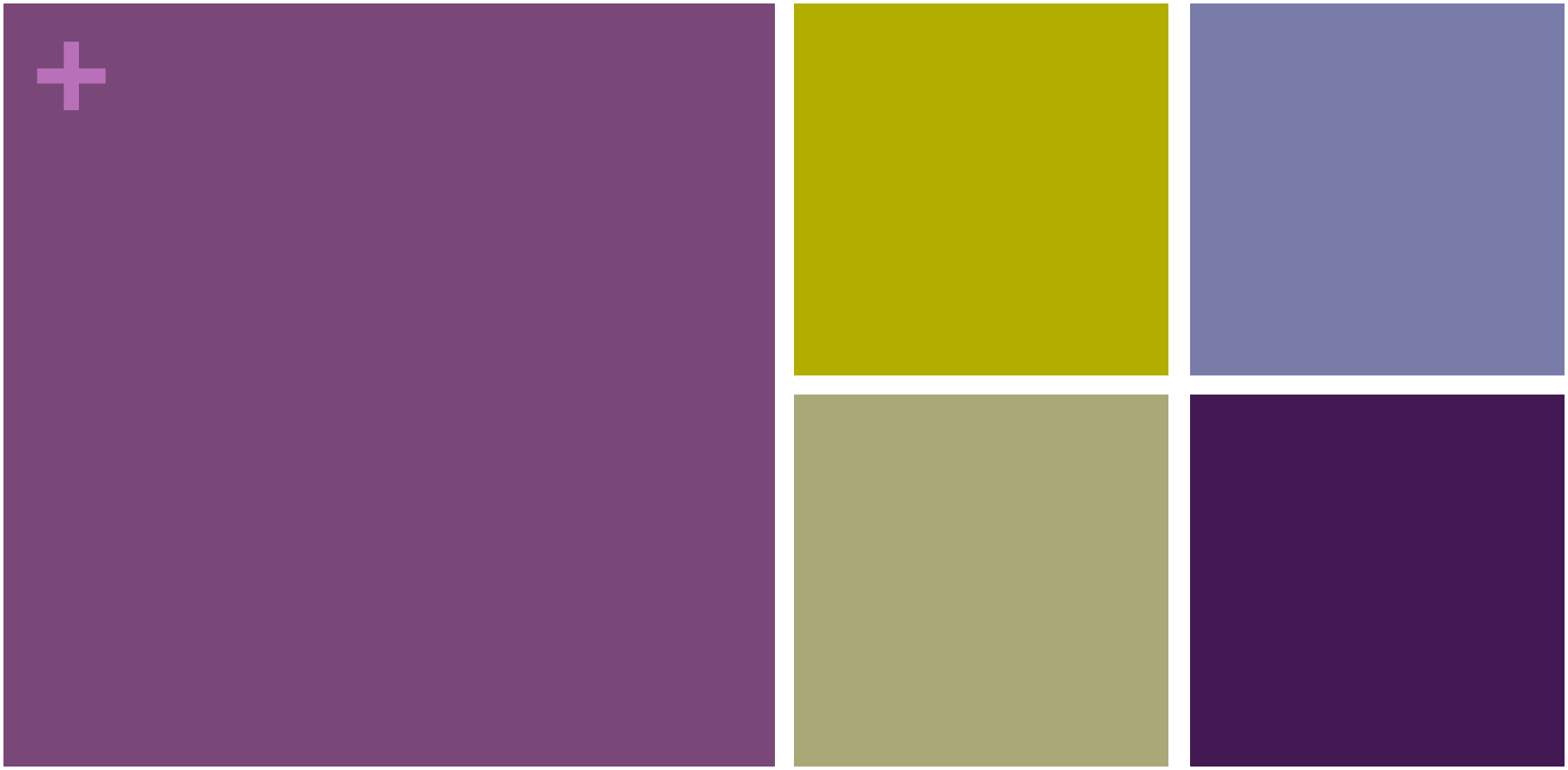
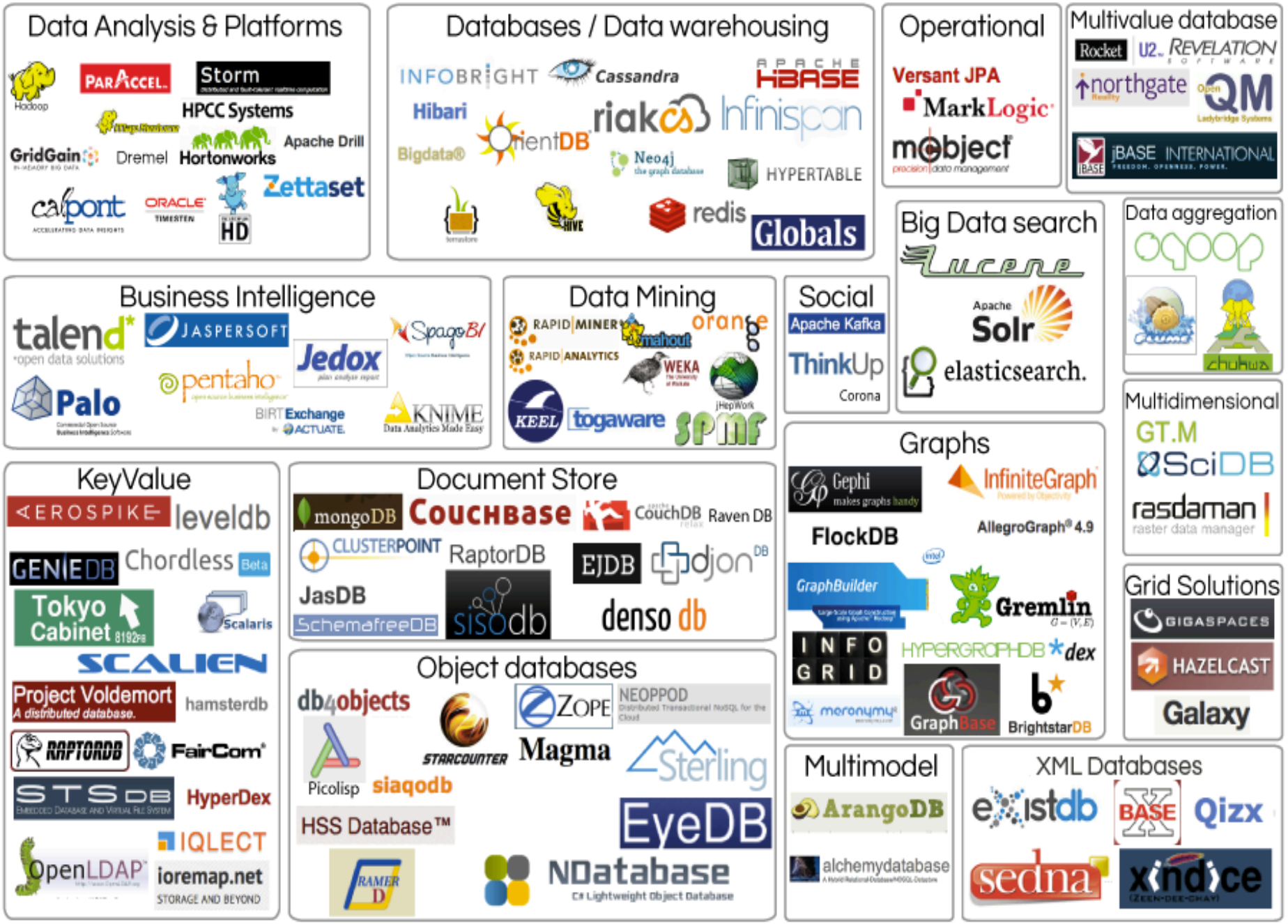


