CS 444: Big Data Systems

Chapter 3. Overview of Big Data Ecosystem

Chase Wu

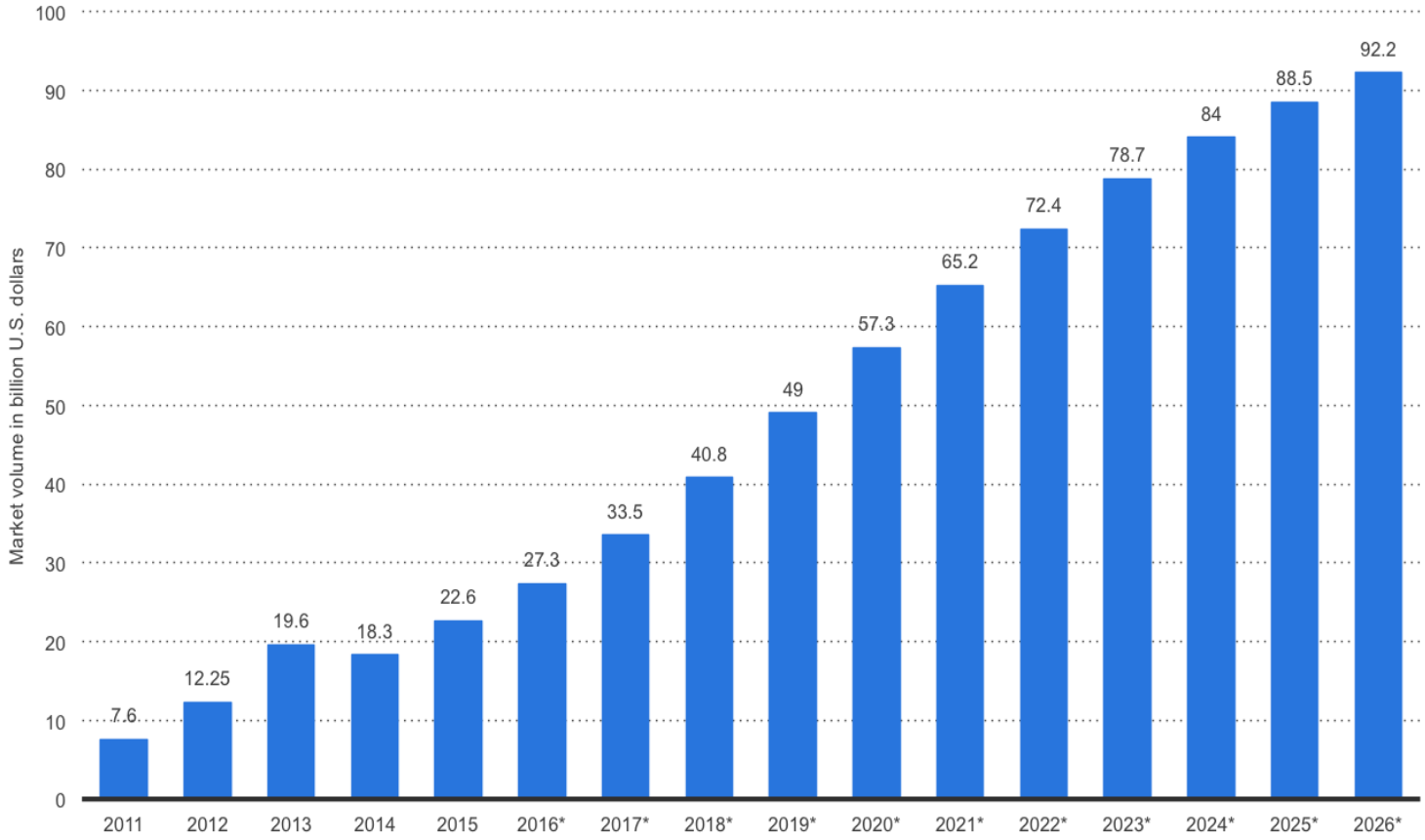
New Jersey Institute of Technology

Some of the slides were provided through the courtesy of Dr. Ching-Yung Lin at Columbia University

Big Data: An illustration created by ChatGPT

Big data market size revenue forecast worldwide from 2011 to 2026 (in billion U.S. dollars)

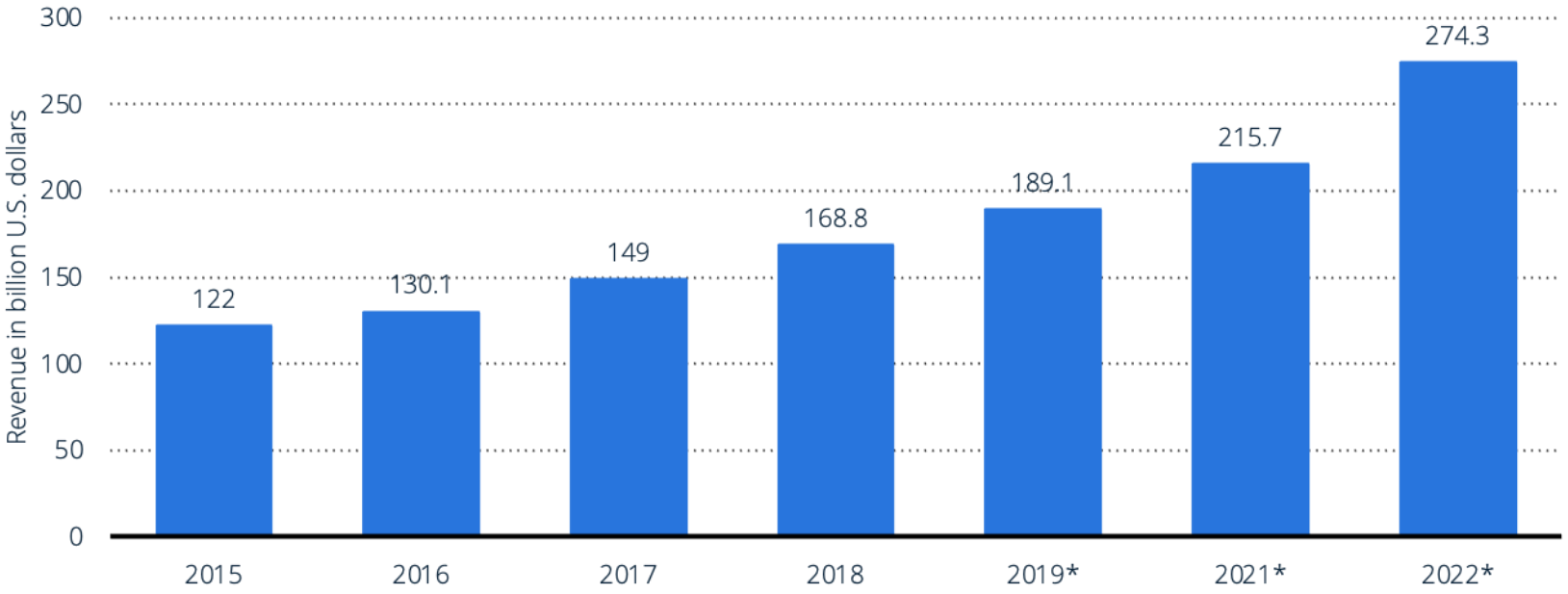
The Big Data market is exploding, not only in terms of marketing hype, but also in real revenue



Note: Worldwide; 2014 to 2016

Source: Wikibon; ID 254266

Real Revenue from big data and business analytics worldwide from 2015 to 2022 (in billion U.S. dollars)

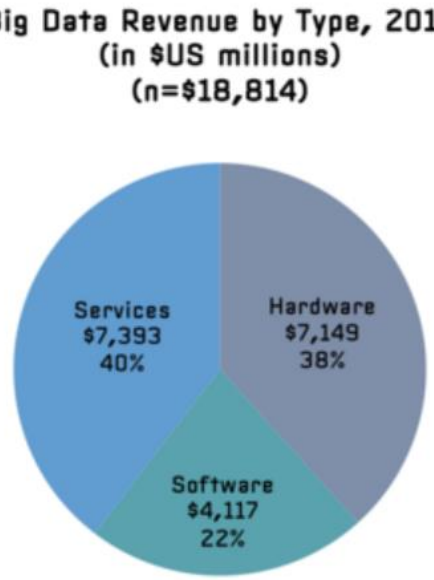


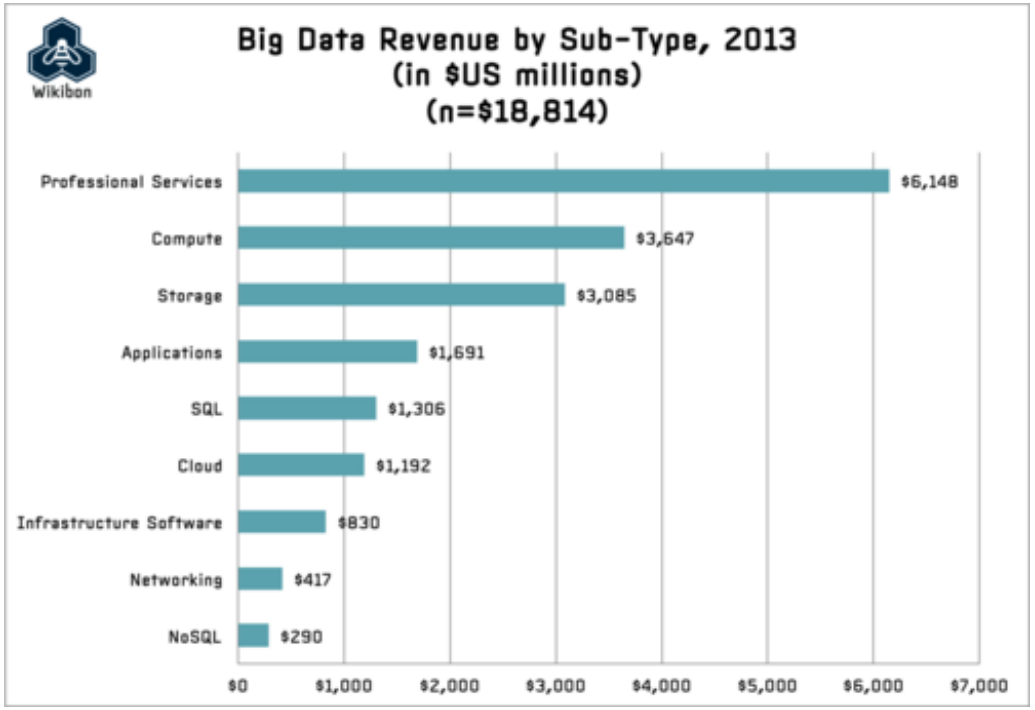
2022: ~4 times more than predicted!

Note(s): Worldwide; 2015 to 2021
Source(s): IDC; ID 551501

statista

Big Data Revenue By Type





http://wikibon.org/wiki/v/Big_Data_Vendor_Revenue_and_Market_Forecast_2013-2017

5 Key Big Data Use Case Categories – IBM's Perspective

Big Data Exploration

Find, visualize, and understand all big data to improve decision making

Enhanced 360º View of the Customer

Extend existing customer views (MDM, CRM, etc.) by incorporating additional internal and external information sources

Security/Intelligence Extension

Lower risk, detect fraud and monitor cyber security in real-time

Operations Analysis

Analyze a variety of machine data for improved business results

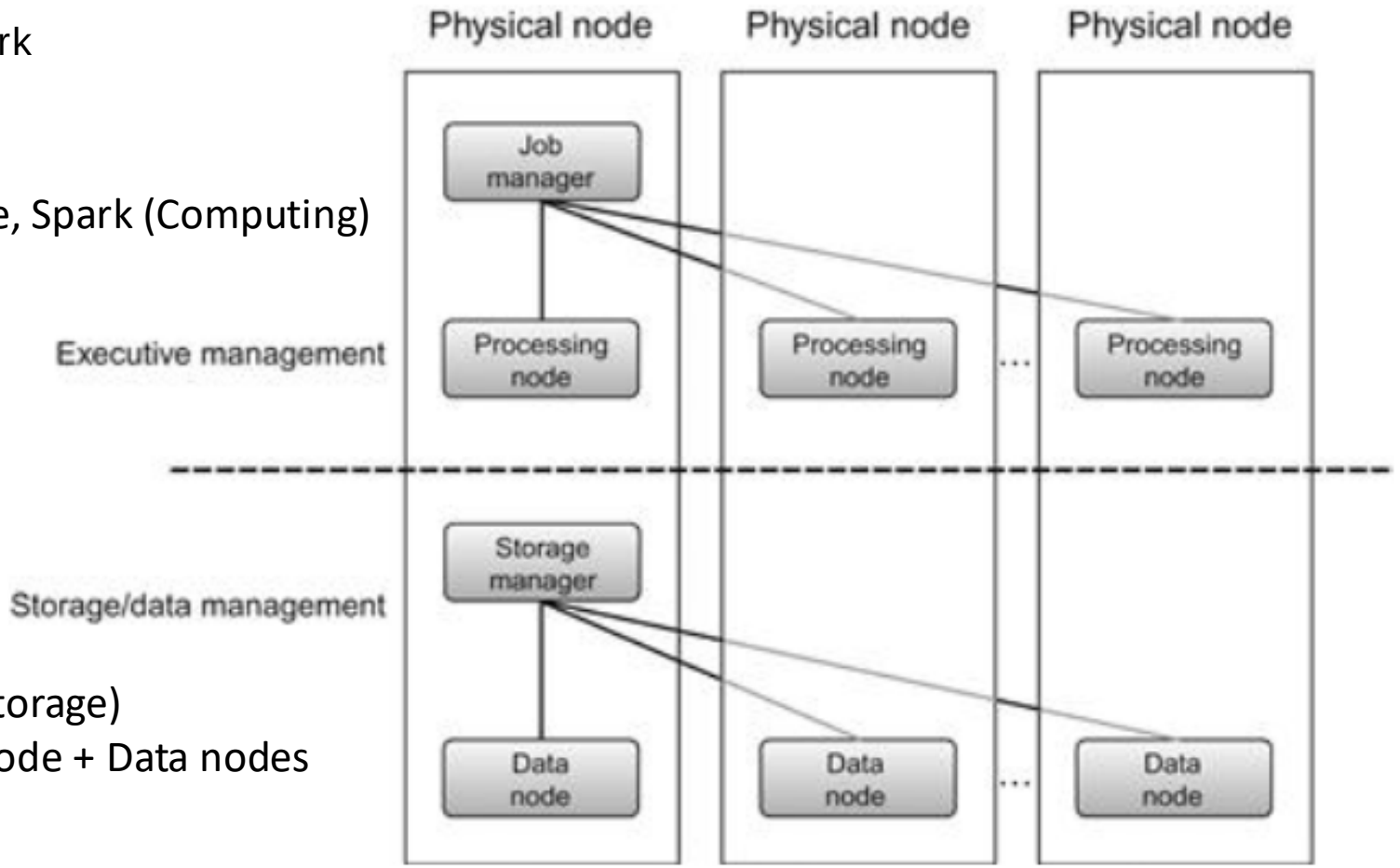
Data Warehouse Augmentation

Integrate big data and data warehouse capabilities to increase operational efficiency

Key Computing Resources for Big Data

- Processing capability: CPU, multi/many-core processor, or node
- Memory
- Storage
- Network

MapReduce, Spark (Computing)



HDFS (Storage)
Name node + Data nodes

"Big Data Analytics", David Loshin, 2013

Techniques towards Big Data

- Massive Parallelism
- Huge Data Volumes Storage
- Data Distribution
- High-Speed Networks
- High-Performance Computing
- Task and Thread Management
- Data Mining and Analytics
- Data Retrieval
- Machine Learning
- Data Visualization

➔ Techniques exist for years to decades. Why did Big Data become hot now?

Why Big Data now?

- More data are being collected and stored
- Open-source code
- Commodity hardware
- Successful applications of data-driven AI and ML techniques, such as the recent GPTs.

The driving force behind big data is quantification of information.

- **In the past, you would just go for a morning jog.**
- **Today, you know it was 7.6km long, you took 11,341 steps and burned 612 calories because of it.**

Definition and Characteristics of Big Data

"*Big data is high-**volume**, high-**velocity** and/or high-**variety** information assets that demand **cost-effective**, **innovative** forms of information processing* that enable ***enhanced insight, decision making, and process automation***."

– Gartner, Inc.

which was derived from:

"*While enterprises struggle to consolidate systems and collapse redundant databases to enable greater operational, analytical, and collaborative consistencies, changing economic conditions have made this job more difficult. E-commerce, in particular, has exploded data management challenges along three dimensions: **volumes, velocity and variety**. In 2001/02, IT organizations much compile a variety of approaches to have at their disposal for dealing each*."

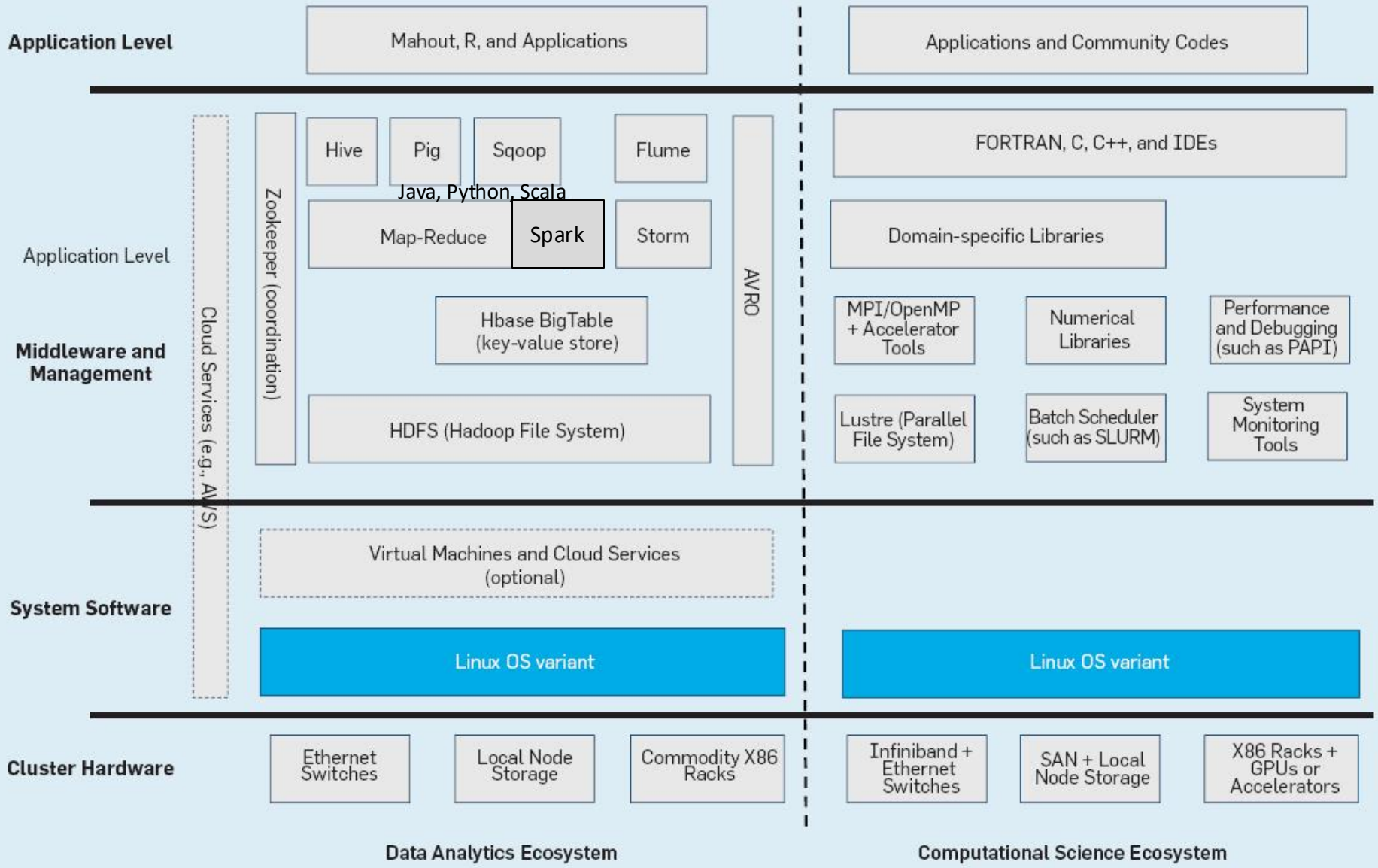
– Doug Laney

Comparison of Approaches in Adopting High-Performance Capabilities

Aspect	Typical Scenario	Big Data
Application development	Applications that take advantage of massive parallelism developed by specialized developers skilled in high-performance computing, performance optimization, and code tuning	A simplified application execution model encompassing a distributed file system, application programming model, distributed database, and program scheduling is packaged within Hadoop, an open source framework for reliable, scalable, distributed, and parallel computing
Platform	Uses high-cost massively parallel processing (MPP) computers, utilizing high-bandwidth networks, and massive I/O devices	Innovative methods of creating scalable and yet elastic virtualized platforms take advantage of clusters of commodity hardware components (either cycle harvesting from local resources or through cloud-based utility computing services) coupled with open source tools and technology
Data management	Limited to file-based or relational database management systems (RDBMS) using standard row-oriented data layouts	Alternate models for data management (often referred to as NoSQL or "Not Only SQL") provide a variety of methods for managing information to best suit specific business process needs, such as in-memory data management (for rapid access), columnar layouts to speed query response, and graph databases (for social network analytics)
Resources	Requires large capital investment in purchasing high-end hardware to be installed and managed in-house	The ability to deploy systems like Hadoop on virtualized platforms allows small and medium businesses to utilize cloud-based environments that, from both a cost accounting and a practical perspective, are much friendlier to the bottom line

"Big Data Analytics", David Loshin, 2013

Comparison of Data Analytics and Computing Ecosystems



Apache Hadoop

The Apache™ Hadoop® project develops open-source software for reliable, scalable, distributed computing.

The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than relying on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.

The project includes these modules:

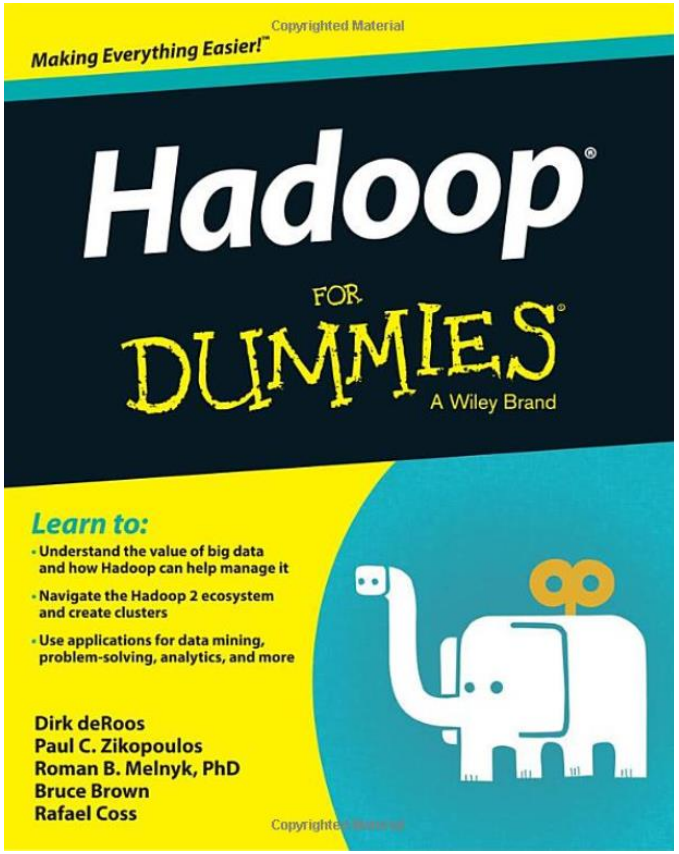
- **Hadoop Common**: The common utilities that support the other Hadoop modules.
- **Hadoop Distributed File System (HDFS™)**: A distributed file system that provides high-throughput access to application data.
- **Hadoop MapReduce**: A YARN-based system for parallel processing of large data sets.
- **Hadoop YARN (starting from the 2nd generation)**: A framework for job scheduling and cluster resource management.

http://hadoop.apache.org

Hadoop-related Apache Projects: Hadoop Ecosystem

- **Ambari™**: A web-based tool for provisioning, managing, and monitoring Hadoop clusters. It also provides a dashboard for viewing cluster health and ability to view MapReduce, Pig and Hive applications visually.
- **Avro™**: A data serialization system.
- **Cassandra™**: A scalable multi-master database with no single points of failure.
- **Chukwa™**: A data collection system for managing large distributed systems.
- **HBase™**: A scalable, distributed database that supports structured data storage for large tables.
- **Hive™**: A data warehouse infrastructure that provides data summarization and ad hoc querying.
- **Mahout™**: A scalable machine learning and data mining library.
- **Pig™**: A high-level data-flow language and execution framework for parallel computation.
- **Spark™**: A fast and general compute engine for Hadoop data. Spark provides a simple and expressive programming model that supports a wide range of applications, including ETL, machine learning, stream processing, and graph computation.
- **Tez™**: A generalized data-flow programming framework, built on Hadoop YARN, which provides a powerful and flexible engine to execute an arbitrary DAG of tasks to process data for both batch and interactive use-cases.
- **Zookeeper™**: A high-performance coordination service for distributed applications.

Reading Reference



Big Data (Hadoop) Ecosystem

Big Data Applications/Domains
(Healthcare, insurance, finance, social networks, transportation, sciences, etc.

Big Data Analytics
(Methods: AI, machine learning, visualization, etc.
Modules: Pig, Hive, Mahout, etc.)

Big Data Computing
(MapReduce, Spark, Storm, Oozie, etc.)

Resource Management and Scheduling
(YARN, Kubernetes, Mesos)

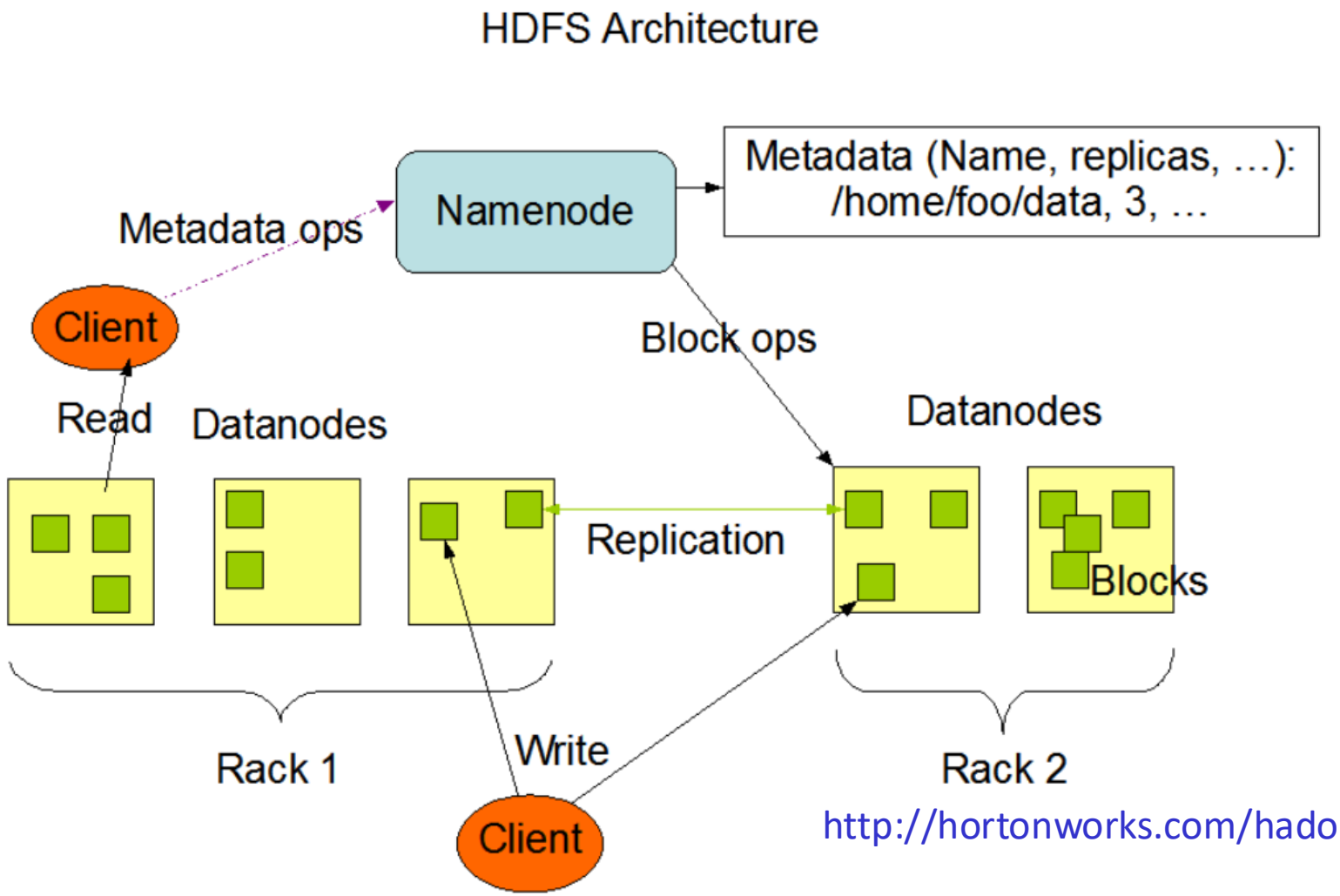
Big Data Management
(NoSQL: RDBMS, Key-Value, Document, Graph, etc.
Systems: SQL, MongoDB, HBase, Cassandra, etc.)

