Enterprise Software
Big Data Analytics

Rakesh Ranjan
Rakesh.ranjan@sjsu.edu

Some slides are adopted from IBM Research and BigData group presentations.

Facebook Information Platform

By Jeff Hammerbacher in "Beautiful Data"

Our first attempt at an offline repository of information involved a Python script for farming queries out to Facebook's tier of MySQL servers and a daemon process, written in C++, for processing our event logs in real time. When the scripts worked as planned, we collected about 10 gigabytes a day. I later learned that this aspect of our system is commonly termed the "ETL" process, for "Extract, Transform, and Load."

Once our Python scripts and C++ daemon had siphoned the data from Facebook's source systems, we stuffed the data into a MySQL database for offline querying. We also had some scripts and queries that ran over the data once it landed in MySQL to aggregate it into more useful representations. It turns out that this offline database for decision support is better known as a "Data Warehouse."

Facebook Information Platform

By Jeff Hammerbacher in "Beautiful Data"

Finally, we had a simple PHP script to pull data from the offline MySQL database and display summaries of the information we had collected to internal users. For the first time, we were able to answer some important questions about the impact of certain site features on user activity. Early analysis looked at maximizing growth through several channels: the layout of the default page for logged-out users, the source of invitations, and the design of the email contact importer. In addition to analysis, we started to build simple products using historical data, including an internal project to aggregate features of sponsored group members that proved popular with brand advertisers.

I didn't realize it at the time, but with our ETL framework, Data Warehouse, and internal dashboard, we had built a simple "Business Intelligence" system.

Modern day data challenges - Facebook story

- Facebook continues to run on MySQL (1 terabyte) until one day mysqld crashed. Recovery took 3 days
- Facebook started migrating to Oracle (few TB of warehouse)
- Clickstream logging – 400 GB of unstructured data on first day
- Facebook developed a parallelized clickstream processing system called "Cheetah" – dumped in favor of Hadoop
- 2005 – Apache Hadoop project started by by Doug Cutting and Mike Cafarella
- 2006 – Yahoo hired Cutting and spent considerably on improving Hadoop. Hadoop was able to sort 1.9 TB in 47 hrs. using 188 servers.
- 2008 – Hadoop was able to sort 1 TB in 209 seconds using 901 servers.
- 2008 – Facebook implements its first 60 node Hadoop cluster, later they moved transactional data collection to Hadoop as well. At its current capacity, the cluster holds nearly 2.5 petabytes of data, and new data is added at a rate of 15 terabytes per day. Over 3,000 MapReduce jobs are run every day, processing 55 terabytes of data.
- 2015 – Facebook adds 4 petabytes of data everyday

Facebook's Top Open Data Problems

- Small Data Challenges
  - Mobile devices – push vs pull information
  - Reducing replication (round trips between application and data layer)
- Big Data Challenges
  - Trading storage spaces with CPU
  - Reliability of pipelines
  - Globally distributed data ware house (Can the query optimizer choose where to replicate data and where to run the query?)

https://research.facebook.com/blog/facebook-s-top-open-data-problems/

Big Data Customer Use Cases

Source: IDC, 2012

Big Data Analytics problems in the Healthcare

1. Predict high-risk populations and begin early interventions for wellness management
2. App for evidence-based medicine, facilitating the practice of personalized medicine
3. View and analyze aggregated data sets by specific populations for cohort management, disease registries, clinical guidelines and patient safety
4. Provide security-rich and quick access to aggregated information and deliver the right information to the right people at the right time

BigData use case: Log Analysis

Transaction error analysis

- **Use scenario**
  - Using a federated system of applications to provide account status and related information for customers. The federated application approach works very well in terms of ensuring near real time response to customers when they use the portal.
  - There are occasions, however, when there are errors in transactions and customers call in with issues as soon as they occur.
- **Pain points**
  - The current system of tracking these issues requires help desk analysts to look at the internal log portal and find out what happened. This usually takes about more than ***24 hours*** to respond to the customer
  - The log portal depends on a system that collects log data from various application systems and aggregates them.
    - The logs have different formats and correlating errors across applications can be tedious and error prone.
    - The bank tried to use a warehousing approach but could not succeed since the cost of such a solution was prohibitive and the response time demands could not be met.
- **Requirement**
  - Need a cost-effective solution to reduce the response time to ***2 minutes***

BigData use case: Piracy of Video Streams

- Use scenario
  - MBC is facing a global increase in the unauthorized live streaming and re-broadcast of its content. These illegal sites allow users to either directly produce/watch live streaming video, link to other streaming sites, or share streams through P2P software.
  - MBC also loses valuable content due to hijacking of any related comments / posts / conversations
  - Not only do these illegal sites represent a potential revenue loss but also a potential threat to MBC's overall business model and future growth.
  - Although MBC knows these illegal sites exist, it does not have an analytical based assessment of how big the problem is, how the market is changing, how to quantify business impact or what the best options are to address this problem. They need the insights into the problem they have.

- Requirement
  - How to analyze large volume of unstructured data. Capture data from multiple sources. Analyze the scope and potential impact of unauthorized streaming and deliver strategic recommendations on how to best confront theses pirating entities and overall trends.
    - How prevalent is the streaming of unauthorized content?
    - What are the major URLs, platforms and where are these sites located?
    - How are users discovering unauthorized content?
    - How can usage trends be identified?
    - What factors are driving consumer participation (e.g. sport, event, schedule, geographic location, access, etc.)?

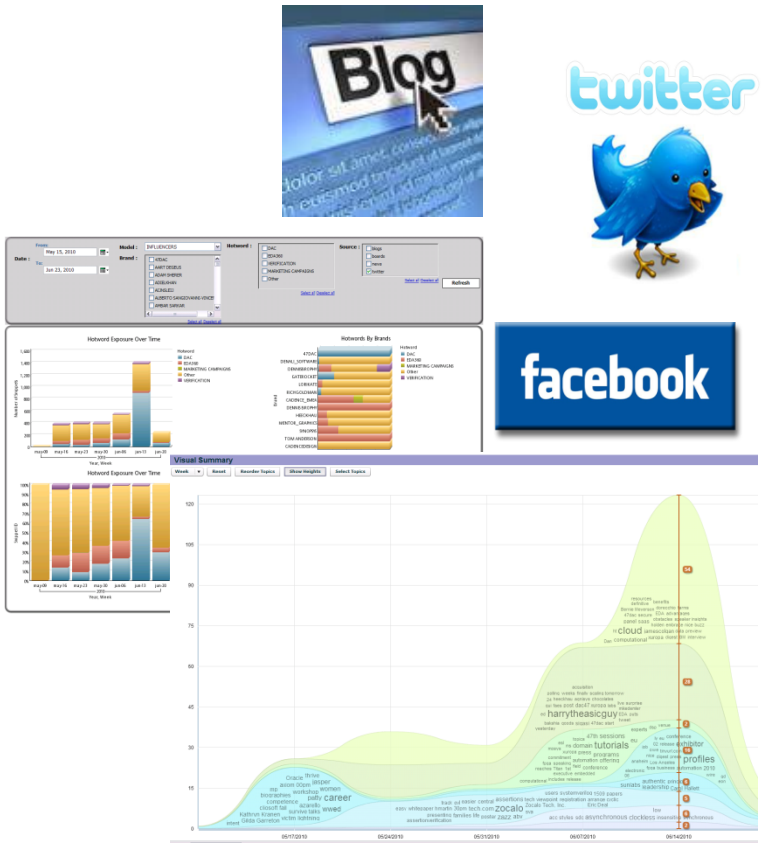
Sentiment Analysis for Products, Services and Brands

Scenario

- Monitor data from various sources such as blogs, boards, news feeds, tweets, and social medias for information pertinent to brand and products, as well as competitors

Requirement

- Extract and aggregate relevant topics, relationships, discover patterns and reveal up-and-coming topics and trends



Customer Acquisition and Retention

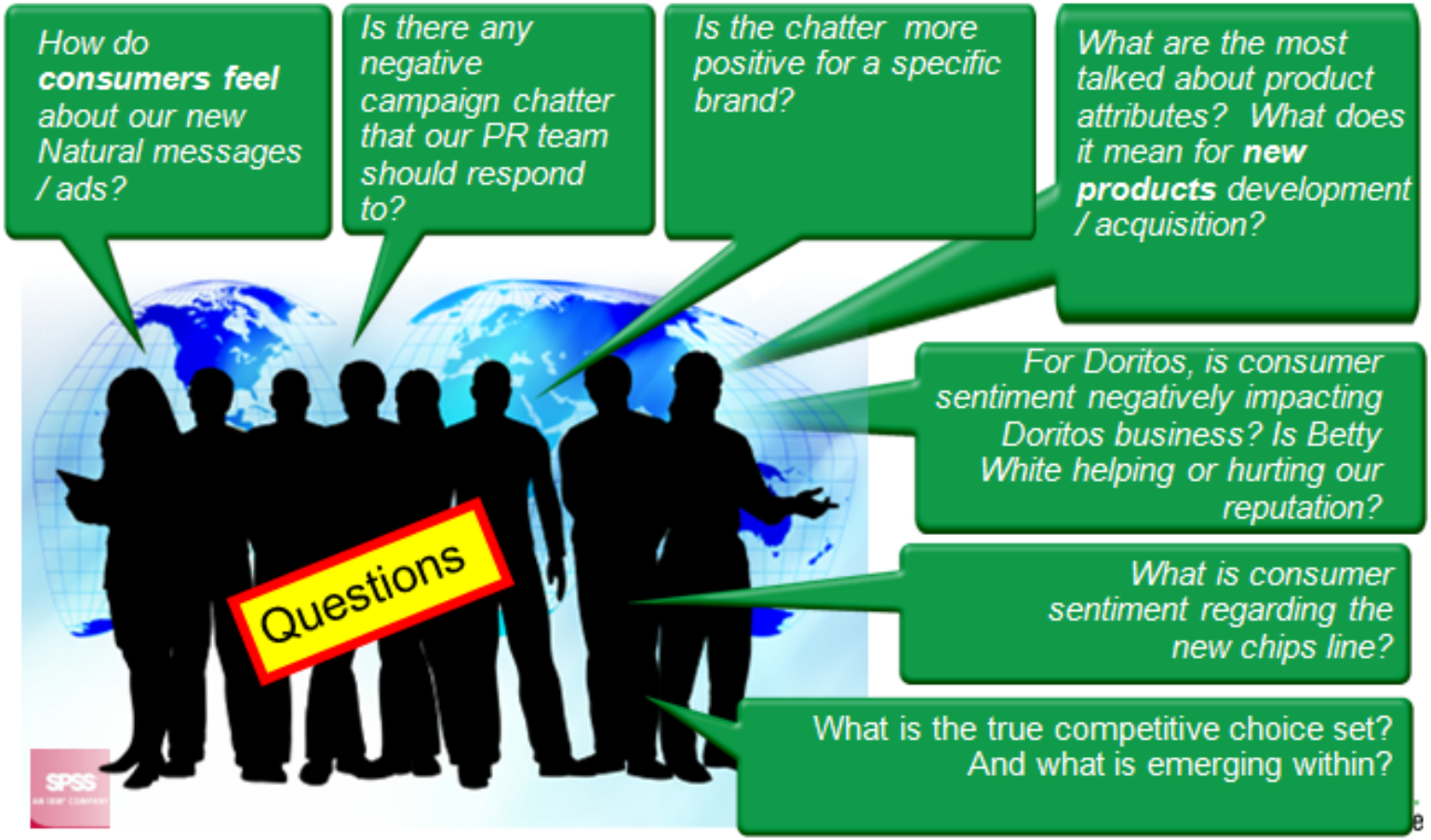
Scenario

- Reconcile what business know about a customer's behavior in physical stores with web stores
- Take action based on insights to enable new levels of customer services

