20090505-000000 aa Main_Page 2 9980
```


Caching Results

>>> pagecounts.count()

- May take a long time

>>> enPages = pagecounts.filter(lambda x: x.split(" ")[1] == "en").cache()

- doesn't actually do anything

>>> enPages.count()

- slow the first time, fast in later calls

Histogram of page views

- First, divide the data

```
>>> enTuples = enPages.map(lambda x: x.split(" "))
```

- And create a count for each date

```
>>> enKeyValuePairs = enTuples.map(lambda x:
(x[0][:8], int(x[3])))
```

- Then combine

```
>>> enKeyValuePairs.reduceByKey(lambda x, y: x + y,
1).collect()
[(u'20090507', 6175726), (u'20090505', 7076855)]
```


Single command to do it all (and only return where >200k)

```
>>> enPages.map(lambda x: x.split(" ")).
map(lambda x: (x[2],int(x[3]))).
reduceByKey(lambda x, y: x + y, 40).
filter(lambda x: x[1] > 200000).
map(lambda x: (x[1], x[0])).collect()

[(451126, u'Main_Page'), (1066734,
u'404_error/'), (468159, u'Special:Search')]
```

PURDUE UNIVERSITY

CS34800
Information Systems

Big Data

Prof. Chris Clifton

4 November 2016

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
- Example: Hadoop
  - Slides courtesy Yahoo!

Introduction to Hadoop

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

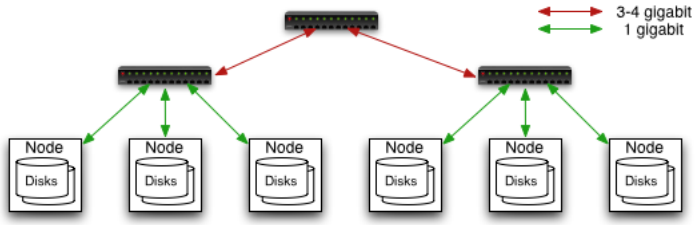
Problem

- 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

Commodity Hardware Cluster



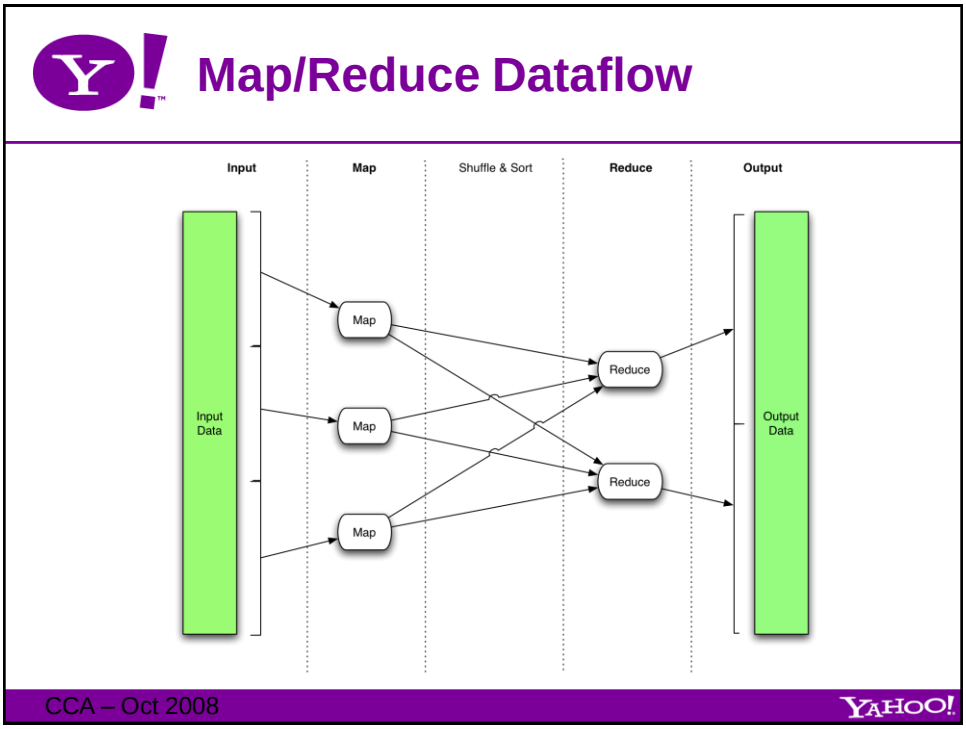
- Typically in 2 level architecture
  - Nodes are commodity PCs
  - 40 nodes/rack
  - Uplink from rack is 8 gigabit
  - Rack-internal is 1 gigabit

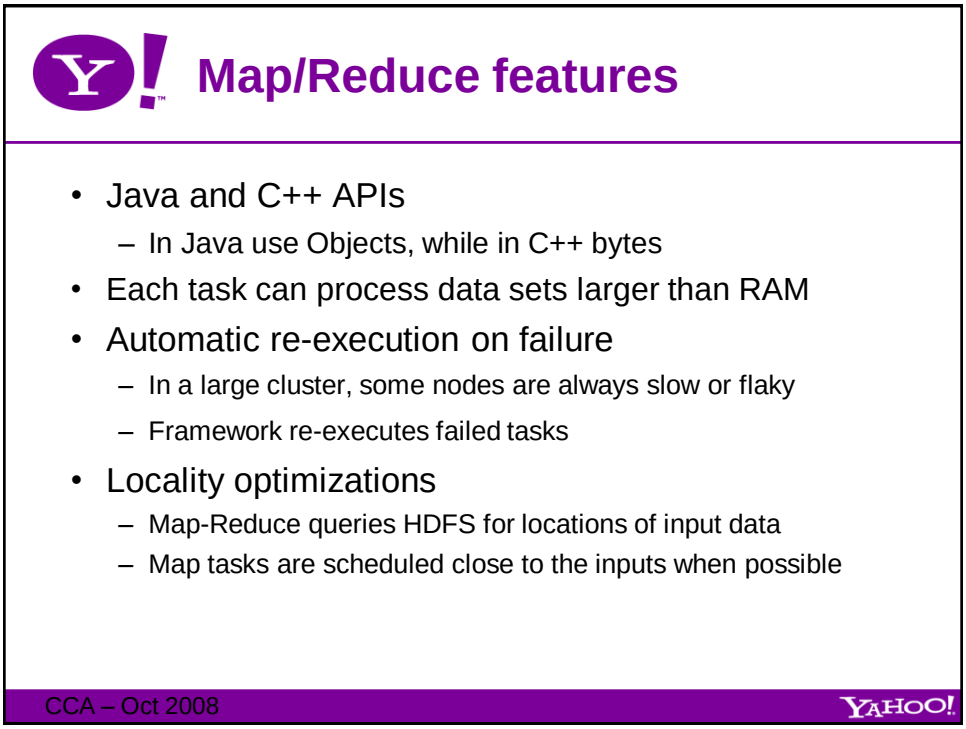
Distributed File System

- 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.

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





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));
}
}
```

```
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);
}
}
```