Big Data Storage
(HDFS)

Big Data Networking
(HPN, SDN, etc.)

Hadoop Distributed File System (HDFS)

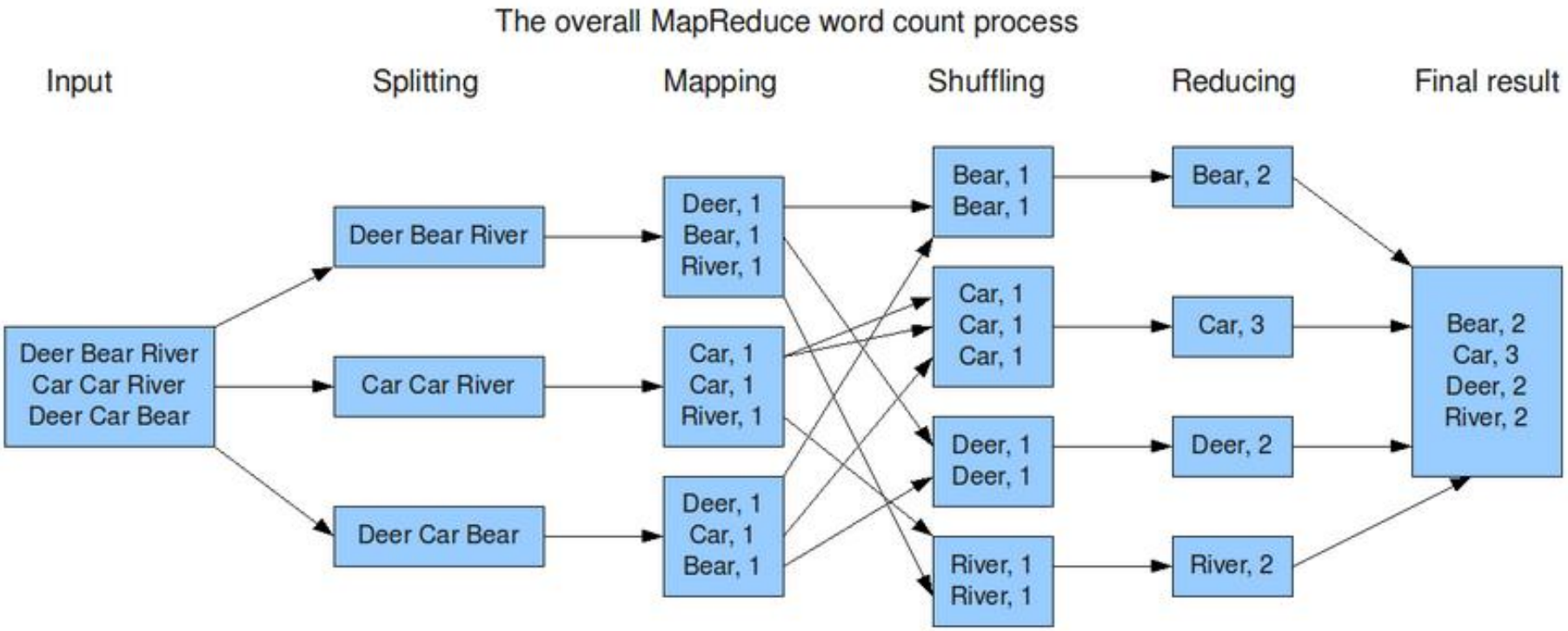
- HDFS is a java-based file system that provides the scalable, fault-tolerant, cost-efficient storage for big data
  - The file content is split into large blocks (typically 128 megabytes), each of which is independently replicated at multiple DataNodes
  - The NameNode maintains the namespace tree (in RAM) and the mapping of blocks to DataNodes



http://hortonworks.com/hadoop/hdfs/

WordCouting: “Hello World” in MapReduce

Basic data structure: (key, value)



http://www.alex-hanna.com

Set Up the Hadoop Environment

- Local (standalone) mode
- Pseudo-distributed mode
- Fully-distributed mode

Setting Up the Hadoop Environment – Pseudo-distributed mode

Setup passphraseless ssh

Now check that you can ssh to the localhost without a passphrase:

```
$ ssh localhost
```

If you cannot ssh to localhost without a passphrase, execute the following commands:

```
$ ssh-keygen -t dsa -P '' -f ~/.ssh/id_dsa
$ cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys
```

Configuration

Use the following:

conf/core-site.xml:

```
<configuration>
     <property>
          <name>fs.defaultFS</name>
          <value>hdfs://localhost:9000</value>
     </property>
</configuration>
```

conf/hdfs-site.xml:

```
<configuration>
     <property>
          <name>dfs.replication</name>
          <value>1</value>
     </property>
</configuration>
```

conf/mapred-site.xml:

```
<configuration>
     <property>
          <name>mapred.job.tracker</name>
          <value>localhost:9001</value>
     </property>
</configuration>
```

On the SSH server
authorized_keys:
used by the SSH server to store the public keys of clients for client authentication

On the SSH client
known_hosts:
used by the SSH client to store the public keys of servers for server authentication

http://hadoop.apache.org/docs/stable2/hadoop-project-dist/hadoop-common/SingleNodeSetup.html

Set Up the Hadoop Environment – Pseudo-distributed mode

Execution

Format a new distributed-filesystem:

```
$ bin/hadoop namenode -format
```

Start the hadoop daemons:

```
$ bin/start-all.sh
```

The hadoop daemon log output is written to the `$HADOOP_LOG_DIR` directory (defaults to `$HADOOP_PREFIX/logs`).

Browse the web interface for the NameNode and the JobTracker; by default they are available at:

- NameNode - `http://localhost:50070/`
- JobTracker - `http://localhost:50030/`

Copy the input files into the distributed filesystem:

```
$ bin/hadoop fs -put conf input
```

Run some of the examples provided:

```
$ bin/hadoop jar hadoop-*-examples.jar grep input output 'dfs[a-z.]+'
```

Examine the output files:

Copy the output files from the distributed filesystem to the local filesytem and examine them:

```
$ bin/hadoop fs -get output output
$ cat output/*
```

or

View the output files on the distributed filesystem:

```
$ bin/hadoop fs -cat output/*
```

Word Count Problem: Hands-on MapReduce Programing Guide -Configuration

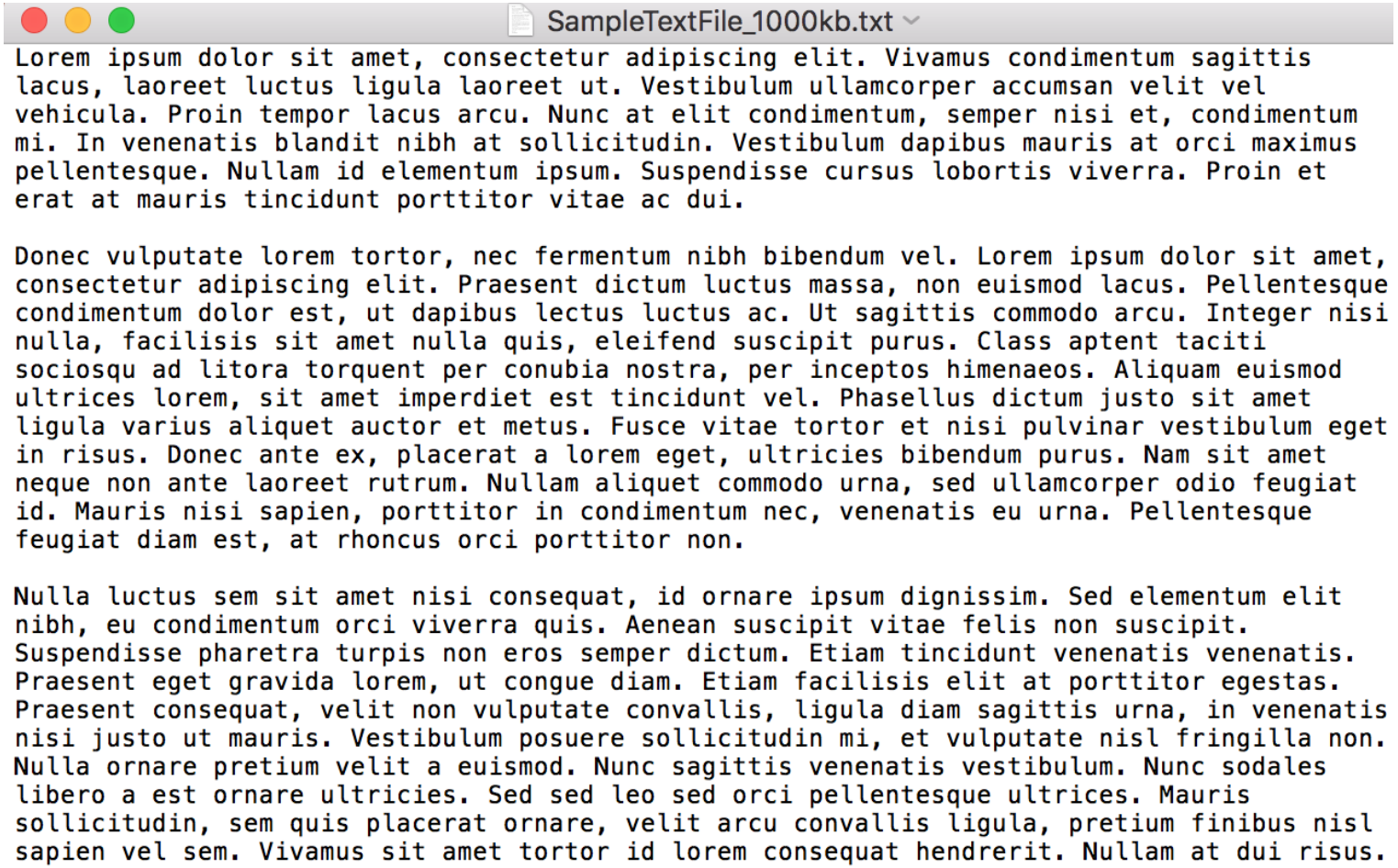
Version: Hadoop 1.2.1

Mode: Pseudo-Distributed Mode

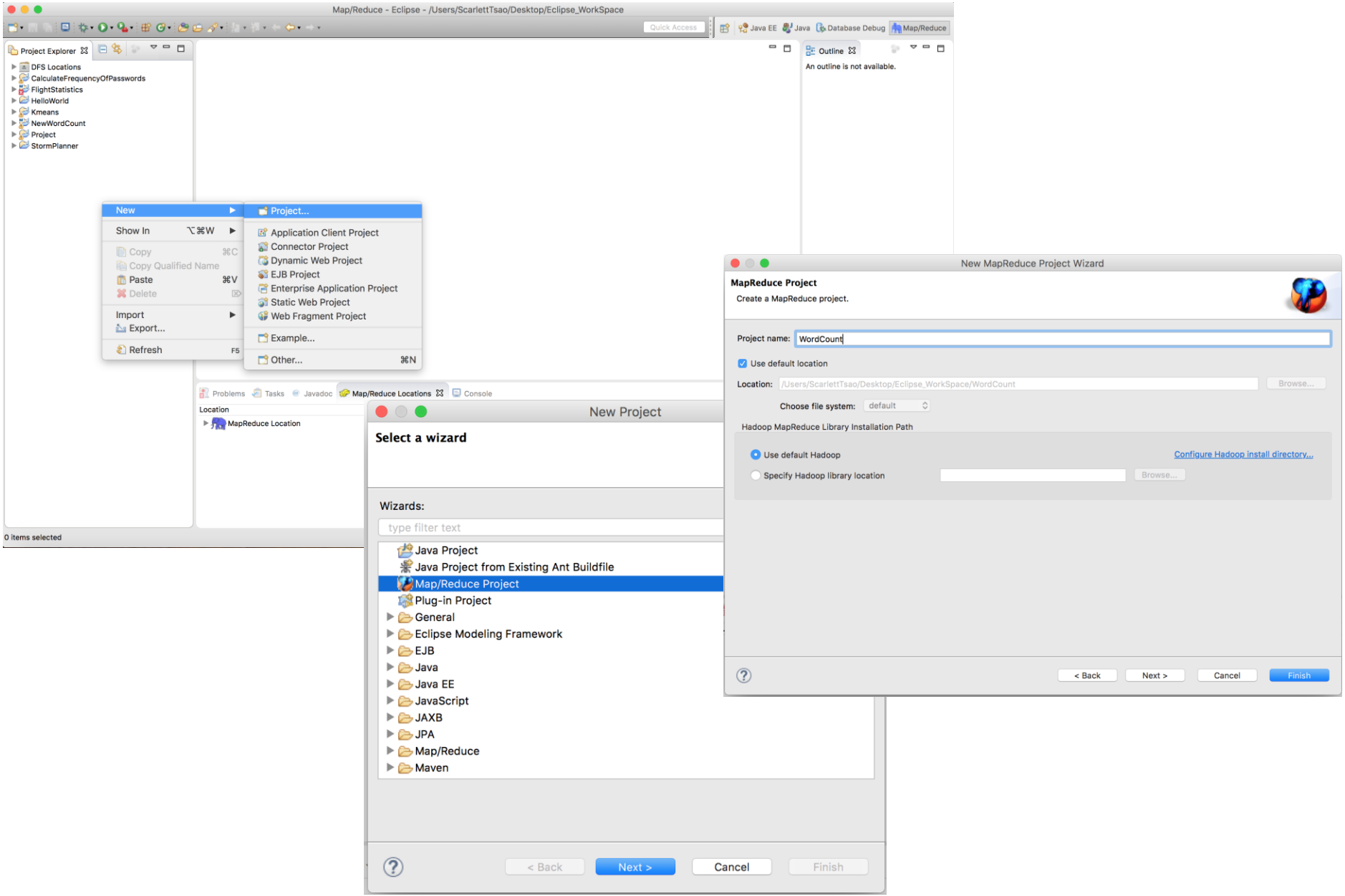
IDE: Eclipse

Word Count Problem
-Input

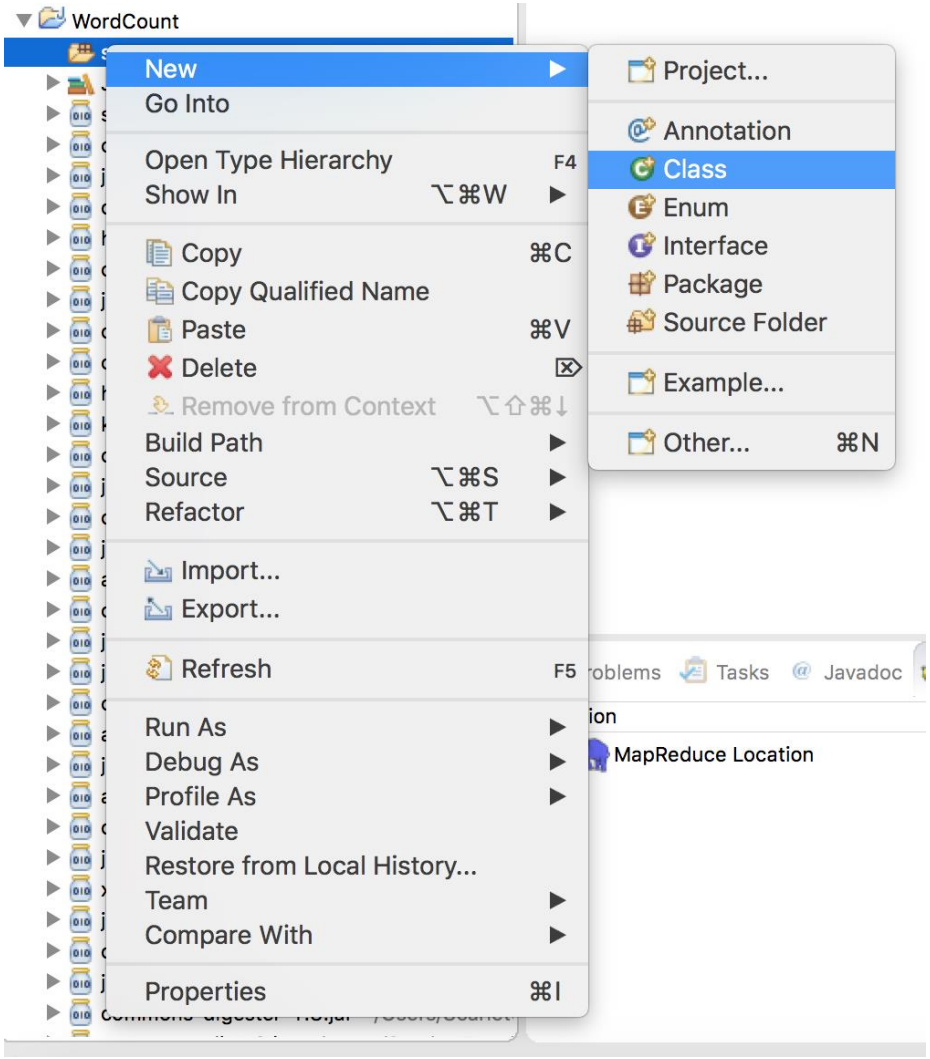
Locally stored file: SampleTextFile_1000kb.txt

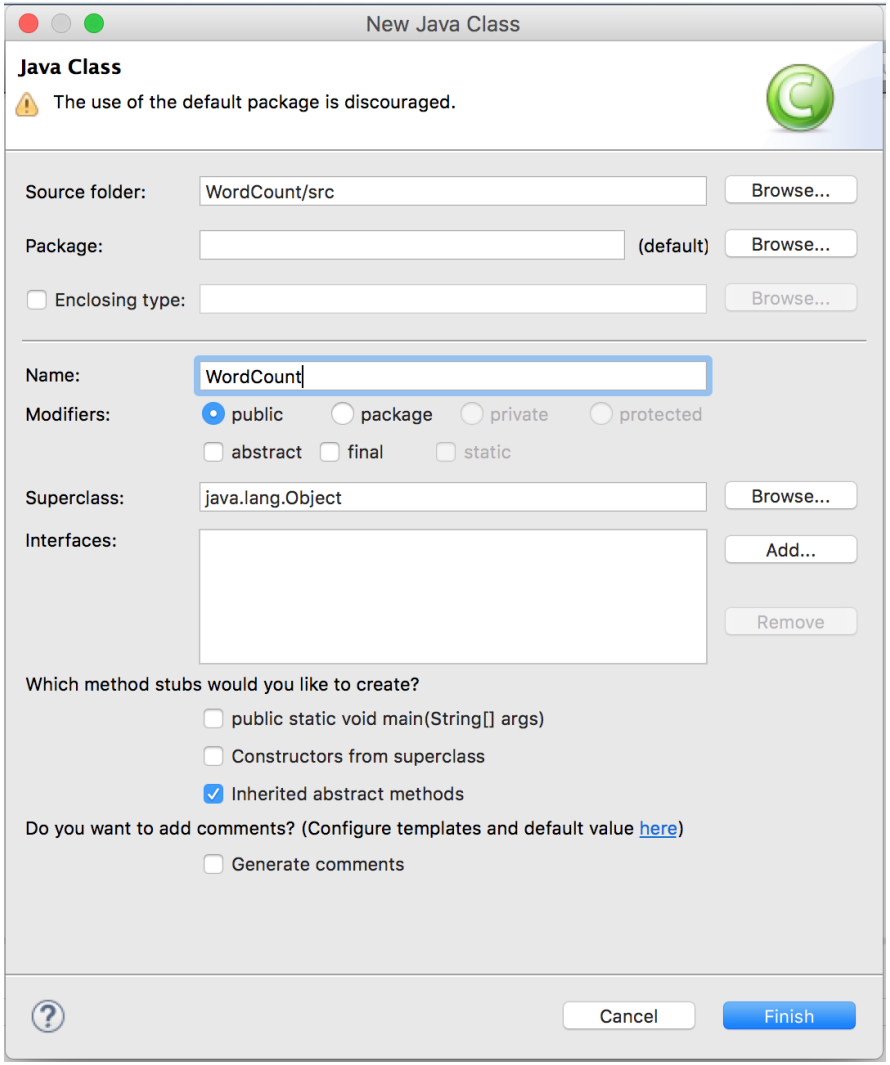


Word Count Problem
-Create a MapReduce Project



Word Count Problem
-Create a class





Word Count Problem
-MapReduce Program

Mapper Function:

*WordCount.java

```
1  import java.io.IOException;
2  import java.util.StringTokenizer;
3
4  import org.apache.hadoop.conf.Configuration;
5  import org.apache.hadoop.fs.Path;
6  import org.apache.hadoop.io.IntWritable;
7  import org.apache.hadoop.io.Text;
8  import org.apache.hadoop.mapreduce.Job;
9  import org.apache.hadoop.mapreduce.Mapper;
10 import org.apache.hadoop.mapreduce.Reducer;
11 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
12 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
13 import org.apache.hadoop.util.GenericOptionsParser;
14
15 public class WordCount {
16
17   public static class TokenizerMapper extends Mapper<Object, Text, Text, IntWritable>{
18     //mapper function
19     private final static IntWritable one = new IntWritable(1);
20     private Text word = new Text();
21
22     public void map(Object key, Text value, Context context) throws IOException, InterruptedException {
23       StringTokenizer itr = new StringTokenizer(value.toString());//convert text into string token iterator
24       while (itr.hasMoreTokens()) {//for each word generate the pair <word, one> as output context
25         word.set(itr.nextToken());
26         context.write(word, one);
27       }
28     }
29   }
30
```

Word Count Problem
-MapReduce Program

Reducer Function:

*WordCount.java

```
30
31  public static class IntSumReducer extends Reducer<Text,IntWritable,Text,IntWritable> {
32    //reducer function
33    private IntWritable result = new IntWritable();
34
35    public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {
36      int sum = 0;
37      for (IntWritable val : values) {//for each word, collect the "ones" and output the pair <keyword, result>
38        sum += val.get();
39      }
40      result.set(sum);
41      context.write(key, result);
42    }
43  }
44
```

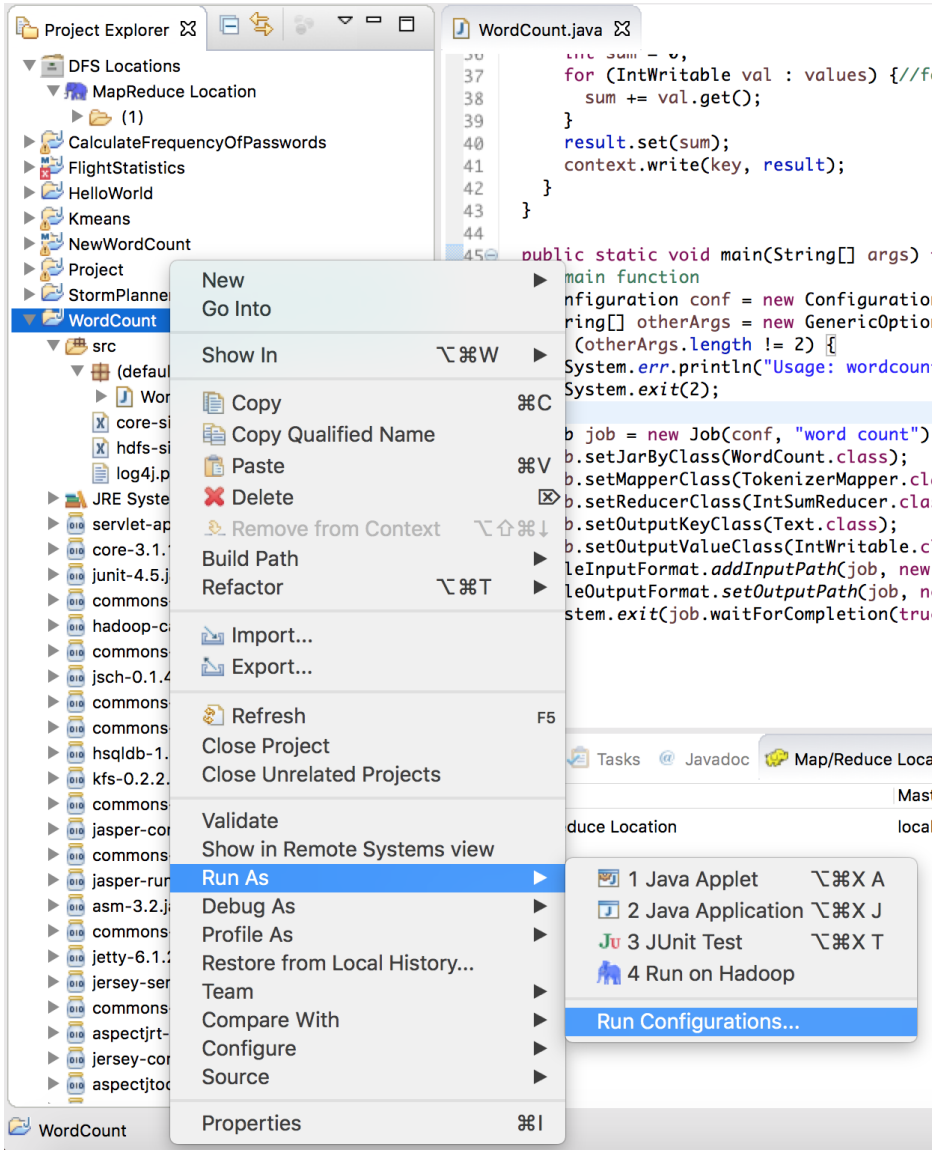
Word Count Problem
-MapReduce Program

Main Function:

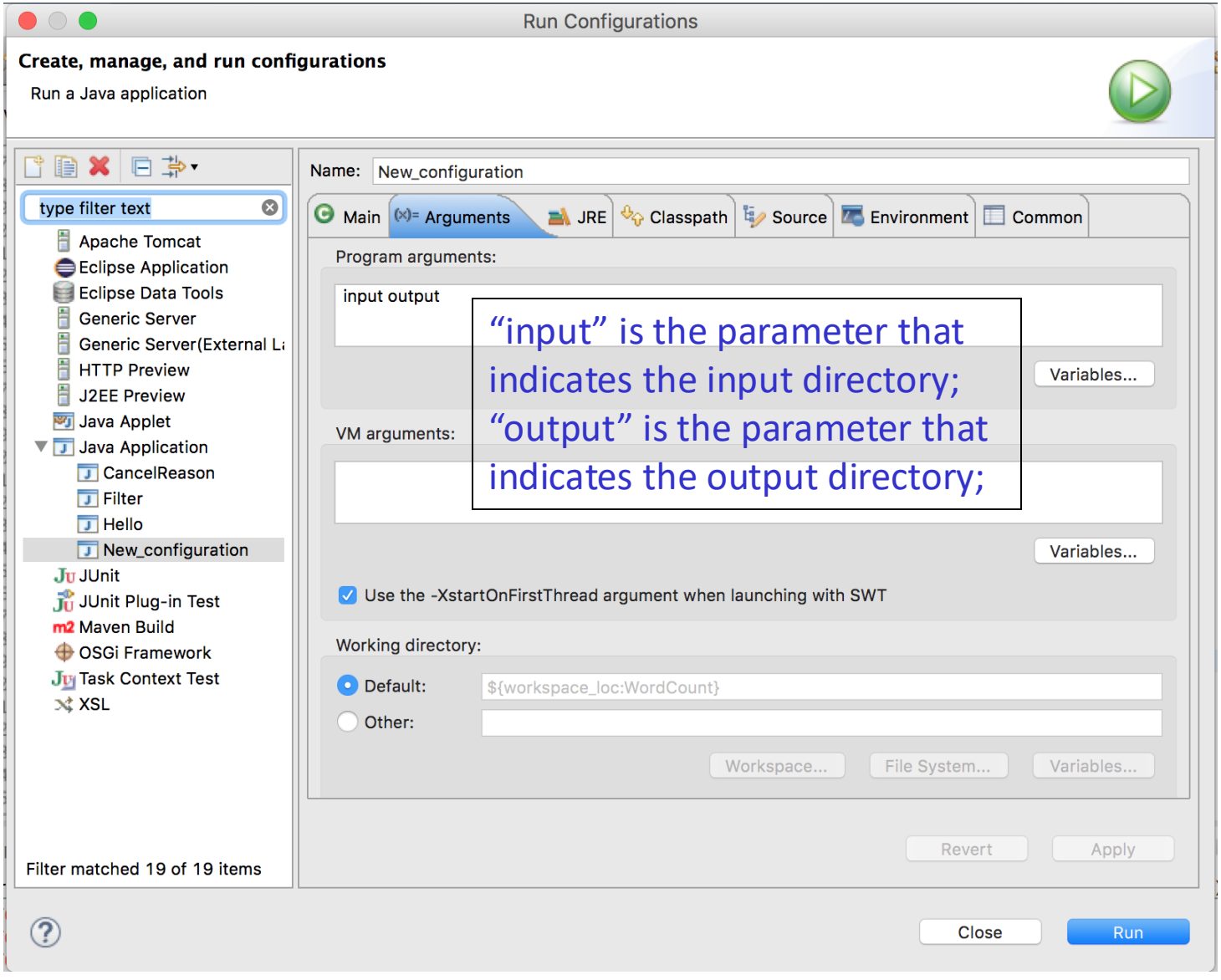
*WordCount.java

```
45  public static void main(String[] args) throws Exception {
46    //main function
47    Configuration conf = new Configuration();
48    String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
49    if (otherArgs.length != 2) {
50      System.err.println("Usage: wordcount <in> <out>");
51      System.exit(2);
52    }
53    Job job = new Job(conf, "word count");
54    job.setJarByClass(WordCount.class);
55    job.setMapperClass(TokenizerMapper.class);
56    job.setReducerClass(IntSumReducer.class);
57    job.setOutputKeyClass(Text.class);
58    job.setOutputValueClass(IntWritable.class);
59    FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
60    FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
61    System.exit(job.waitForCompletion(true) ? 0 : 1);
62  }
63 }
```