Requirement

- Weblog and click-stream analysis
- Integrated view between behavior data and transaction histories

Sentiment Analysis



The World is Changing and Becoming More…

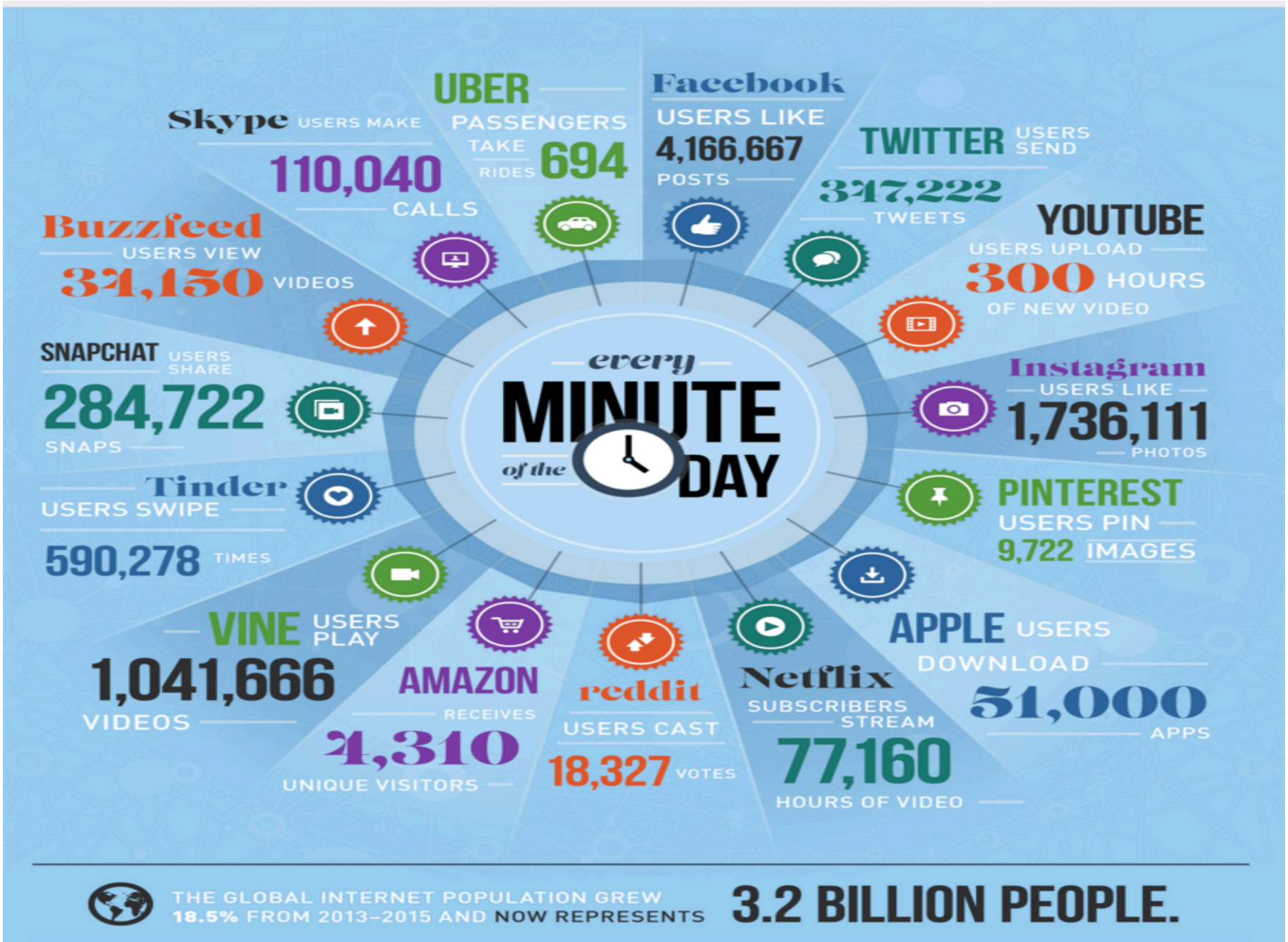
INSTRUMENTED

INTERCONNECTED

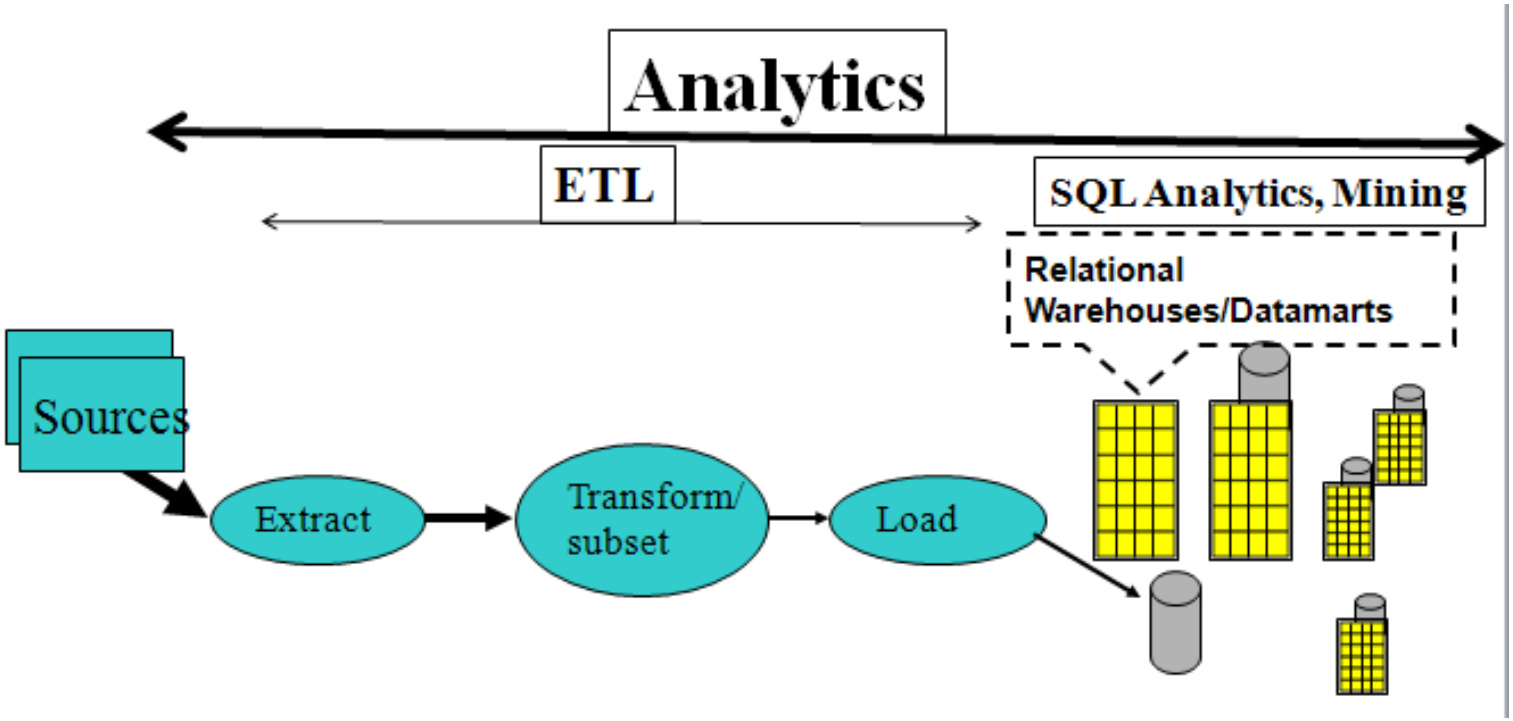
INTELLIGENT

The resulting explosion of information creates a need for a new kind of intelligence