BigData Tools

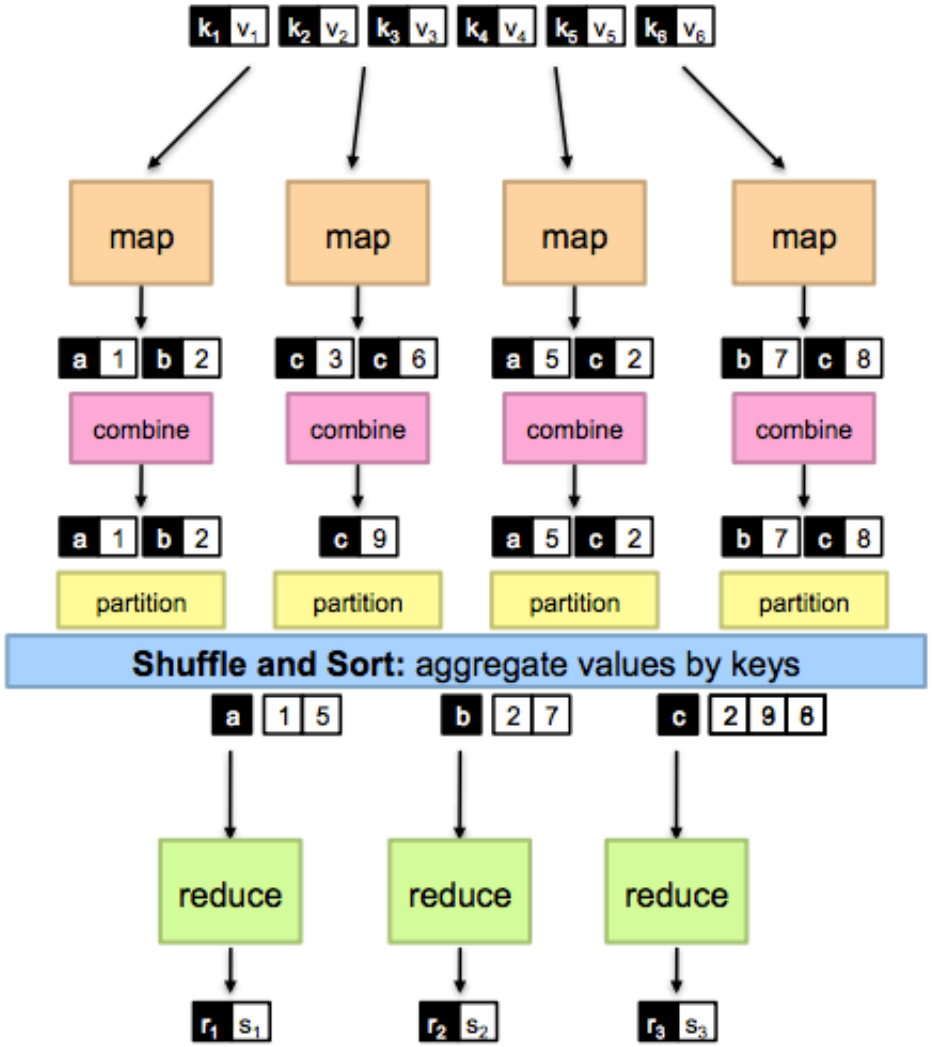
Abhijit Bendale (abendale@vast.uccs.edu)
04/15/2014

BigData Tools

- Data Analysis and Platforms
- Business Intelligence
- Document Store
- Twitter Case Study


Data Analysis & Platforms
Hadoop
ParAccel
Storm
HPCC Systems
Apache Drill
GridGain
Dremel
Hortonworks
calpont
Oracle TimesTen
HD
Zettaset
Databases / Data warehousing
Infobright
Cassandra
HBase
Hibari
Bigdata®
orientDB
riak
Infinispan
Neo4j
HYPERTABLE
redis
Globals
Operational
Versant JPA
MarkLogic
mobject
Multivalue database
Rocket
U2
Revelation
northgate
QM
jBASE International
Big Data search
Lucene
Apache Solr
elasticsearch.
Data aggregation
sqoop
Chukwa
Business Intelligence
talend
open data solutions
Jaspersoft
SpagoBI
Jedox
pentaho
Palo
BIRT Exchange
by ACTUATE
KNIME
Data Analytics Made Easy
Data Mining
RAPID MINER
mahout
orange
RAPID ANALYTICS
WEKA
KEEL
togaware
SPMF
Social
Apache Kafka
ThinkUp
Corona
Multidimensional
GT.M
SciDB
rasdaman
raster data manager
KeyValue
Aerospike
leveldb
GenieDB
Chordless
Beta
Tokyo Cabinet
8192PB
Scalaris
SCALIEN
Project Voldemort
A distributed database.
hamsterdb
RaptorDB
FairCom
STSDB
Embedded Database and Virtual File System
HyperDex
IQLECT
OpenLDAP
ioremap.net
STORAGE AND BEYOND
Document Store
mongoDB
Couchbase
CouchDB
Raven DB
CLUSTERPOINT
RaptorDB
EJDB
djon DB
JasDB
SchemafreeDB
sisodb
denso db
Graphs
Gephi
makes graphs handy
InfiniteGraph
Powered by Objectivity
AllegroGraph® 4.9
FlockDB
GraphBuilder
Gremlin
INFO GRID
HYPERGRAPHDB
dex
meronymy
GraphBase
BrightstarDB
Grid Solutions
GIGASPACES
HAZELCAST
Galaxy
Object databases
db4objects
Picolisp
siaqodb
STARCOUNTER
ZOPE
NEOPPOD
Distributed Transactional NoSQL for the Cloud
Magma
Sterling
HSS Database™
EyeDB
RAMER D
NDatabase
C# Lightweight Object Database
Multimodel
ArangoDB
alchemydatabase
XML Databases
existdb
BASE X
Qizx
sedna
xindice
(ZEEN-DEE-CHAY)
Created by: www.bigdata-startups.com