Word Count Problem
-Execute MapReduce Program on Eclipse

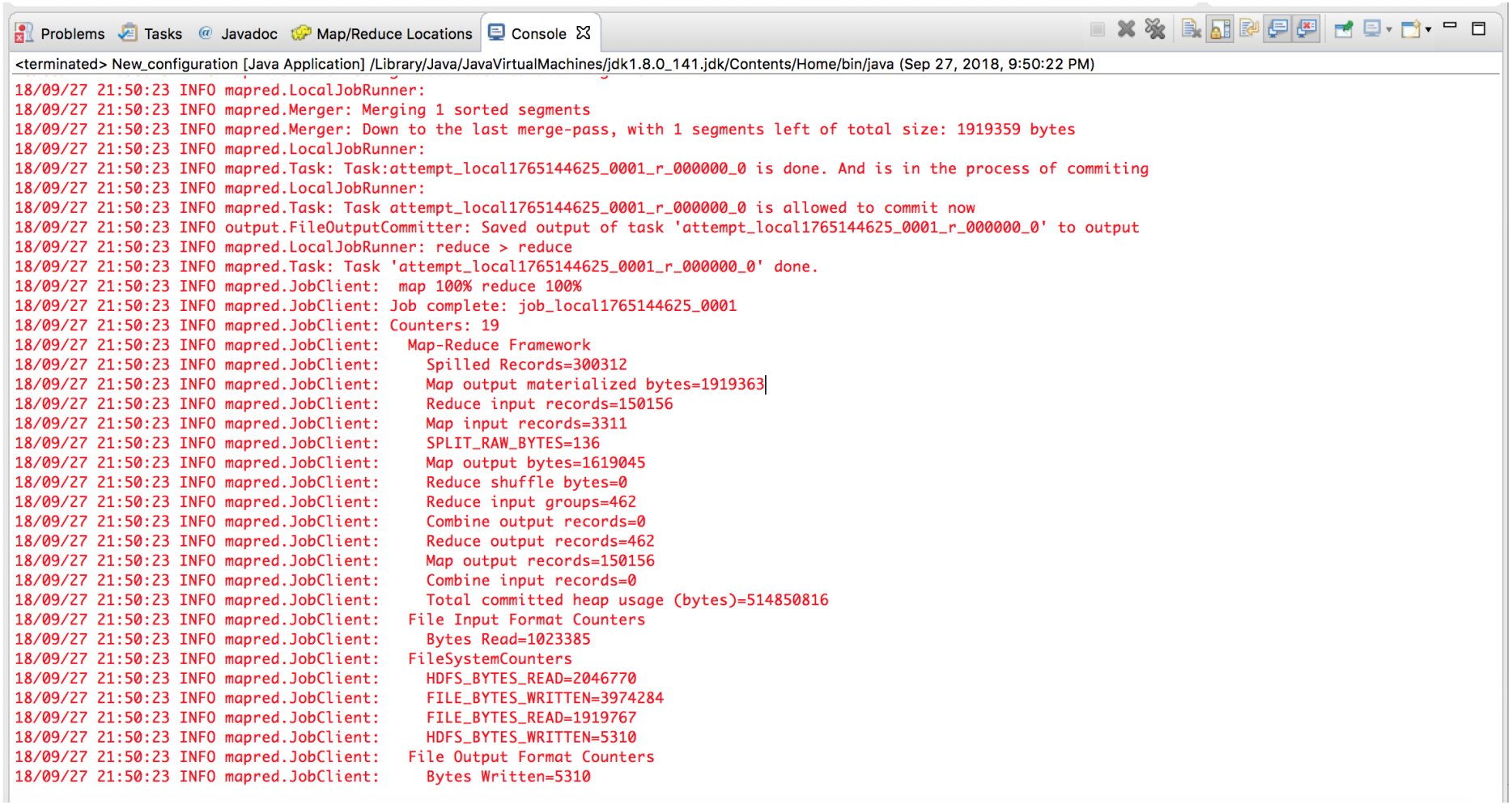


Word Count Problem
-Execute MapReduce Program on Eclipse



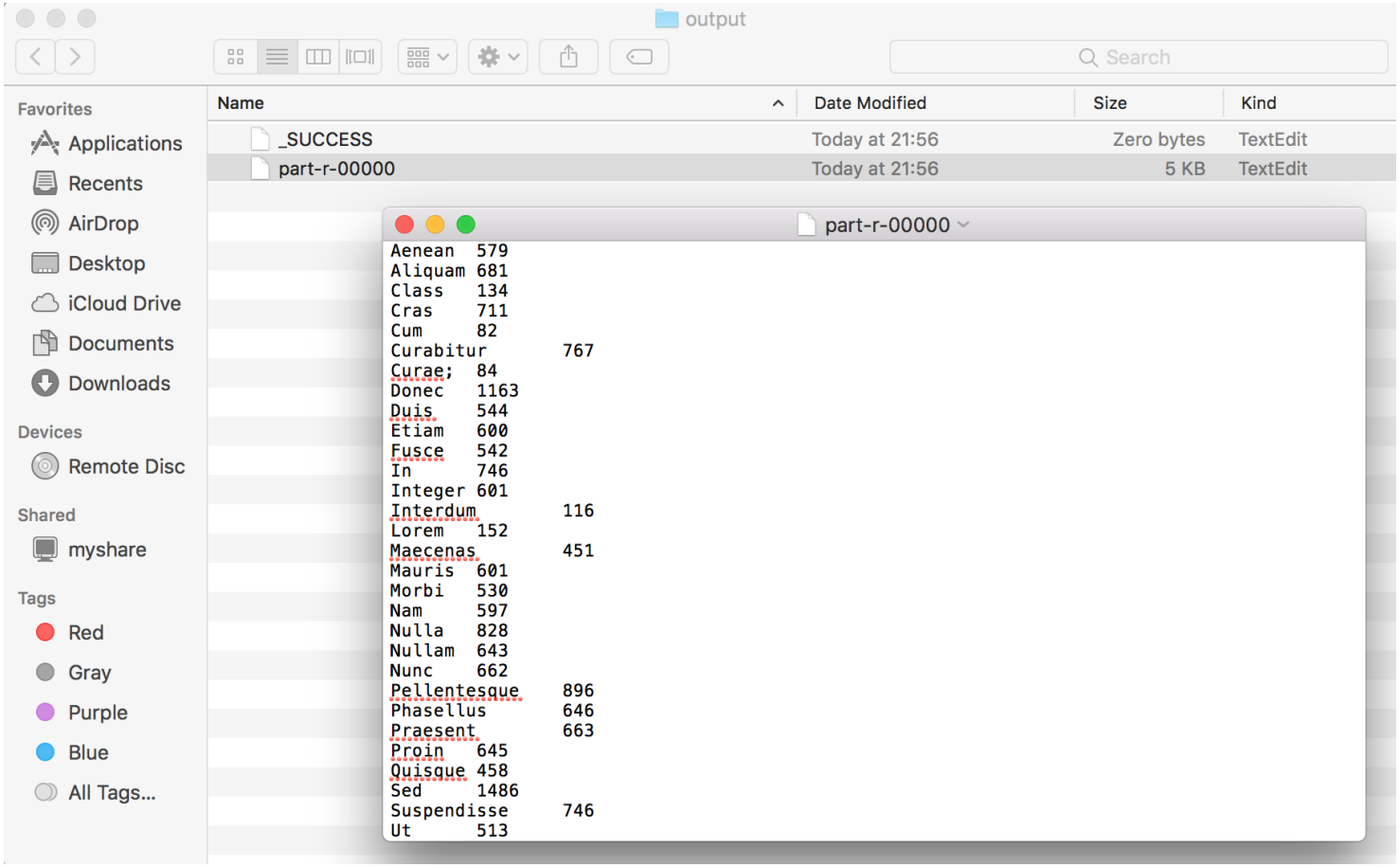
Word Count Problem
-Execute MapReduce Program on Eclipse

When the program is successfully executed:



Word Count Problem
-Check the output

```
Scarletts-MBP:hadoop-1.2.1 ScarlettTsao$ hadoop fs -get output ~/Desktop/
Warning: $HADOOP_HOME is deprecated.

Scarletts-MBP:hadoop-1.2.1 ScarlettTsao$
```



Word Count Problem -Configuration

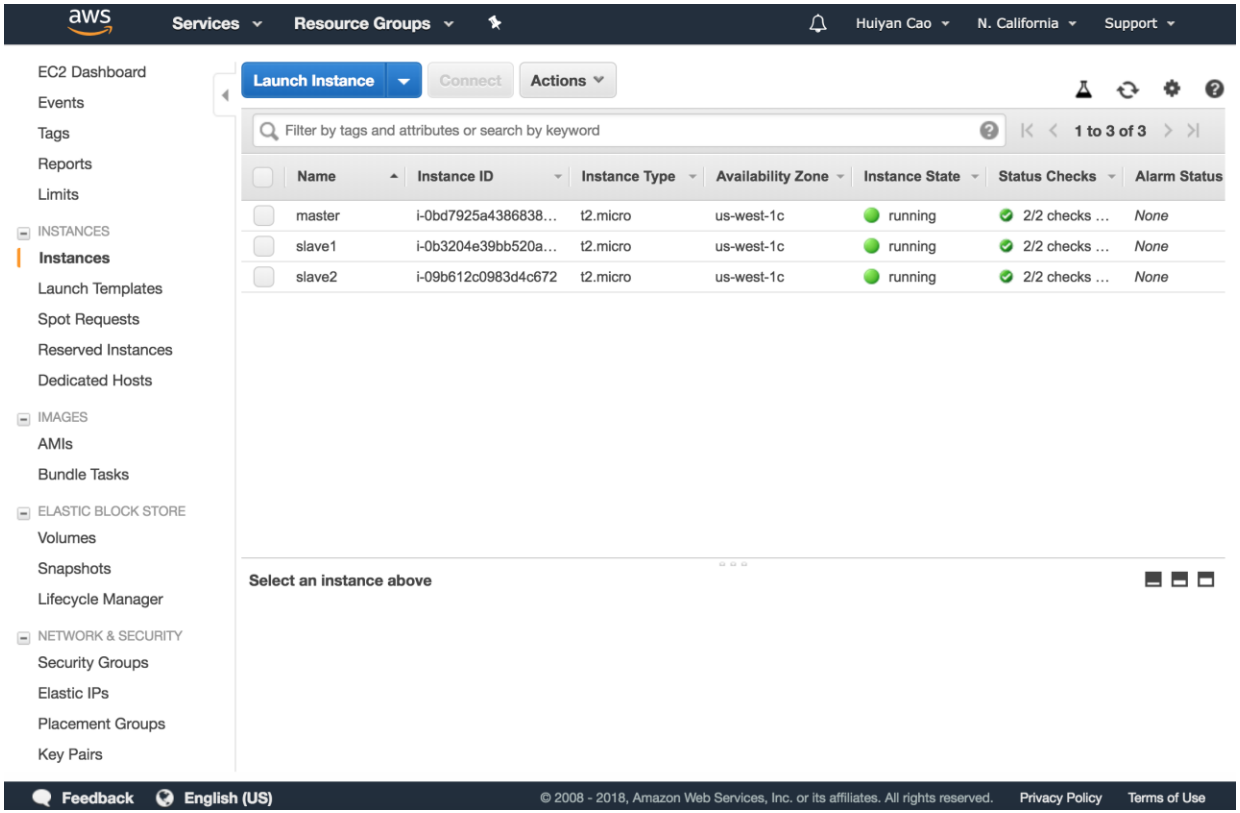
Version: Hadoop 1.2.1

Mode: Fully-Distributed Mode

Cloud: Amazon Web Service

Word Count Problem -Configuration

Three homogenous VM instances: one master node, and two slave nodes.



Instance Type — Edit instance type

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t2.micro	Variable	1	1	EBS only	-	Low to Moderate

Security Groups — Edit security groups

Instance Details — Edit instance details

Storage — Edit storage

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encrypted
Root	/dev/xvda	snap-0c44a2efc327e7ee6	8	gp2	100 / 3000	N/A	Yes	Not Encrypted

Word Count Problem
-Input

Create the input directory in HDFS and upload the input data from local to this directory. The input size is 108MB.

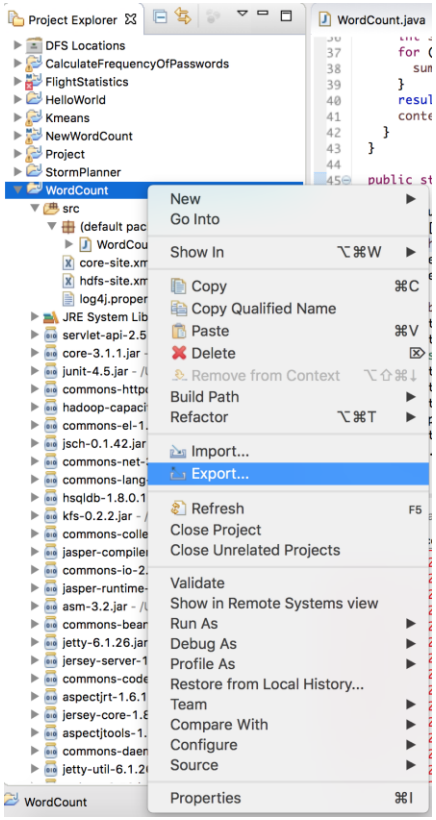
Word Count Problem
-Export JAR file with Eclipse

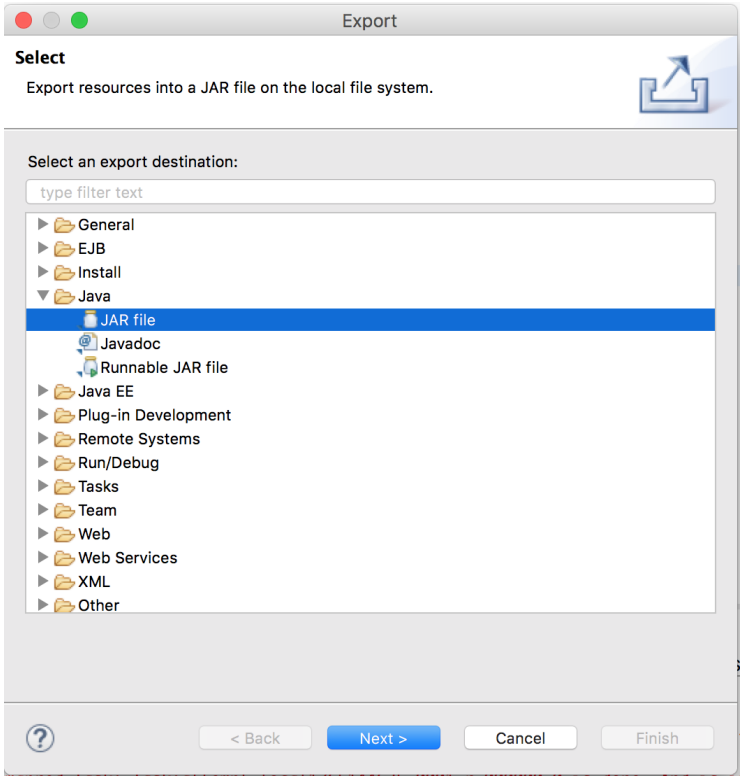
We use Eclipse to test the program locally (Stand-alone mode or Pseudo-Distributed mode).

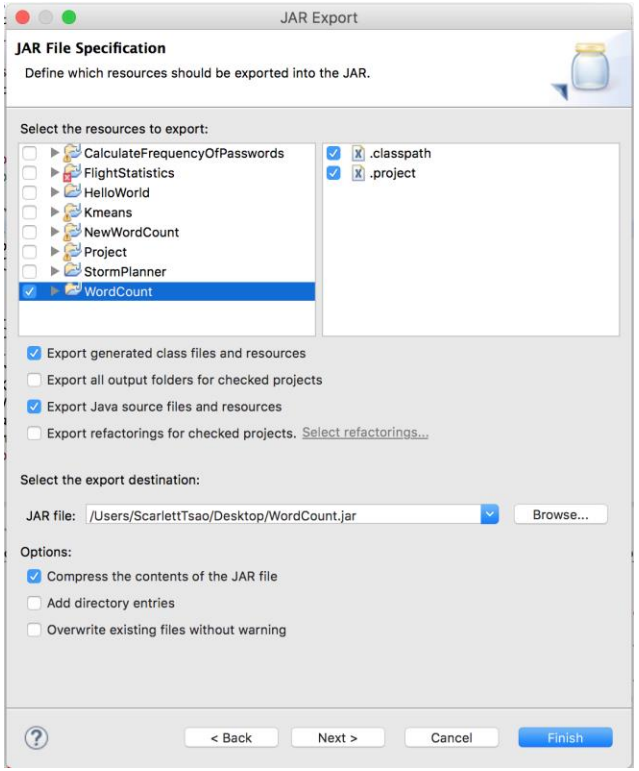
If we want to run a MapReduce program in a Fully-Distributed Mode on a Hadoop cluster, for example, in a public cloud environment, we can upload the JAR file to the master node of the cluster and execute the program by using the following command:

$ bin/hadoop jar WordCount.jar WordCount /user/user_name/wordcount/input /user/user_name/wordcount/output

Word Count Problem
-Export JAR file with Eclipse







Word Count Problem -Execution

Execute JAR on the cluster.

.ssh — hadoop@ip-10-0-0-67:~ — ssh -i hadoop.pem ec2-user@master — 127×56

```
[hadoop@ip-10-0-0-67 ~]$ hadoop jar WordCount.jar WordCount input output
Warning: $HADOOP_HOME is deprecated.

18/10/01 00:53:13 INFO input.FileInputFormat: Total input paths to process : 1
18/10/01 00:53:13 INFO util.NativeCodeLoader: Loaded the native-hadoop library
18/10/01 00:53:13 WARN snappy.LoadSnappy: Snappy native library not loaded
18/10/01 00:53:14 INFO mapred.JobClient: Running job: job_201810010050_0001
18/10/01 00:53:15 INFO mapred.JobClient:  map 0% reduce 0%
18/10/01 00:53:27 INFO mapred.JobClient:  map 18% reduce 0%
18/10/01 00:53:28 INFO mapred.JobClient:  map 50% reduce 0%
18/10/01 00:53:30 INFO mapred.JobClient:  map 61% reduce 0%
18/10/01 00:53:31 INFO mapred.JobClient:  map 80% reduce 0%
18/10/01 00:53:33 INFO mapred.JobClient:  map 92% reduce 0%
18/10/01 00:53:36 INFO mapred.JobClient:  map 100% reduce 0%
18/10/01 00:53:42 INFO mapred.JobClient:  map 100% reduce 16%
18/10/01 00:53:45 INFO mapred.JobClient:  map 100% reduce 70%
18/10/01 00:53:48 INFO mapred.JobClient:  map 100% reduce 82%
18/10/01 00:53:51 INFO mapred.JobClient:  map 100% reduce 93%
18/10/01 00:53:54 INFO mapred.JobClient:  map 100% reduce 100%
18/10/01 00:53:55 INFO mapred.JobClient: Job complete: job_201810010050_0001
18/10/01 00:53:55 INFO mapred.JobClient: Counters: 30
18/10/01 00:53:55 INFO mapred.JobClient:   Job Counters
18/10/01 00:53:55 INFO mapred.JobClient:     Launched reduce tasks=1
18/10/01 00:53:55 INFO mapred.JobClient:     SLOTS_MILLIS_MAPS=38850
18/10/01 00:53:55 INFO mapred.JobClient:     Total time spent by all reduces waiting after reserving slots (ms)=0
18/10/01 00:53:55 INFO mapred.JobClient:     Total time spent by all maps waiting after reserving slots (ms)=0
18/10/01 00:53:55 INFO mapred.JobClient:     Rack-local map tasks=1
18/10/01 00:53:55 INFO mapred.JobClient:     Launched map tasks=2
18/10/01 00:53:55 INFO mapred.JobClient:     Data-local map tasks=1
18/10/01 00:53:55 INFO mapred.JobClient:     SLOTS_MILLIS_REDUCES=19652
18/10/01 00:53:55 INFO mapred.JobClient:   File Output Format Counters
18/10/01 00:53:55 INFO mapred.JobClient:     Bytes Written=135481920
18/10/01 00:53:55 INFO mapred.JobClient:   FileSystemCounters
18/10/01 00:53:55 INFO mapred.JobClient:     FILE_BYTES_READ=513974282
18/10/01 00:53:55 INFO mapred.JobClient:     HDFS_BYTES_READ=108389844
18/10/01 00:53:55 INFO mapred.JobClient:     FILE_BYTES_WRITTEN=703824550
18/10/01 00:53:55 INFO mapred.JobClient:     HDFS_BYTES_WRITTEN=135481920
18/10/01 00:53:55 INFO mapred.JobClient:   File Input Format Counters
18/10/01 00:53:55 INFO mapred.JobClient:     Bytes Read=108389632
18/10/01 00:53:55 INFO mapred.JobClient:   Map-Reduce Framework
18/10/01 00:53:55 INFO mapred.JobClient:     Map output materialized bytes=189674700
18/10/01 00:53:55 INFO mapred.JobClient:     Map input records=13548192
18/10/01 00:53:55 INFO mapred.JobClient:     Reduce shuffle bytes=189674700
18/10/01 00:53:55 INFO mapred.JobClient:     Spilled Records=50260615
18/10/01 00:53:55 INFO mapred.JobClient:     Map output bytes=162578304
18/10/01 00:53:55 INFO mapred.JobClient:     Total committed heap usage (bytes)=336011264
18/10/01 00:53:55 INFO mapred.JobClient:     CPU time spent (ms)=39640
18/10/01 00:53:55 INFO mapred.JobClient:     Combine input records=0
18/10/01 00:53:55 INFO mapred.JobClient:     SPLIT_RAW_BYTES=212
18/10/01 00:53:55 INFO mapred.JobClient:     Reduce input records=13548192
18/10/01 00:53:55 INFO mapred.JobClient:     Reduce input groups=13548192
18/10/01 00:53:55 INFO mapred.JobClient:     Combine output records=0
18/10/01 00:53:55 INFO mapred.JobClient:     Physical memory (bytes) snapshot=509358080
18/10/01 00:53:55 INFO mapred.JobClient:     Reduce output records=13548192
18/10/01 00:53:55 INFO mapred.JobClient:     Virtual memory (bytes) snapshot=2244427776
18/10/01 00:53:55 INFO mapred.JobClient:     Map output records=13548192
```

Word Count Problem -Output

Download the output folder from HDFS to the master node.

Finds out the size of the output, which is 130MB.

Word Count Problem
-Output Location

Note that by default the data block size is 64MB in Hadoop 1.2.1, and 128MB in Hadoop 2.

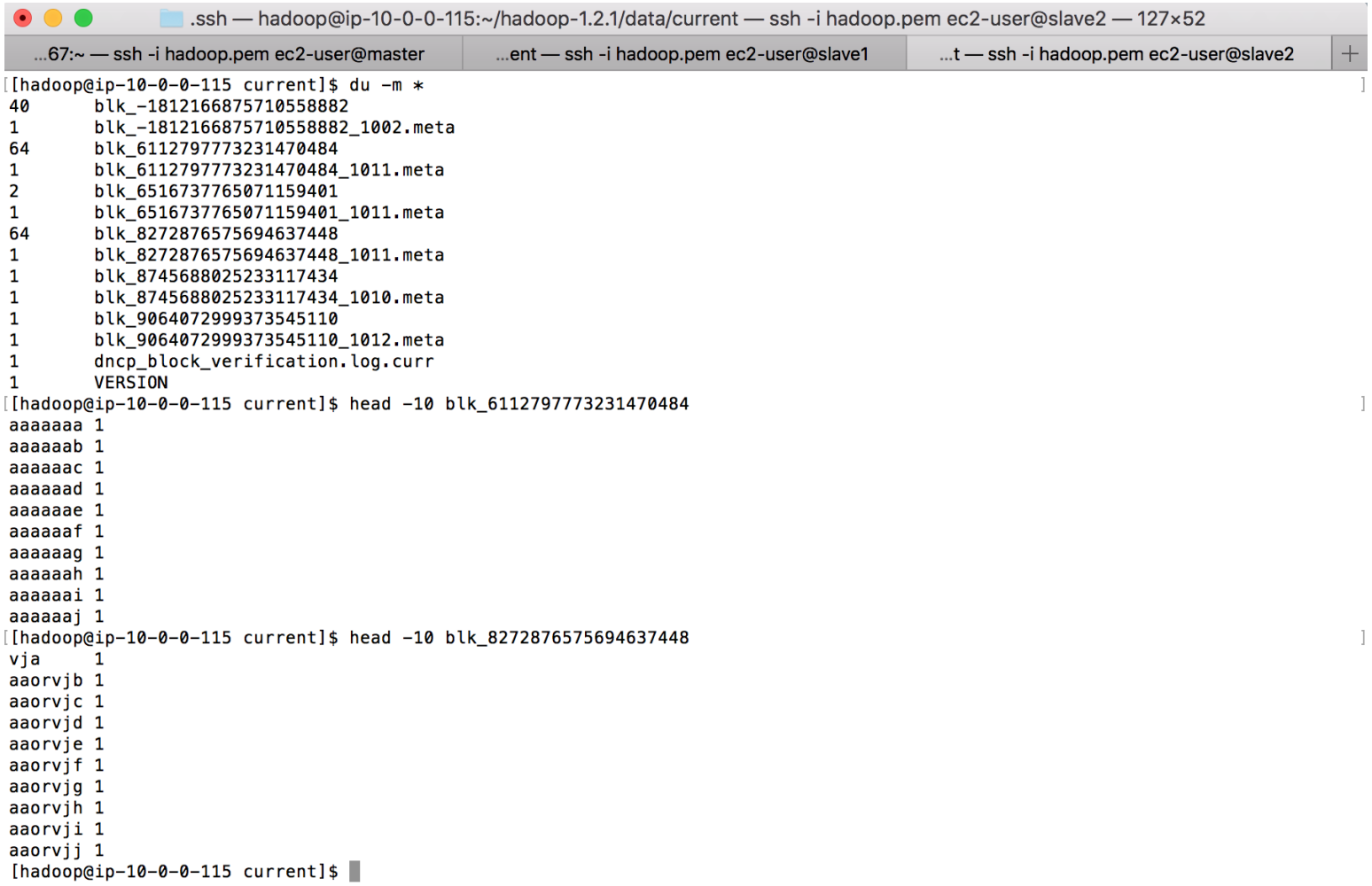
Check slave node 1: there is only one block stored on this node.

By checking the first 10 lines of the data block's contents, we see that the file stored on slave node 1 contains the mapping keys.

Word Count Problem
-Output Location

Check slave node 2: there are two data blocks stored on this node.

By checking the first 10 lines of these two data blocks' contents, we see that the data blocks stored on slave node 2 contain the output.