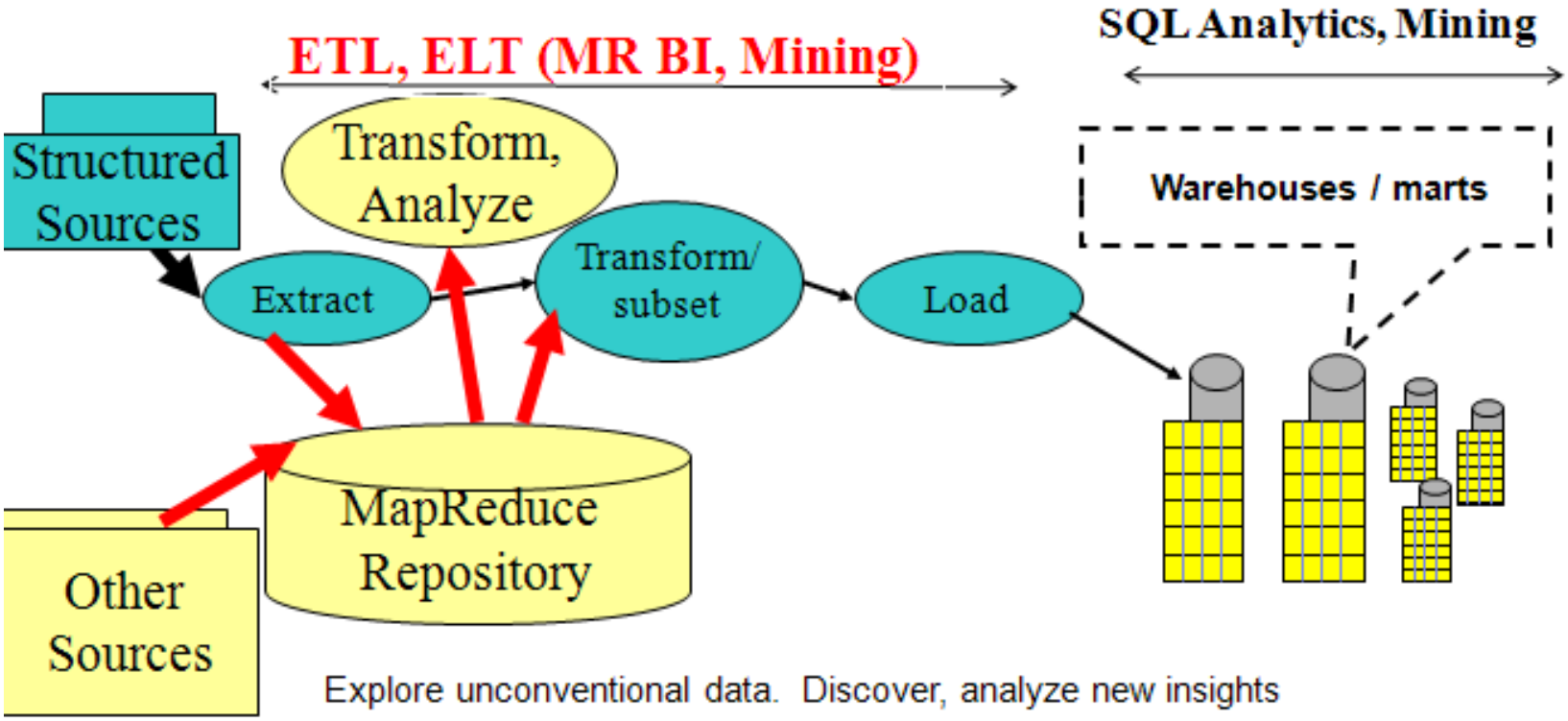
The BigData Challenge



Traditional approach to Analytics



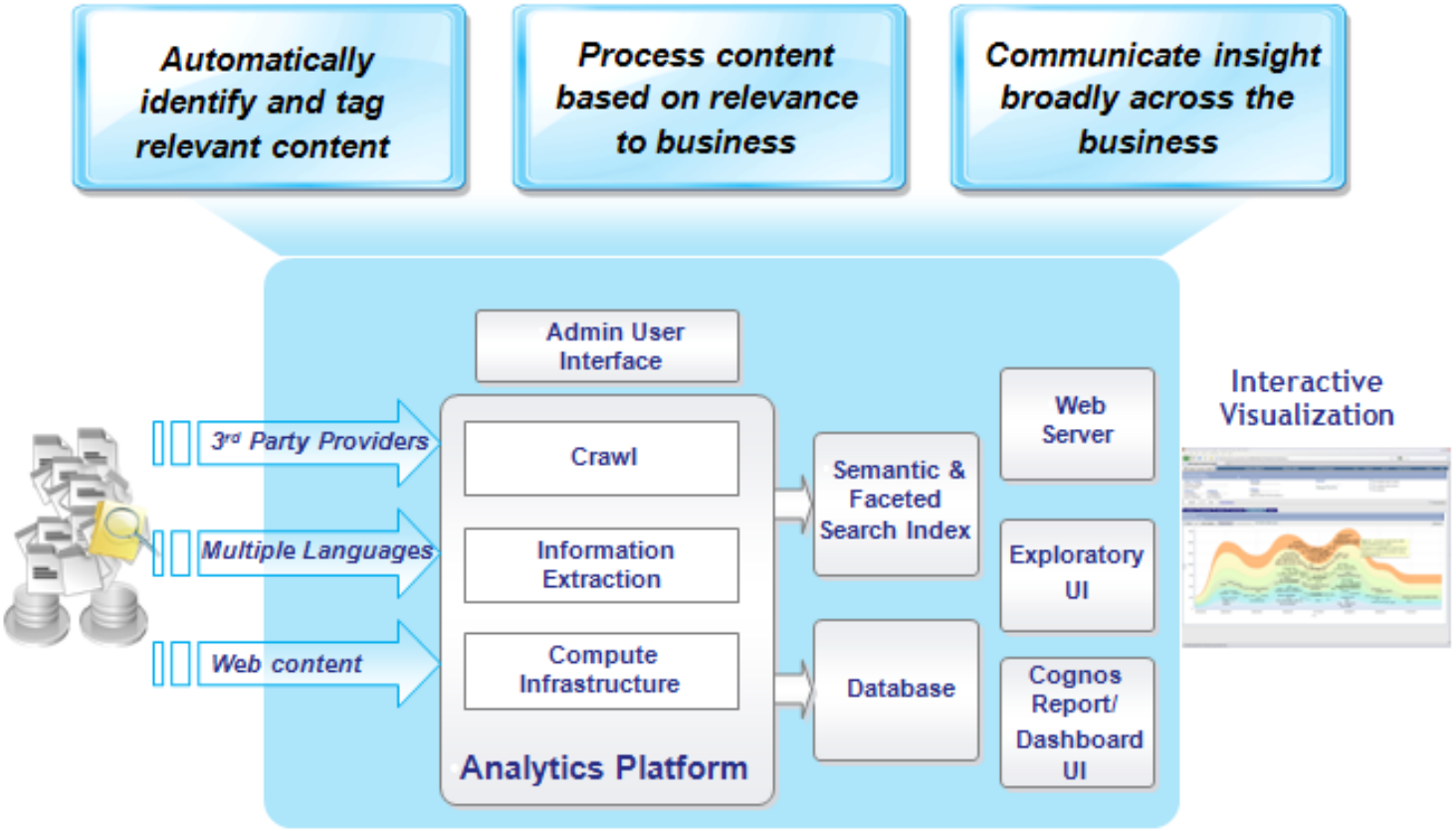
Emerging requirements for analytics



Explore unconventional data. Discover, analyze new insights

MapReduce repository for petabyte scale

Leveraging Big Data Analytics



http://www.hadoopworld.com/agenda/

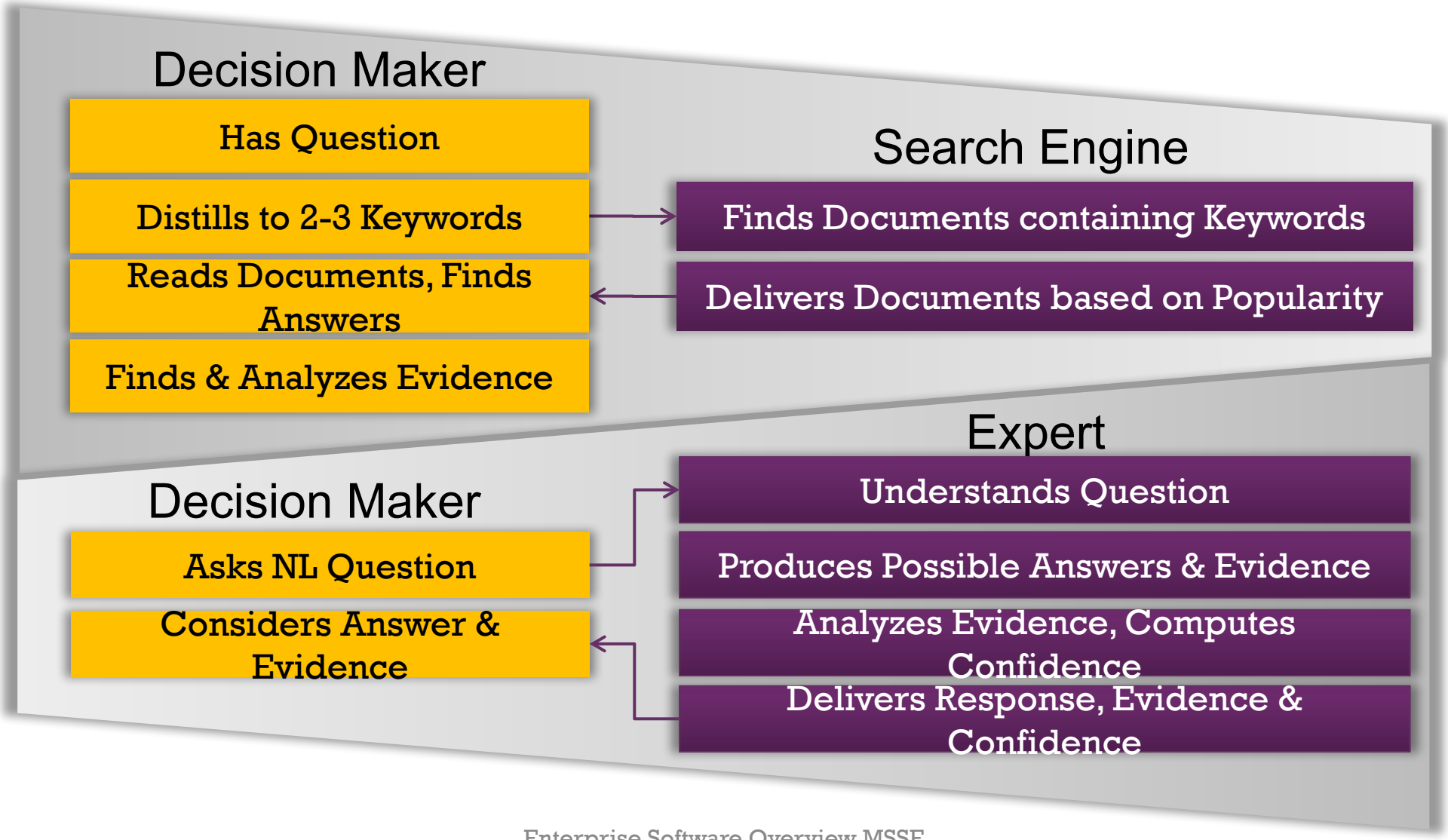
Watson & Power of Big Data Analytics

How Watson Uses Big Data

Think about it for a second: what would a *machine* have to do to replicate what a *human* does when we answer questions? It turns out, quite a bit: first the person must hear the question and process what he/she hears into concepts that can be understood--or natural language processing, in computer parlance. Then the person must consult their stored knowledge to see if they can find enough information to either make an educated guess or not. Once the person has determined their level of confidence in their answer, they need to decide if they are confident enough to ring in and put some money on the line. At that point the person, if they decide to ring in, must process their answer into speech--and frame that speech, in Jeopardy's case, into a question.

http://www.youtube.com/watch?v=lI-M7O_bRNg&feature=relmfu

Search Vs. Expert Q&A



Open-Domain Q&A

A Long-Standing Challenge in Artificial Intelligence to emulate human expertise

- Given
  - Rich Natural Language Questions
  - Over a Broad Domain of Knowledge

- Deliver
  - Precise Answers: Determine what is being asked & give precise response
  - Accurate Confidences: Determine likelihood answer is correct
  - Consumable Justifications: Explain why the answer is right
  - Fast Response Time: Precision & Confidence in <3 seconds

Some Basic Jeopardy's clues

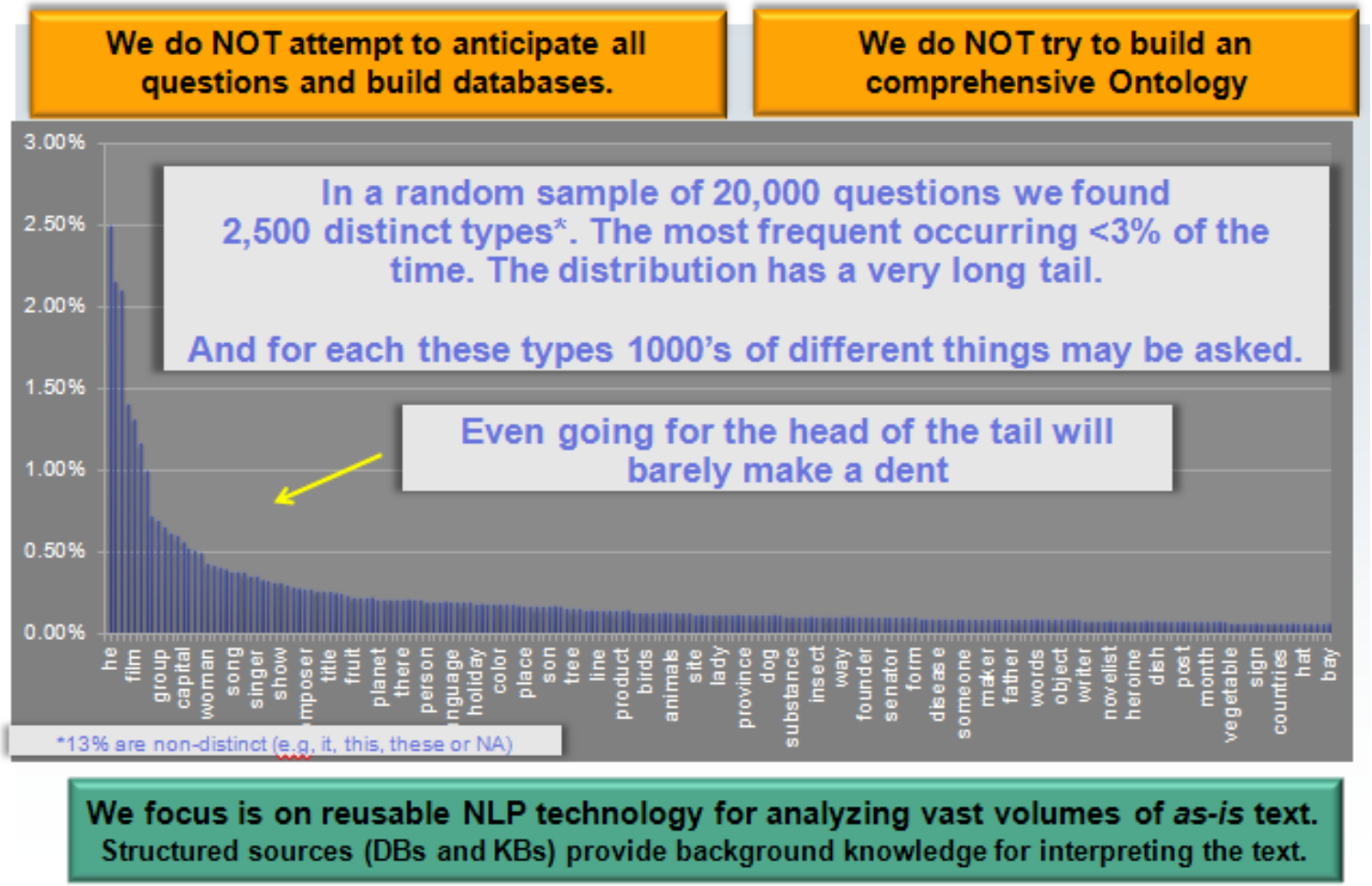
The *type* of thing being asked for is often indicated but can go from specific to very vague

