



Big Data
Too Big To Ignore



Geert



- ⚙️ Big Data Consultant and Manager
- ⚙️ Currently finishing a 3rd Big Data project
- ⚙️ IBM & Cloudera Certified
- ⚙️ IBM & Microsoft Big Data Partner

Agenda

- ⚙️ Defining Big Data
- ⚙️ Introduction to Hadoop





Volume

Big Data

Velocity

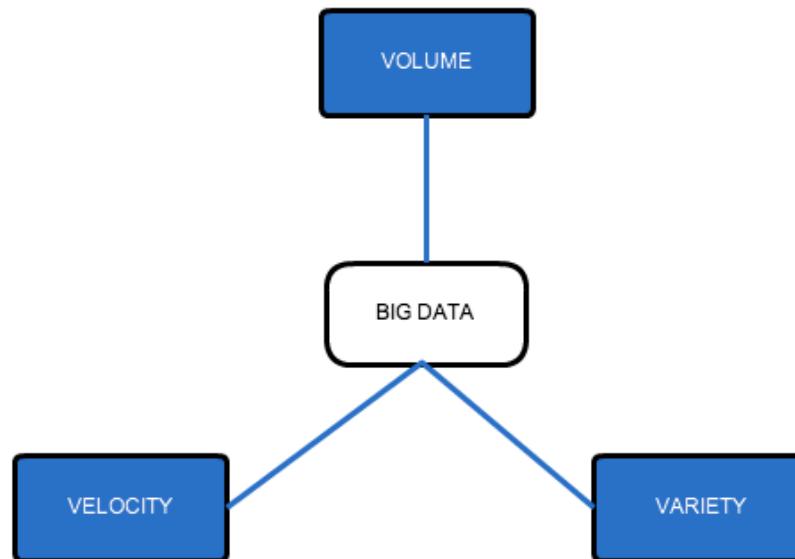


Variety

Big Data Technical Drivers



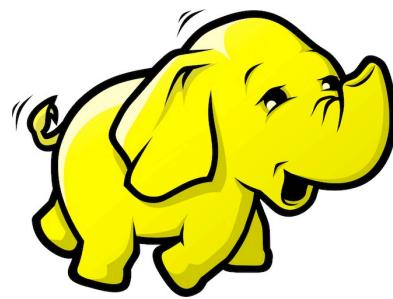
From Terrabytes to Petabytes



From Batch to Near Real-Time

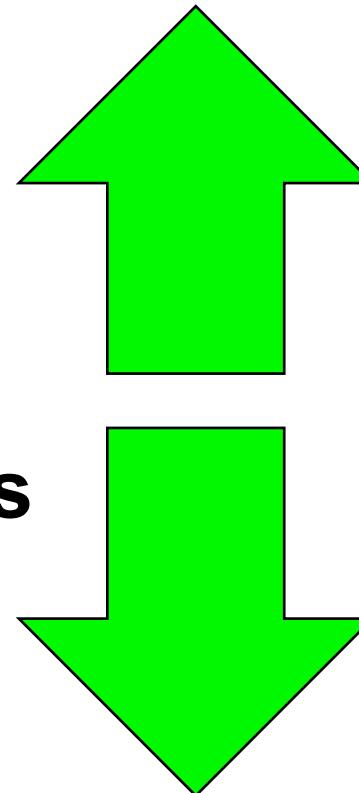
*From Structured to
Semi-Structured & Unstructured*

Big Data Business Drivers



Do More

with Less



ANALYTICS

COSTS



Analyzing Big Data sets will become a key basis for competition

Leaders in every sector will have to grapple the implications of big data

McKinsey

Big Data analytics are rapidly emerging as the preferred solution to business and technology trends that are disrupting

Enterprises should not delay implementation of big data analytics

Gartner

Use Hadoop to gain a competitive advantage over more risk-averse enterprises

Prioritize Big Data projects that might benefit from Hadoop

Forrester Research

Transformation of Online Marketing



THEN...

Leads

Company	First	Last	Oppy	Created
Acme	Fred	Langan	\$250K	6/08/12
BigCo	Tom	Jones	\$100K	6/17/12
DealCo	Jan	Sedor	\$50K	7/01/12
Stor Works	Liza	Grear	\$750K	7/14/12
RF Group	Carl	Tomer	\$47K	7/18/12

NOW...

Marketing and Sales
Recommendations



BLOGS.FORBES.COM/DAVEFEINLEIB

Transformation of Customer Service

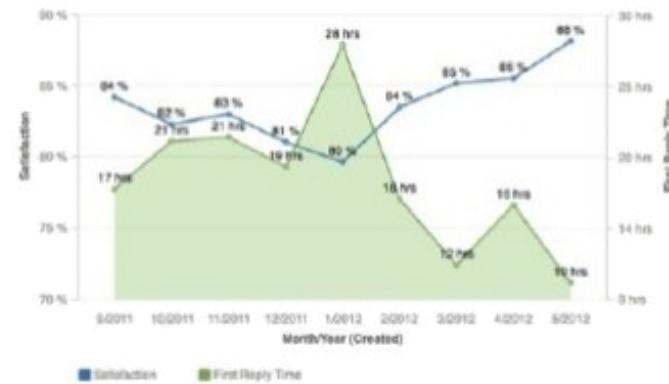
THEN...

Unhappy
customers



NOW...

Customer insight



BLOGS.FORBES.COM/DAVEFEINLEIB

Big Data Definition



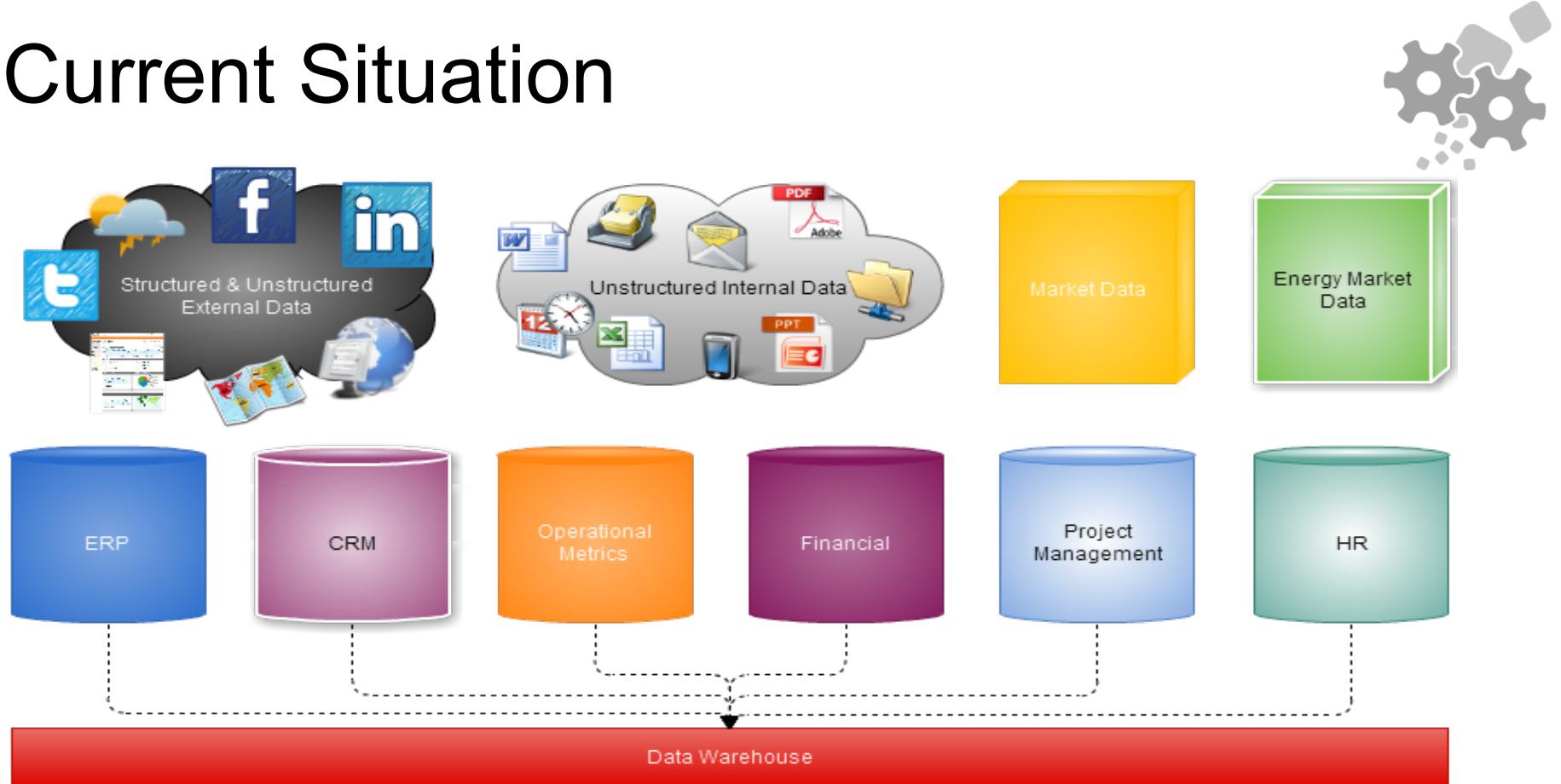
Big Data Technologies allow you to implement Use Cases which Legacy Technologies can't.