MapReduce flow



Map

- Independent record transformations
  - And deletions and replications
- (K1, V1) → list(K2, V2)

Receives a key value pair
And outputs a 0 or more
key-value pairs

Reduce

- Aggregate results from map phase
- (K2, list(V2)) → list(K3, V3)

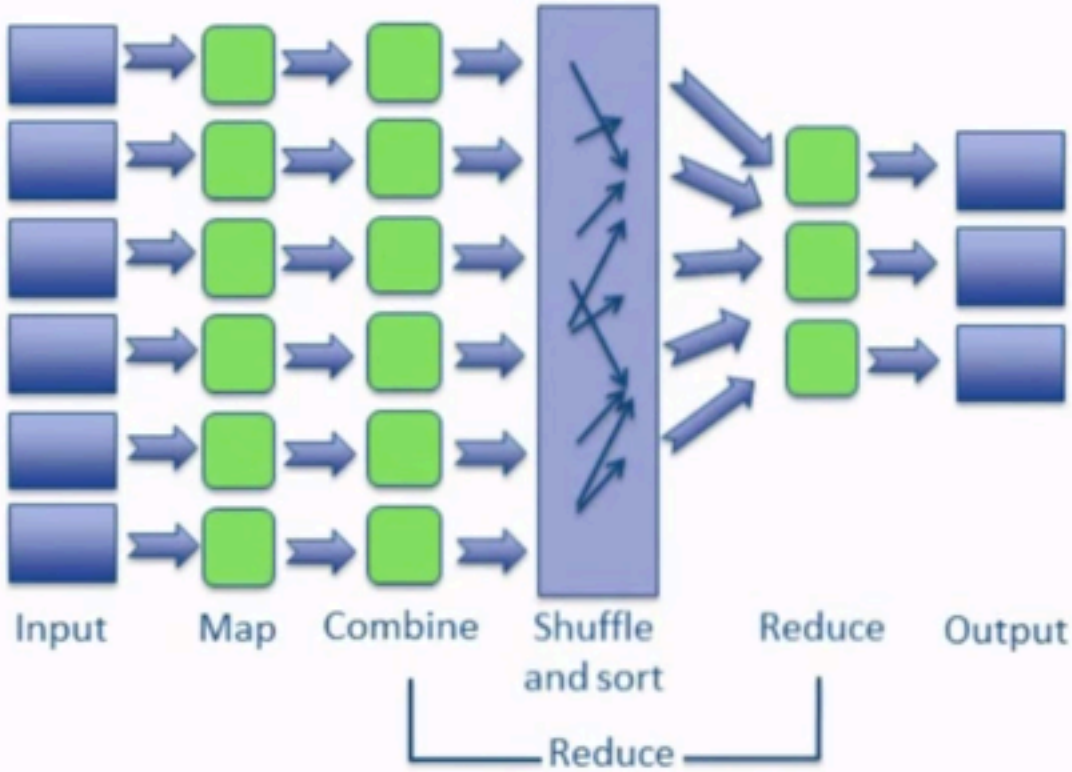
Reduce all the key-pairs with key
K2 to a new reduced key-value pair
K3, V3

Framework

- Schedules and re-runs tasks
- Splits the input
- Moves map outputs to reduce inputs
- Receives the results