- This fish was thought to be extinct millions of years ago until one was found off South Africa in 1938
- Category: ENDS IN "TH"
- Answer: coelacanth

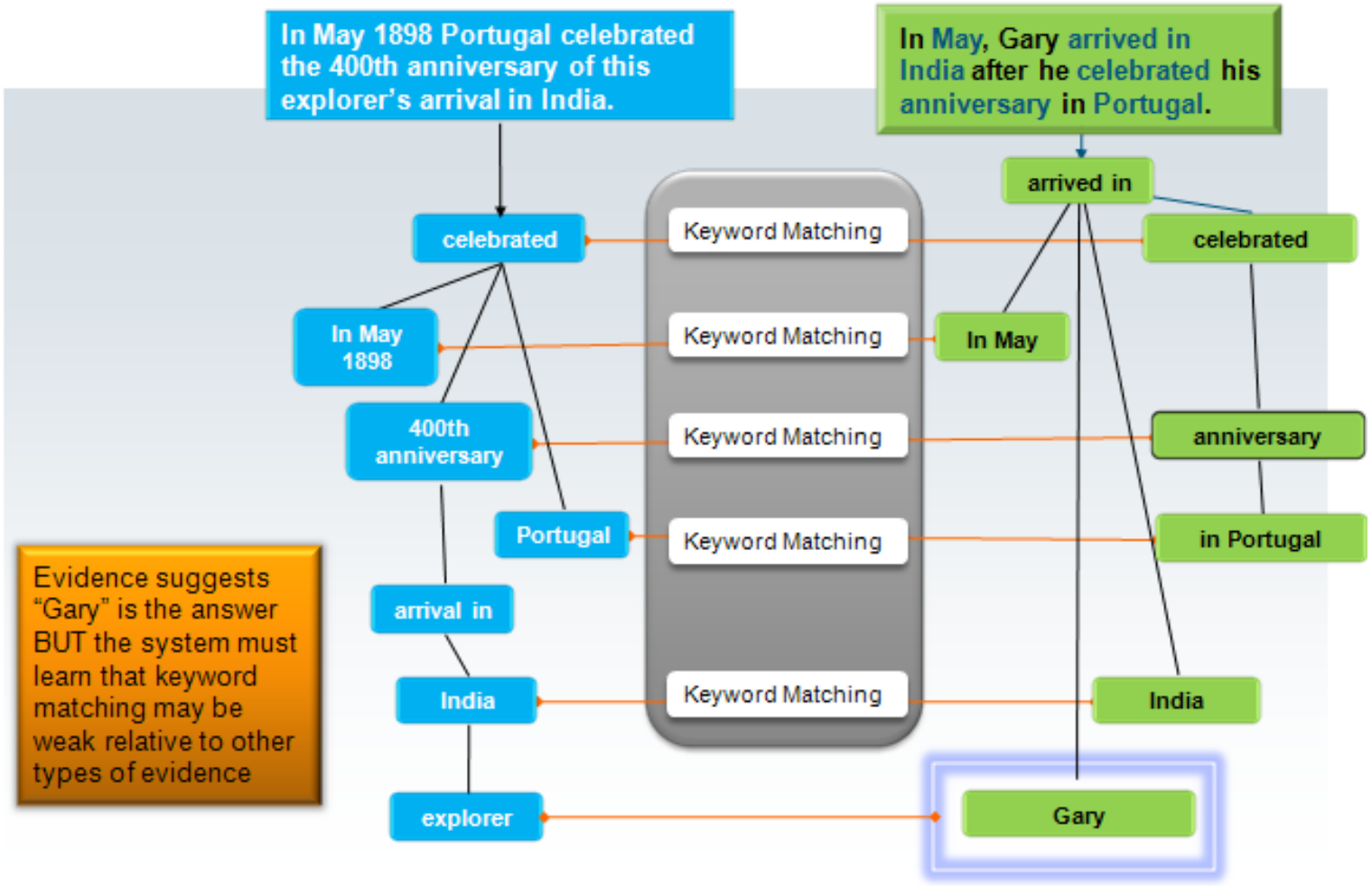
- When hit by electrons, a phosphor gives off electromagnetic energy in this form
- Category: General Science
- Answer: light (or photons)

- Secy. Chase just submitted this to me for the third time--guess what, pal. This time I'm accepting it
- Category: Lincoln Blogs
- Answer: his resignation

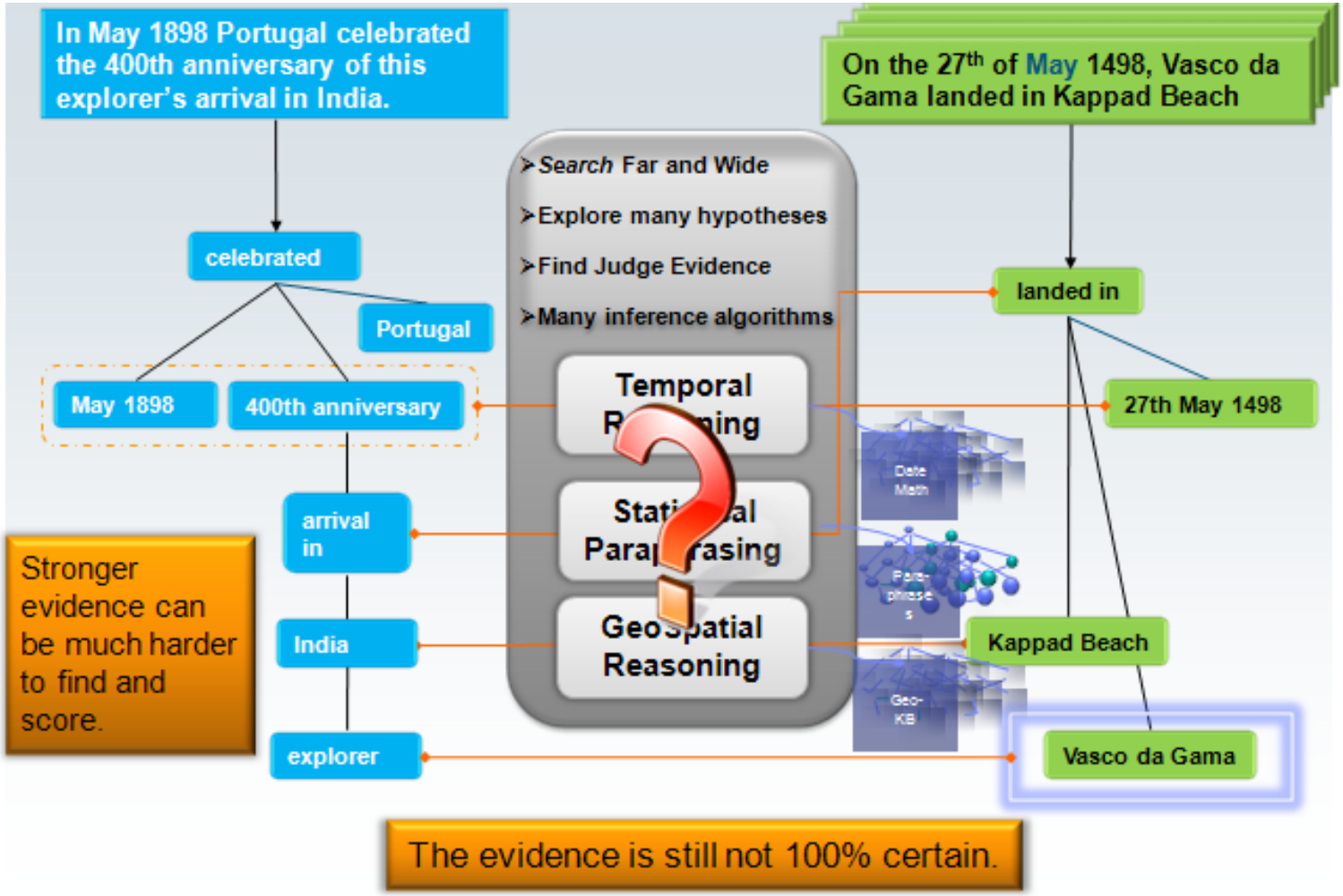
Broad Q&A domain - solution



Keyword Evidence in Q&A



Deeper Evidences



What data did Watson use?

- Wikipedia from DBPedia
- NY Times
- Encyclopedia
- World Book
- Bible
- Thesauri
- Dictionaries
- Wordnet
- Specific databases
  - Geography
  - Entertainment
  - News articles
  - Classic books

Hadoop distributed file system (HDFS)

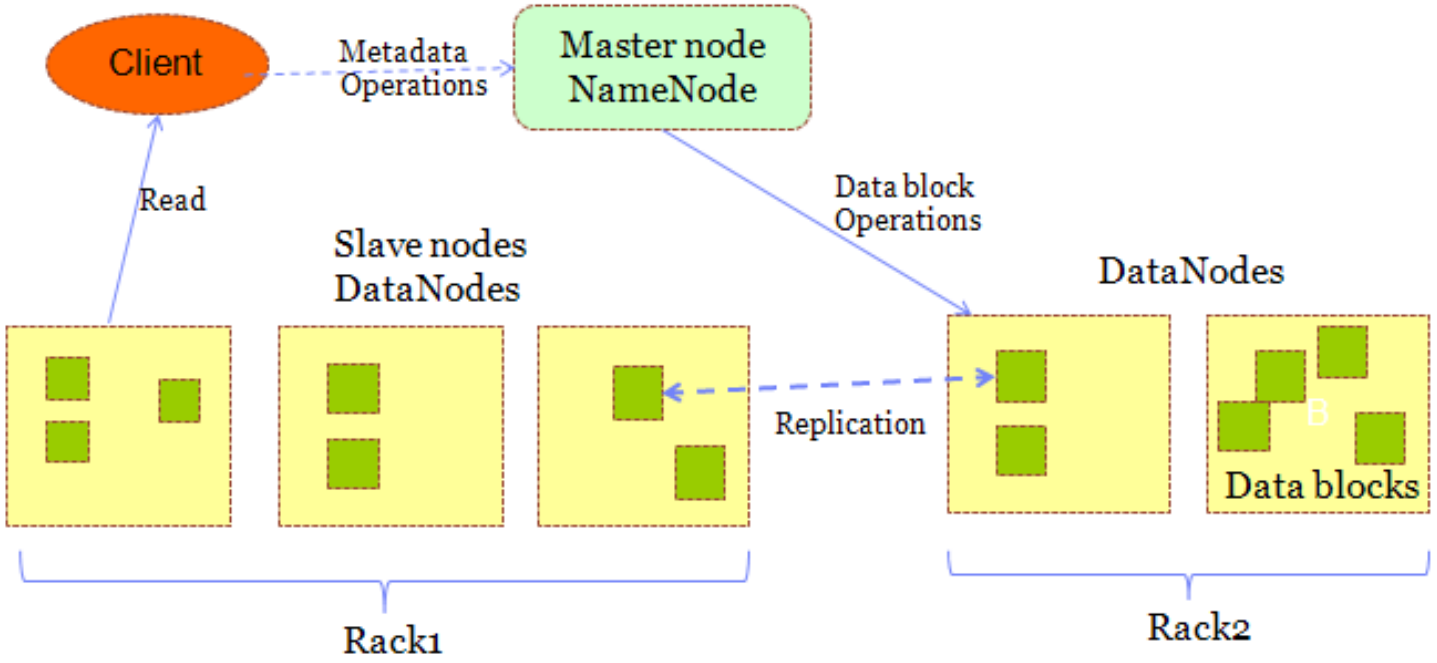
- Build for batch processing on BigData

- Optimized to support very large files
  - File size = MBs to TBs
  - Modest number of large files
  - Subset of POSIX features supported