Implementing Big Data

Our Vision on Data

Current Situation



Our Vision #1



Focus on Data not on Derived Data

Our Vision #2



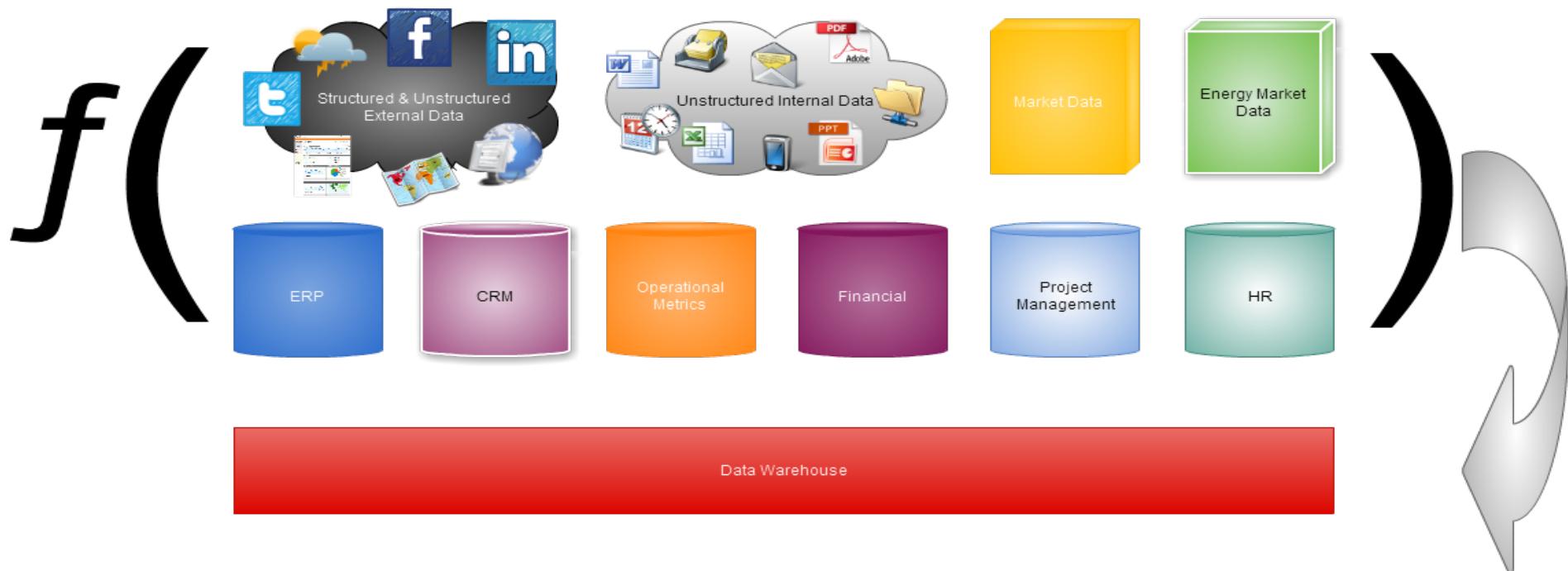
Data is immutable

Our Vision #3



Query = function (all data)

Concept

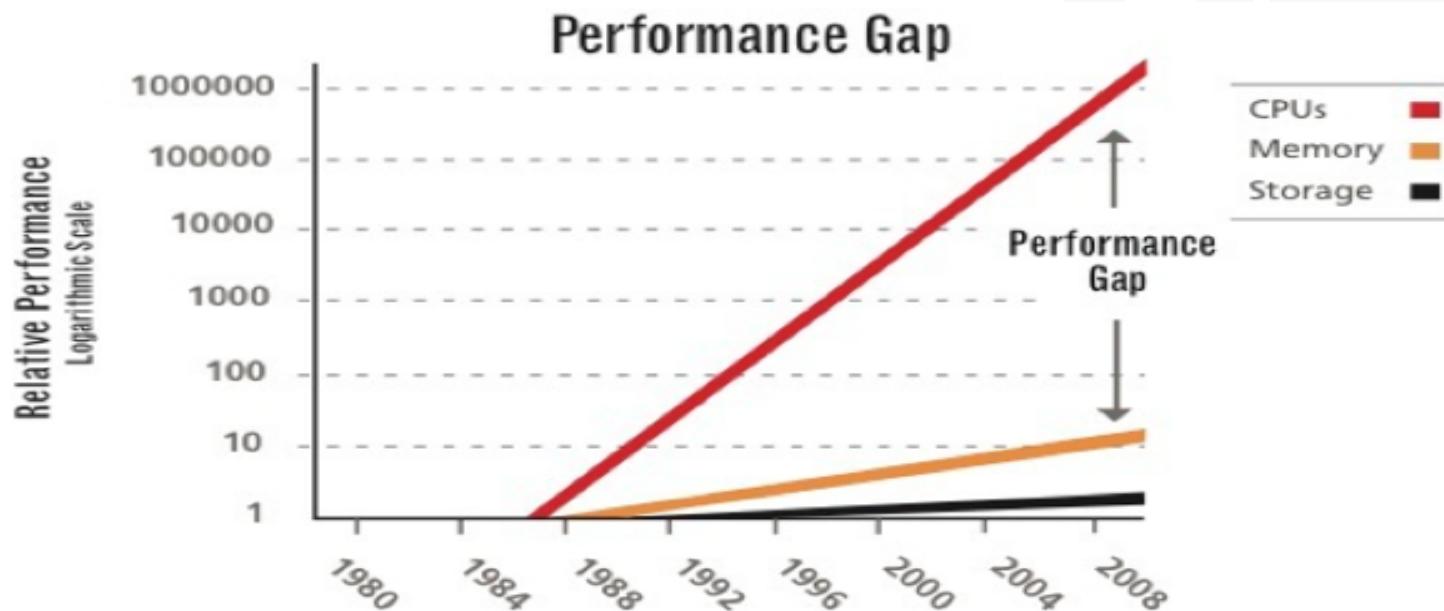




Introducing The Hadoop Ecosystem



Context: Performance Gap Trend



Context: Exponential for Decades



- Abundance of
 - computing & storage
 - generated data (estimated 8ZB in '15)
 - things
- More data provides greater value
- Traditional data doesn't scale well
- It's time for a new approach!

New Hardware Approach

Traditional

- Exotic HW
 - big central servers
 - SAN
 - RAID
- Hardware reliability
- Limited scalability
- Expensive



Big Data

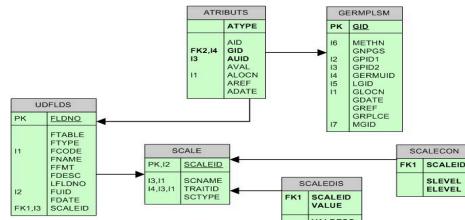
- Commodity HW
 - racks of pizza boxes
 - Ethernet
 - JBOD
- Unreliable HW
- Scales further
- Cost effective