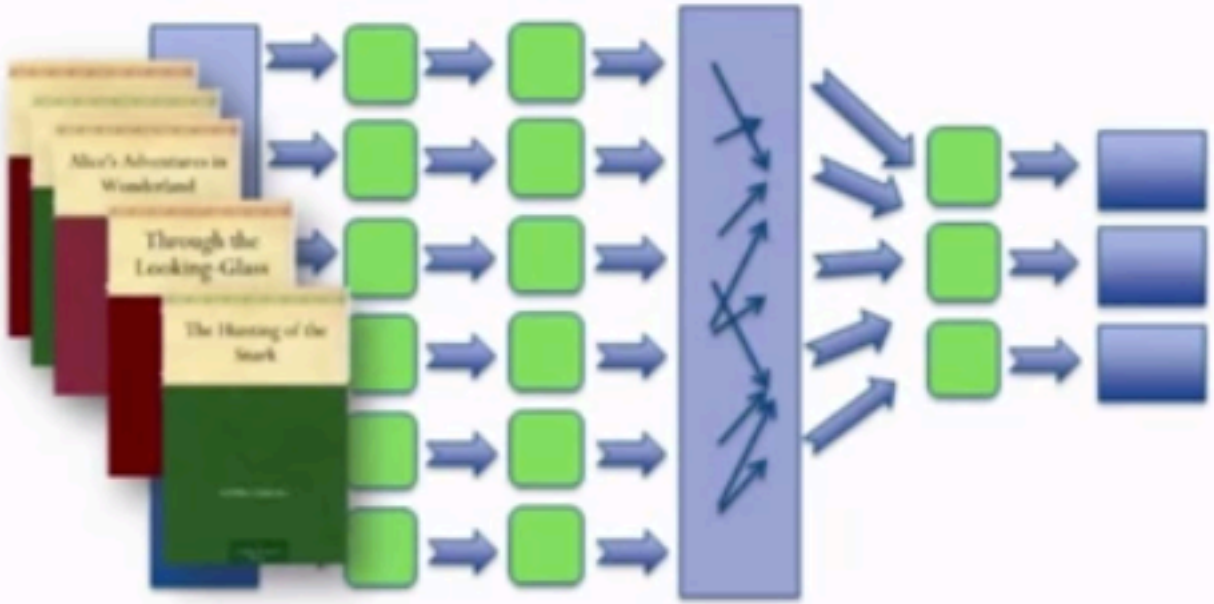
Majority of what Hadoop does..!