- Built-in fault tolerance
  - Presumes periodic hardware failures (common with low-cost commodity hardware)
  - Replicates data blocks across nodes

- Data distribution and efficiency for MapReduce
  - Data spread across nodes at load time for effective parallel processing
  - Time to read whole data set more important than time to read first record
  - Simplified concurrency model: write once, read many orientation (although Hadoop 0.21 supports appends)

HDFS Architecture – Master/Slaves



In Hadoop terminology, *node* = *machine*

HDFS - NameNode

- Manages the file system namespace
  - Maintains file system tree and meta data for all files/directories in the tree
  - Two persistent files (namespace image and edit log) plus additional in-memory data
  - Essential for file system access.
  - Single point of failure. Name node loss renders file system inaccessible
- Centralizes and manages file system metadata in memory
  - Maps blocks to DataNodes, filenames, etc
  - Metadata size limited to available RAM of NameNode.
  - Bias toward modest number of large files, not large number of small files (where metadata can grow too sizeable)
  - NameNode will crash if it runs out of RAM
- Runs on a master node
  - Coordinates access to DataNodes but data never goes on NameNode
  - Hadoop has no built-in failover mechanism for NameNode

HDFS - DataNode (Slave)

- Files on HDFS are chopped into blocks and stored on DataNodes
  - Size of blocks is configurable
  - Different blocks from the same file might be stored on different DataNodes
- Serves read and write requests to clients
- Performs block creation, deletions, and replication as instructed by NameNode
  - Replication factor is configurable
- One instance of DataNode per slave node is recommended in real deployment

HDFS - Datablocks

- Much larger than traditional file system blocks
  - 64MB by default. Increase to 128MB for very large files.
  - If chunk of file is smaller than HDFS block size, only needed space is used

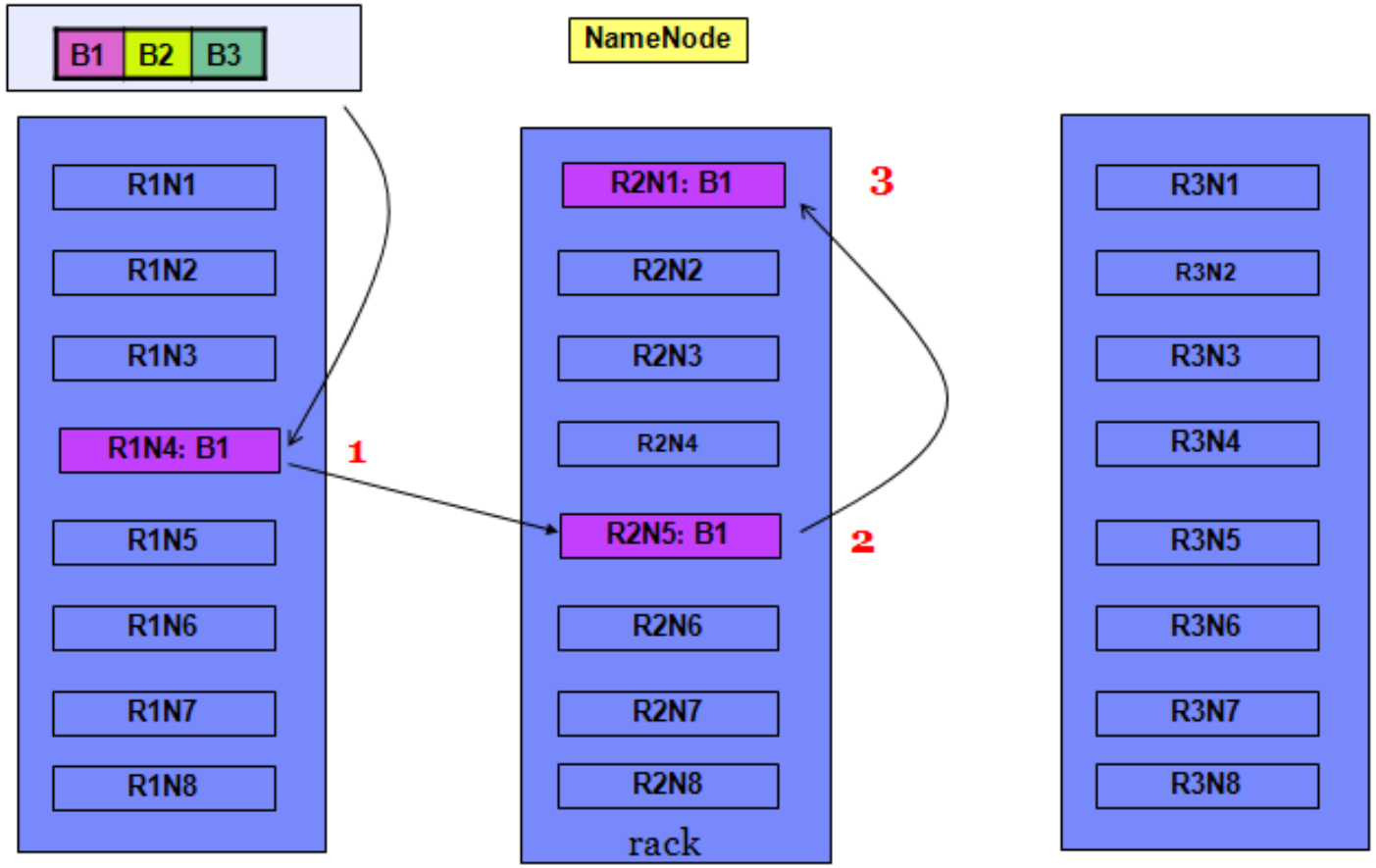
- Trade-off: block size and MapReduce parallelism
  - Map tasks in MapReduce normally operate on one block at a time
  - so if you have too few tasks (fewer than nodes in the cluster), your jobs will run slower than they could otherwise

- Advantages of HDFS's data block approach
  - Simplifies replication, providing fault tolerance and reliability
  - Shields users from storage subsystem details

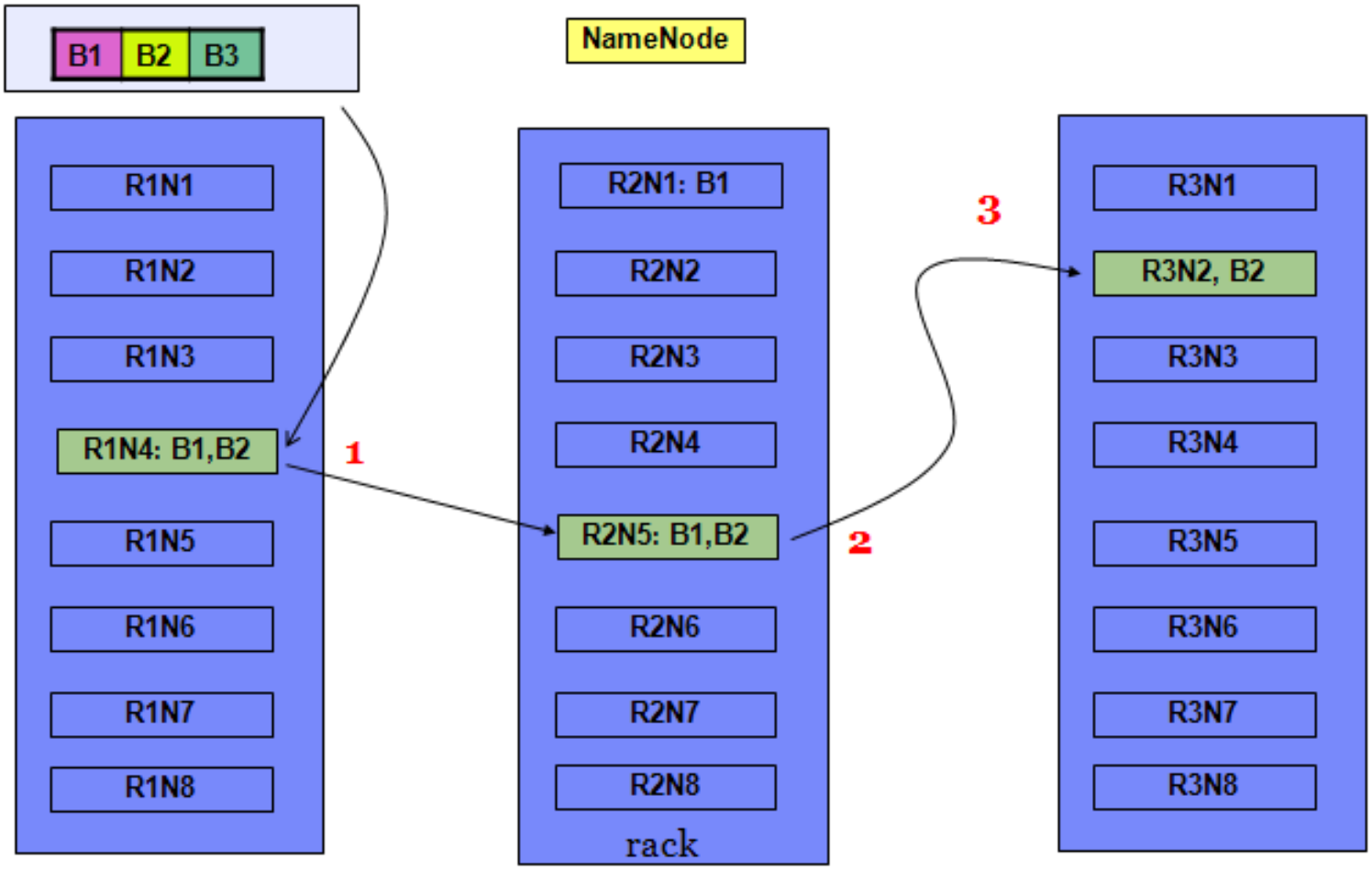
Reliability & Datablock replication

- Each block replicated across 3 DataNodes (by default)
  - 1st replica placed on same node as client
  - 2nd replica placed on different rack from 1st rack
  - 3rd replica placed on same rack as 2nd rack, but on a different node and striped
- Tradeoff between reliability and write/read bandwidth