New Software Approach

Traditional

- Monolithic
 - Centralized
 - RDBMS
- Schema first
- Proprietary



Big Data

- Distributed
 - storage & compute nodes
- Raw data
- Open source

0265740 072451 007735 147620 061064 157435 113057 155356 114603
0265760 107204 102816 171451 046040 120223 001774 030477 045673
0266000 171317 116055 155117 134444 167210 041405 147122 050505
0266020 070176 047705 13754 157477 105592 076515 177356 056383
0266040 070176 047705 13754 157477 105592 076515 177356 056383
0266060 041023 074017 127113 003214 037025 037640 066173 123424
0266100 087001 0105 140090 065341 070415 100002 056615 056615
0266120 056745 074447 157445 157445 065341 070415 100002 056615
0266140 024451 007424 114200 077733 024430 012546 172404 102345
0266160 040223 050170 055164 164634 047151 126525 112514 032315
0266180 062276 062276 157455 157455 065341 070415 100002 056615
0266200 117156 030746 154234 152001 151144 163706 136237 164376
0266220 137055 062276 163755 115456 005322 132567 073215 002655
0266240 037657 037657 175837 104754 036436 072172 160750 043643 145410
0266320 072074 000007 040627 070652 173011 002215 125132 140214
0266340 120601 170106 040457 057227 056631 041462 106321 056631
0266360 120601 170106 040457 057227 056631 041462 106321 056631
0266400 020243 005602 004146 121574 124651 006634 071332 102070
0266420 157504 160307 168589 074051 065520 116438 167273 030635
0266440 157504 160307 168589 074051 065520 116438 167273 030635
0266460 026630 007210 000630 121234 076033 140764 000737 003276
0266500 114050 042647 104475 110537 066710 104754 075447 112254
0266520 146207 015135 024446 30101 072457 040764 165513 156412
0266540 146207 015135 024446 30101 072457 040764 165513 156412
0266560 166410 067251 158516 106405 136770 030515 064740 022032
0266620 142134 067251 158516 106405 136770 076350 031115 015243
0266620 093074 016294 044055 107230 110063 033950 032765 172463

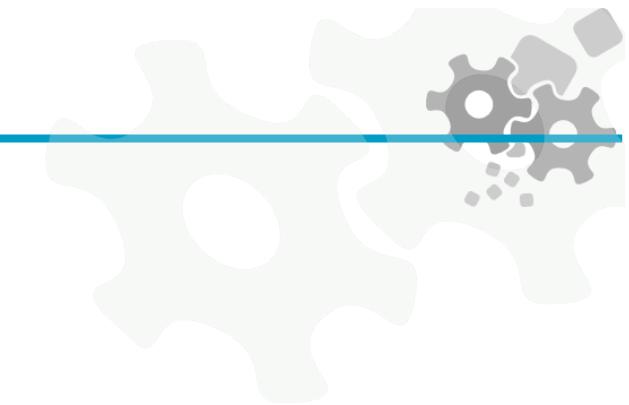
Hadoop

- De facto big data industry standard (batch)
- Vendor adoption
 - IBM, Microsoft, Oracle, EMC, ...
- A collection of projects at Apache
 - HDFS, MapReduce, Hive, Pig, Hbase, Flume, Oozie, ...
- Main components
 - HDFS
 - MapReduce
- Cluster
 - Set of machines running HDFS and MapReduce



Distributions

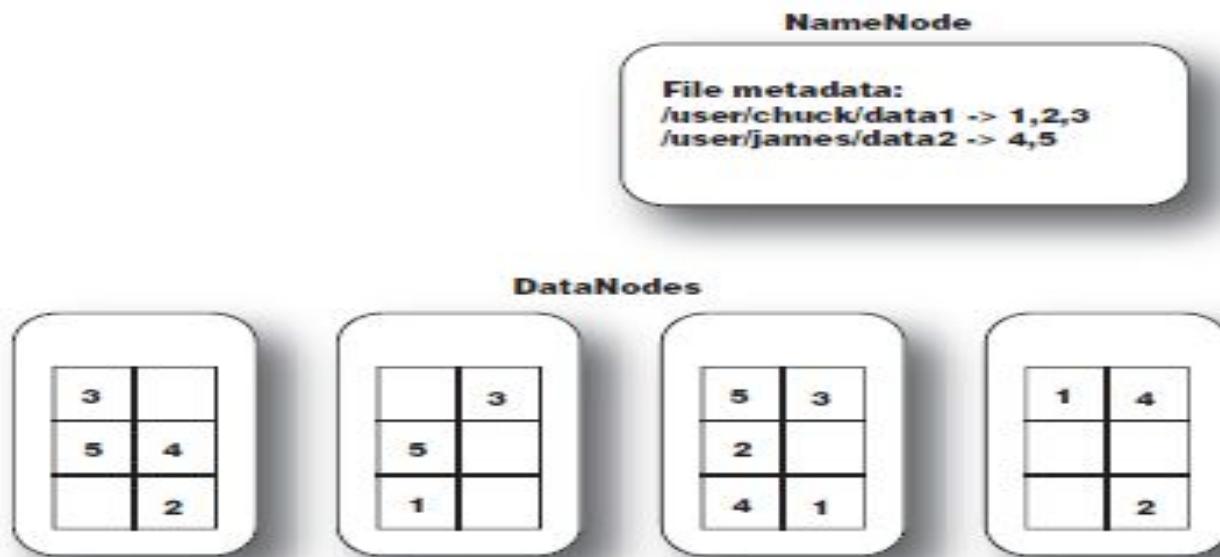
- Cloudera
 - www.cloudera.com
 - Cloudera Enterprise subscription
 - Currently CDH3
 - Linux package
 - Virtual machine
 - Cloud
 - Stack
 - hadoop, hbase, hive, pig, mahout, flume, ...
 - Cloudera SCM
 - Connectors for Teradata, Netezza, Microstrategy and Quest



Distributions

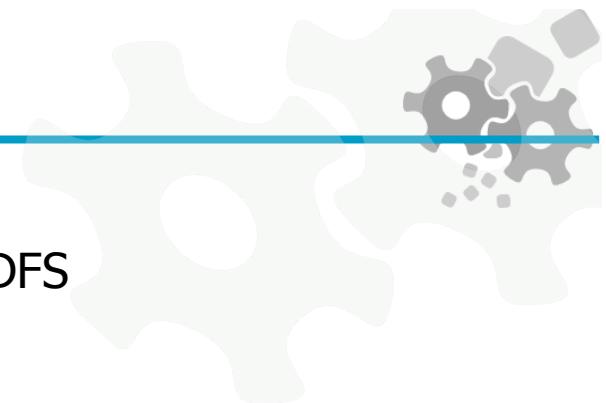
- Hortonworks
 - www.hortonworks.com
 - Hortonworks Data Platform Powered by Apache Hadoop, 100% opensource solution
- IBM
 - Offers a derivative version of Apache Hadoop that IBM supports on IBM JVMs on a number of platforms
- Microsoft
 - Hadoop on Azure & Hadoop on Premise
- Apache Hadoop
 - Distribution for individual components only

HDFS



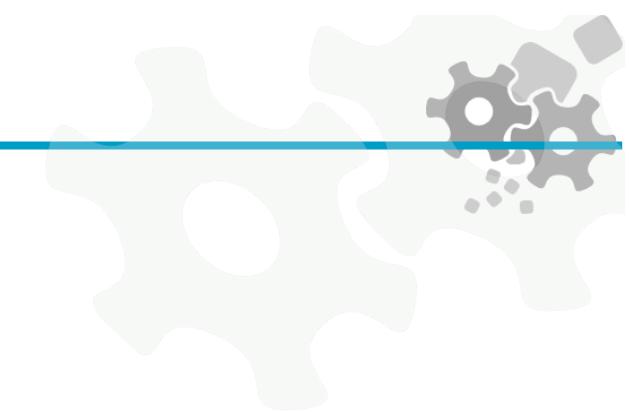
Access to HDFS

- The HDFS java client api's can be used
- Typically files are moved from local filesystem into HDFS
- Using hadoop fs commands
- Through Hue (Cloudera SCM)
- Fuse
- HDFS DAV



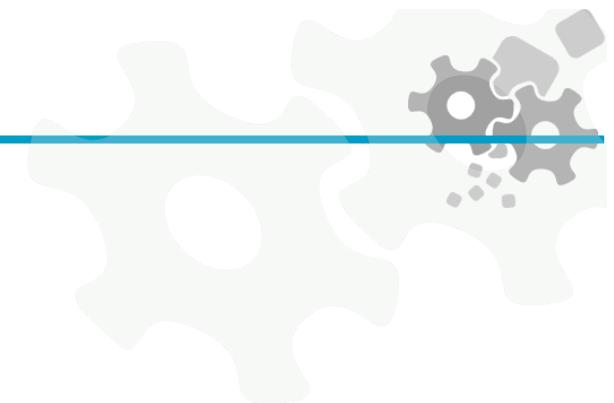
hadoop fs examples

- Get directory listing of user home directory in hdfs
 - hadoop fs -ls
- Get directory listing of root directory
 - hadoop fs -ls /
- Copy a local file to hdfs (user home directory)
 - hadoop fs -copyFromLocal foo.txt foo.txt
- Move a file from hdfs to local disk
 - hadoop fs -copyToLocal /user/geert/foo.txt foo.txt
- Display the contents of a file
 - hadoop fs -cat /data/as400/customers.txt
- Make a new directory (user home directory)
 - hadoop fs -mkdir output