Word Count Problem -Configuration

Version: Hadoop 2.6.0

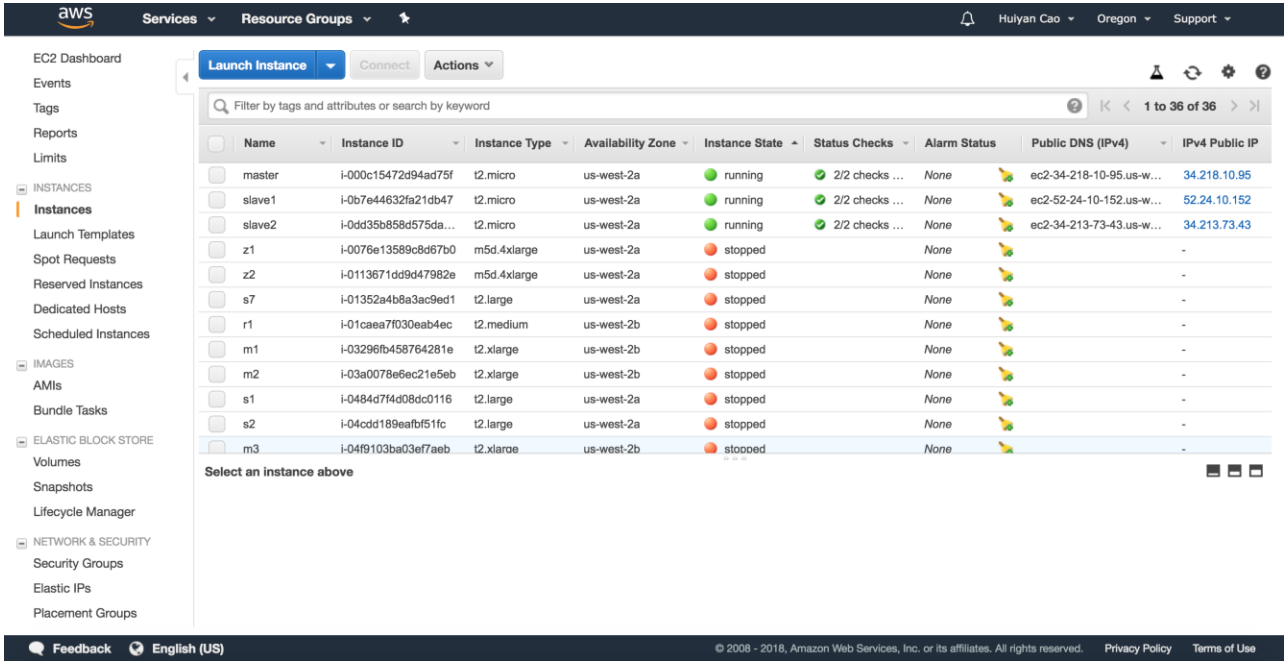
Mode: Fully-Distributed Mode

Cloud: Amazon Web Service

Word Count Problem -Configuration

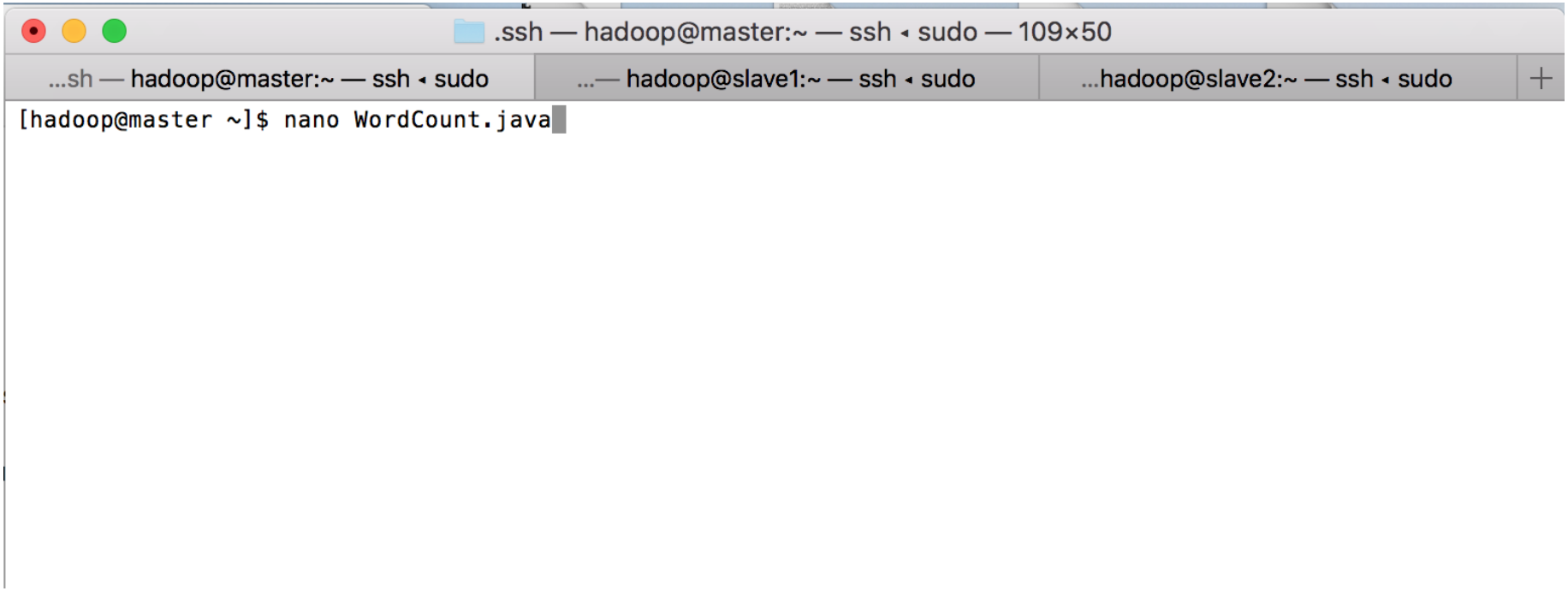
Three homogenous Virtual Machine instances:

- One master node
- Two slave nodes



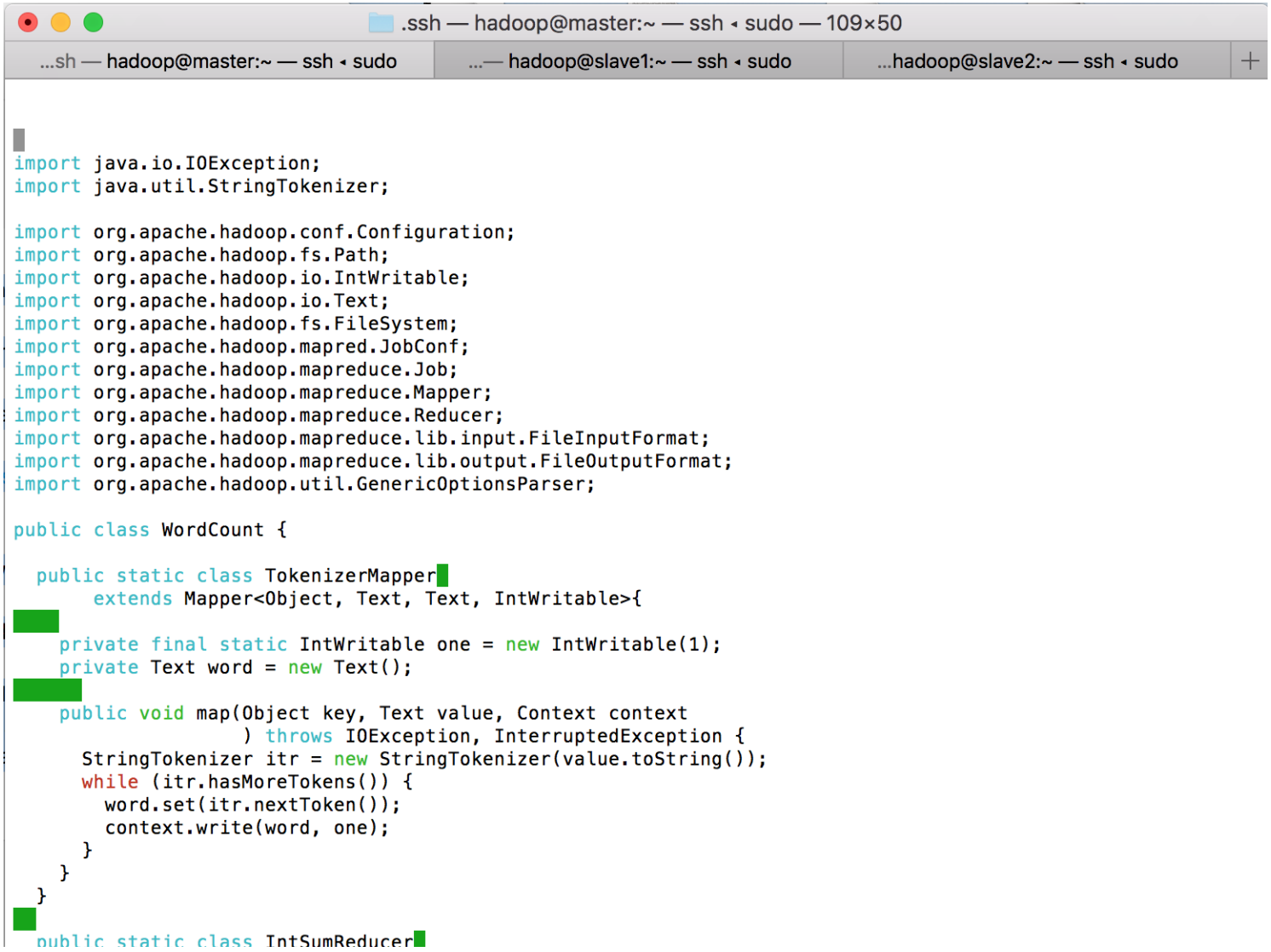
Word Count Problem
-MapReduce Program

Create a WordCount java program.



Word Count Problem
-MapReduce Program

Mapper Function:




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

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

public class WordCount {

  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
  }

  public static class IntSumReducer
```

Word Count Problem
-MapReduce Program

Reducer Function:

```
public static class IntSumReducer
     extends Reducer<Text,IntWritable,Text,IntWritable> {
  private IntWritable result = new IntWritable();

  public void reduce(Text key, Iterable<IntWritable> values,
                     Context context
                     ) throws IOException, InterruptedException {
    int sum = 0;
    for (IntWritable val : values) {
      sum += val.get();
    }
    result.set(sum);
    context.write(key, result);
  }
}
```

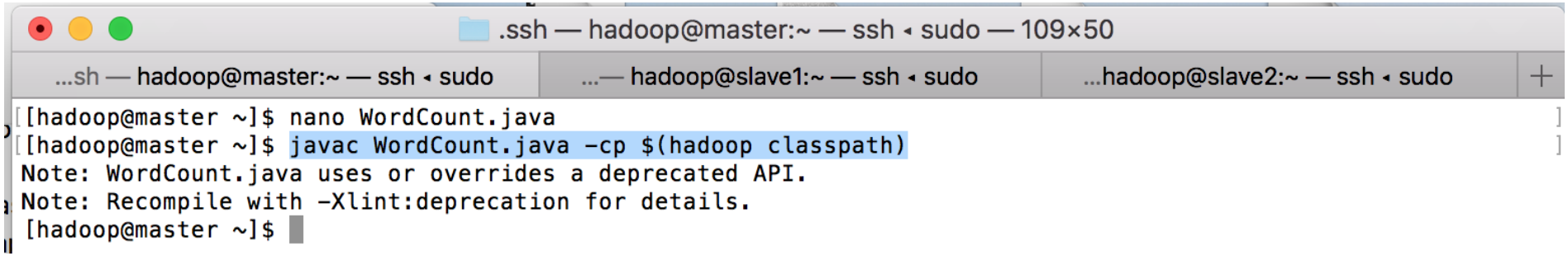
Word Count Problem
-MapReduce Program

Main Function:

```
public static void main(String[] args) throws Exception {
  Configuration conf = new Configuration();
 String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
  if (otherArgs.length != 2) {
    System.err.println("Usage: wordcount <in> <out>");
    System.exit(2);
  }

  FileSystem hdfs =FileSystem.get(conf);
  Path findf=new Path(otherArgs[1]);
  boolean isExists=hdfs.exists(findf);
  System.out.println("exit?"+isExists);
  if(isExists)
  {
      hdfs.delete(findf, true);
      System.out.println("delete output");
  }

  Job job = new Job(conf, "word count");