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

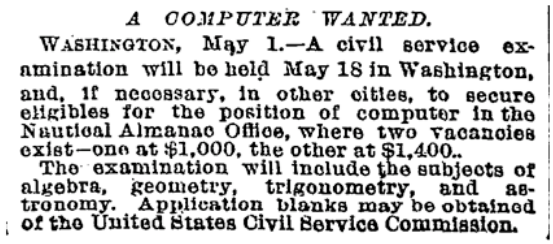
- Search needs a graph of the “known” web
  - Invert edges, compute link text, whole graph heuristics
- Periodic batch job using Map/Reduce
  - Uses a chain of ~100 map/reduce jobs
- Scale
  - 1 trillion edges in graph
  - Largest shuffle is 450 TB
  - Final output is 300 TB compressed
  - Runs on 10,000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

Research Clusters

- The grid team runs the research clusters as a service to Yahoo researchers
- Mostly data mining/machine learning jobs
- Most research jobs are *not* Java:
  - 42% Streaming
    - Uses Unix text processing to define map and reduce
  - 28% Pig
    - Higher level dataflow scripting language
  - 28% Java
  - 2% C++

NY Times

- Needed offline conversion of public domain articles from 1851-1922.
- Used Hadoop to convert scanned images to PDF
- Ran 100 Amazon EC2 instances for around 24 hours
- 4 TB of input
- 1.5 TB of output



A COMPUTER WANTED.

WASHINGTON, May 1.—A civil service examination will be held May 18 in Washington, and, if necessary, in other cities, to secure eligibles for the position of computer in the Nautical Almanac Office, where two vacancies exist—one at $1,000, the other at $1,400.

The examination will include the subjects of algebra, geometry, trigonometry, and astronomy. Application blanks may be obtained of the United States Civil Service Commission.