hadoop fs examples

- Remove a directory
 - hadoop fs -rm /business/data
- Remove a directory and all its content (recursive)
 - hadoop fs -rmr /business/data
- More information about hadoop fs command options
 - http://hadoop.apache.org/common/docs/r0.20.2/hdfs_shell.html



hadoop fs examples

```
training@ubuntu: ~
File Edit View Terminal Help
training@ubuntu:~$ hadoop fs -ls /
Found 3 items
drwxrwxr-x  - root      supergroup          0 2011-12-18 06:28 /tmp
drwxr-xr-x  - training   supergroup          0 2011-01-05 06:36 /user
drwxr-xr-x  - hadoop    supergroup          0 2011-01-05 06:43 /var
training@ubuntu:~$ hadoop fs -ls
Found 8 items
drwxr-xr-x  - training   supergroup          0 2011-12-18 05:38 /user/training/count2
drwxr-xr-x  - training   supergroup          0 2011-12-18 05:39 /user/training/count3
drwxr-xr-x  - training   supergroup          0 2011-12-18 06:01 /user/training/length
drwxr-xr-x  - training   supergroup          0 2011-12-18 06:18 /user/training/movie
drwxr-xr-x  - training   supergroup          0 2011-12-18 06:26 /user/training/movierating
drwxr-xr-x  - training   supergroup          0 2011-12-18 05:37 /user/training/pwords
drwxr-xr-x  - training   supergroup          0 2011-12-18 05:28 /user/training/shakespeare
drwxr-xr-x  - training   supergroup          0 2011-12-18 05:33 /user/training/wordcounts
training@ubuntu:~$ whoami
training
training@ubuntu:~$ hadoop fs -copyFromLocal pig_1324218511127.log pig_1324218511127.log
training@ubuntu:~$ hadoop fs -ls
Found 9 items
drwxr-xr-x  - training   supergroup          0 2011-12-18 05:38 /user/training/count2
drwxr-xr-x  - training   supergroup          0 2011-12-18 05:39 /user/training/count3
drwxr-xr-x  - training   supergroup          0 2011-12-18 06:01 /user/training/length
drwxr-xr-x  - training   supergroup          0 2011-12-18 06:18 /user/training/movie
drwxr-xr-x  - training   supergroup          0 2011-12-18 06:26 /user/training/movierating
drwxr-xr-x  - training   supergroup          0 2011-12-18 05:37 /user/training/pwords
drwxr-xr-x  - training   supergroup          0 2011-12-18 05:28 /user/training/shakespeare
drwxr-xr-x  - training   supergroup          0 2011-12-18 05:33 /user/training/wordcounts
-rw-r--r--  1 training   supergroup  5917 2011-12-18 09:08 /user/training/pig_1324218511127.log
training@ubuntu:~$
```

Hadoop Namenode webpage

NameNode 'node1.c.foundation.local:8020'

Started:	Tue Dec 13 19:28:33 CET 2011
Version:	0.20.2-cdh3u2, r95a824e4005b2a94fe1c11f1ef9db4c672ba43cb
Compiled:	Thu Oct 13 21:51:41 PDT 2011 by root from Unknown
Upgrades:	There are no upgrades in progress.

[Browse the filesystem](#)
[Namenode Logs](#)

Cluster Summary

4813 files and directories, 2798 blocks = 7611 total. Heap Size is 174.38 MB / 602.69 MB (28%)

Configured Capacity	:	6.97 TB
DFS Used	:	100.09 GB
Non DFS Used	:	478.95 GB
DFS Remaining	:	6.41 TB
DFS Used%	:	1.4 %
DFS Remaining%	:	91.89 %
Live Nodes	:	4
Dead Nodes	:	0
Decommissioning Nodes	:	0
Number of Under-Replicated Blocks	:	37

NameNode Storage:

Storage Directory	Type	State
/dfs/hn	IMAGE_AND_EDITS	Active

Cloudera's Distribution including Apache Hadoop, 2011.

NameNode 'node1.c.foundation.local:8020'

Started:	Tue Dec 13 19:28:33 CET 2011
Version:	0.20.2-cdh3u2, r95a824e4005b2a94fe1c11f1ef9db4c672ba43cb
Compiled:	Thu Oct 13 21:51:41 PDT 2011 by root
Upgrades:	There are no upgrades in progress.

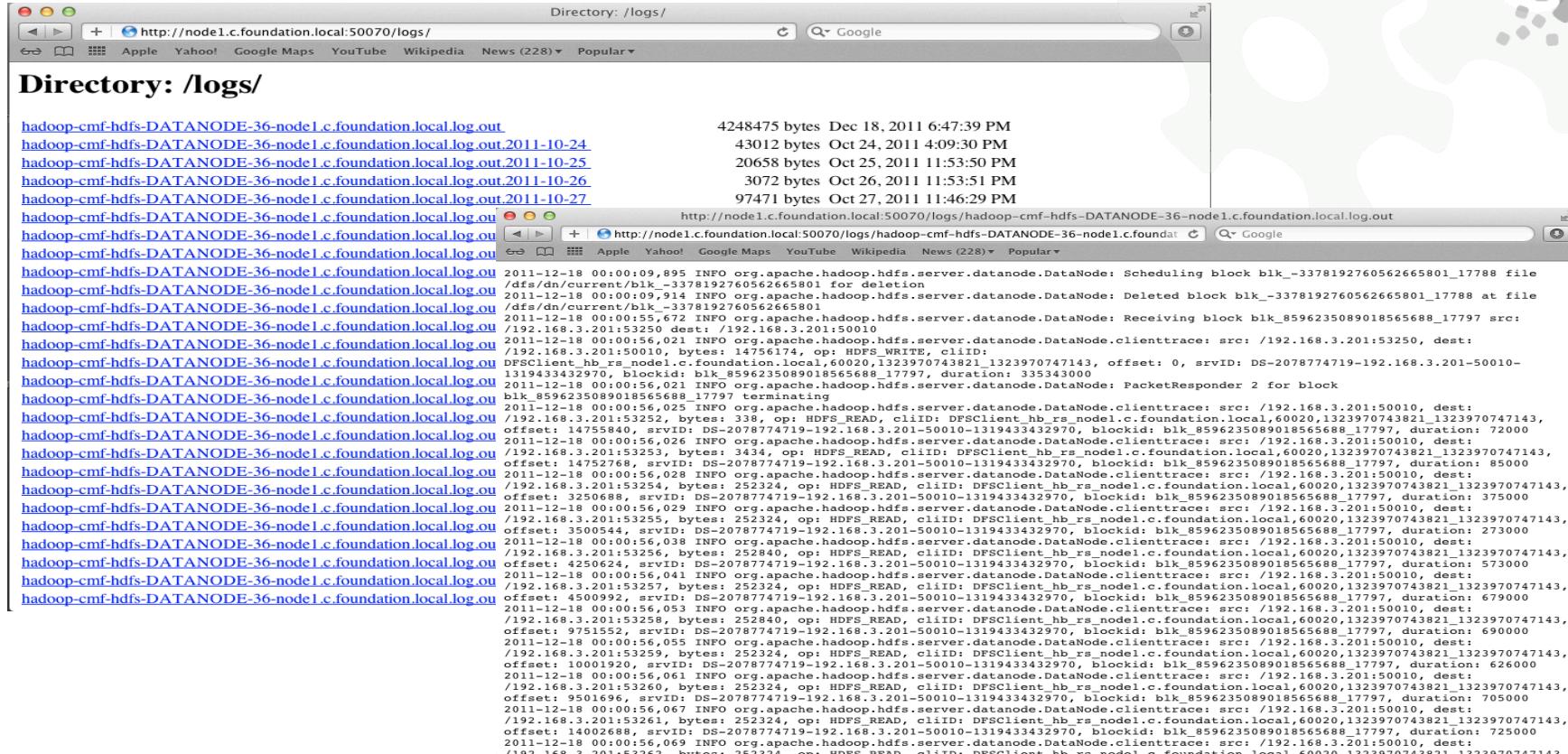
[Browse the filesystem](#)
[Namenode Logs](#)
[Go back to DFS home](#)