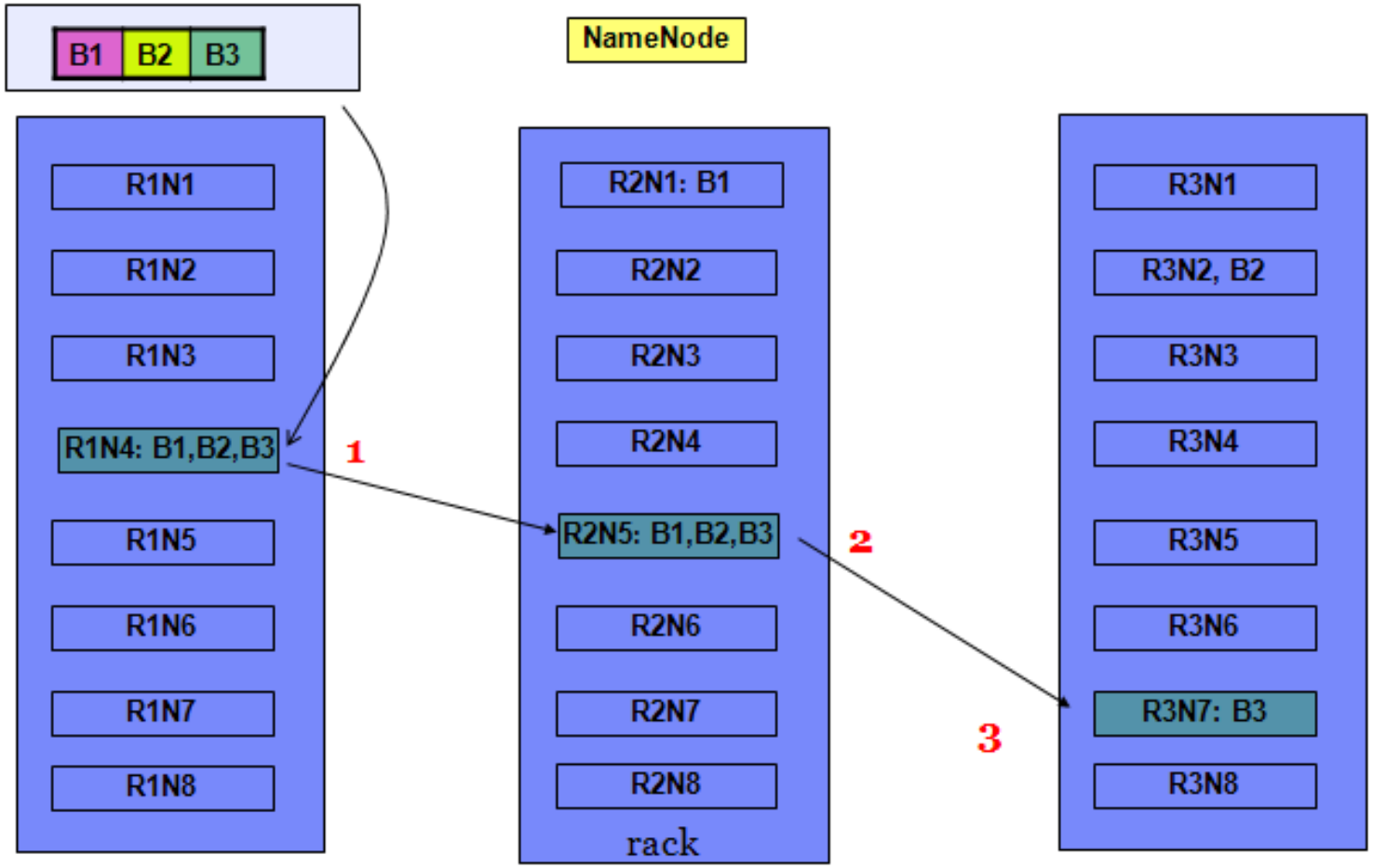
Data Replication Pipeline



Data Replication Pipeline#2



Data Replication Pipeline#3



Recovery using Replication

- DataNode sends heartbeat to NameNode periodically
- No heartbeat? NameNode marks DataNode as dead
  - Stops forwarding any new I/O requests to the DataNode
  - Data registered to a dead DataNode is "lost" to HDFS
- NameNode's process for data recovery:
  - Determine which blocks were on the lost node
  - Find other DataNodes with copies of these blocks
  - Instruct these DataNodes to copy the blocks to other nodes (whenever possible)

Cluster maintenance - Rebalancing

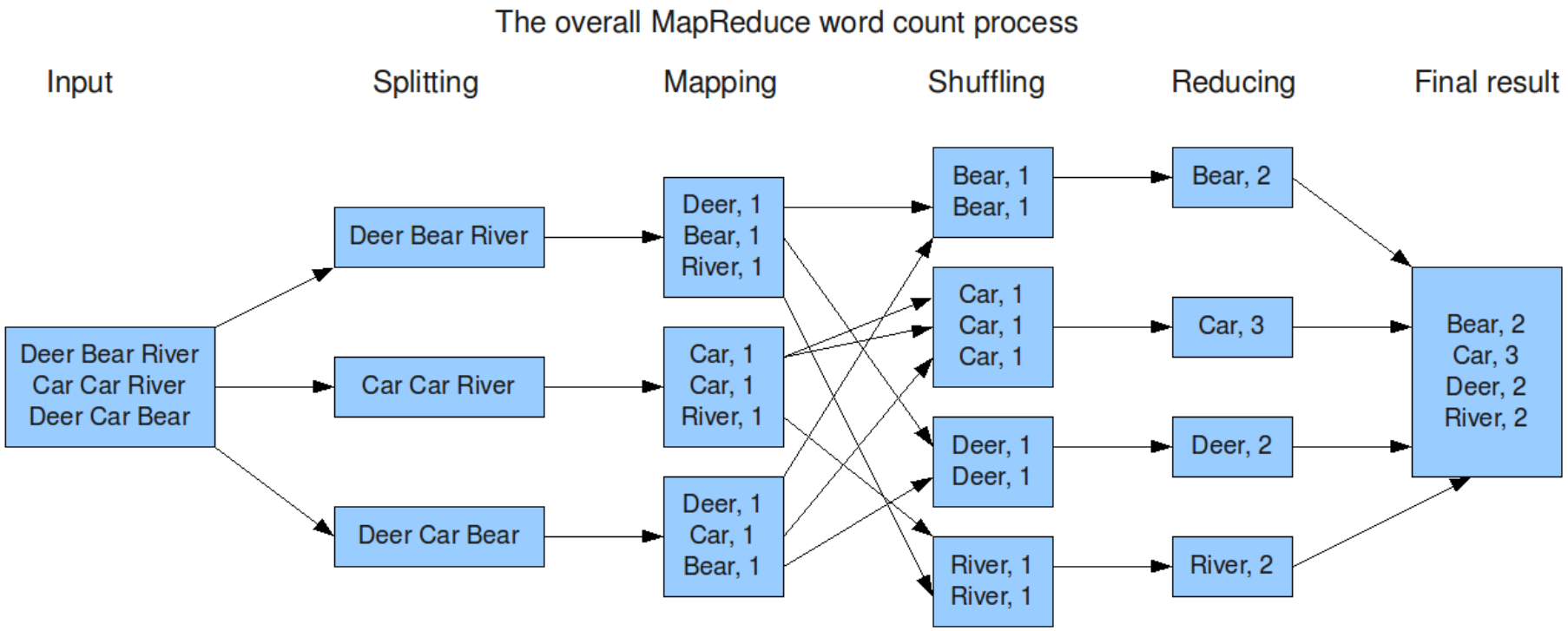
- Adding DataNodes can cause clusters to be 'unbalanced'
  - New files evenly distributed by HDFS
  - Old files retain existing distribution…..New node will have far less data than the others
    - During MapReduce processing, this Node will use much more network bandwidth as it retrieves data from other Nodes
- Use balancer utility to rebalance cluster
  - Immediately after adding new nodes to cluster
  - Periodically (e.g., weekly) for maintenance
  - Using existing data block replication strategy

Map-Reduce overview

- Benefits of MapReduce
  - Simple programming model (2 concepts – map and reduce)
  - Computing system framework automatically handles
    - Large scale data
    - Parallelization and distributed execution
    - Fault tolerance
    - Job management and monitoring

- Characteristics
  - Batch-oriented data processing
  - Storage system independent
    - For parallelism, will require a distributed file system (e.g., HDFS)

Map Reduce



Source: http://blog.jteam.nl/2009/08/04/introduction-to-hadoop/

Example mapper()

```
public class MapClass extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable> {

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

    public void map(LongWritable key, Text value,
                    OutputCollector<Text, IntWritable> output,
                    Reporter reporter) throws IOException {
        String line = value.toString();
        StringTokenizer itr = new StringTokenizer(line);
        while (itr.hasMoreTokens()) {
            word.set(itr.nextToken());
            output.collect(word, one);
        }
    }
}
```

Example reducer()

```
public class Reduce extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable> {

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

}
```

Word counting app

```
public class WordCount {
   public static void main(String[] args) throws Exception {
      JobConf conf = new JobConf(WordCount.class);
      conf.setJobName("wordcount");

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

      conf.setMapperClass(Map.class);
      conf.setCombinerClass(Reduce.class);
      conf.setReducerClass(Reduce.class);

      conf.setInputFormat(TextInputFormat.class);
      conf.setOutputFormat(TextOutputFormat.class);

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

Hive & Pig

- Hive is a data warehouse system for Hadoop that facilitates easy data summarization, ad-hoc queries, and the analysis of large datasets stored in Hadoop compatible file systems
- Hive provides a mechanism to project structure onto this data and query the data using a SQL-like language called HiveQL
- Developed and used as primary platform by Facebook
- Pig is a platform for analyzing large data sets that consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs
- Developed and used as primary platform by Yahoo!

Word counting ex using Pig

```
lines = LOAD '../data/words.txt' USING TextLoader() AS (sentence:chararray);
words = FOREACH lines GENERATE FLATTEN(TOKENIZE(sentence)) AS word;
groupedWords = GROUP words BY word;
counts = FOREACH groupedWords GENERATE group, COUNT(words);
STORE counts INTO 'output/wordcounts' USING PigStorage();
```

Where words.txt is:

```
Deer Bear River
Car Car River
Deer Car Bear
```

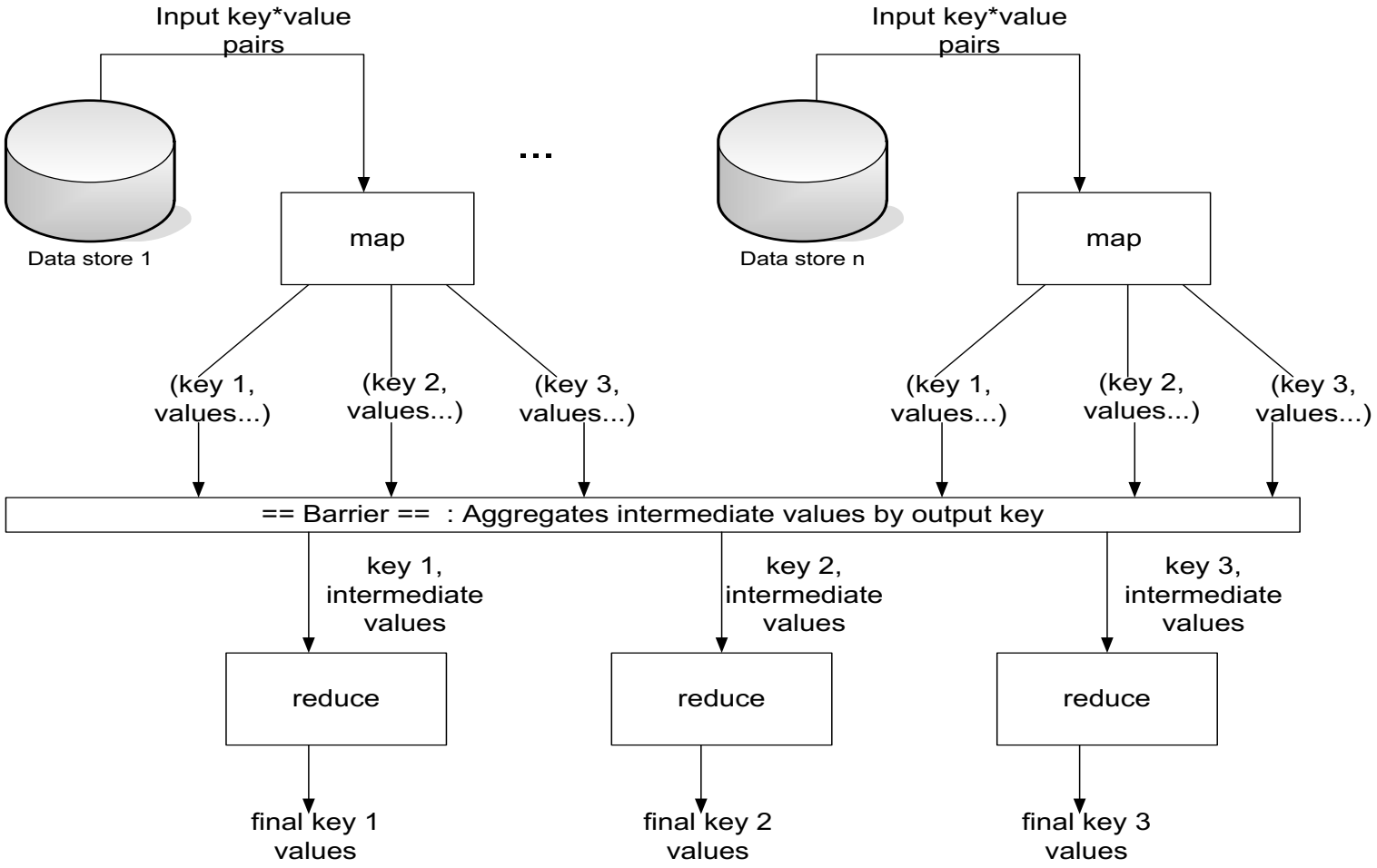
Generates:

```
Car     3
Bear    2
Deer    2
River   2
```

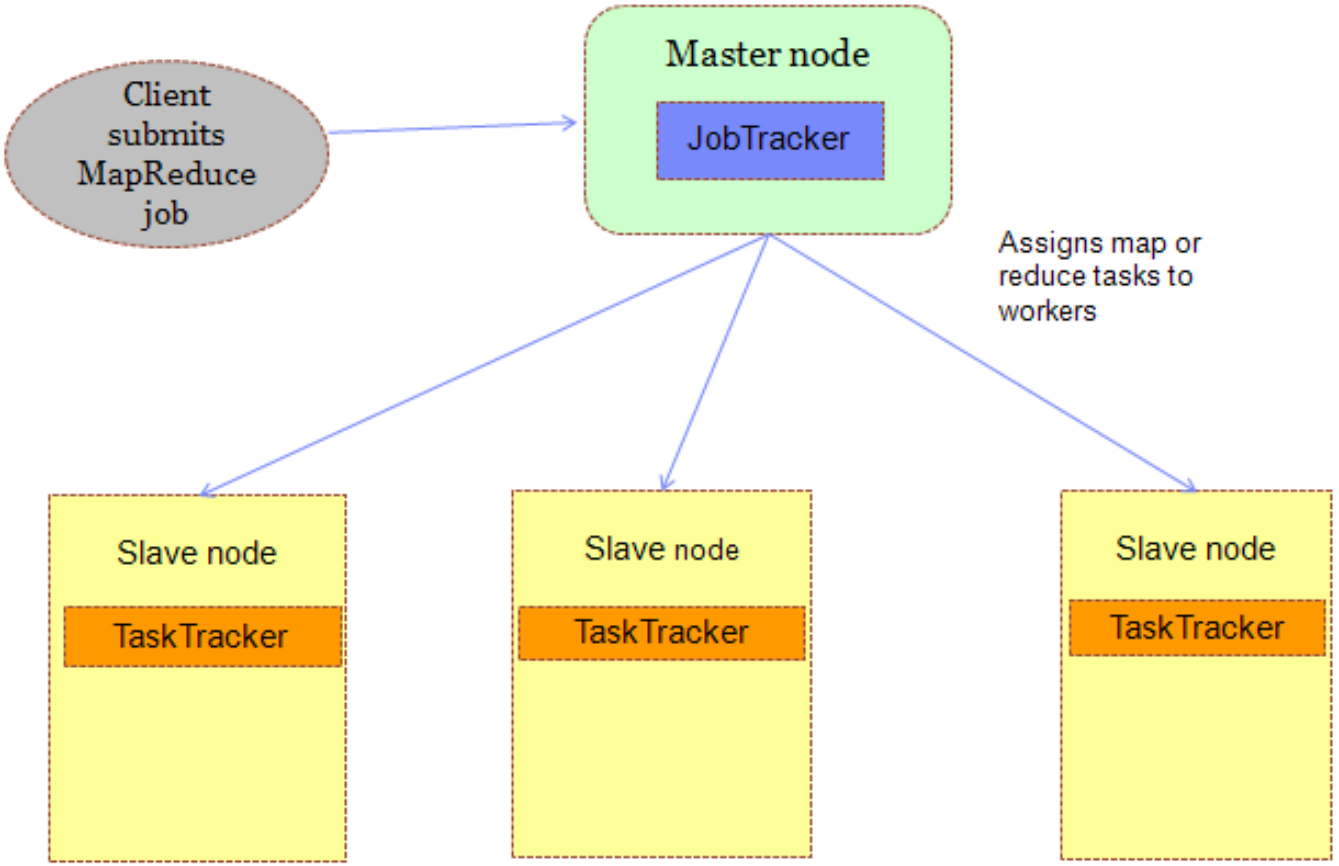
Wordcount using hive

```
create table textlines(text string);

load data local inpath 'C:\work\ClearPoint\Data20\data\words.txt' overwrite into table textlines;

create table words(word string);

insert overwrite table words select explode(split(text, '[ \t]+')) word from textlines;