MapReduce Flow

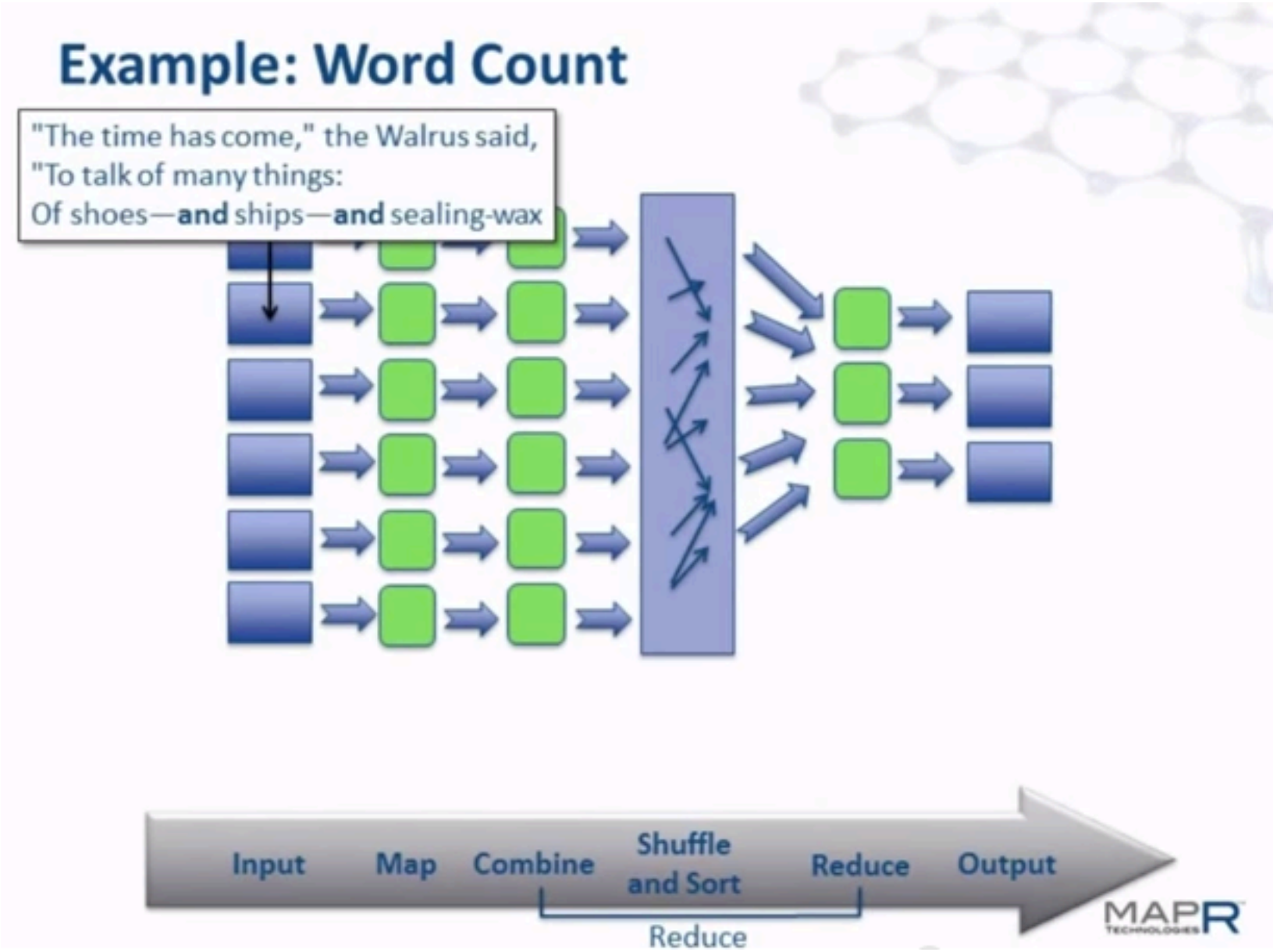


MAPR TECHNOLOGIES

Example: Word Count

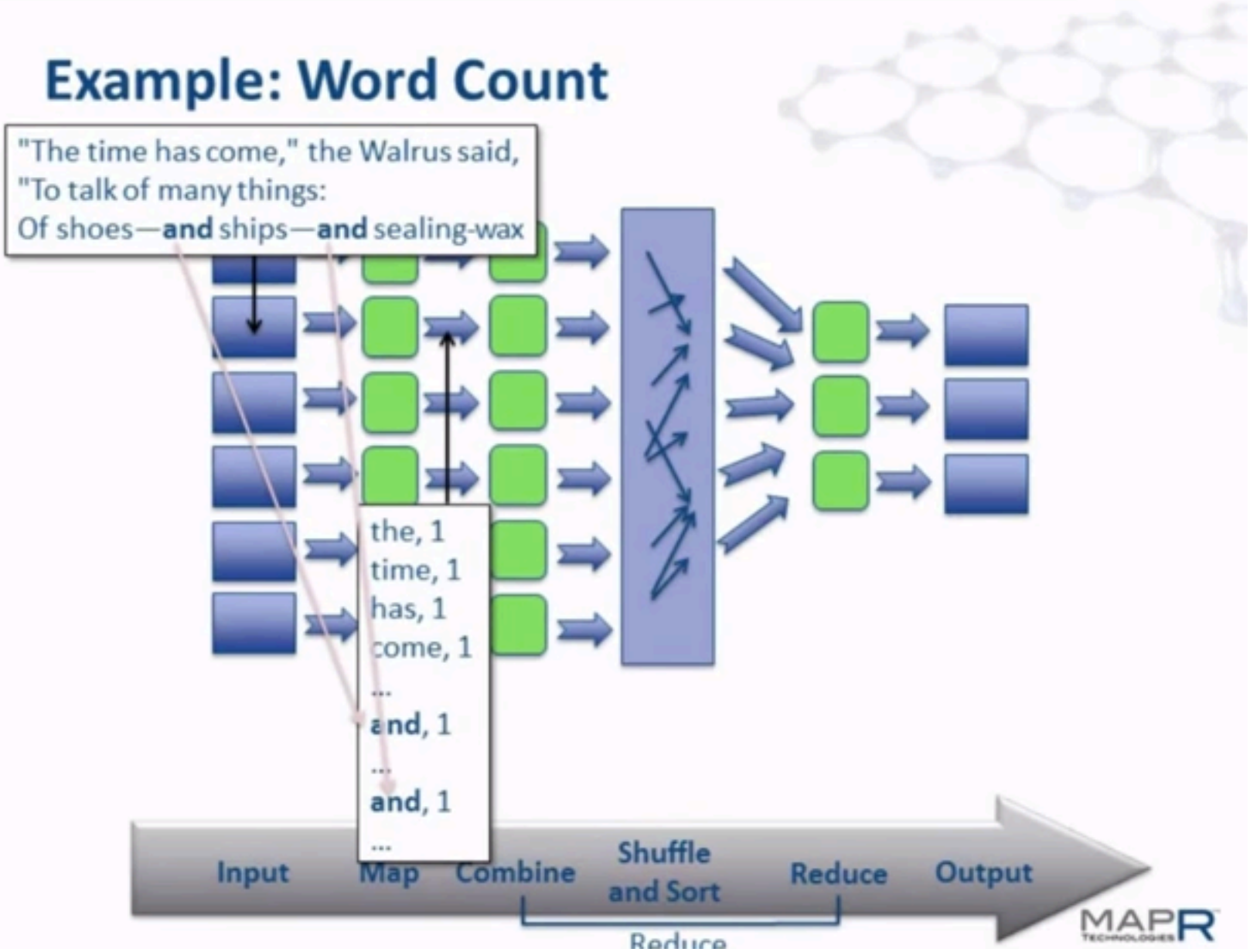


Goal: Consider we have a list of books and we want to count occurrences of ever word. Hadoop will distribute this task.