Live Datanodes : 4

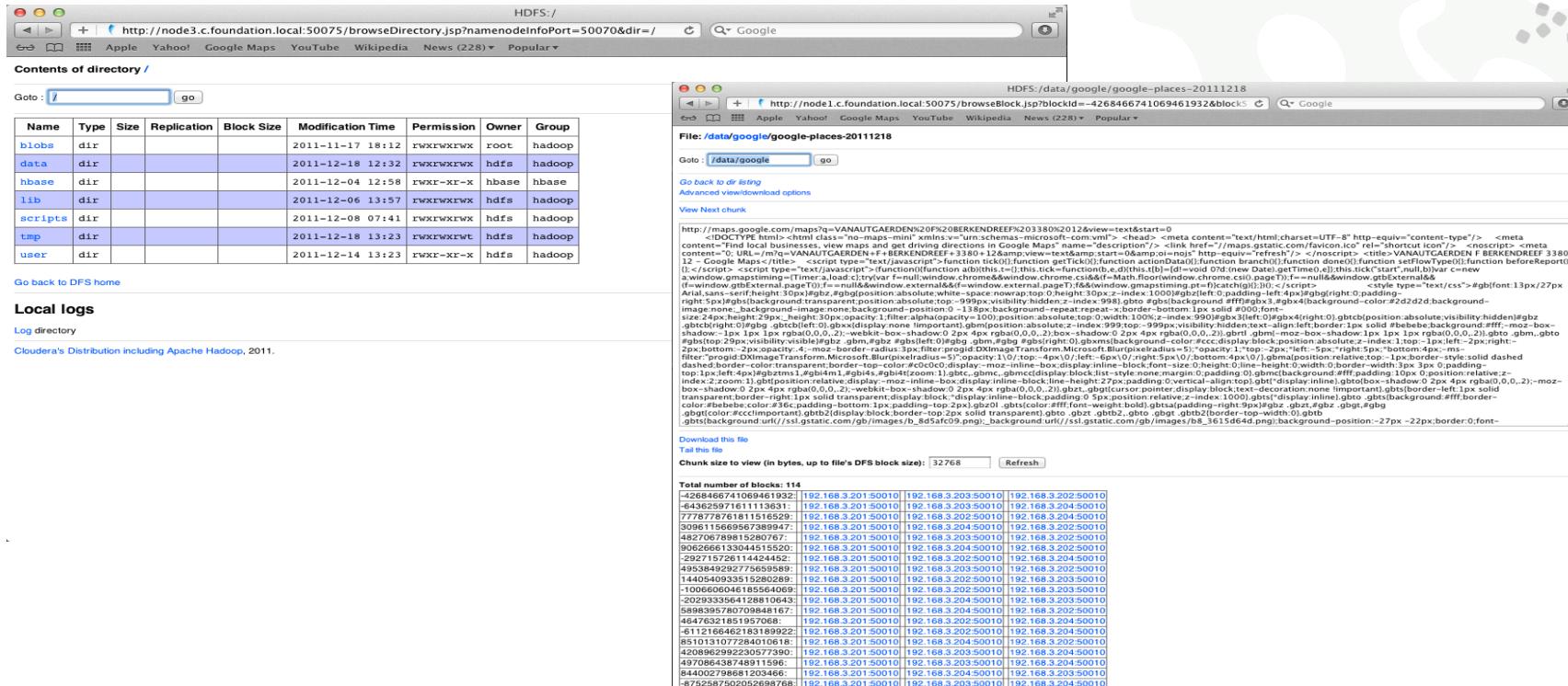
Node	Last Contact	Admin State	Configured Capacity (TB)	Used (TB)	Non DFS Used (TB)	Remaining (TB)	Used (%)	Used (%)	Remaining (%)	Blocks	Failed Volumes
node1	2	In Service	1.74	0.02	0.22	1.51	0.87	<div style="width: 86.74%;"></div>	86.74	898	0
node2	2	In Service	1.74	0.03	0.08	1.63	1.72	<div style="width: 93.41%;"></div>	93.41	2754	0
node3	2	In Service	1.74	0.03	0.08	1.63	1.68	<div style="width: 93.53%;"></div>	93.53	2742	0
node4	2	In Service	1.74	0.02	0.08	1.64	1.33	<div style="width: 93.89%;"></div>	93.89	2040	0

Cloudera's Distribution including Apache Hadoop, 2011.

Hadoop Namenode webpage

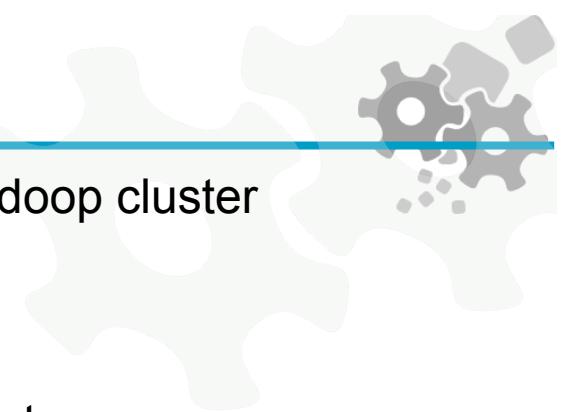


Hadoop Namenode webpage



MapReduce

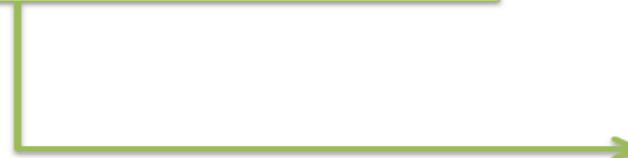
- MapReduce is the system used to process data in the Hadoop cluster
- Consists of two phases
 - Map & Reduce
 - Between the two is a stage known as the shuffle and sort
- Data Locality
 - Each Map task operates on a discrete portion of the overall dataset
 - Typically one HDFS block of data
- After all Maps are complete, the MapReduce system distributes the intermediate data to nodes which perform the Reduce phase



MapReduce

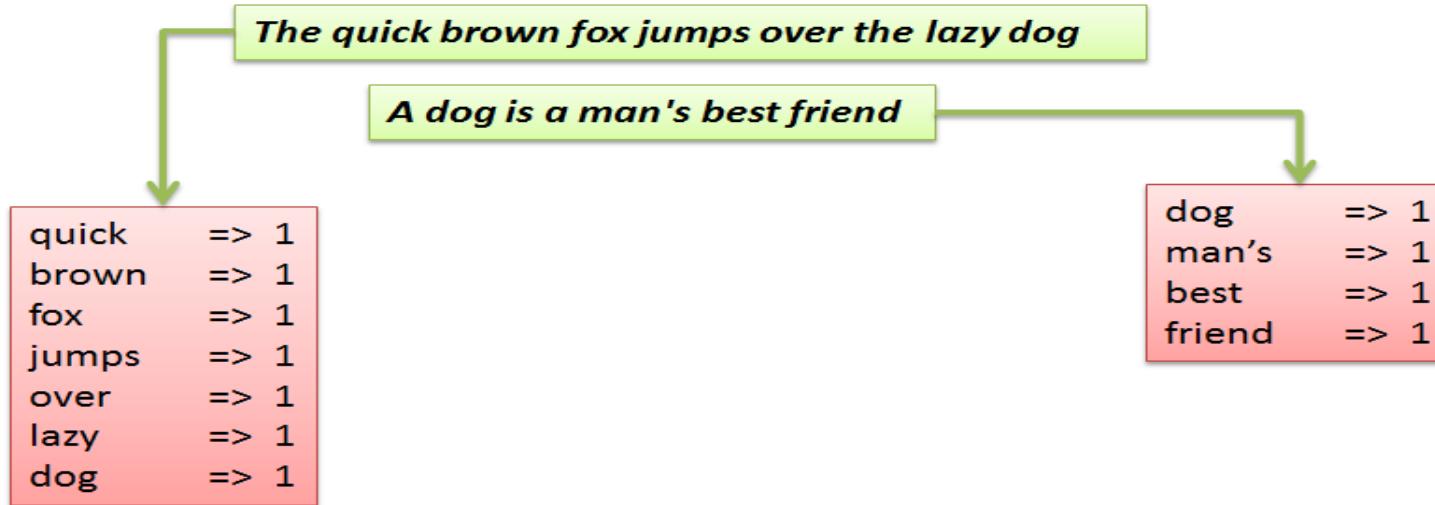
Count the number of words in the following paragraph.
Make sure you do not count “Stop Words” (the, a, is)