  job.setJarByClass(WordCount.class);
  job.setMapperClass(TokenizerMapper.class);
  job.setCombinerClass(IntSumReducer.class);
  job.setReducerClass(IntSumReducer.class);
  job.setOutputKeyClass(Text.class);
  job.setOutputValueClass(IntWritable.class);
  FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
  FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
  System.exit(job.waitForCompletion(true) ? 0 : 1);
 }
}
```

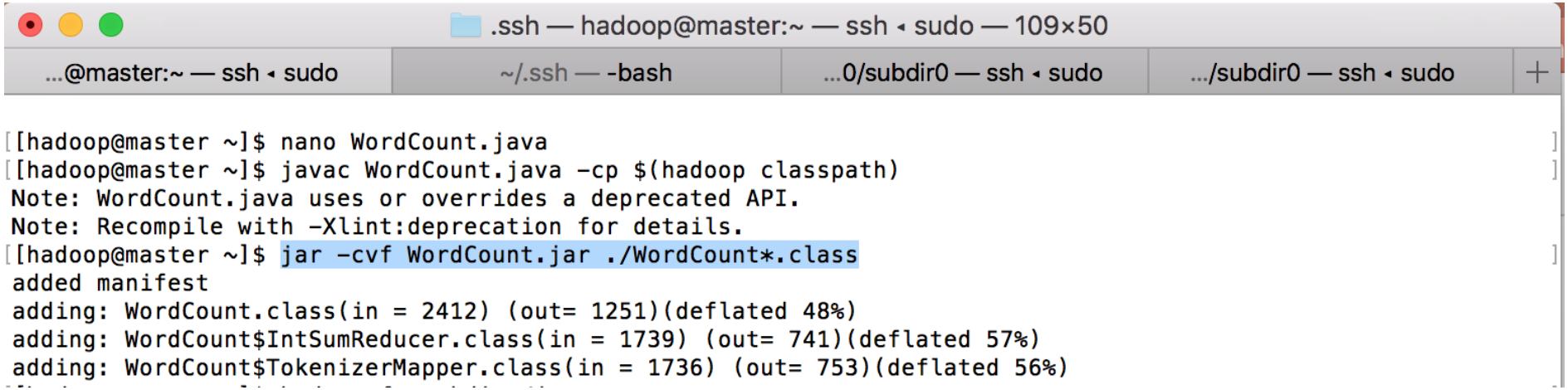
Word Count Problem
-Compile

Compile.



Word Count Problem -Compile

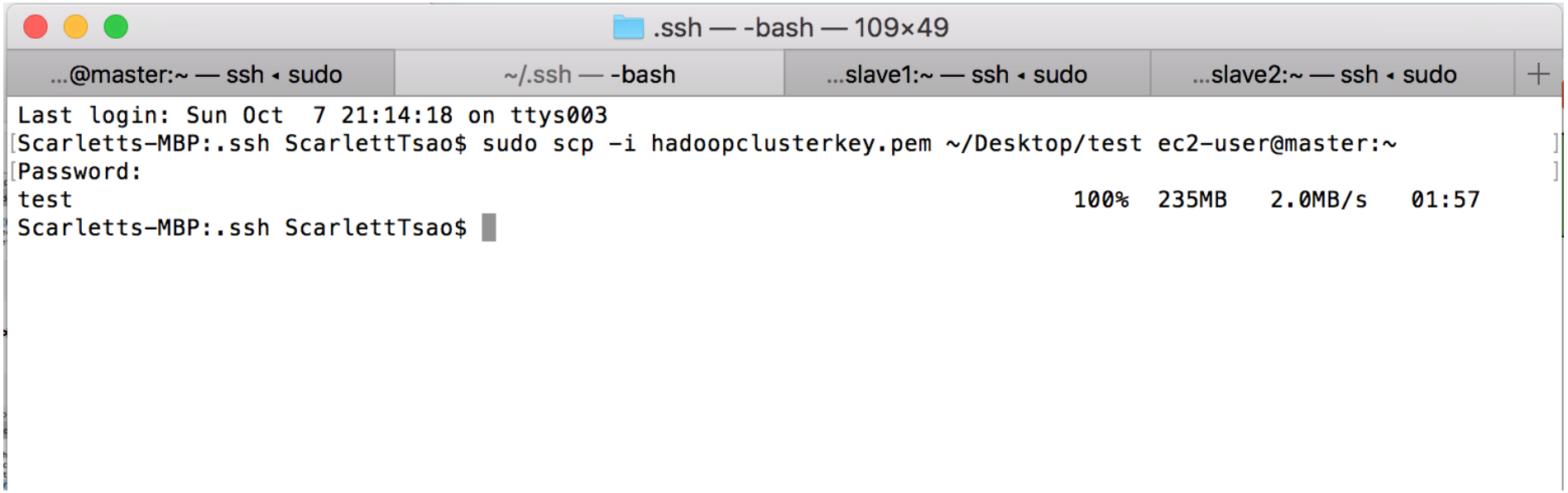
Export JAR.



Word Count Problem
-Input

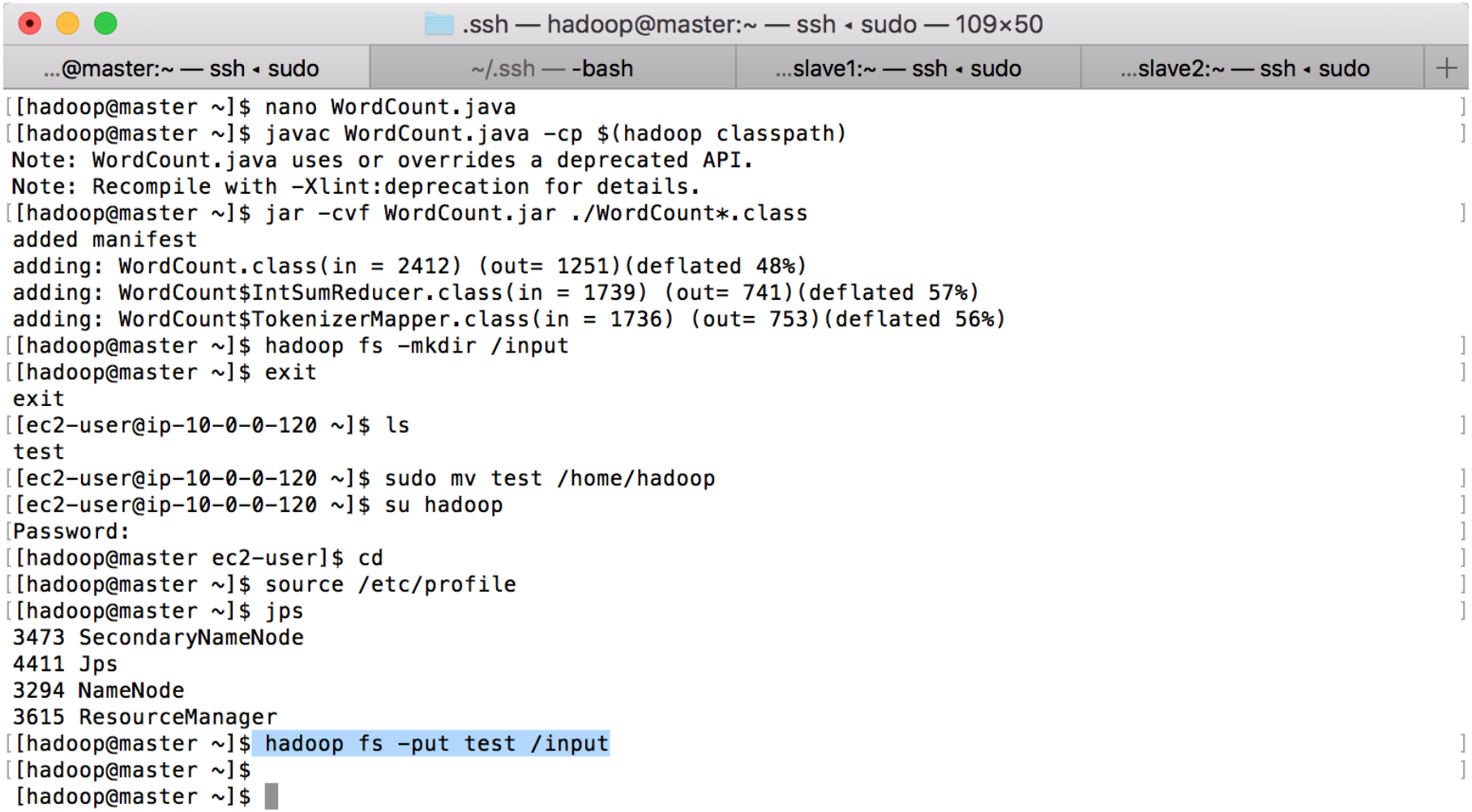
Upload the input data from local to the master node.

The input size is 235MB.



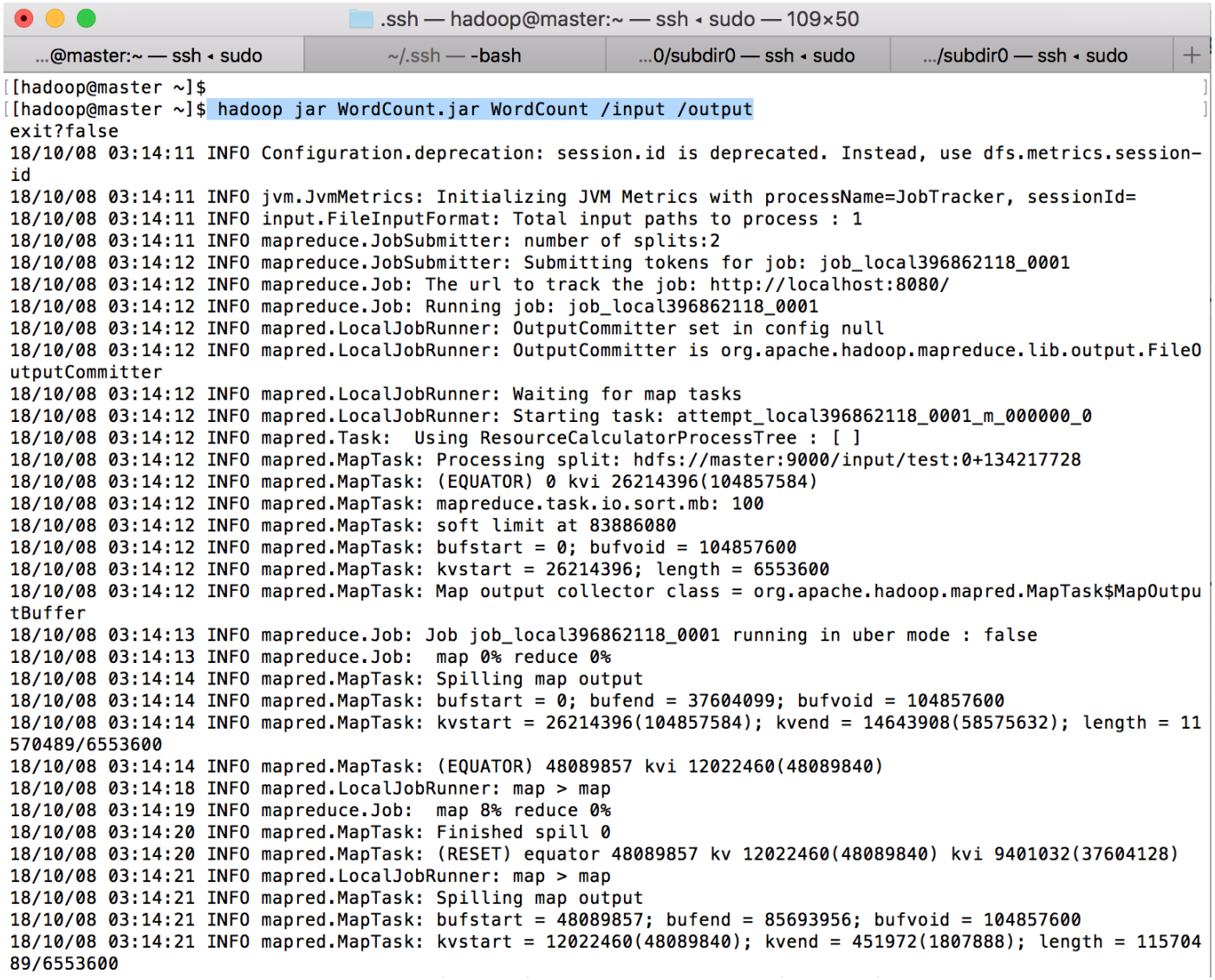
Word Count Problem
-Input

Create the input directory in HDFS and place the input file in it.



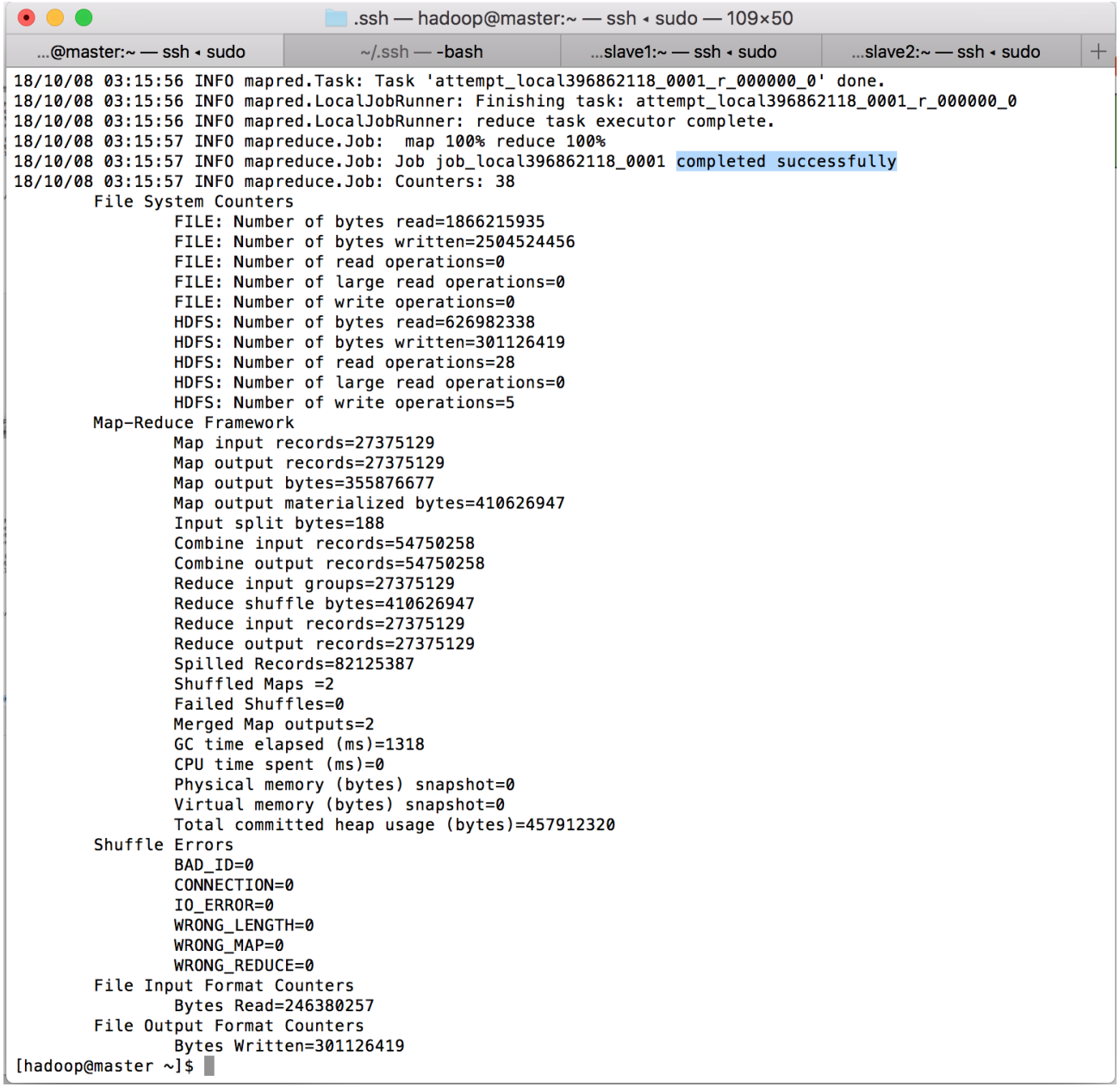
Word Count Problem -Execution

Execute JAR file on the cluster.



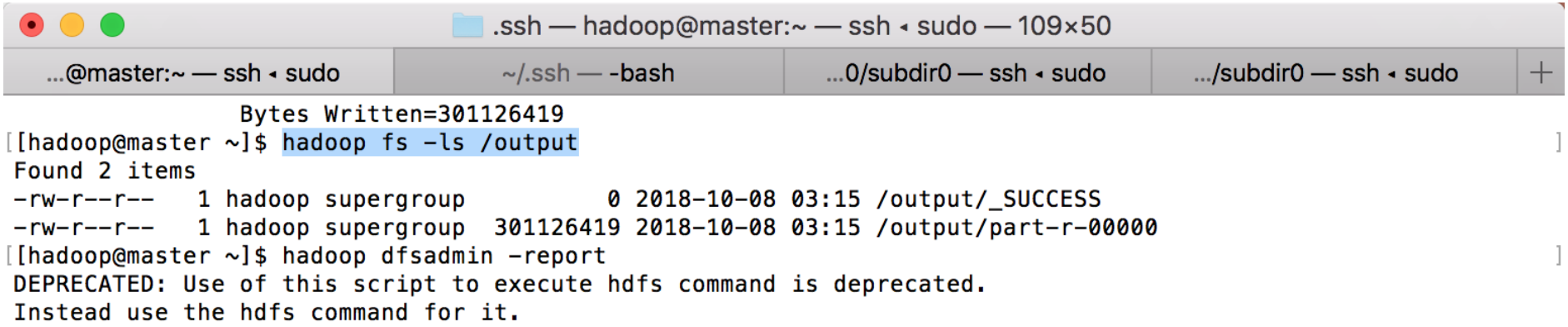
Word Count Problem -Execution

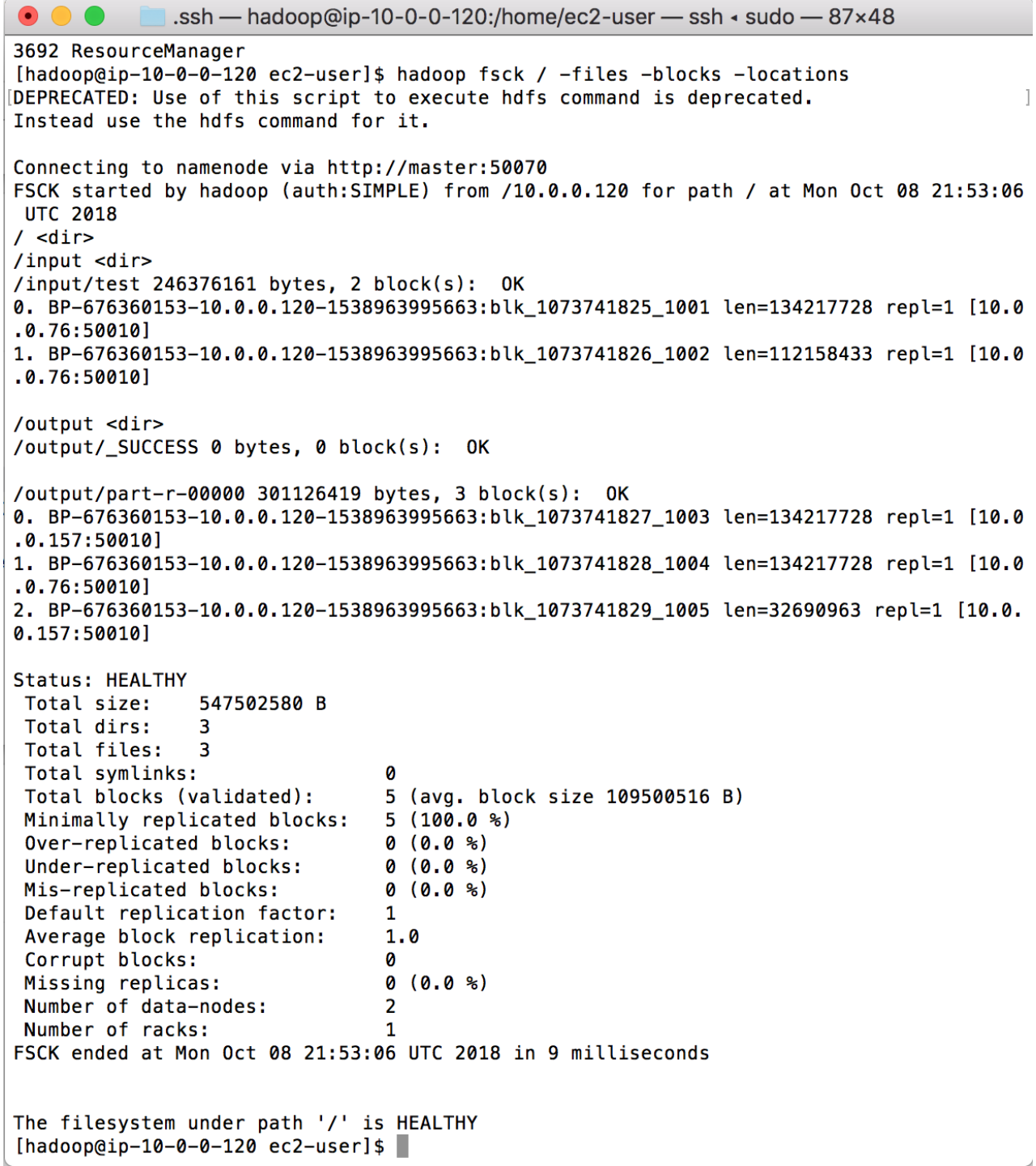
Execute successfully.



Word Count Problem -Output

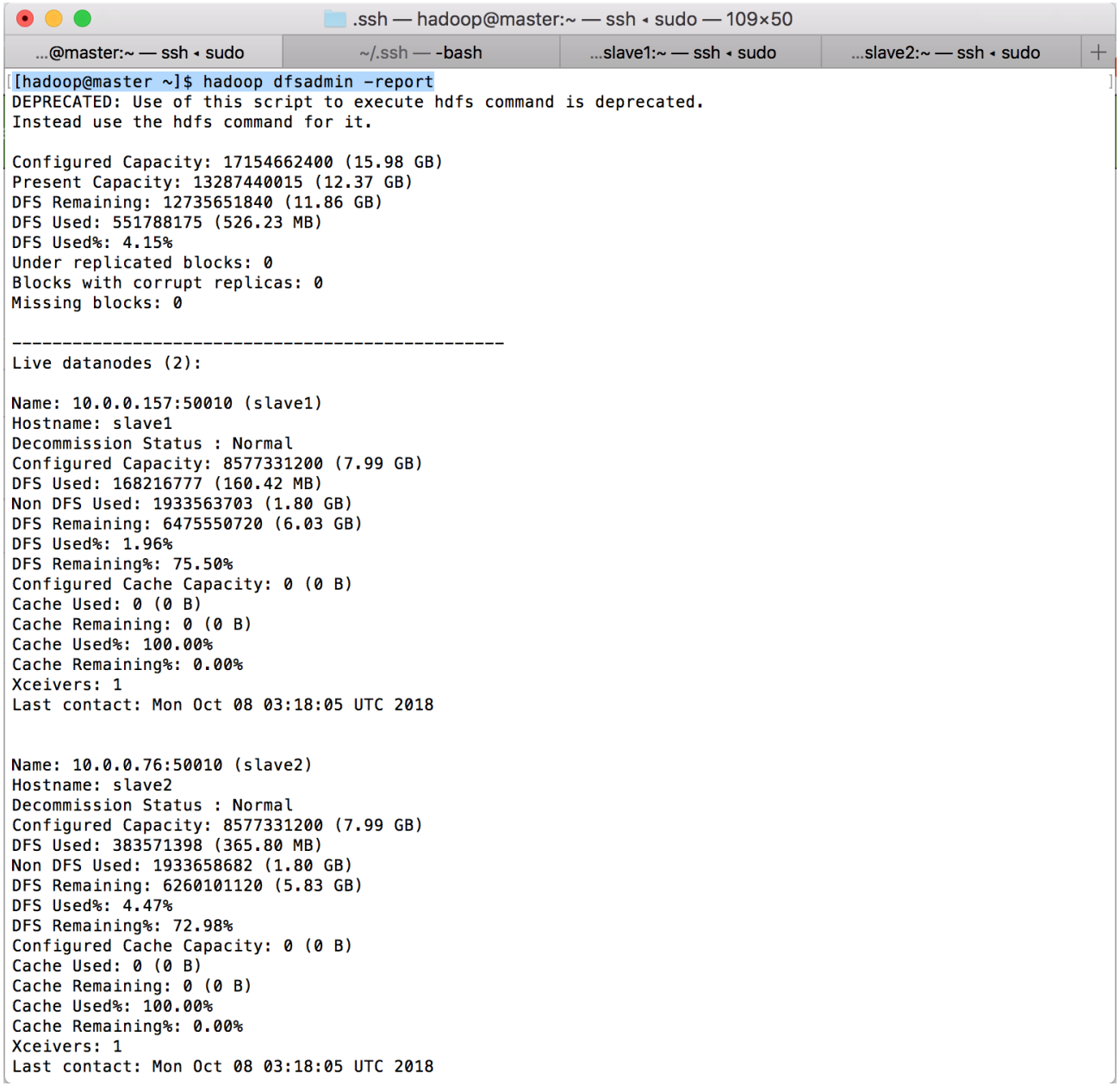
Finds out the size of the output, which is 287MB.




.ssh — hadoop@ip-10-0-0-120:/home/ec2-user — ssh ◂ sudo — 87×48
3692 ResourceManager
[hadoop@ip-10-0-0-120 ec2-user]$ hadoop fsck / -files -blocks -locations
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.
Connecting to namenode via http://master:50070
FSCK started by hadoop (auth:SIMPLE) from /10.0.0.120 for path / at Mon Oct 08 21:53:06 UTC 2018
/ <dir>
/input <dir>
/input/test 246376161 bytes, 2 block(s): OK
0. BP-676360153-10.0.0.120-1538963995663:blk_1073741825_1001 len=134217728 repl=1 [10.0.0.76:50010]
1. BP-676360153-10.0.0.120-1538963995663:blk_1073741826_1002 len=112158433 repl=1 [10.0.0.76:50010]
/output <dir>
/output/_SUCCESS 0 bytes, 0 block(s): OK
/output/part-r-00000 301126419 bytes, 3 block(s): OK
0. BP-676360153-10.0.0.120-1538963995663:blk_1073741827_1003 len=134217728 repl=1 [10.0.0.157:50010]
1. BP-676360153-10.0.0.120-1538963995663:blk_1073741828_1004 len=134217728 repl=1 [10.0.0.76:50010]
2. BP-676360153-10.0.0.120-1538963995663:blk_1073741829_1005 len=32690963 repl=1 [10.0.0.157:50010]
Status: HEALTHY
Total size: 547502580 B
Total dirs: 3
Total files: 3
Total symlinks: 0
Total blocks (validated): 5 (avg. block size 109500516 B)
Minimally replicated blocks: 5 (100.0 %)
Over-replicated blocks: 0 (0.0 %)
Under-replicated blocks: 0 (0.0 %)
Mis-replicated blocks: 0 (0.0 %)
Default replication factor: 1
Average block replication: 1.0
Corrupt blocks: 0
Missing replicas: 0 (0.0 %)
Number of data-nodes: 2
Number of racks: 1
FSCK ended at Mon Oct 08 21:53:06 UTC 2018 in 9 milliseconds
The filesystem under path '/' is HEALTHY
[hadoop@ip-10-0-0-120 ec2-user]$

Word Count Problem -Output Location

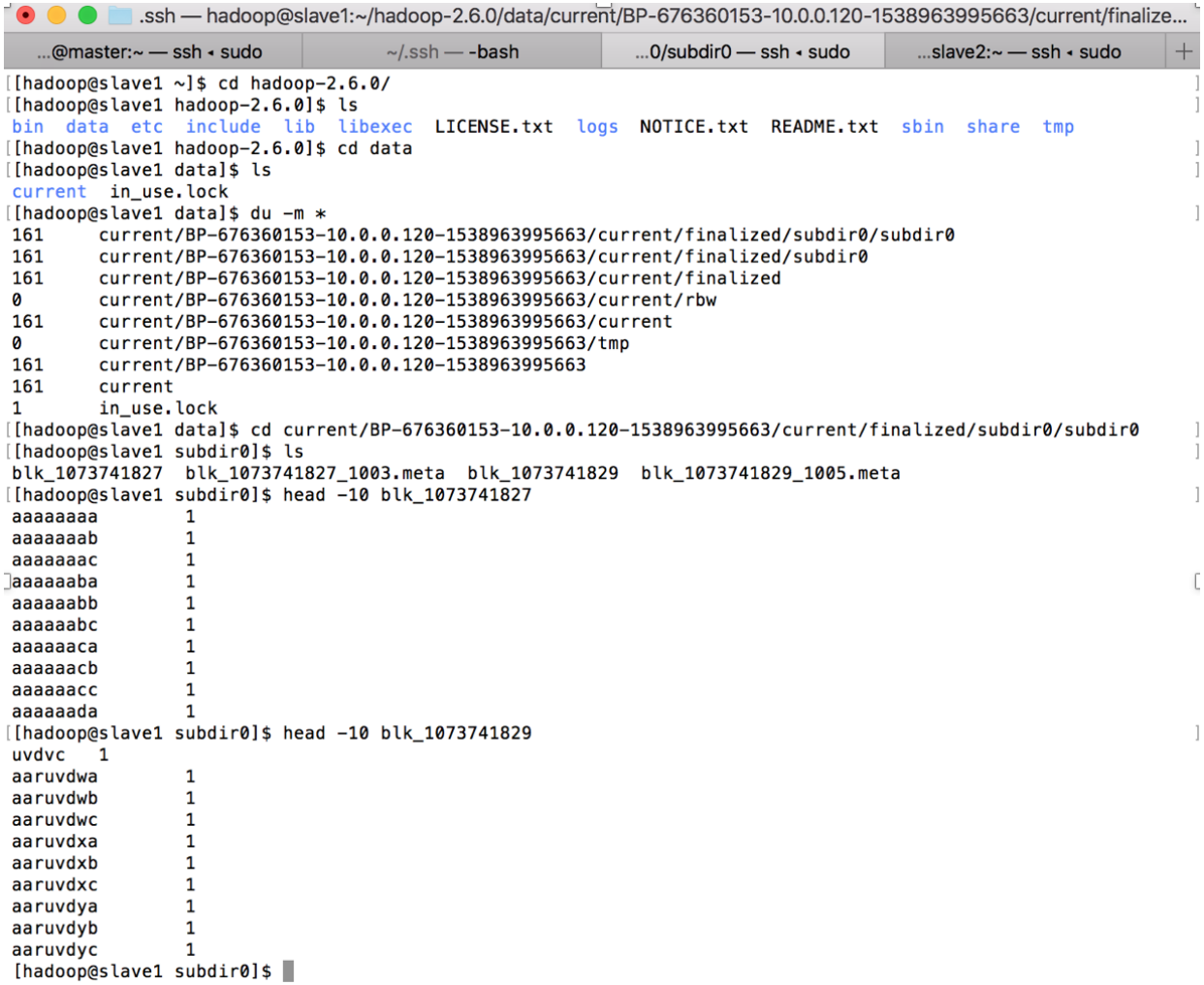
Two datablocks on slave 1.
（160.42MB）
Three datablocks on slave 2.
（365.80MB）



Word Count Problem
-Output Location

Check slave node 1: There are two data blocks stored on this node. The data block size is 128MB.

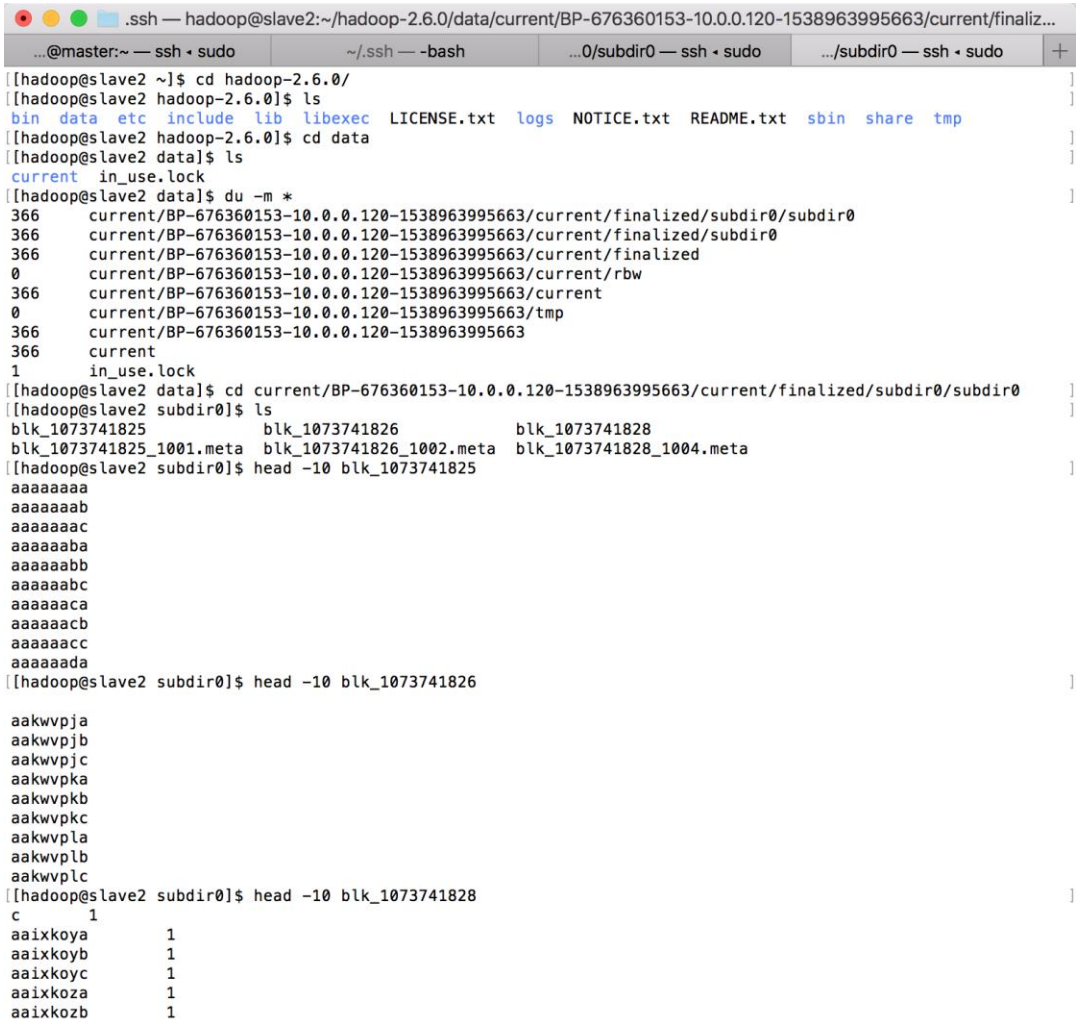
By checking the first 10 lines of the two datablocks' contents, we see that both datablocks stored on slave node 1 are the output.



Word Count Problem
-Output Location

Check slave node 2: There are three data blocks stored on this node. The data block size is 128MB.

By checking the first 10 lines of the three datablocks' contents, we see that the third datablock stored on slave node 2 is the output, while the other two store the keys.



Hadoop configuration

- Standalone Mode

https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/SingleCluster.html#Configuration

- Pseudo-Distributed Mode

https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/SingleCluster.html#Pseudo-Distributed_Operation

- Fully-Distributed Mode

https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/ClusterSetup.html

How to install Hadoop on Amazon AWS (step by step):
https://www.youtube.com/watch?v=a-DXDkK1i08

Another useful tutorial:
https://www.edureka.co/blog/install-hadoop-single-node-hadoop-cluster

Java programming

How to MapReduce programming with Apache:

https://www.javaworld.com/article/2077907/open-source-tools/mapreduce-programming-with-apache-hadoop.html

If you need more details, the following book helps:

https://eecs.wsu.edu/~yinghui/mat/courses/fall%202015/resources/Hadoop%20the%20definitive%20guide.pdf

The following tutorial shows you how to use Eclipse to write, compile, execute and export .jar file for the word counting problem in Hadoop in detail:

https://www.dezyre.com/hadoop-tutorial/hadoop-mapreduce-wordcount-tutorial

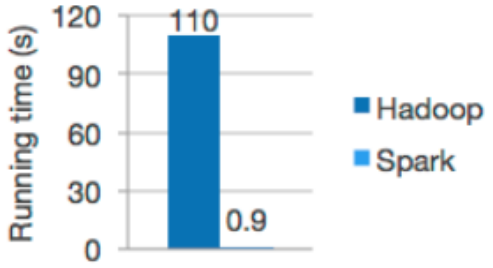
Apache Spark

Built on top of HDFS

Speed

Run programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk.

Spark has an advanced DAG execution engine that supports cyclic data flow and in-memory computing.



Logistic regression in Hadoop and Spark

Ease of Use

Write applications quickly in Java, Scala or Python.

Spark offers over 80 high-level operators that make it easy to build parallel apps. And you can use it *interactively* from the Scala and Python shells.

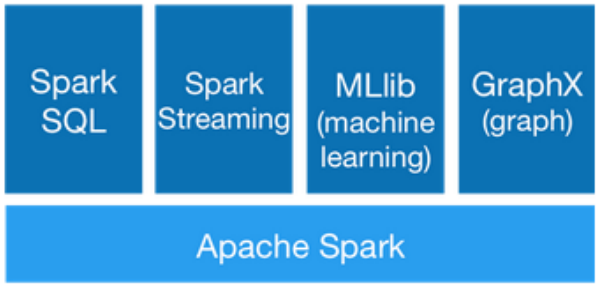
```
file = spark.textFile("hdfs://...")