Published 1892, copyright New York Times

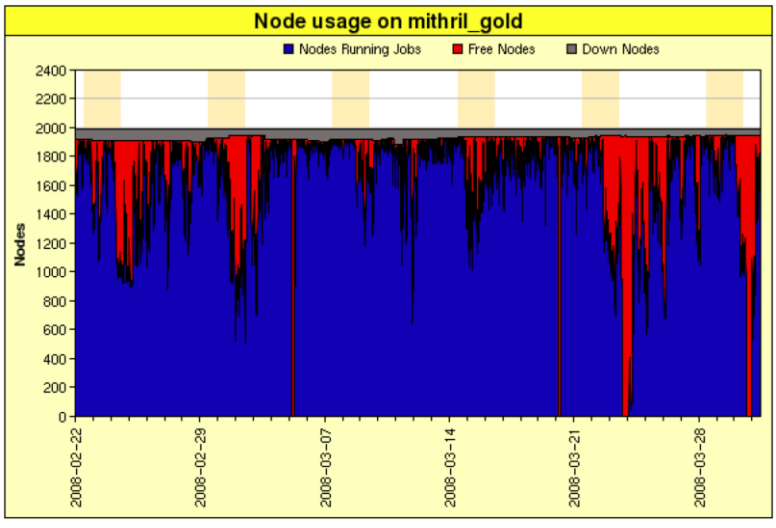
Terabyte Sort Benchmark

- Started by Jim Gray at Microsoft in 1998
- Sorting 10 billion 100 byte records
- Hadoop won the general category in 209 seconds
  - 910 nodes
  - 2 quad-core Xeons @ 2.0Ghz / node
  - 4 SATA disks / node
  - 8 GB ram / node
  - 1 gb ethernet / node
  - 40 nodes / rack
  - 8 gb ethernet uplink / rack
- Previous records was 297 seconds
- Only hard parts were:
  - Getting a total order
  - Converting the data generator to map/reduce

Hadoop clusters

- We have ~20,000 machines running Hadoop
- Our largest clusters are currently 2000 nodes
- Several petabytes of user data (compressed, unreplicated)
- We run hundreds of thousands of jobs every month

Research Cluster Usage

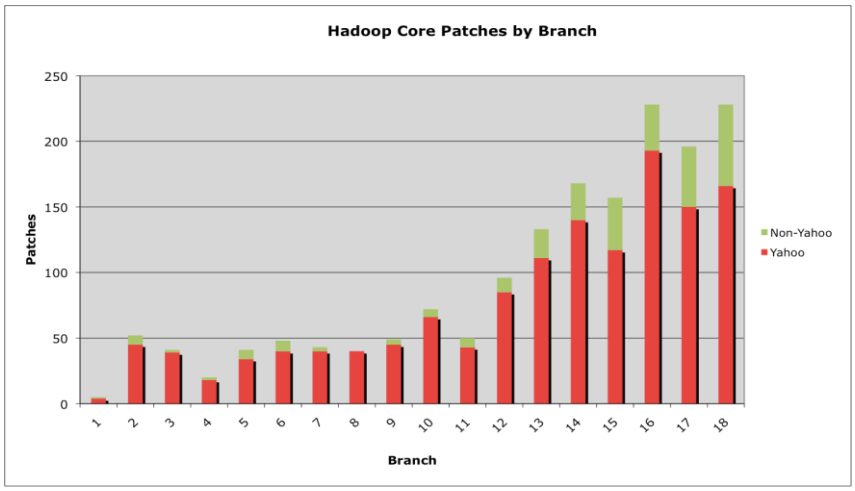


CCA – Oct 2008 YAHOO!

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!

Size of Releases



Who Uses Hadoop?

- Amazon/A9
- AOL
- Facebook
- Fox interactive media
- Google / IBM
- New York Times
- PowerSet (now Microsoft)
- Quantcast
- Rackspace/Mailtrust
- Veoh
- Yahoo!
- More at http://wiki.apache.org/hadoop/PoweredBy

What's Next?

- Better scheduling
  - Pluggable scheduler
  - Queues for controlling resource allocation between groups
- Splitting Core into sub-projects
  - HDFS, Map/Reduce, Hive
- Total Order Sampler and Partitioner
- Table store library
- HDFS and Map/Reduce security
- High Availability via Zookeeper
- Get ready for Hadoop 1.0

HIVE: RDBMS on Hadoop

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

But what about…

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

HIVE: RDBMS on Hadoop

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

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