***The quick brown fox jumps over the lazy dog
A dog is a man's best friend***

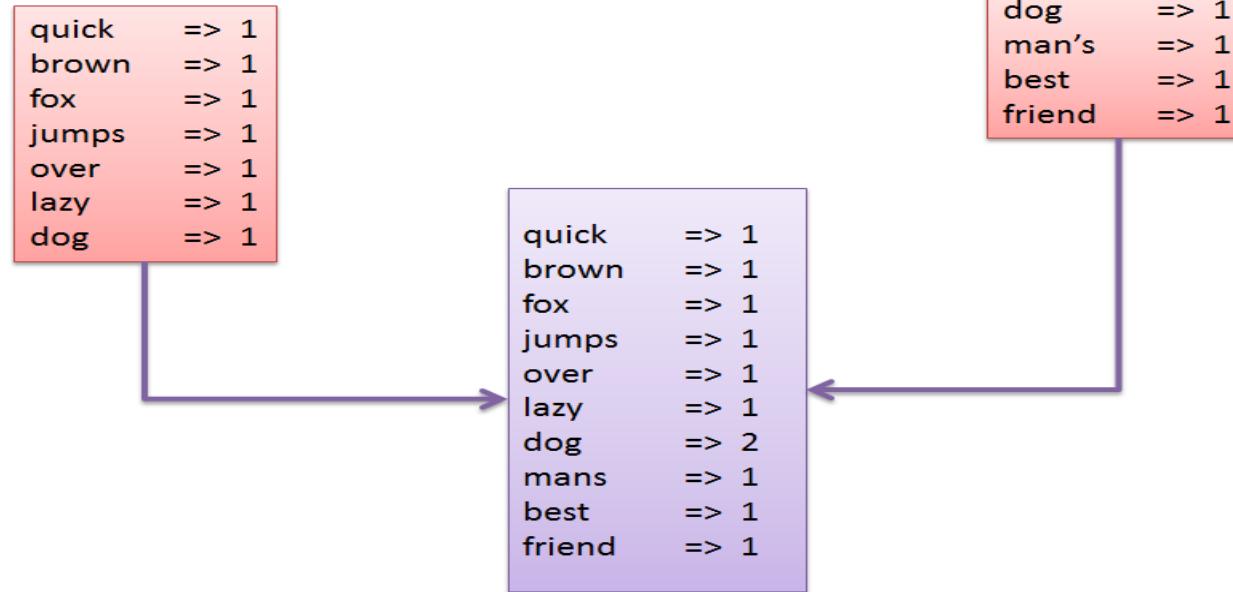


quick	=> 1
brown	=> 1
fox	=> 1
jumps	=> 1
over	=> 1
lazy	=> 1
dog	=> 2
mans	=> 1
best	=> 1
friend	=> 1

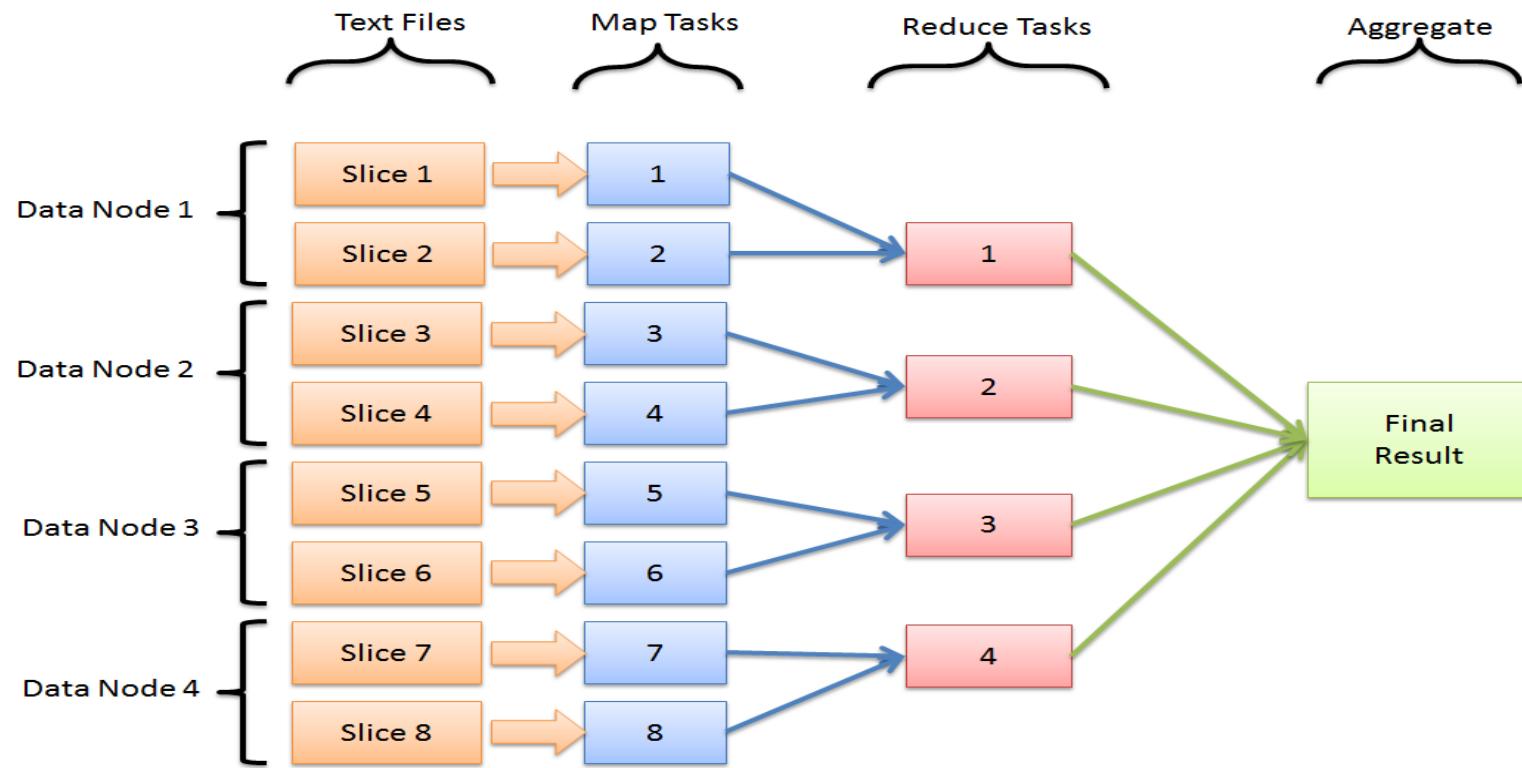
MapReduce



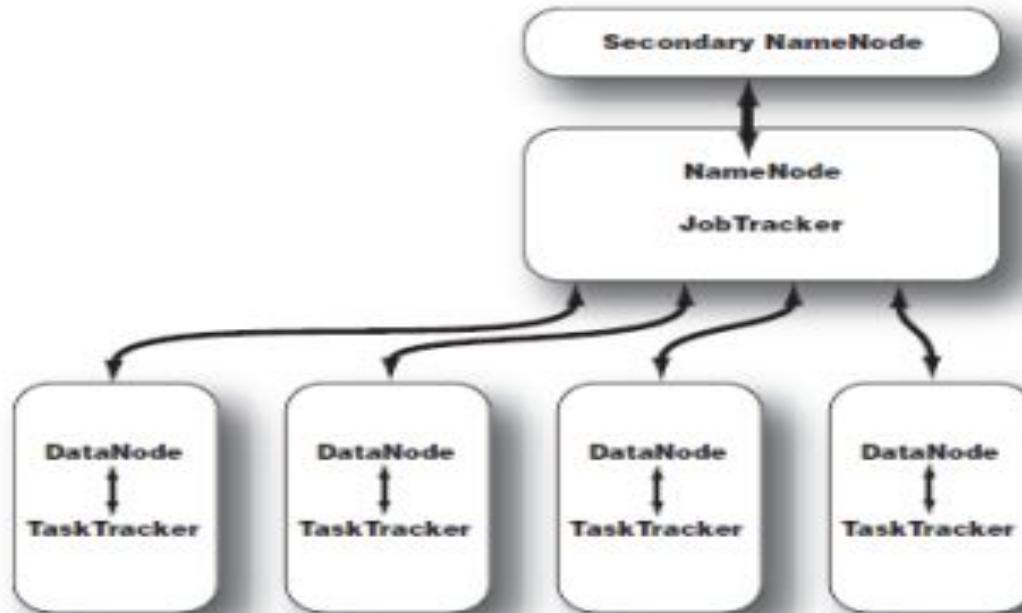
MapReduce



MapReduce



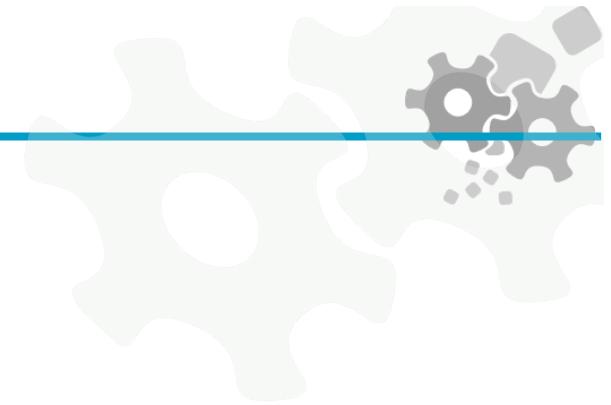
Hadoop Architecture



MapReduce In Action

- Calculate average word length per first letter of word
 - AverageWordLength.java: launches job
 - LetterMapper.java: mapper per first letter
 - AverageReducer.java: calculates average length