file.flatMap(lambda line: line.split())
    .map(lambda word: (word, 1))
    .reduceByKey(lambda a, b: a+b)
```

Word count in Spark's Python API

Generality

Combine SQL, streaming, and complex analytics.

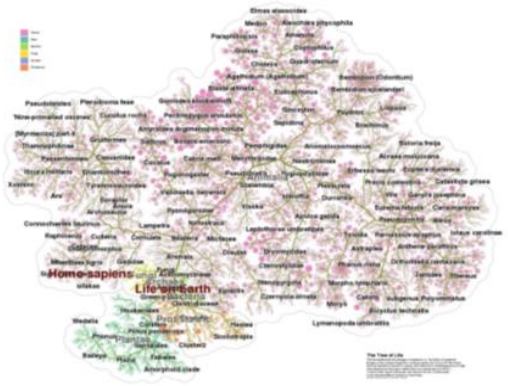
Spark powers a stack of high-level tools including Spark SQL, MLlib for machine learning, GraphX, and Spark Streaming. You can combine these frameworks seamlessly in the same application.



Big Data Visualization

- Graph Database
- Visual Analytics

76,425 species



Tree of Life by Dr. Yifan Hu

14.8 million tweets

The information diffusion graph of the death of Osama bin Laden by Gilad Lotan

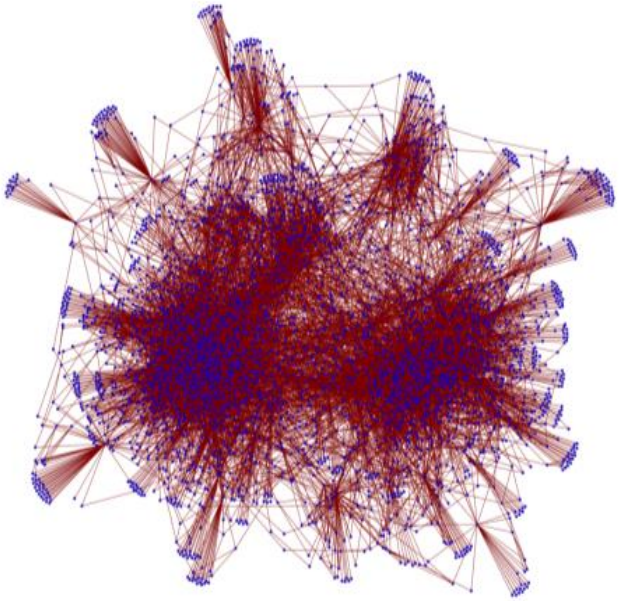
500 million users

Facebook friendship graph by Paul Butler

Challenging Task :

Squeezing millions and even billions of records into million pixels (1600 X 1200 ≈ 2 million pixels)

Visualization Key Challenges



Visual clutter

How can we encode the information intuitively?

Performance issues

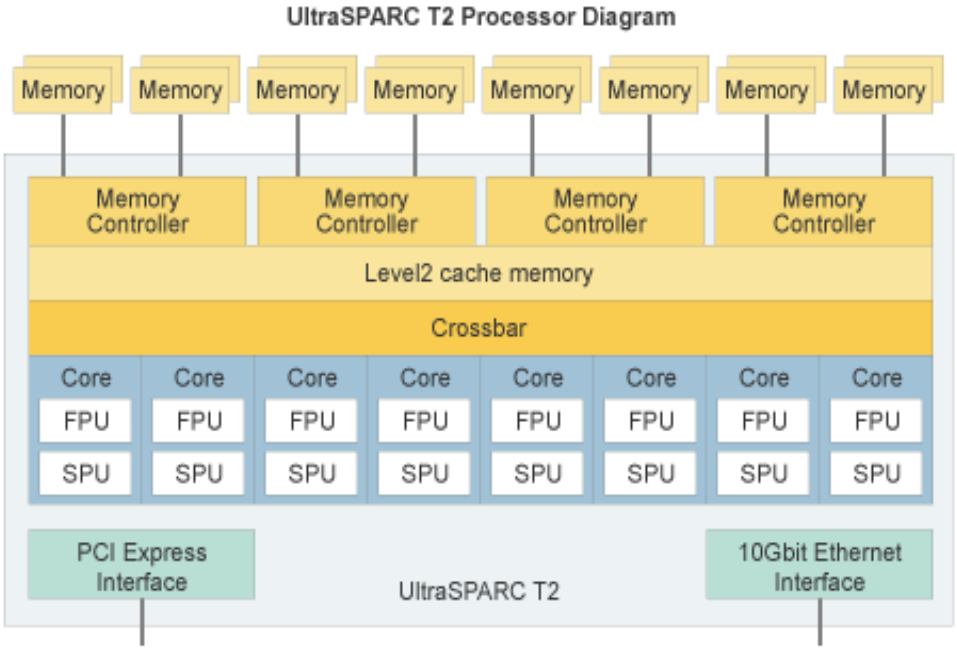
How can we render the huge datasets in real time with rich interactions?

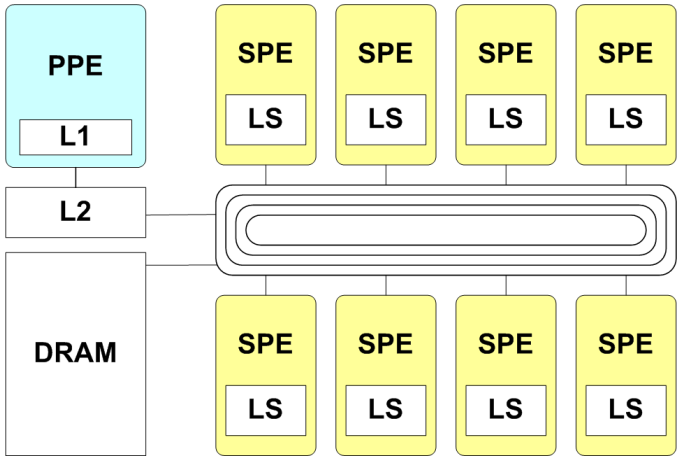
Cognition

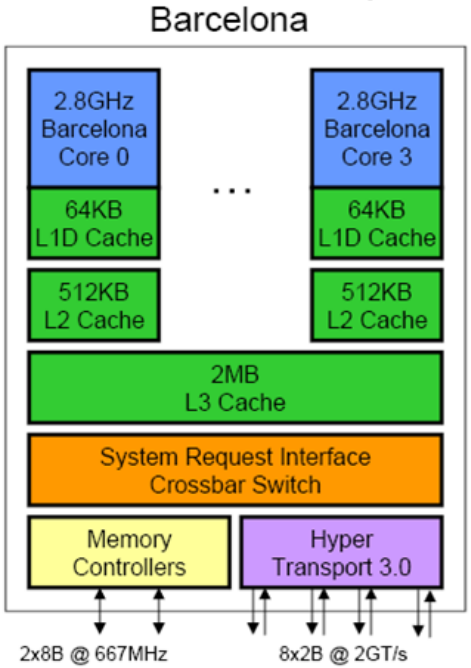
How can users understand the visual representation when the information is overwhelming?

Platform Dependent Graphical Models

- Homogeneous **multicore** processors
  - Intel Xeon E5335 (Clovertown)
  - AMD Opteron 2347 (Barcelona)
  - *Netezza (FPGA, multicore)*
- Homogeneous **manycore** processors
  - Sun UltraSPARC T2 (Niagara 2), GPGPU
- **Heterogeneous** multicore processors
  - Cell Broadband Engine
- **Clusters**
  - HPCC, DataStar, *BlueGene,* etc.

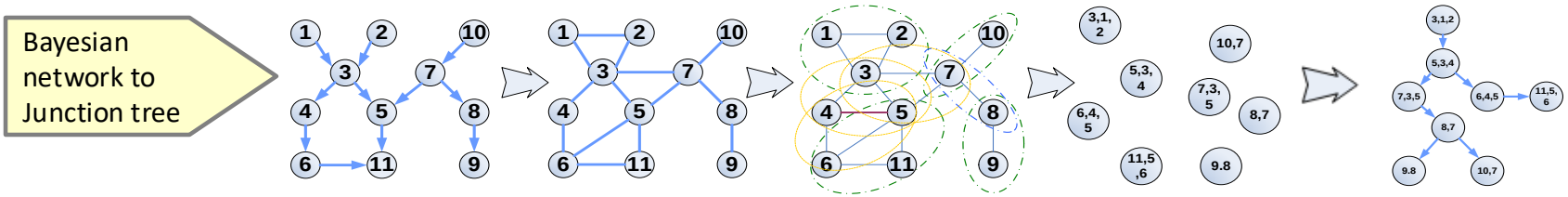






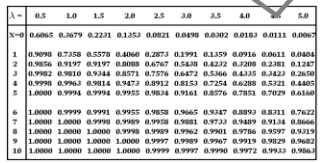
Graph Workload Types

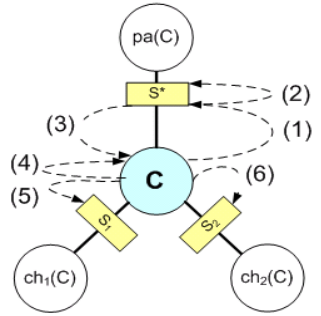
- Type 1: Computations on graph structures / topologies
  - Example → converting Bayesian network into junction tree, graph traversal (BFS/DFS), etc.
  - Characteristics → Poor locality, irregular memory access, limited numeric operations



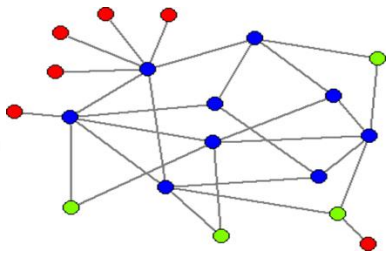
- Type 2: Computations on graphs with rich properties
  - Example → Belief propagation: diffuse information through a graph using statistical models
  - Characteristics
    - Locality and memory access pattern depend on vertex models
    - Typically a lot of numeric operations
    - Hybrid workload

$$\psi^*_{\mathcal{S}} = \sum_{\mathcal{Y}\setminus\mathcal{S}} \psi^*_{\mathcal{Y}}, \quad \psi^*_{\mathcal{X}} = \psi_{\mathcal{X}} \frac{\psi^*_{\mathcal{S}}}{\psi_{\mathcal{S}}}$$





- Type 3: Computations on dynamic graphs
  - Example → streaming graph clustering, incremental k-core, etc.
  - Characteristics
    - Poor locality, irregular memory access
    - Operations to update a model (e.g., cluster, sub-graph)
    - Hybrid workload



3-core subgraph

Large-scale graph benchmark – Graph 500 complementing Top 500

Breadth-First Search (BFS), Single Source Shortest Path (SSSP)

November 2024

RANK	MACHINE	VENDOR	INSTALLATION SITE	LOCATION	COUNTRY	YEAR	NUMBER OF NODES	NUMBER OF CORES	SCALE	GTEPS
1	Wuhan Supercomputer	HUST	Wuhan Supercomputing Center	Wuhan	China	2023	252	6999552	41	15335.9
2	Pengcheng Cloudbrain-II	HUST-Pengcheng Lab-HUAWEI	Pengcheng Lab	ShenZhen	China	2022	488	93696	40	11529.7
3	Supercomputer Fugaku	Fujitsu	RIKEN Center for Computational Science (R-CCS)	Kobe Hyogo	Japan	2020	82944	3981312	39	2126.45
4	Tianhe Exascale Prototype Upgrade System	National University of Defense Technology	National Supercomputer Center in Tianjin	Tianjin	China	2021	2048	131072	34	2054.35
5	SuperMUC-NG	Lenovo	Leibniz Rechenzentrum	Garching	Germany	2018	4096	196608	37	1053.93
6	NERSC Cori - 1024 haswell partition	Cray	NERSC/LBNL	DOE/SC/LBNL/NERSC	United States	2017	1024	32768	36	558.833
7	Nurion	Cray	Korea Institute of Science and Technology Information	Daejeon	Korea Republic Of	2018	1024	65536	36	337.239
8	NERSC Cori - 512 KNL partition	Cray	NERSC/LBNL	DOE/SC/LBNL/NERSC	United States	2017	512	32768	35	229.188
9	Lise	Atos	Zuse Institute Berlin (ZIB)	Berlin	Germany	2019	1270	121920	38	197.7
10	Undisclosed Cray XE6	Cray	National Computing Facility	University	United States	2013	512	16384	34	134.173

Common Use Cases for Big Data in Hadoop

- Log Data Analysis
  - most common, fits perfectly for HDFS scenario: Write once & Read often.
- Data Warehouse Modernization
- Fraud Detection
- Risk Modeling
- Social Sentiment Analysis
- Image Classification
- Graph Analysis
- Beyond

D. deRoos *et al*, Hadoop for Dummies, John Wiley & Sons, 2014

Big Data Analytics Example Use Cases

System G

1. Social Network Analysis
2. Recommendation
3. Commerce
4. Financial Analysis
5. Social Media Monitoring
6. Telco Customer Analysis
7. Watson
8. Data Exploration and Visualization
9. Personalized Search
10. Anomaly Detection (Espionage, Sabotage, etc.)
11. Fraud Detection
12. Cybersecurity
13. Sensor Monitoring (Smarter another Planet)
14. Celluar Network Monitoring
15. Cloud Monitoring
16. Code Life Cycle Management
17. Traffic Navigation
18. Image and Video Semantic Understanding
19. Genomic Medicine
20. Brain Network Analysis
21. Data Curation
22. Near Earth Object Analysis

Use Case 1: Social Network Analysis in Enterprise for Productivity

Production Live System used by IBM GBS since 2009 – verified ~$100M contribution

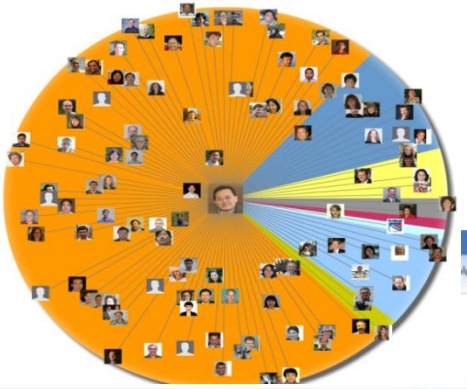
15,000 contributors in 76 countries; 92,000 annual unique IBM users

25,000,000+ emails & SameTime messages (incl. Content features)

1,000,000+ Learning clicks; 14M KnowledgeView, SalesOne, …, access data

1,000,000+ Lotus Connections (blogs, file sharing, bookmark) data

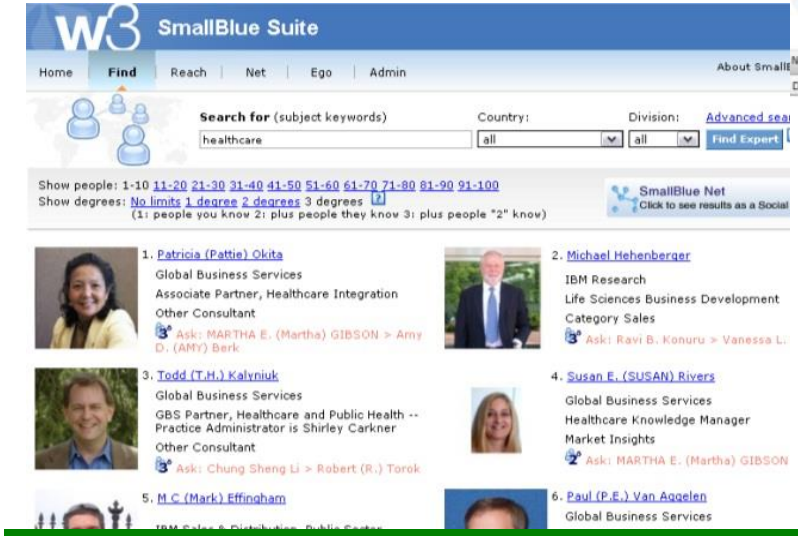
200,000 people's consulting project & earning data

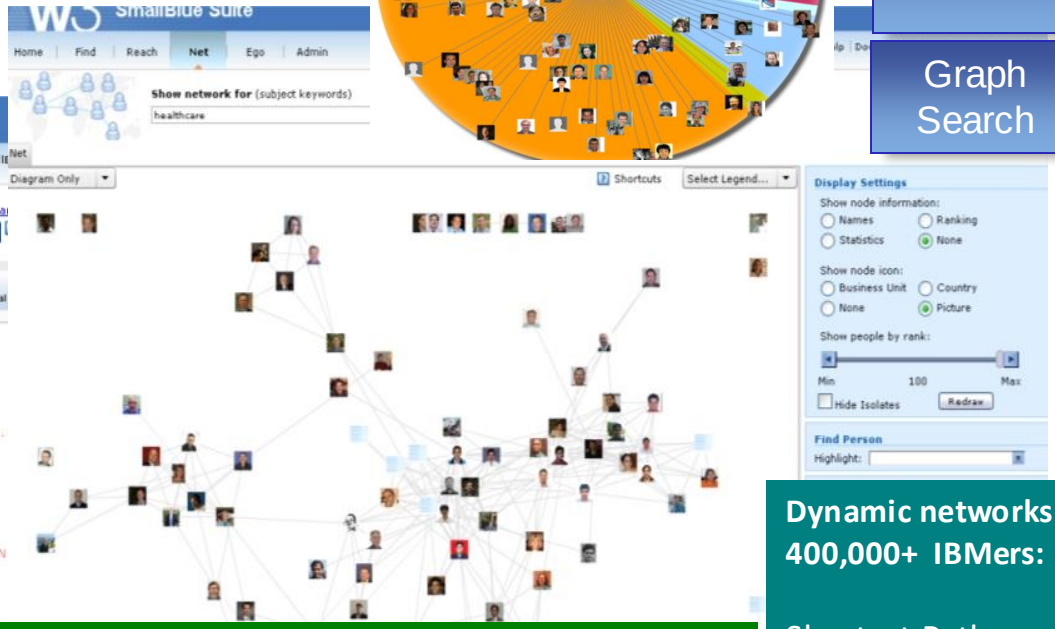


Shortest Paths