select word, count(*) from words group by word;
```

MR Paradigm


Input key*value pairs
Input key*value pairs
...
Data store 1
Data store n
map
map
(key 1, values...)
(key 2, values...)
(key 3, values...)
(key 1, values...)
(key 2, values...)
(key 3, values...)
== Barrier == : Aggregates intermediate values by output key
key 1, intermediate values
key 2, intermediate values
key 3, intermediate values
reduce
reduce
reduce
final key 1 values
final key 2 values
final key 3 values

MR – Job Execution



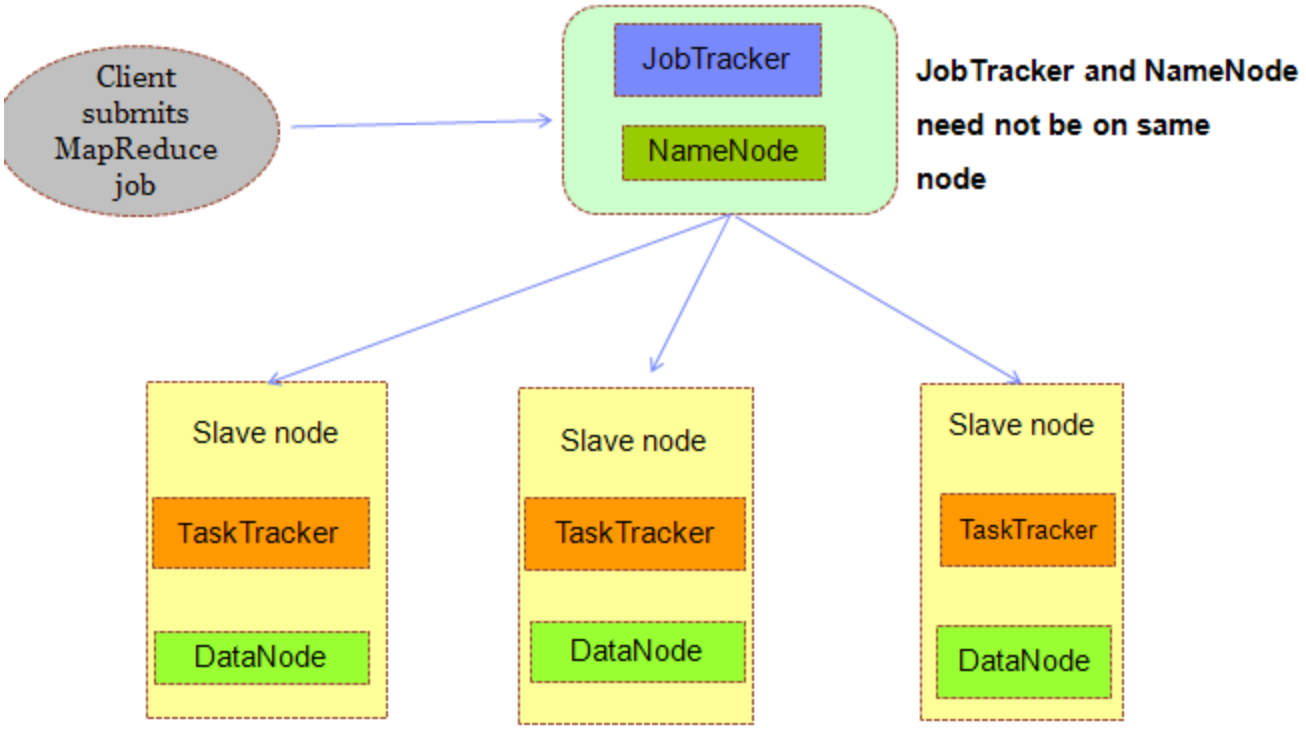
MR – Job Execution(behind the scene)

- A *job* is composed of
  - An unordered set of *Map tasks* which have locality preferences
  - An unordered set of *Reduce tasks*

- Tasks are scheduled by the JobTracker
  - They are then executed by TaskTrackers
  - One TaskTracker per node
  - Each TaskTracker has a fixed number of *slots* for Map and Reduce tasks
    - This may differ per node – a node with a powerful processor may have more slots than one with a slower CPU

- TaskTracker periodically sends heartbeat to JobTracker to indicate
  - It's alive
  - Report the availability of free task slots for new tasks

MR HDFS Co-located



Fault Tolerance

- Task processes send heartbeats to the TaskTracker (slave)
  - Failure presumed if no heartbeat in 10 minutes
  - JVM for task's killed by the TaskTracker

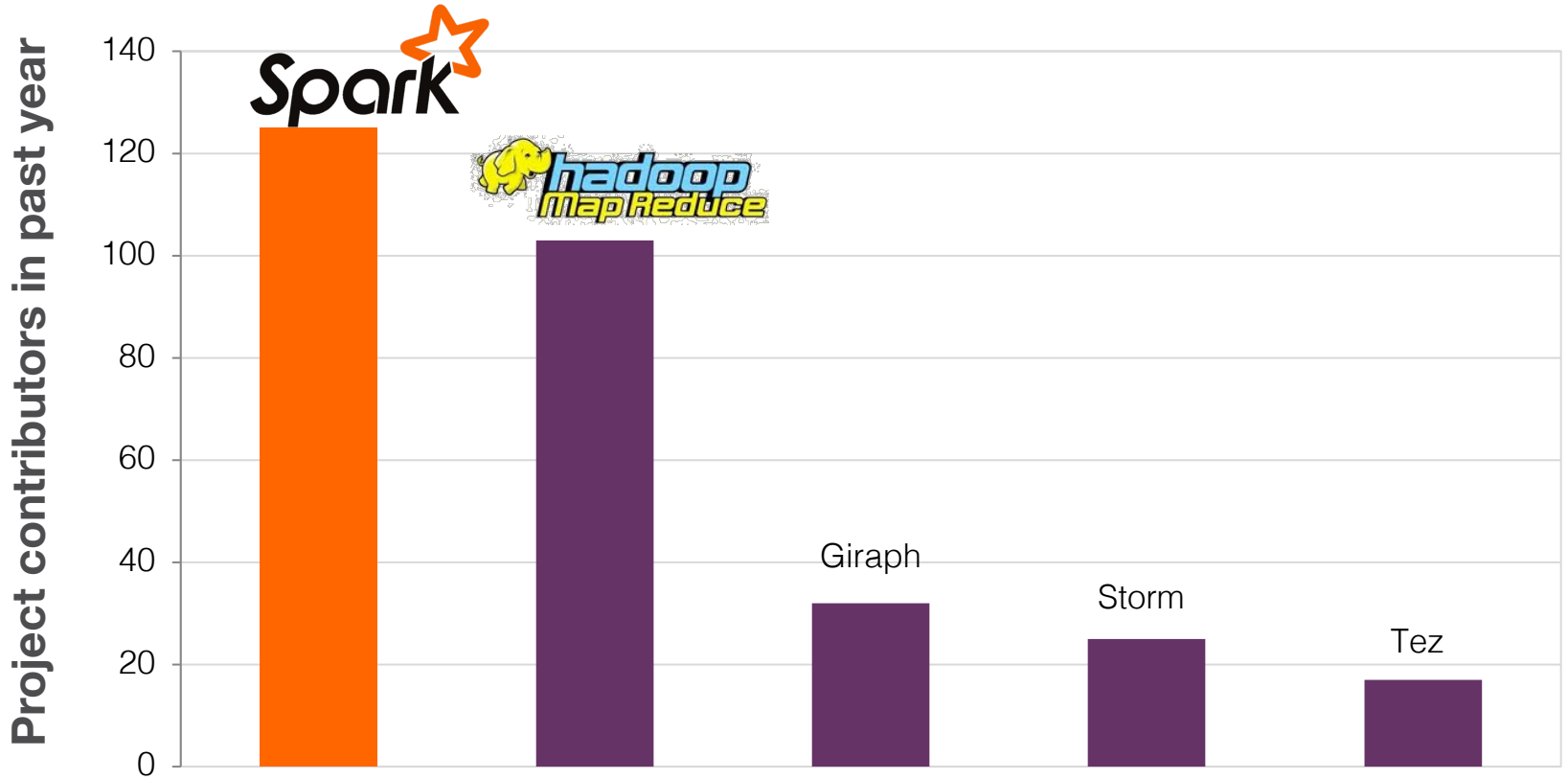
- TaskTrackers send heartbeats and info on failed tasks to the JobTracker (master)
  - JobTracker reschedules failed tasks, typically via a different TaskTracker (running on a different node)
  - Any TaskTracker reporting a high number of failed tasks is blacklisted to prevent the node from blocking the entire job

- Exception handling:
  - Any task that throws an exception = failed task
  - If certain input key/values cause exceptions in map tasks, JobTracker will skip those values on re-execution.
  - Effect: Can work around bugs in third-party libraries!

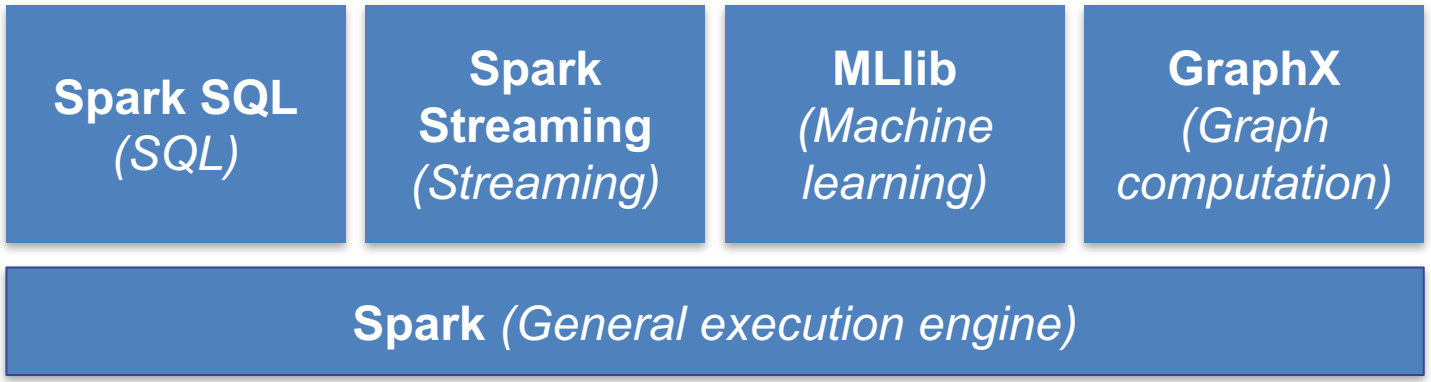
Spark

- Originally developed in 2009 in UC Berkeley's AMP Lab
  Fully open sourced in 2010 – now a Top Level Project at the Apache Software Foundation

spark.apache.org
github.com/apache/spark
user@spark.apache.org



Unified Platform



Continued innovation bringing new functionality, e.g.:

- **Java 8** (Closures, Lamba Expressions)
- **BlinkDB** (Approximate Queries)
- **SparkR** (R wrapper for Spark)

Machine Learning - MLlib

- K-Means
- L_1 and L_2-regularized Linear Regression
- L_1 and L_2-regularized Logistic Regression
- Alternating Least Squares
- Naive Bayes
- Stochastic Gradient Descent
- And more..

Data Sources

- Local Files
  - file:///opt/httpd/logs/access_log
- S3
- swift
- Hadoop Distributed Filesystem
  - Regular files, sequence files, any other Hadoop InputFormat
- HBase

Deploying Spark – Cluster Manager Types

- Mesos
- EC2
- GCE
- Standalone mode
- YARN

Easy and Fast Big Data

- **Easy to Develop**
  - Rich APIs in Java, Scala, Python, R
  - Interactive shell

2-5× less code

- **Fast to Run**
  - General execution graphs
  - In-memory storage

Up to 10× faster on disk, 100× in memory

Resilient Distributed Datasets (RDD)

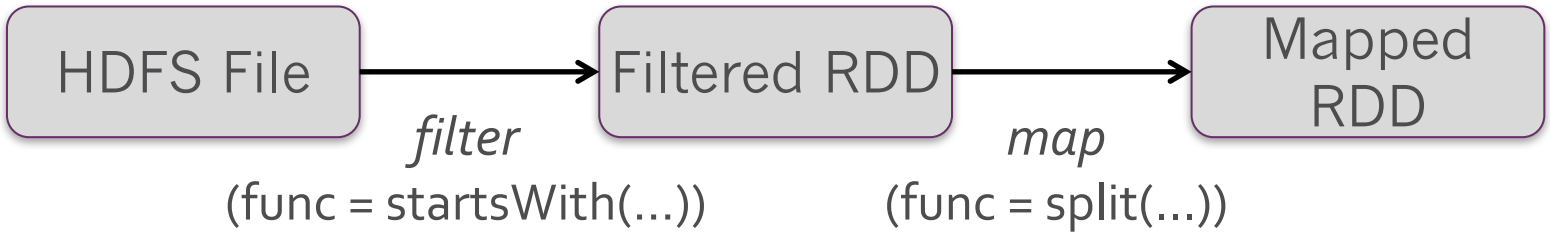
- Spark revolves around RDDs
- Fault-tolerant collection of elements that can be operated on in parallel
  - Parallelized Collection: Scala collection which is run in parallel
  - Hadoop Dataset: records of files supported by Hadoop

http://www.cs.berkeley.edu/~matei/papers/2012/nsdi_spark.pdf

RDD Fault Recovery

RDDs track *lineage* information that can be used to efficiently re compute lost data

```
msgs = textFile.filter(lambda s: s.startsWith("ERROR"))
               .map(lambda s: s.split("\t")[2])
```



RDD Operations

- Transformations
  - Creation of a new dataset from an existing
    - map, filter, distinct, union, sample, groupByKey, join, etc…
- Actions
  - Return a value after running a computation
    - collect, count, first, takeSample, foreach, etc…

Check the documentation for a complete list

http://spark.apache.org/docs/latest/scala-programming-guide.html#rdd-operations

RDD Persistence / Caching

- Variety of storage levels
  - memory_only (default), memory_and_disk, etc…
- API Calls
  - persist(StorageLevel)
  - cache() – shorthand for persist(StorageLevel.MEMORY_ONLY)
- Considerations
  - Read from disk vs. recompute (memory_and_disk)
  - Total memory storage size (memory_only_ser)
  - Replicate to second node for faster fault recovery (memory_only_2)
    - Think about this option if supporting a web application

http://spark.apache.org/docs/latest/scala-programming-guide.html#rdd-persistence

Interactive Shell

- Iterative Development
  - Cache those RDDs
  - Open the shell and ask questions
    - We have all wished we could do this with MapReduce
  - Compile / save your code for scheduled jobs later
- Scala – spark-shell
- Python – pyspark
- R - SparkR

Word Count

- Java MapReduce (~15 lines of code)
- Java Spark (~ 7 lines of code)
- Scala and Python (4 lines of code)
  - interactive shell: skip line 1 and replace the last line with counts.collect()
- Java8 (4 lines of code)

```
val sc = new SparkContext(master, appName, [sparkHome], [jars])
```

```
val file = sc.textFile("hdfs://...")
```

```
val counts = file.flatMap(line => line.split(" "))
      .map(word => (word, 1))
      .reduceByKey(_ + _)
```

```
counts.saveAsTextFile("hdfs://...")
```

```
counts.saveAsTextFile("hdfs://...");
```

Word Count – Spark Streaming

```
// Create the context with a 1 second batch size
val ssc = new StreamingContext(args(0), "NetworkWordCount",
Seconds(1),
System.getenv("SPARK_HOME"),
StreamingContext.jarOfClass(this.getClass))
```

```
// Create a NetworkInputDStream on target host:port and count
the
// words in input stream of \n delimited text (eg. generated by
'nc')
val lines = ssc.socketTextStream("localhost", 9999,
StorageLevel.MEMORY_ONLY_SER)
```

```
val words = lines.flatMap(_.split(" "))
```

```
val wordCounts = words.map(x => (x, 1)).reduceByKey(_ + _)
```

```
wordCounts.print()
```

```
ssc.start()
```

Configuration

http://spark.apache.org/docs/latest/

Most Important

- Application Configuration
  http://spark.apache.org/docs/latest/configuration.html

- Standalone Cluster Configuration
  http://spark.apache.org/docs/latest/spark-standalone.html

- Tuning Guide
  http://spark.apache.org/docs/latest/tuning.html

Team Assignment

1. In Bluemix, acquire Data Science Experience service
2. Pick an analytic use case of your choice from my blog: https://ranjanr.blogspot.com/2015/12/spark-use-cases.html
3. Submit your team's jupyter notebook link for evaluation in your team github (use a separate folder)