AverageWordLength



```
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.FloatWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;

import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class AverageWordLength extends Configured implements Tool {

    public int run(String[] args) throws Exception {
        if (args.length != 2) {
            System.out.printf("Usage: %s [generic options] <indir> <outdir>\n",
                getClass().getSimpleName());
            ToolRunner.printGenericCommandUsage(System.out);
            System.exit(-1);
        }

        JobConf conf = new JobConf(getConf(), AverageWordLength.class);
        conf.setJobName("AverageWordLength");

        FileInputFormat.setInputPaths(conf, new Path(args[0]));
        FileOutputFormat.setOutputPath(conf, new Path(args[1]));

        conf.setMapperClass(LetterMapper.class);
        conf.setReducerClass(AverageReducer.class);

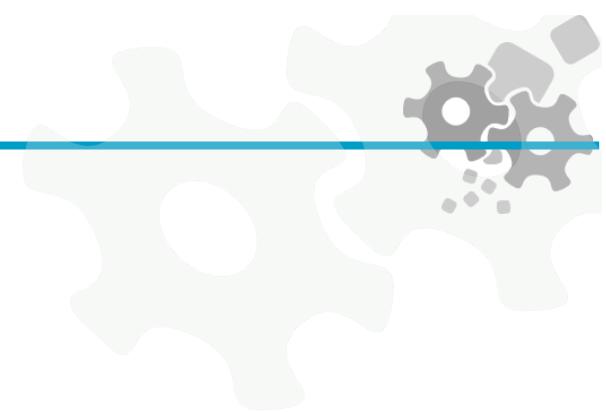
        conf.setMapOutputKeyClass(Text.class);
        conf.setMapOutputValueClass(IntWritable.class);

        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(FloatWritable.class);

        JobClient.runJob(conf);
        return 0;
    }

    public static void main(String[] args) throws Exception {
        int exitCode = ToolRunner.run(new AverageWordLength(), args);
        System.exit(exitCode);
    }
}
```

LetterMapper



```
import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;

public class LetterMapper extends MapReduceBase
    implements Mapper<Object, Text, Text, IntWritable> {

    private Text letter = new Text();
    private IntWritable wordLength = new IntWritable(1);

    public void map(Object key,
                    Text value,
                    OutputCollector<Text, IntWritable> output,
                    Reporter reporter)
    throws IOException {

        // Break line into words for processing
        StringTokenizer wordList =
            new StringTokenizer(value.toString());

        while (wordList.hasMoreTokens()) {
            String word = wordList.nextToken();
            letter.set(word.substring(0, 1));
            wordLength.set(word.length());
            output.collect(letter, wordLength);
        }
    }
}
```

AverageReducer

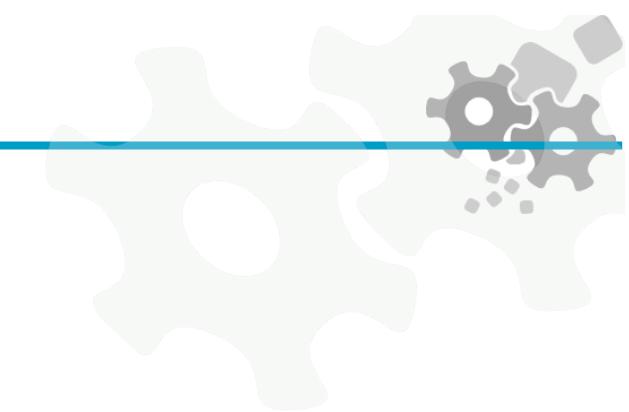
```
import java.io.IOException;
import java.util.Iterator;

import org.apache.hadoop.io.FloatWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;

public class AverageReducer extends MapReduceBase
    implements Reducer<Text, IntWritable, Text, FloatWritable> {

    private FloatWritable avg = new FloatWritable();

    public void reduce(Text key,
                       Iterator<IntWritable> values,
                       OutputCollector<Text, FloatWritable> output,
                       Reporter reporter)
    throws IOException {
        int wordCount = 0;
        int letterCount = 0;
        while (values.hasNext()) {
            wordCount++;
            letterCount += values.next().get();
        }
        avg.set((float) letterCount / wordCount);
        output.collect(key, avg);
    }
}
```



MapReduce In Action

```
training@ubuntu:~/git/exercises/averagewordlength/java$ ls
AverageReducer.class AverageReducer.java AverageWordLength.class AverageWordLength.java LetterMapper.class LetterMapper.java wl.jar
training@ubuntu:~/git/exercises/averagewordlength/java$ hadoop jar wl.jar AverageWordLength shakespeare report
11/12/18 13:26:54 INFO mapred.FileInputFormat: Total input paths to process : 4
11/12/18 13:26:54 INFO mapred.JobClient: Running job: job_201112180521_0013
11/12/18 13:26:55 INFO mapred.JobClient: map 0% reduce 0%
11/12/18 13:27:05 INFO mapred.JobClient: map 50% reduce 0%
11/12/18 13:27:08 INFO mapred.JobClient: map 100% reduce 0%
11/12/18 13:27:17 INFO mapred.JobClient: map 100% reduce 100%
11/12/18 13:27:19 INFO mapred.JobClient: Job complete: job_201112180521_0013
11/12/18 13:27:19 INFO mapred.JobClient: Counters: 18
11/12/18 13:27:19 INFO mapred.JobClient: Job Counters
11/12/18 13:27:19 INFO mapred.JobClient: Launched reduce tasks=1
11/12/18 13:27:19 INFO mapred.JobClient: Launched map tasks=4
11/12/18 13:27:19 INFO mapred.JobClient: Data-local map tasks=4
11/12/18 13:27:19 INFO mapred.JobClient: FileSystemCounters
11/12/18 13:27:19 INFO mapred.JobClient: FILE_BYTES_READ=12543198
11/12/18 13:27:19 INFO mapred.JobClient: HDFS_BYTES_READ=5284231
11/12/18 13:27:19 INFO mapred.JobClient: FILE_BYTES_WRITTEN=20057238
11/12/18 13:27:19 INFO mapred.JobClient: HDFS_BYTES_WRITTEN=728
11/12/18 13:27:19 INFO mapred.JobClient: Map-Reduce Framework
11/12/18 13:27:19 INFO mapred.JobClient: Reduce input groups=69
11/12/18 13:27:19 INFO mapred.JobClient: Combine output records=0
11/12/18 13:27:19 INFO mapred.JobClient: Map input records=173126
11/12/18 13:27:19 INFO mapred.JobClient: Reduce shuffle bytes=7513912
11/12/18 13:27:19 INFO mapred.JobClient: Reduce output records=69
11/12/18 13:27:19 INFO mapred.JobClient: Spilled Records=2507132
11/12/18 13:27:19 INFO mapred.JobClient: Map output bytes=5635416
11/12/18 13:27:19 INFO mapred.JobClient: Map input bytes=5284231
11/12/18 13:27:19 INFO mapred.JobClient: Combine input records=0
11/12/18 13:27:19 INFO mapred.JobClient: Map output records=939236
11/12/18 13:27:19 INFO mapred.JobClient: Reduce input records=939236
training@ubuntu:~/git/exercises/averagewordlength/java$ hadoop fs -cat report/part-00000 | head -5
&      3.0422535
'      4.9115677
(      8.342302
-      8.333333
.      1.0
training@ubuntu:~/git/exercises/averagewordlength/java$
```

JobTracker page

localhost Hadoop Map/Reduce Administration

Quick Links

State: RUNNING
Started: Sun Dec 18 05:21:40 PST 2011
Version: 0.20.2+320, r9b72d268a0b590b4fd7d13aca17c1c453f8bc957
Compiled: Mon Jun 28 23:17:49 UTC 2010 by root
Identifier: 201112180521

Cluster Summary (Heap Size is 15.19 MB/966.69 MB)

Maps	Reduces	Total Submissions	Nodes	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node	Blacklisted Nodes
0	0	12	1	2	2	4.00	0

Scheduling Information

Queue Name	Scheduling Information
default	N/A

Filter (Jobid, Priority, User, Name)

Example: 'user:smith 3200' will filter by 'smith' only in the user field and '3200' in all fields