Centralities

Graph Search



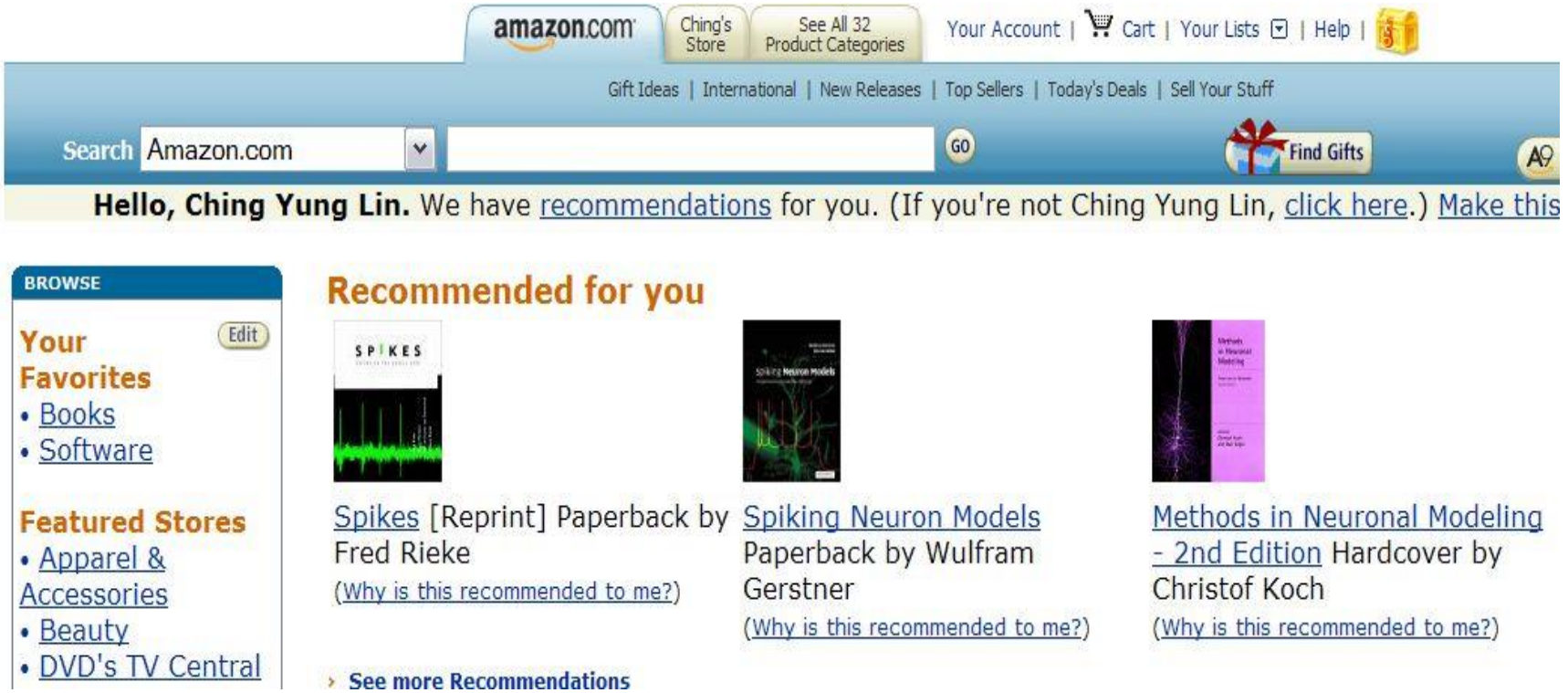


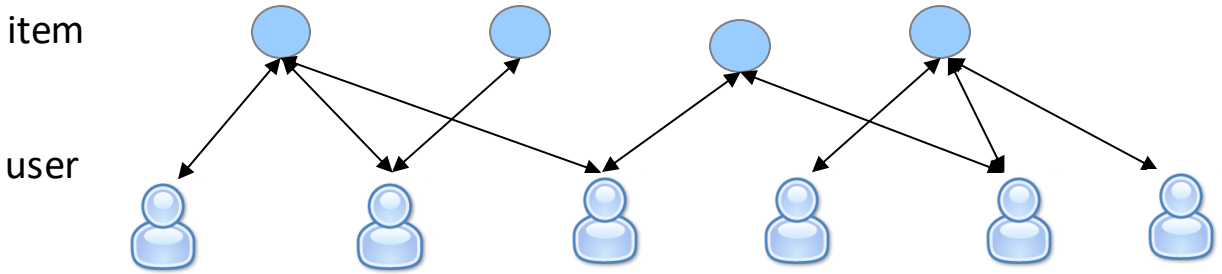
– On BusinessWeek four times, including being the Top Story of Week, April 2009
– Help IBM earned the 2012 Most Admired Knowledge Enterprise Award
– Wharton School study: $7,010 gain per user per year using the tool
– In 2012, contributing about 1/3 of GBS Practitioner Portal $228.5 million savings and benefit
– APQC (WW leader in Knowledge Practice) April 2013:
"The Industry Leader and Best Practice in Expertise Location"

Dynamic networks of 400,000+ IBMers:

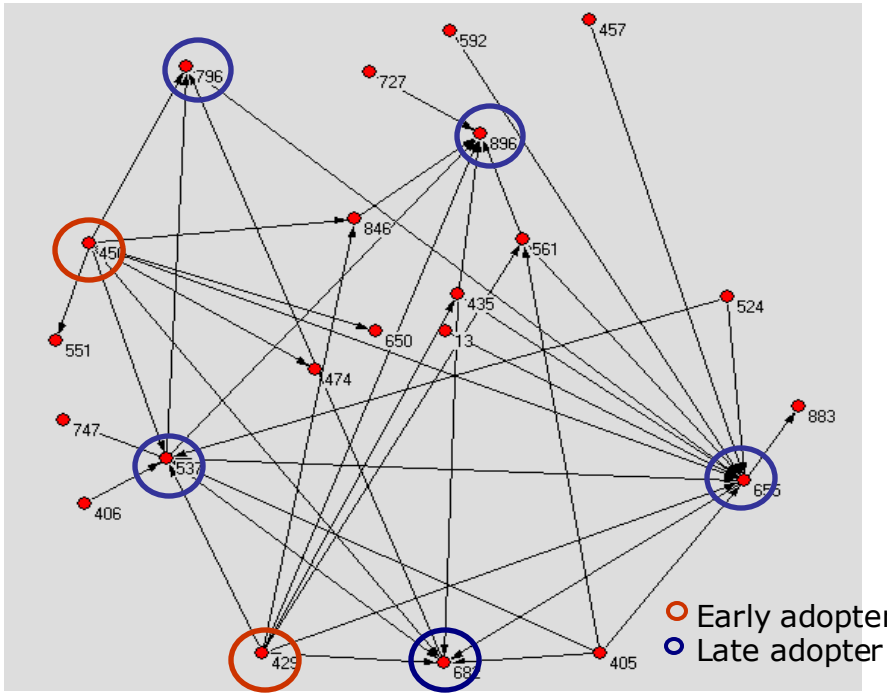
Shortest Paths
Social Capital
Bridges
Hubs
Expertise Search
Graph Search
Graph Recomm.

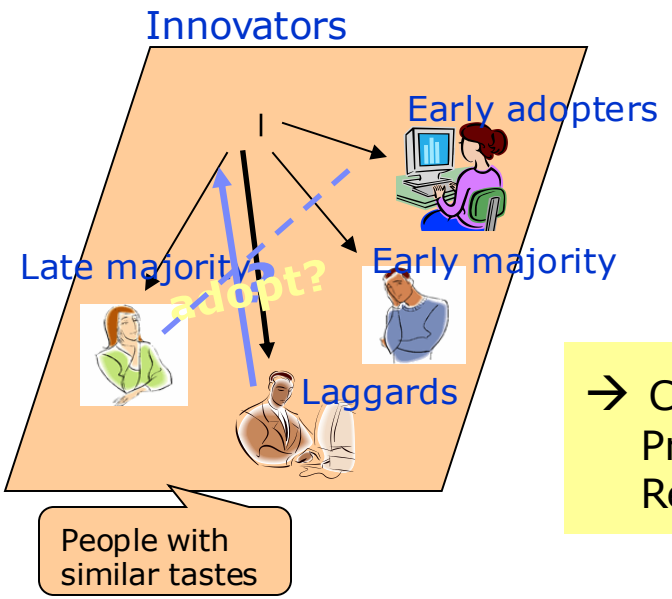
Use Case 2: Recommendation

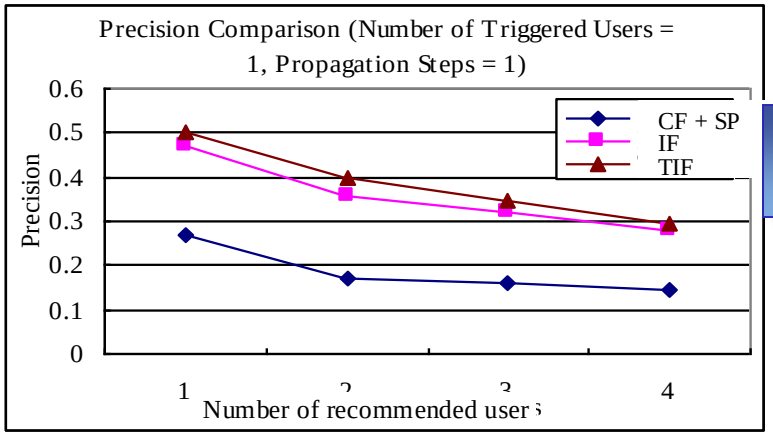


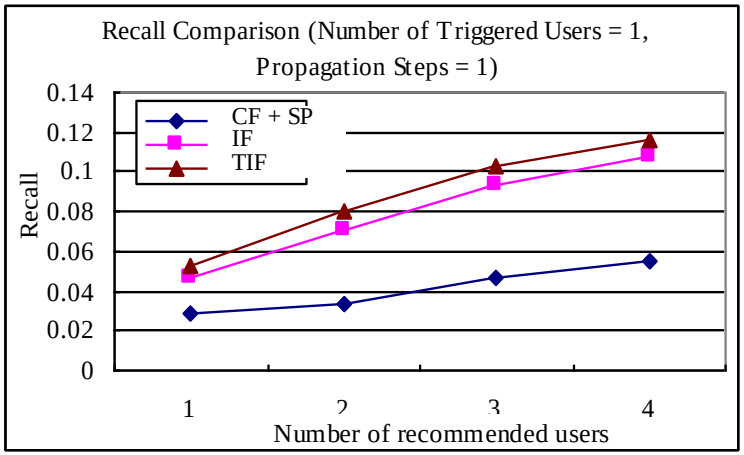


Use Case 3: Recommendation for Commerce









Network Info Flow

Tests:
– 1 month
– 586 new docs
– 1,170 users

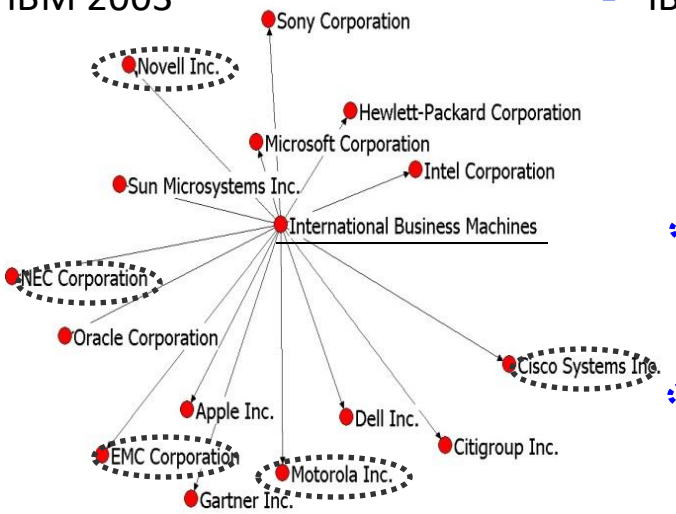
IF: Graphical Information Flow Model
TIF: Joint Topic Detection + Information Flow Model

→ Comparing to Collaborative Filtering (CF) + Similar People
Precision: IF is 91% better, TIF is 108% better
Recall: IF is 87% better, TIF is 113% better

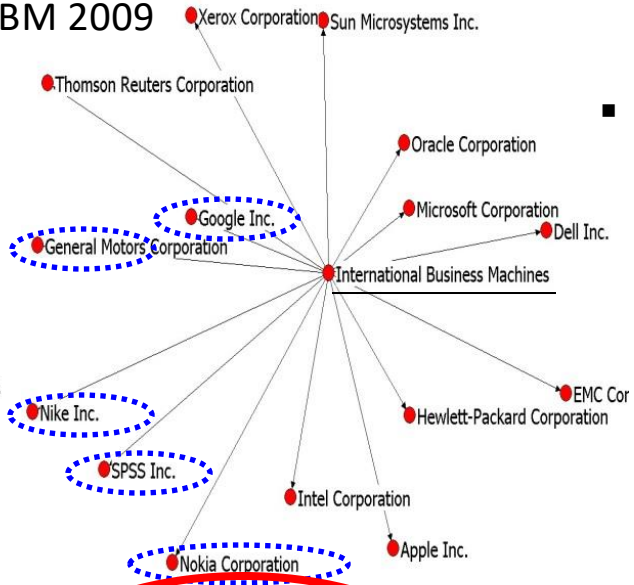
Use Case 4: Graph Analytics for Financial Analysis

Goal: *Injecting Network Graph Effects for Financial Analysis. Estimating company performance considering correlated companies, network properties and evolutions, causal parameter analysis, etc.*

- IBM 2003



- IBM 2009

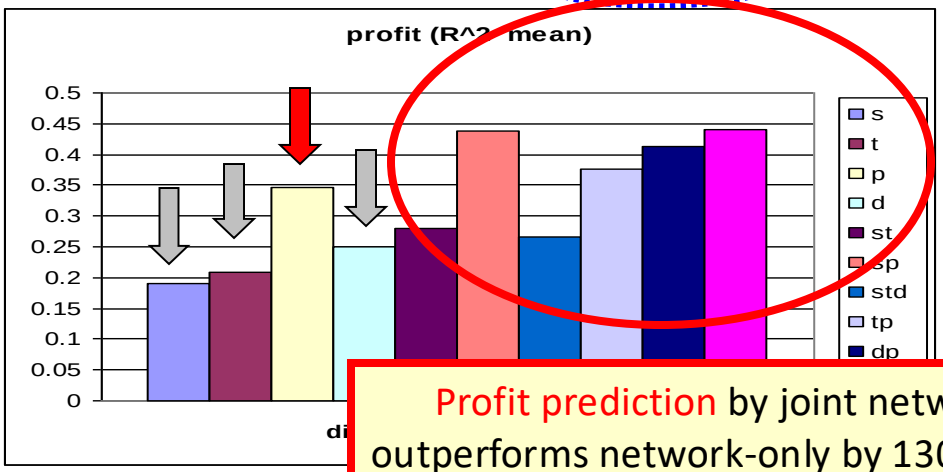


- Data Source:
  - Relationships among 7594 companies, data mining from NYT 1981 ~ 2009

Targets: 20 Fortune companies' normalized Profits

Goal: Learn from previous 5 years, and predict next year

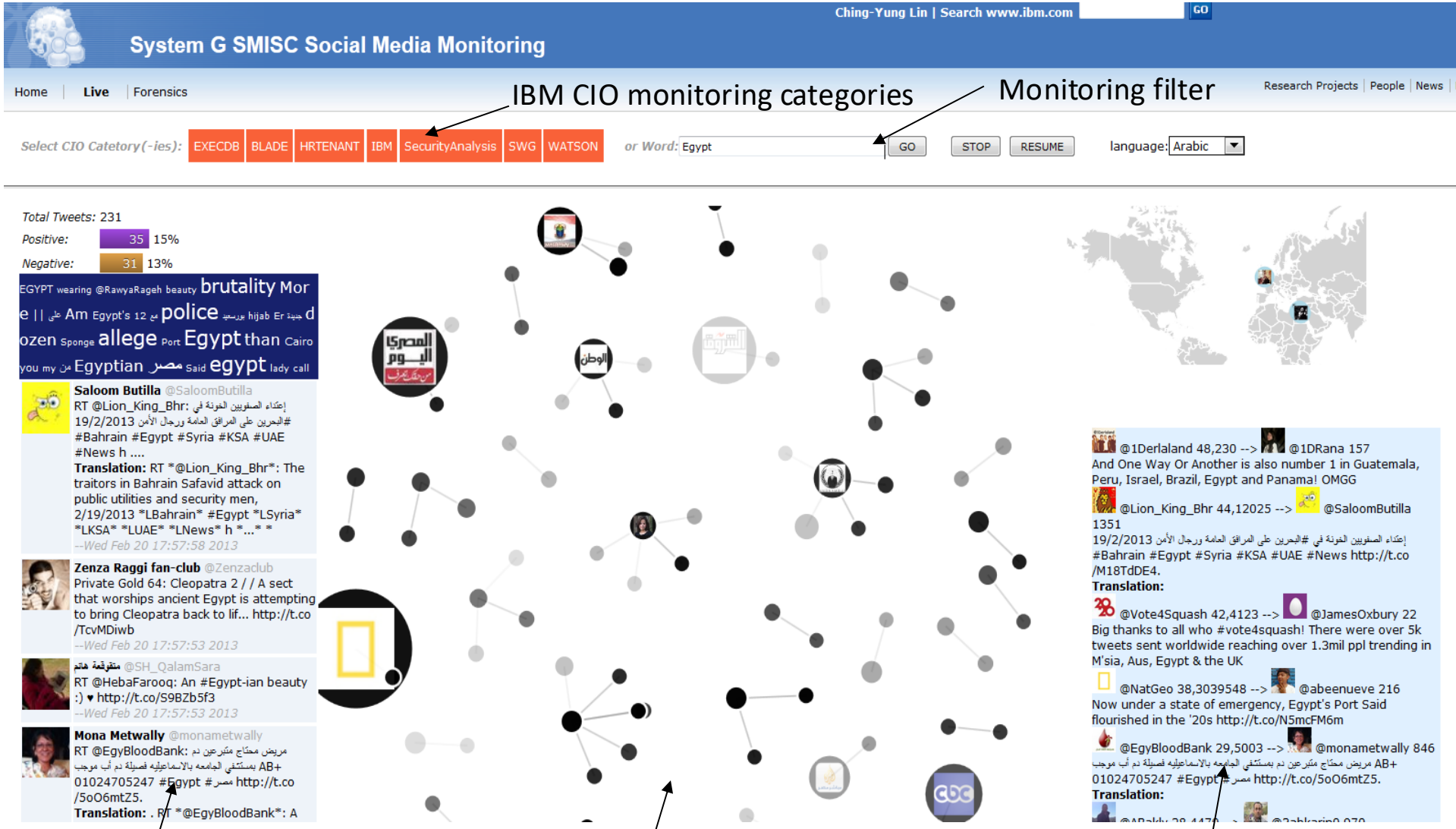
Model: Support Vector Regression (RBF kernel)



Network feature:
s (current year network feature),
t (temporal network feature),
d (delta value of network feature)
Financial feature:
p (historical profits and revenues)

Profit prediction by joint network and financial analysis outperforms network-only by 130% and financial-only by 33%.

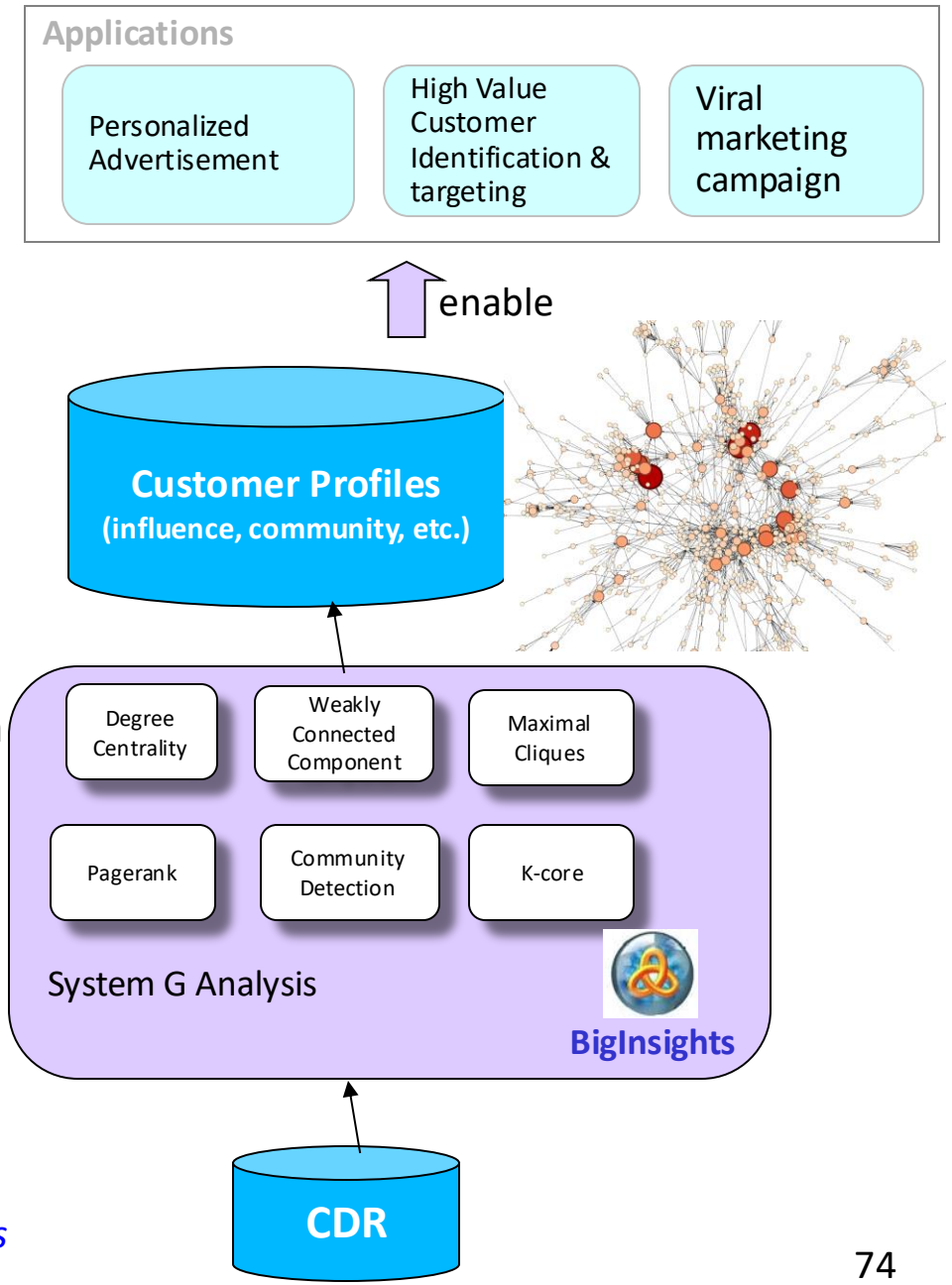
Use Case 5: Social Media Monitoring



Use Case 6: Customer Social Analysis for Telco

Goal: Extract customer social network behaviors to enable Call Detail Records (CDRs) data monetization for Telco.

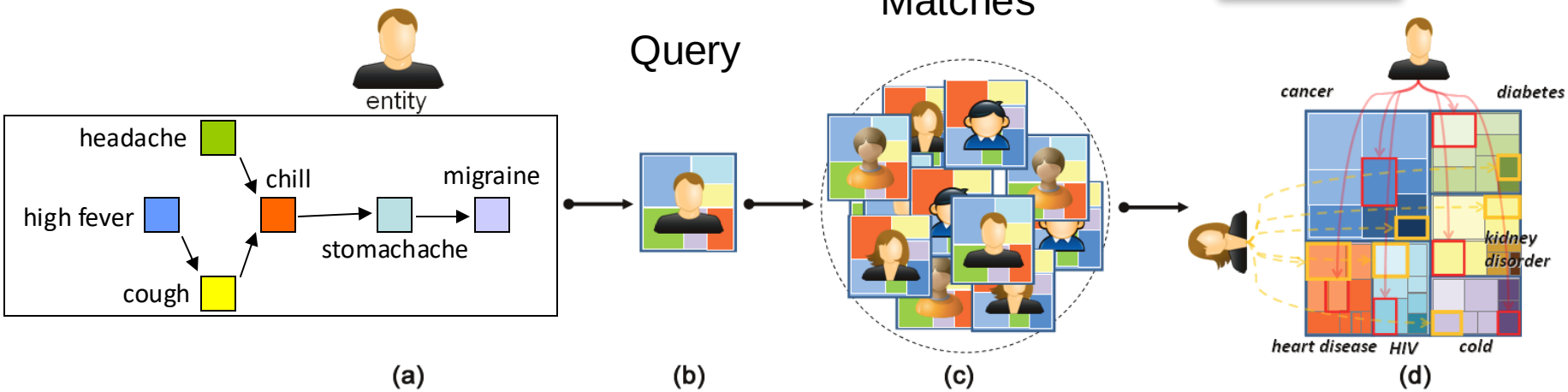
- Applications based on the extracted social profiles
  - Personalized advertisement (beyond the scope of traditional campaign in Telco)
  - High value customer identification and targeting
  - Viral marketing campaign
- Approach
  - Construct social graphs from CDRs based on {caller, callee, call time, call duration}
  - Extract customer social features (e.g., influence, communities, etc.) from the constructed social graph as customer social profiles
  - Build analytics applications (e.g., personalized advertisement) based on the extracted customer social profiles



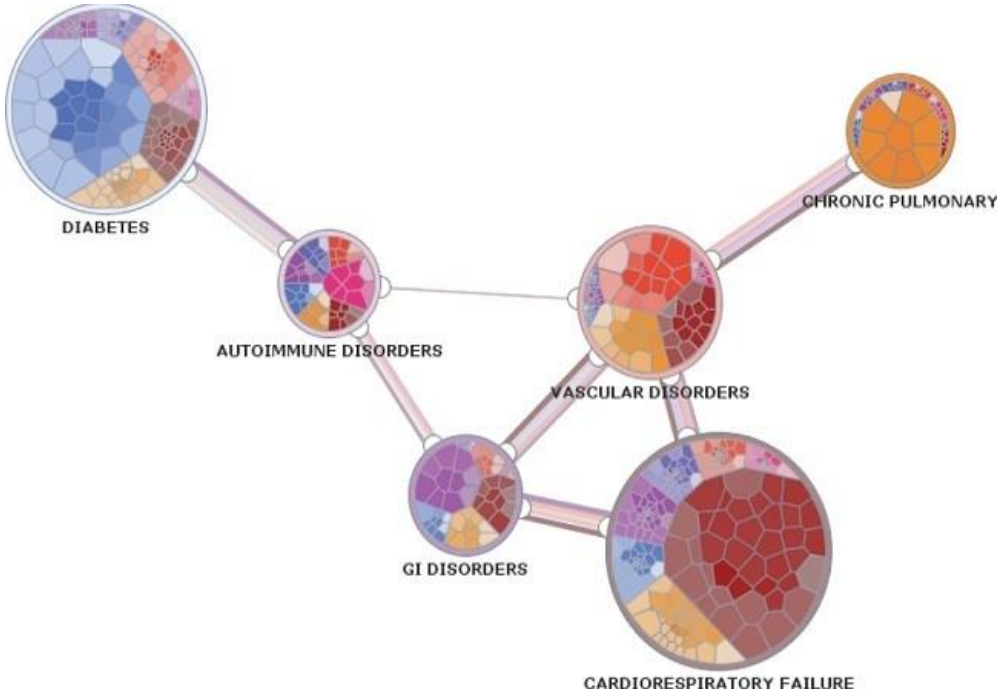
PoCs with Chinese and Indian Telecomm companies

Use Case 7: Graph Analytics and Visualization for Watson

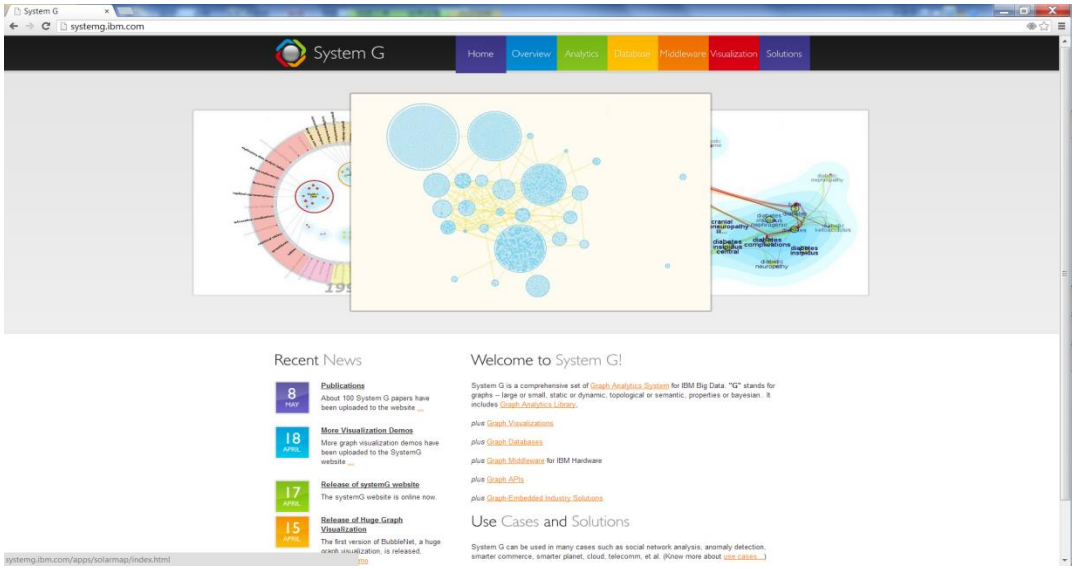
Graph Matching



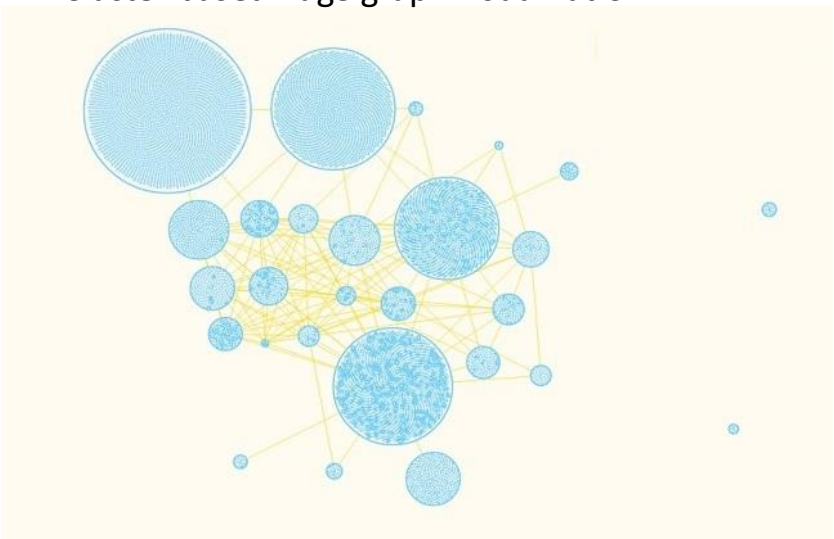
Graph Communities



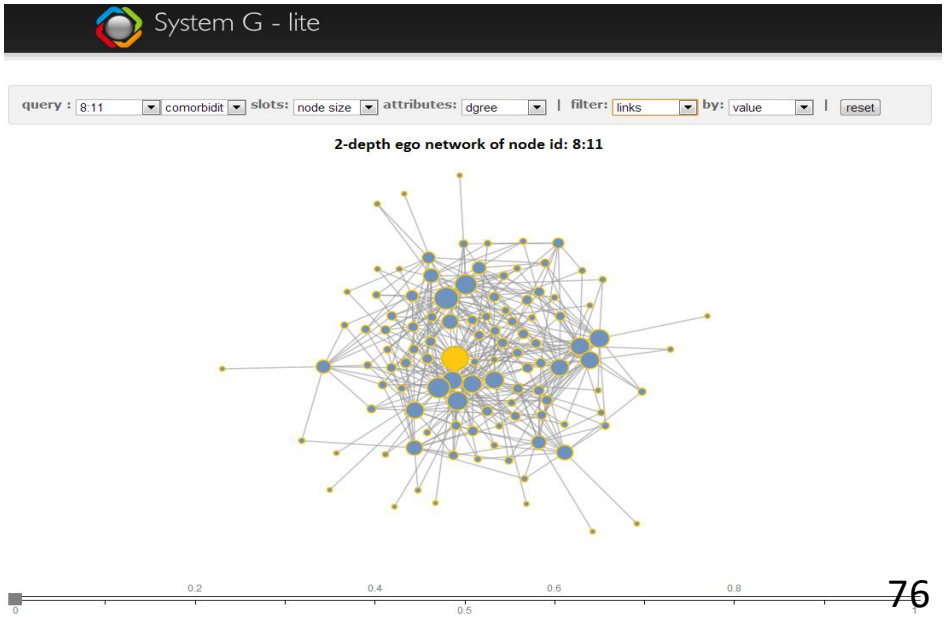
User Case 8: Visualization for Navigation and Exploration



Cluster based huge graph visualization



Query based huge graph visualization

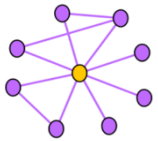


Use Case 9: Graph Search

Graph Search

existing search engine

query



Info-Socio networks

index

ranking

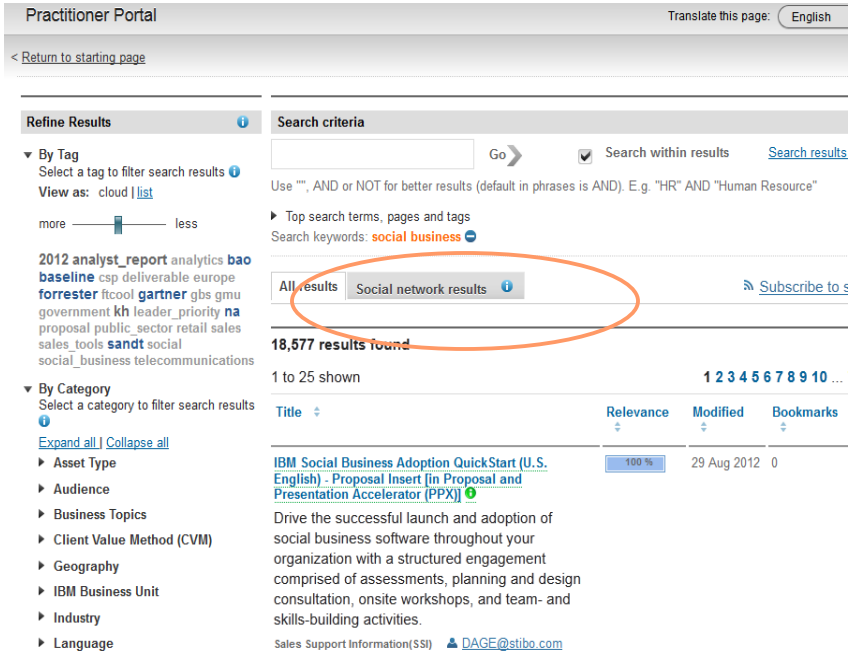
re-ranking

Improved search results

Interest / social network based content recommendations

Graph analysis

query context



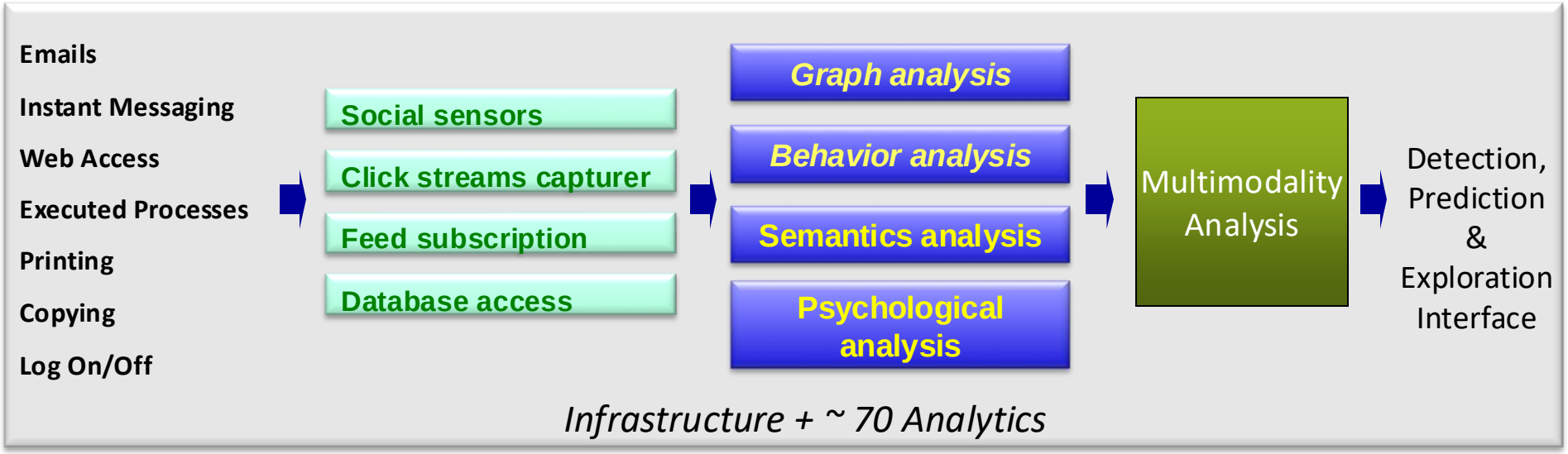
Use Case 10: Anomaly Detection at Multiple Scales

Based on President Executive Order 13587

Goal: System for Detecting and Predicting Abnormal Behaviors in Organization, through large-scale social network & cognitive analytics and data mining, to decrease insider threats such as espionage, sabotage, colleague-shooting, suicide, etc.

"Enterprise Information Leakage Impacted economy and jobs" Feb 2013

"What's emerged is a multibillion dollar detective industry"
npr Jan 10, 2013

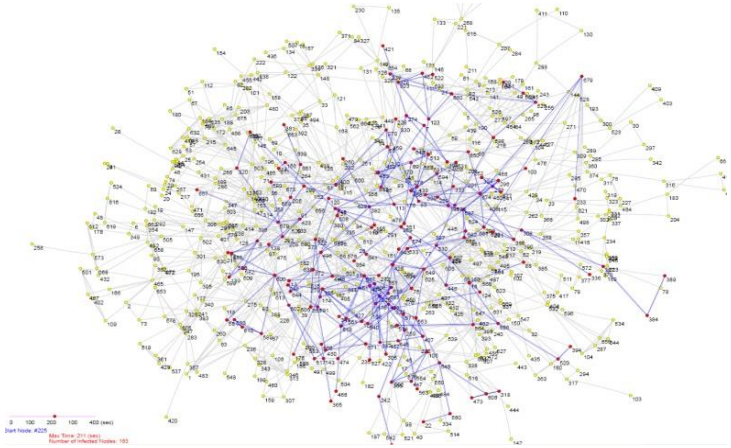


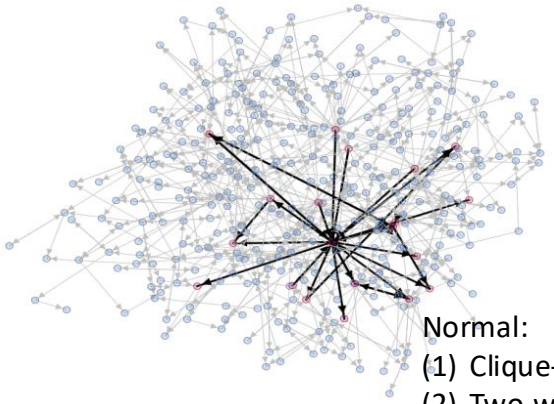
Use Case 11: Fraud Detection for Bank

Network Info Flow

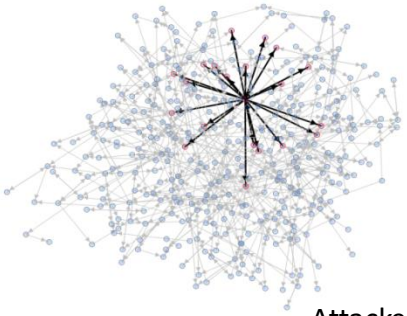
Ego Net Features

Ponzi scheme Detection

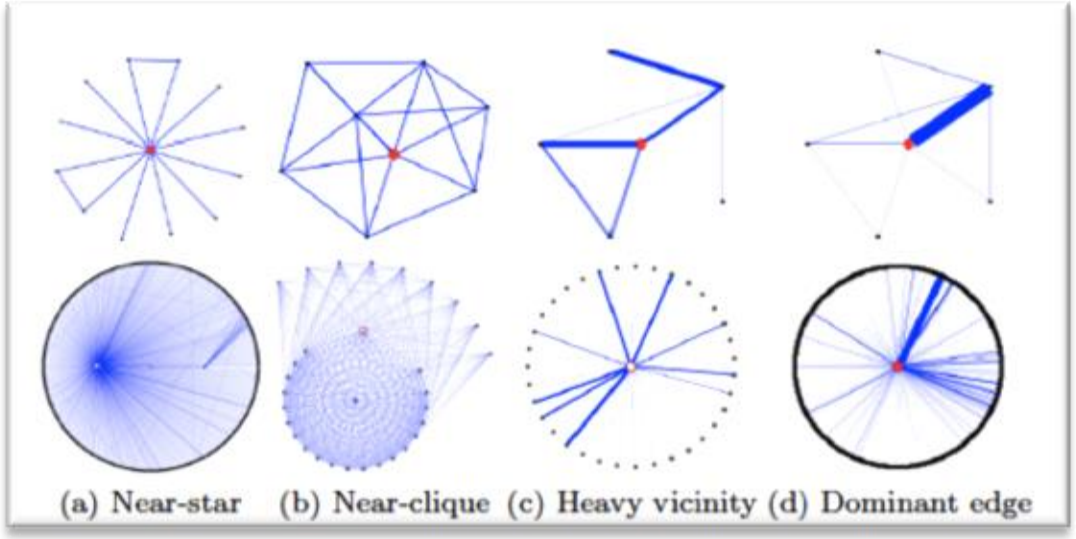




Normal:
(1) Clique-like
(2) Two-way links



Attacker:
Near-Star

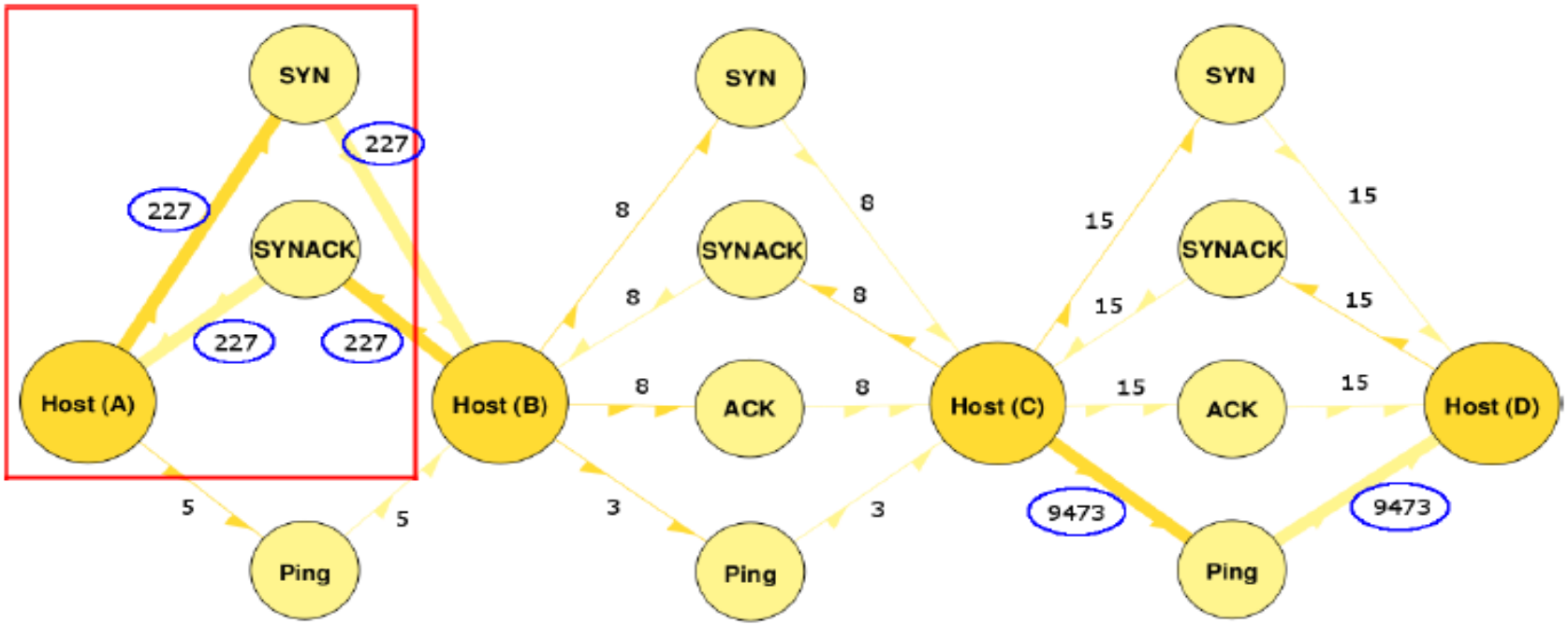


Use Case 12: Detecting Cyber Attacks

Network Info Flow

Ego Net Features

Detecting DoS attack



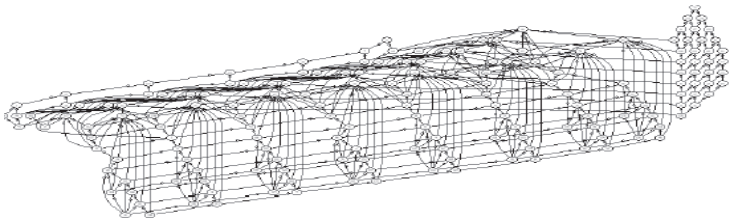
(a) Single large graph representing TCP SYN and ICMP PING network traffic, with two Denial of Service (DoS) attacks taking place.

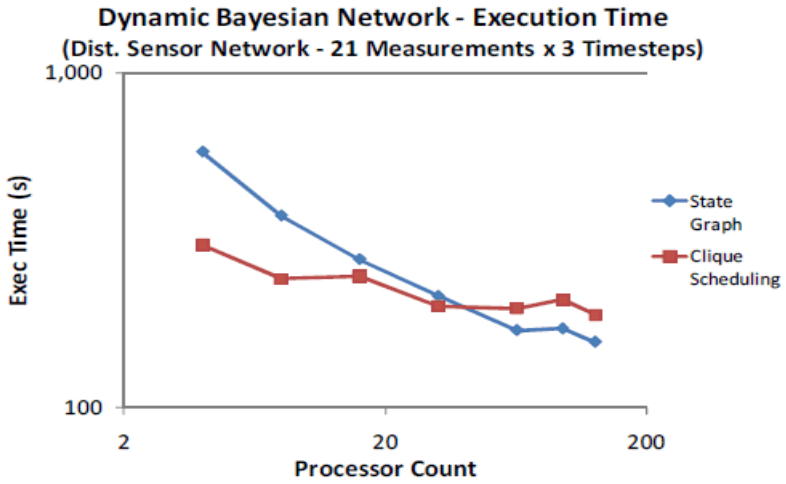
Use Case 13: Smarter *another* Planet

Bayesian Network

Goal: Atmospheric Radiation Measurement (ARM) climate research facility provides *24x7 continuous field observations* of cloud, aerosol and radiative processes. **Graphical models** can automate the validation with improvement efficiency and performance.

Approach: BN is built to represent the dependence among sensors and replicated across timesteps. BN parameters are learned from over *15 years* of ARM climate data to support distributed climate sensor validation. Inference validates sensors in the connected instruments.





Bayesian Network

* 3 timesteps * 63 variables
* 3.9 avg states * 4.0 avg indegree
* 16,858 CPT entries

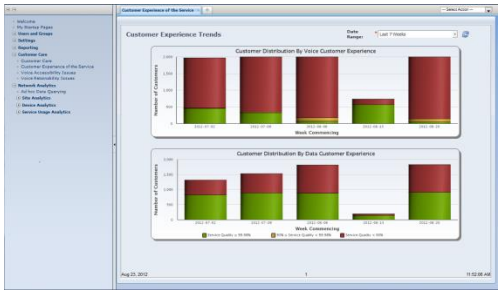
Junction Tree

* 67 cliques