Running Jobs

none

Completed Jobs

Jobid	Priority	User	Name	Map % Complete	Map Total	Maps Completed	Reduce % Complete	Reduce Total	Reduces Completed	Job Scheduling Information
job_201112180521_0001	NORMAL	training	WordCount	100.00%	4	4	100.00%	1	1	NA
job_201112180521_0002	NORMAL	training	WordCount	100.00%	2	2	100.00%	1	1	NA
job_201112180521_0003	NORMAL	training	WordCount	100.00%	2	2	100.00%	1	1	NA
job_201112180521_0004	NORMAL	training	WordCount	100.00%	2	2	100.00%	1	1	NA
job_201112180521_0005	NORMAL	training	AverageWordLength	100.00%	4	4	100.00%	1	1	NA
job_201112180521_0006	NORMAL	training	movie.jar	100.00%	5	5	100.00%	0	0	NA

JobTracker page

Hadoop job_201112180521_0001 on localhost

User: training
Job Name: WordCount
Job File: hdfs://localhost/var/lib/hadoop-0.20/cache/mapred/system/job_201112180521_0001/job.xml
Job Setup: Successful
Status: Succeeded
Started at: Sun Dec 18 05:33:22 PST 2011
Finished at: Sun Dec 18 05:33:48 PST 2011
Finished In: 26sec
Job Cleanup: Successful

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	100.00%	4	0	0	4	0	0 / 0
reduce	100.00%	1	0	0	1	0	0 / 0

Job Counters	Counter	Map	Reduce	Total
	Launched reduce tasks	0	0	1
	Launched map tasks	0	0	4
FileSystemCounters	Data-local map tasks	0	0	4
	FILE_BYTES_READ	7,200,180	10,768,592	17,968,772
	HDFS_BYTES_READ	5,284,231	0	5,284,231
	FILE_BYTES_WRITTEN	17,968,918	10,768,592	28,737,510
Map-Reduce Framework	HDFS_BYTES_WRITTEN	0	753,527	753,527
	Reduce input groups	0	70,139	70,139
	Combine output records	0	0	0
	Map input records	173,126	0	173,126
	Reduce shuffle bytes	0	10,768,610	10,768,610
	Reduce output records	0	70,139	70,139
	Spilled Records	1,567,896	939,236	2,507,132
	Map output bytes	8,890,114	0	8,890,114
	Map input bytes	5,284,231	0	5,284,231
	Map output records	939,236	0	939,236
	Combine input records	0	0	0
	Reduce input records	0	939,236	939,236

Hadoop map task list for job_201112180521_0001 on localhost

All Tasks

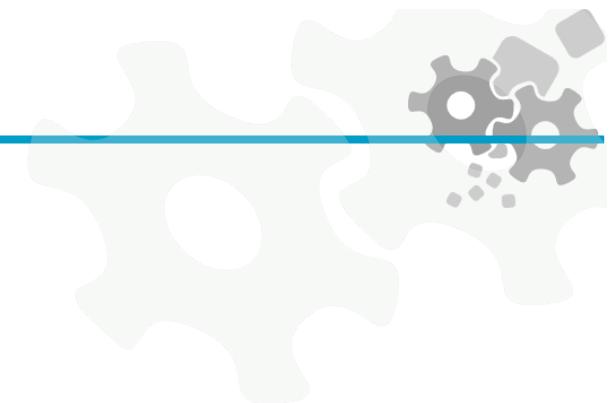
Task	Complete	Status	Start Time	Finish Time	Errors	Counters
task_201112180521_0001_m_000000	100.00%	hdfs://localhost/user/training/shakespeare/comedies:0+1784616	18-Dec-2011 05:33:27	18-Dec-2011 05:33:33 (6sec)		10
task_201112180521_0001_m_000001	100.00%	hdfs://localhost/user/training/shakespeare/tragedies:0+1752440	18-Dec-2011 05:33:27	18-Dec-2011 05:33:33 (6sec)		10
task_201112180521_0001_m_000002	100.00%	hdfs://localhost/user/training/shakespeare/histories:0+1479035	18-Dec-2011 05:33:33	18-Dec-2011 05:33:36 (3sec)		9
task_201112180521_0001_m_000003	100.00%	hdfs://localhost/user/training/shakespeare/poems:0+268140	18-Dec-2011 05:33:33	18-Dec-2011 05:33:36 (3sec)		9

[Go back to JobTracker](#)

Cloudera's Distribution for Hadoop, 2011.

MapReduce

- Abstract Processing Model
 - Distributed sort merge engine
- Implementation
 - Programming
 - Java
 - Python
 - High-level tool using MapReduce Jobs
 - Hive
 - Pig
 - ...



Hive

- Framework for data warehousing on top of Hadoop
- Developed at Facebook for managing and learning from the huge volumes of data Facebook was generating
- Makes it possible for analysts with strong SQL skills to run queries
- Used by many organizations
- SQL is lingua franca in business intelligence tools
- SQL is limited so Hive is not fit for building complex machine learning algorithms
- Generates MR jobs when executing queries

Hive

```
CREATE EXTERNAL TABLE movie (id INT, name STRING, year INT) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' LOCATION '/user/root/movie'

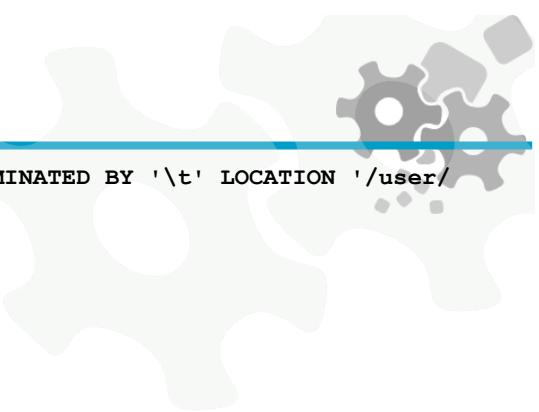
CREATE EXTERNAL TABLE movierating (userid INT, movieid INT, rating INT)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
LOCATION '/user/cloudera/movierating'

SELECT * FROM movie

--Select oldest movie
SELECT *
FROM movie
WHERE year != 0
SORT BY year
LIMIT 1

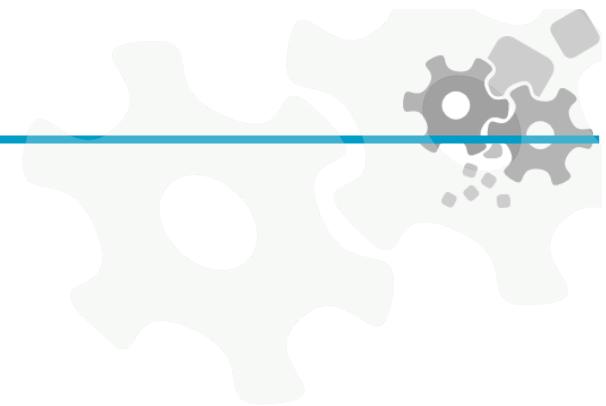
--Select movies without rating
SELECT name, year
FROM movie LEFT OUTER JOIN movierating
ON movie.id = movierating.movieid
WHERE movieid IS NULL

--Update movies with numratings, avgrating
DROP TABLE newmovie
```



Hive

```
root@master ~ # hive
Hive history file=/tmp/root/hive_job_log_root_201108031010_1952745660.txt
hive> select * from movie limit 10;
OK
1      Toy Story      1995
2      Jumanji 1995
3      Grumpier Old Men    1995
4      Waiting to Exhale   1995
5      Father of the Bride Part II 1995
6      Heat      1995
7      Sabrina 1995
8      Tom and Huck     1995
9      Sudden Death    1995
10     GoldenEye      1995
Time taken: 0.067 seconds
hive>
```



Pig

- Abstraction layer for processing large data sets
- 2 Components
 - Pig Latin: the language used to express data flows
 - Grunt: the execution environment
- Pig Program
 - Composed of series of operations, or transformations
 - The operations describe a dataflow that is translated into one or more MapReduce jobs

Pig

```
-- max_temp.pig: Finds the maximum temperature by year
records =
    LOAD 'input/ncdc/micro-tab/sample.txt'
    AS (year:chararray, temperature:int, quality:int);

filtered_records =
    FILTER records
    BY temperature != 9999
    AND ( quality == 0 OR
          quality == 1 OR
          quality == 4 OR
          quality == 5 OR
          quality == 9);

grouped_records =
    GROUP filtered_records
    BY year;

max_temp =
    FOREACH grouped_records
    GENERATE group, MAX(filtered_records.temperature);
DUMP max_temp;
```