* 873,064 PT entries in cliques

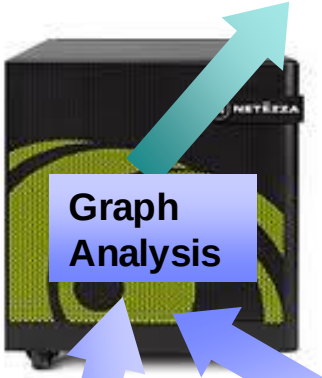
Use Case 14: Cellular Network Analytics in Telco Operation

Goal: Efficiently and uniquely identify *internal* state of Cellular/Telco networks (e.g., performance and load of network elements/links) using probes between monitors placed at selected network elements & endhosts

- Applied Graph Analytics to telco network analytics based on CDRs (call detail records): estimate traffic load on CSP network with low monitoring overhead
  - CDRs, already collected for billing purposes, contain information about voice/data calls
  - Traditional NMS* and EMS** typically lack of end-to-end visibility and topology across vendors
  - Employ graph algorithms to analyze network elements which are not reported by the usage data from CDR information
- Approach
  - Cellular network comprises a hierarchy of network elements
  - Map CDR onto network topology and infer load on each network element using graph analysis
  - Estimate network load and localize potential problems

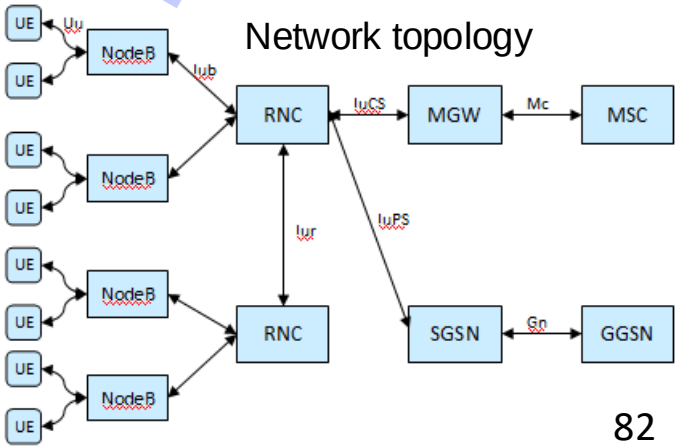


Network load level report



CDR

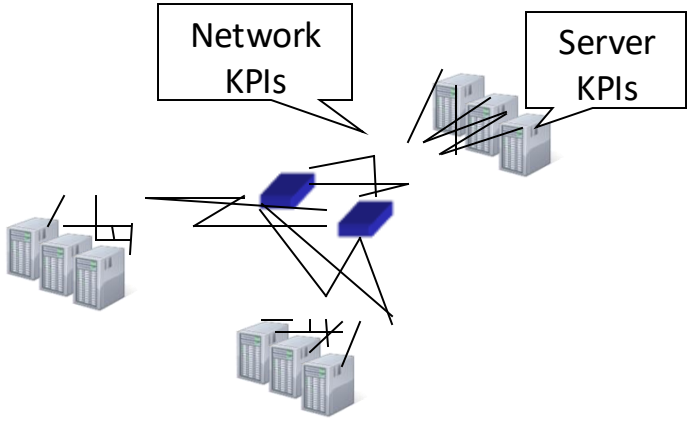
Network topology



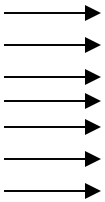
Use Case 15: Monitoring Large Cloud

Goal: *M*onitoring technology that can track the time-varying state (e.g., causality relationships between KPIs) of a large Cloud when the processing power of monitoring system cannot keep up with the scale of the system & the rate of change

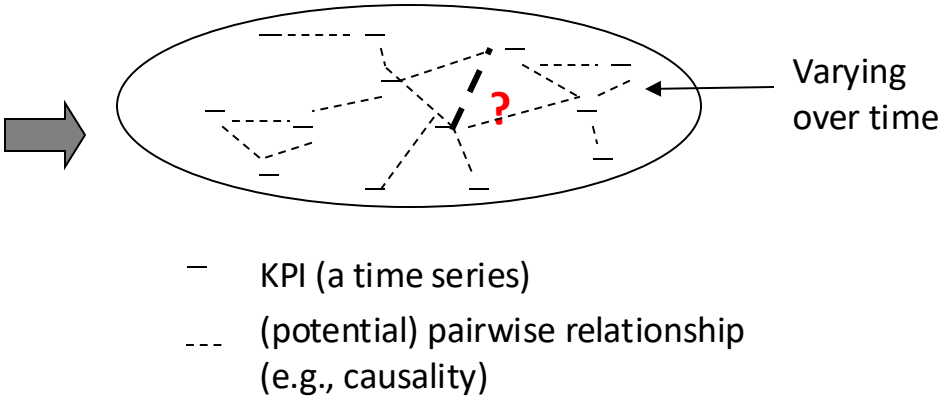
- *Causality relationships (e.g., Granger causality) are crucial in performance monitoring & root cause analysis*
- *Challenge: easy to test pairwise relationship, but hard to test multi-variate relationship (e.g., a large number of KPIs)*



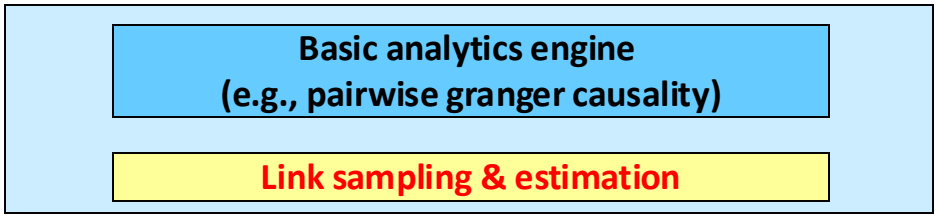
KPI time series (e.g., server performance/load, network performance/load)



Causality analyzer

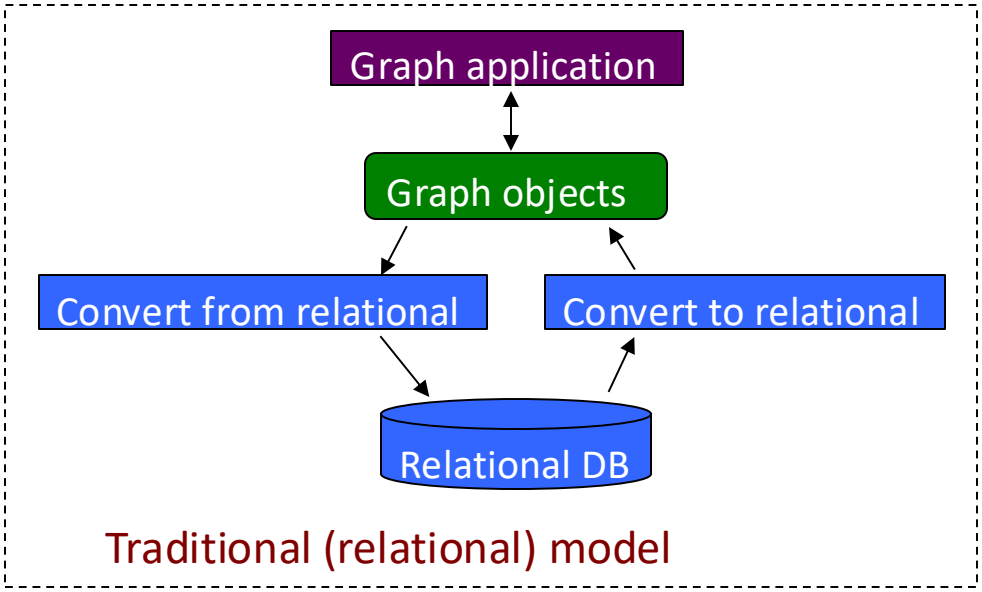


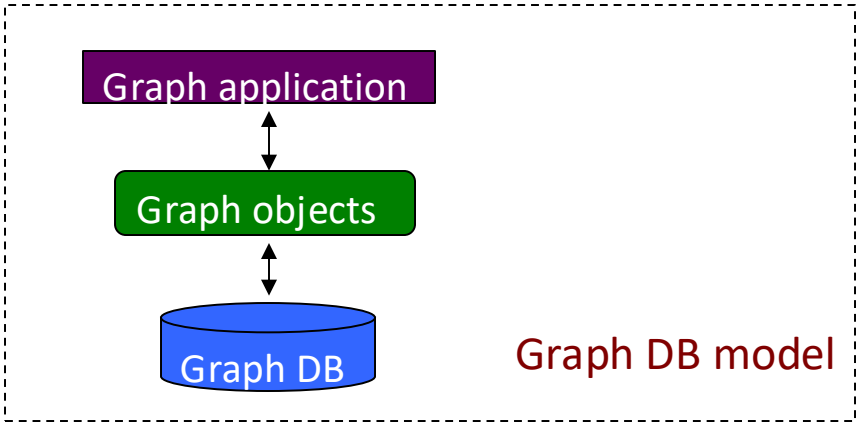
Our approach: Probabilistic monitoring via sampling & estimation



Select KPI pairs (sampling)→ Test link existence → Estimate unsampled links based on history → Overall graph

Use Case 16: Code Life Cycle Improvement





- Advantages of working directly with graph DB for graph applications
  - Smaller and simpler code
  - Flexible schema → easy schema evolution
  - Code is easier and faster to write, debug and manage
  - Code and Data is easier to transfer and maintain

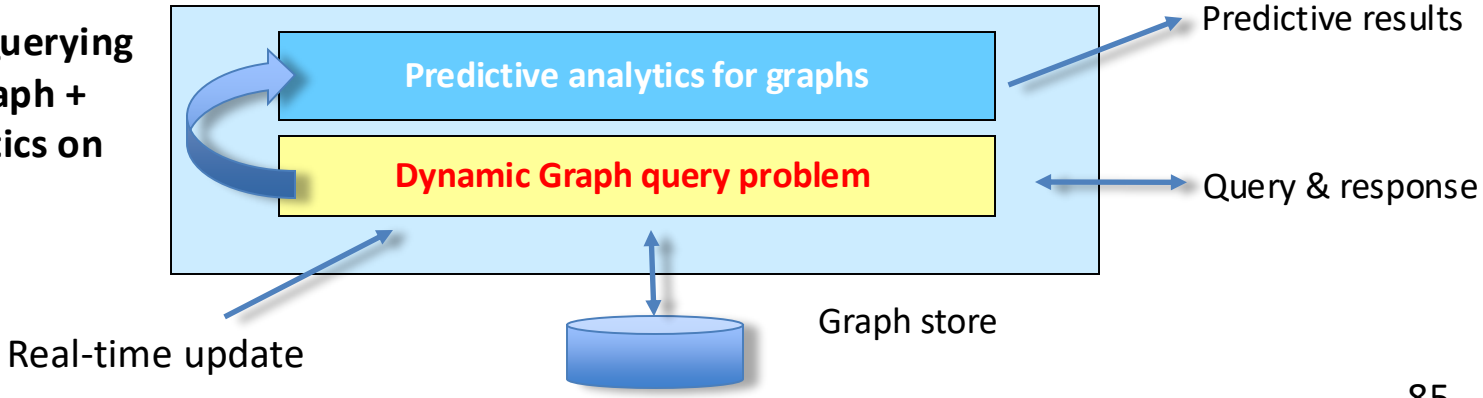
Use Case 17: Smart Navigation Utilizing Real-time Road Information

Goal: *Enable unprecedented level of accuracy in **traffic scheduling** (for a fleet of transportation vehicles) and navigation of individual cars utilizing the **dynamic real-time information** of changing road condition and predictive analysis on the data*

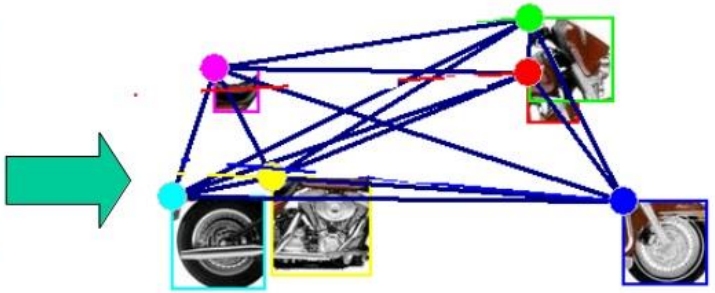
- *Dynamic graph algorithms implemented in System G provide **highly efficient graph query computation** (e.g. shorted path computation) on time-varying graphs (order of magnitudes improvement over existing solutions)*

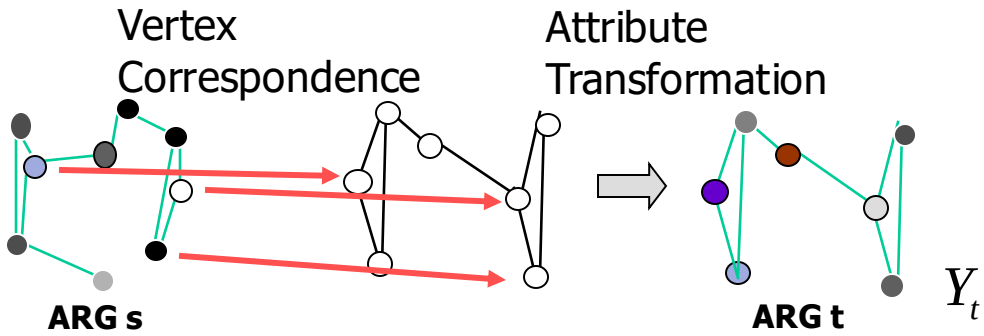
- *High-throughput **real-time predictive analytics** on graph makes it possible to estimate the future traffic condition on the route to make sure that the decision taken now is optimal overall*

Our approach: Querying over dynamic graph + predictive analytics on graph properties



Use Case 18: Graph Analysis for Image and Video Analysis





Use Case 19: Graph Matching for Genomic Medicine

- Ongoing discussions

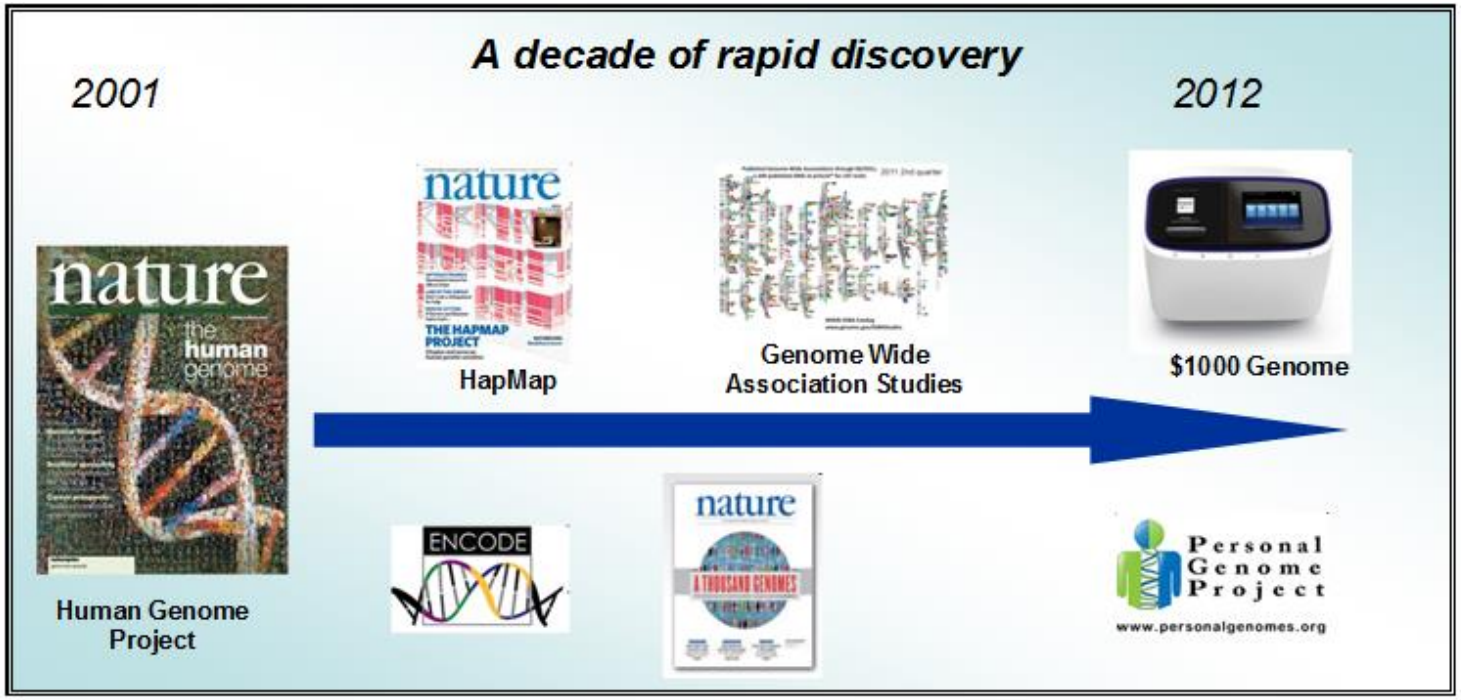
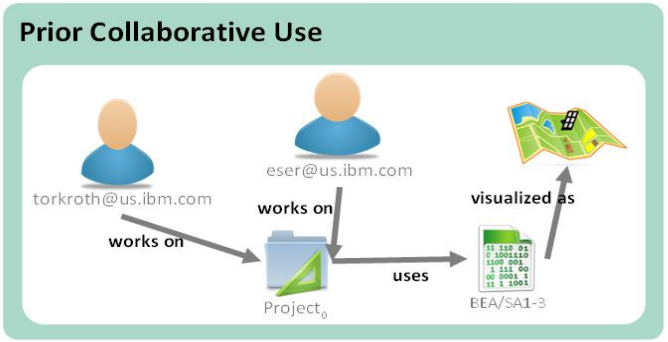
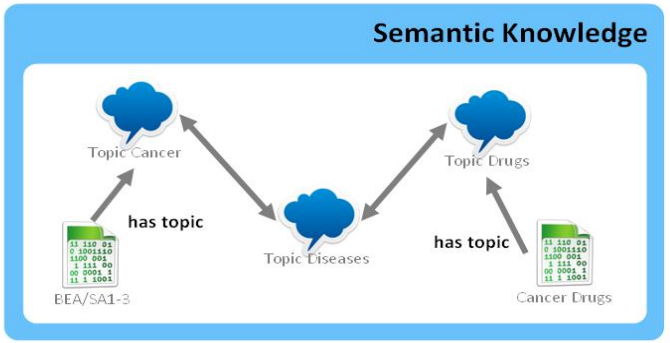
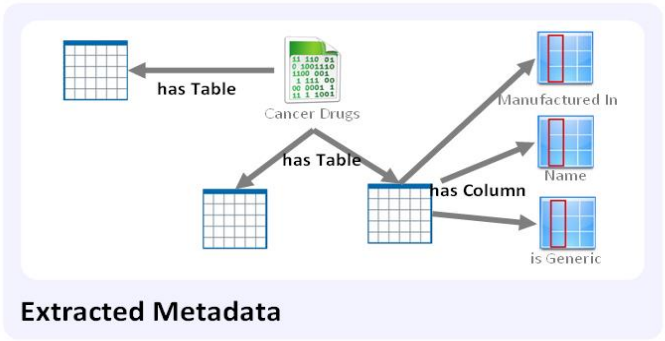


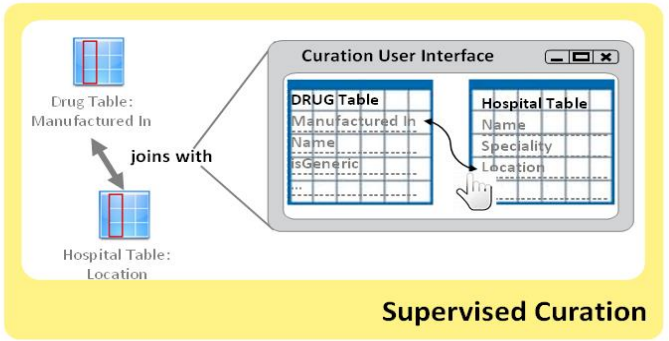
Figure 1: Since the Human Genome Project, various projects have started to reveal the mysteries of genomes and the $1000 Genome is almost reality.

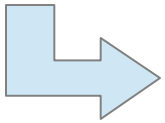
Use Case 20: Data Curation for Enterprise Data Management

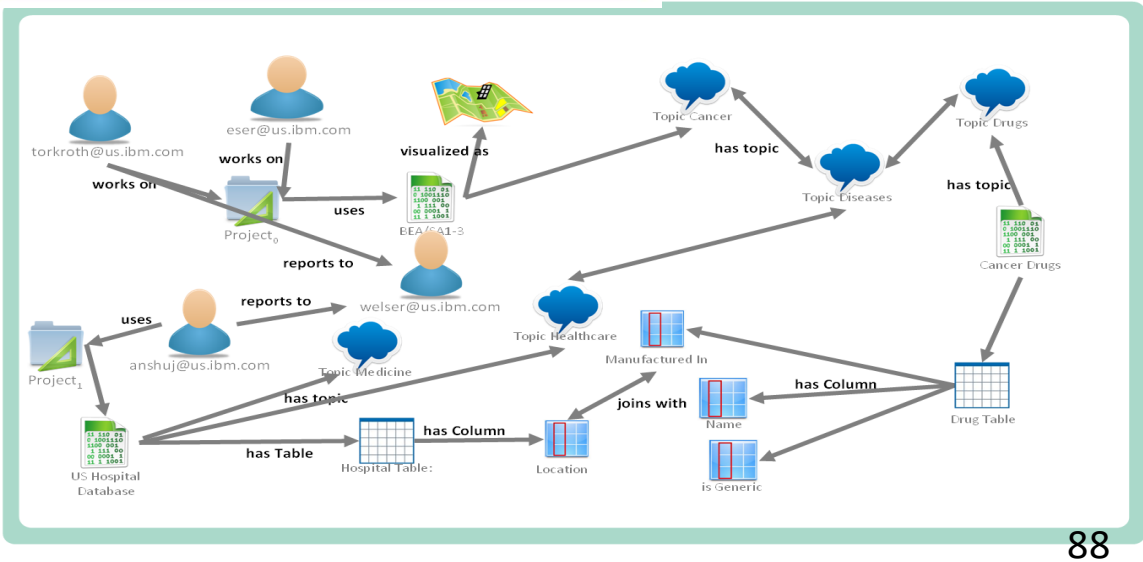




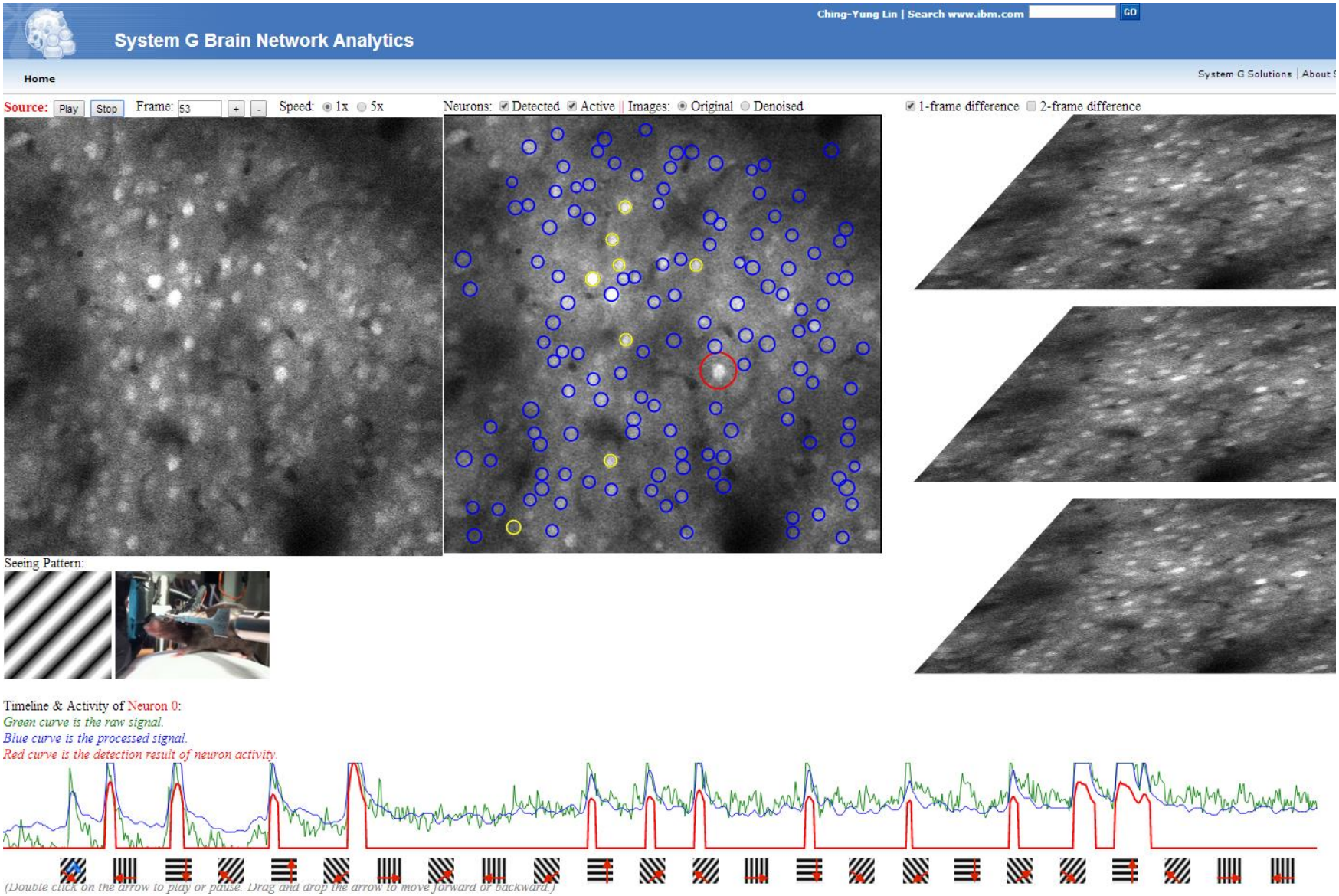






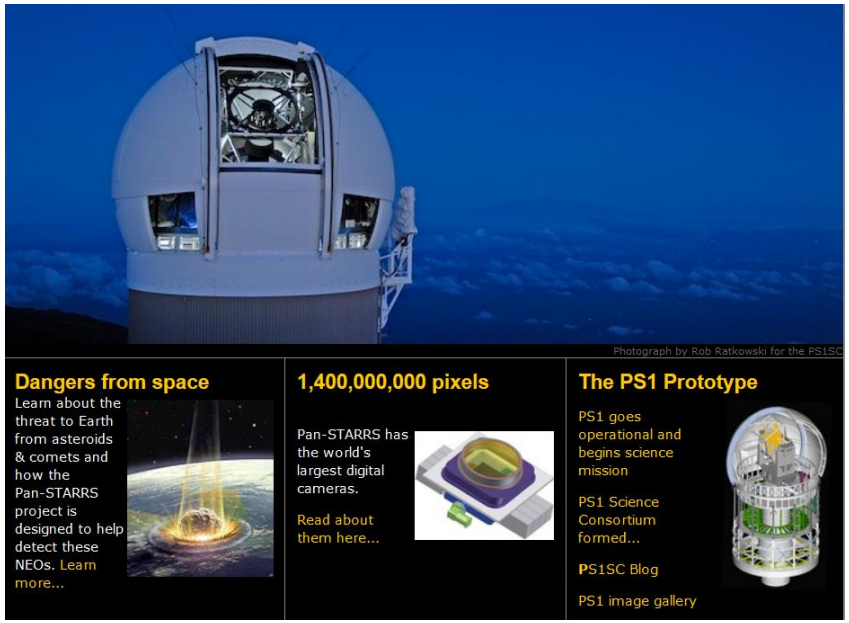


Use Case 21: Understanding Brain Network

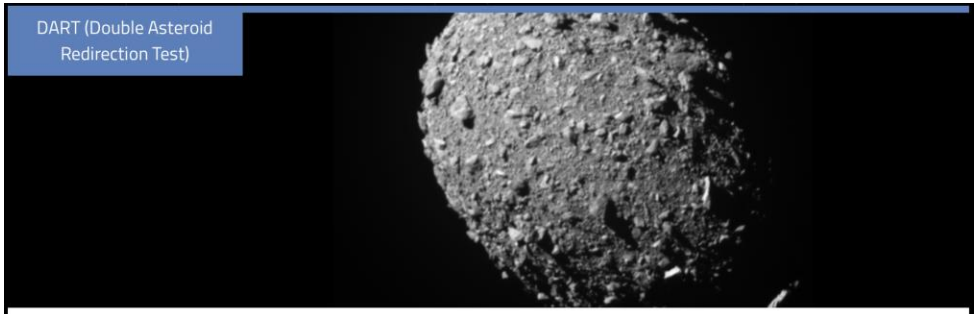


Use Case 22: Planet Security

- Big Data on Large-Scale Sky Monitoring



NASA's DART Mission Hits Asteroid in First-Ever Planetary Defense Test



Sep 26, 2022
RELEASE 22-100

NASA's DART Mission Hits Asteroid in First-Ever Planetary Defense Test

After 10 months flying in space, NASA's Double Asteroid Redirection Test (DART) – the world's first planetary defense technology demonstration – successfully impacted its asteroid target on Monday, the agency's first attempt to move an asteroid in space.

Mission control at the Johns Hopkins Applied Physics Laboratory (APL) in Laurel, Maryland, announced the successful impact at 7:14 p.m. EDT.

As a part of NASA's overall planetary defense strategy, DART's impact with the asteroid Dimorphos demonstrates a viable mitigation technique for protecting the planet from an Earth-bound asteroid or comet, if one were discovered.

"At its core, DART represents an unprecedented success for planetary defense, but it is also a mission of unity with a real benefit for all humanity," said NASA Administrator Bill Nelson. "As NASA studies the cosmos and our home planet, we're also working to protect that home, and this international collaboration turned science fiction into science fact, demonstrating one way to protect Earth."

DART targeted the asteroid moonlet Dimorphos, a small body just 530 feet (160 meters) in diameter. It orbits a larger, 2,560-foot (780-meter) asteroid called Didymos. Neither asteroid poses a threat to Earth.

https://www.nbcnews.com/video/nasa-s-dart-spacecraft-crashes-into-asteroid-149320773570

Questions?