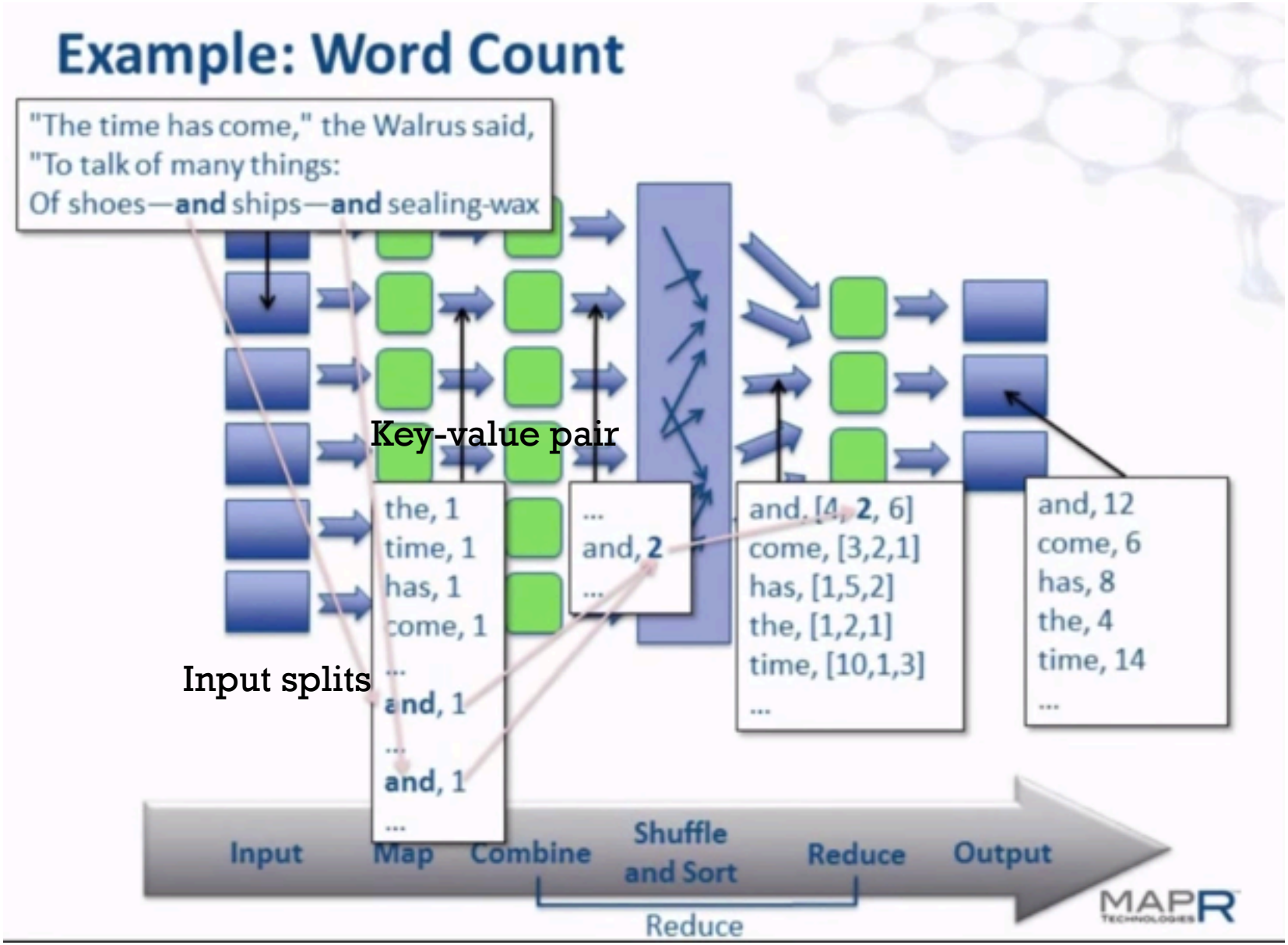


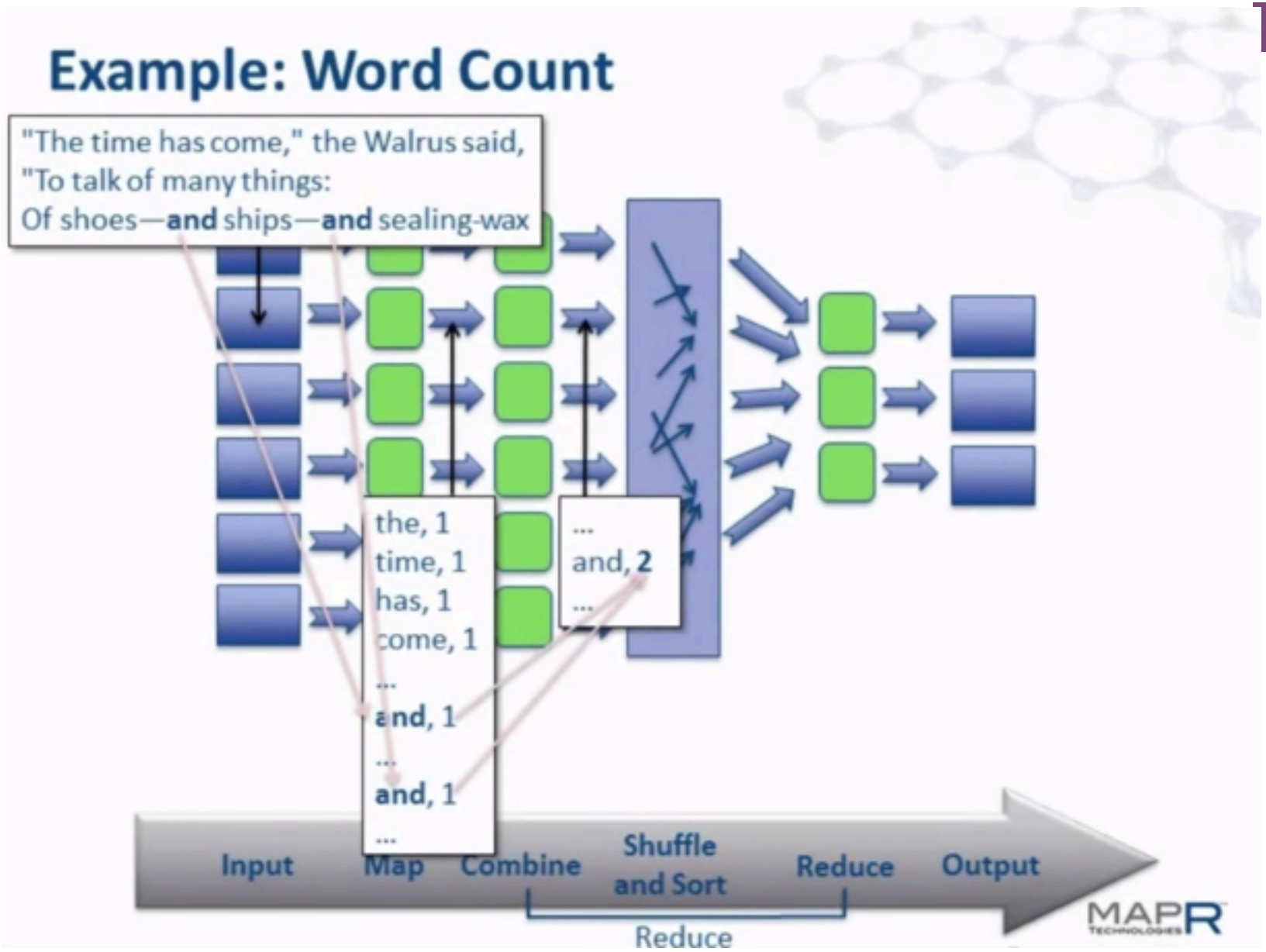
Here Key is byte offset in the file, Value is the text.

Example: Word Count

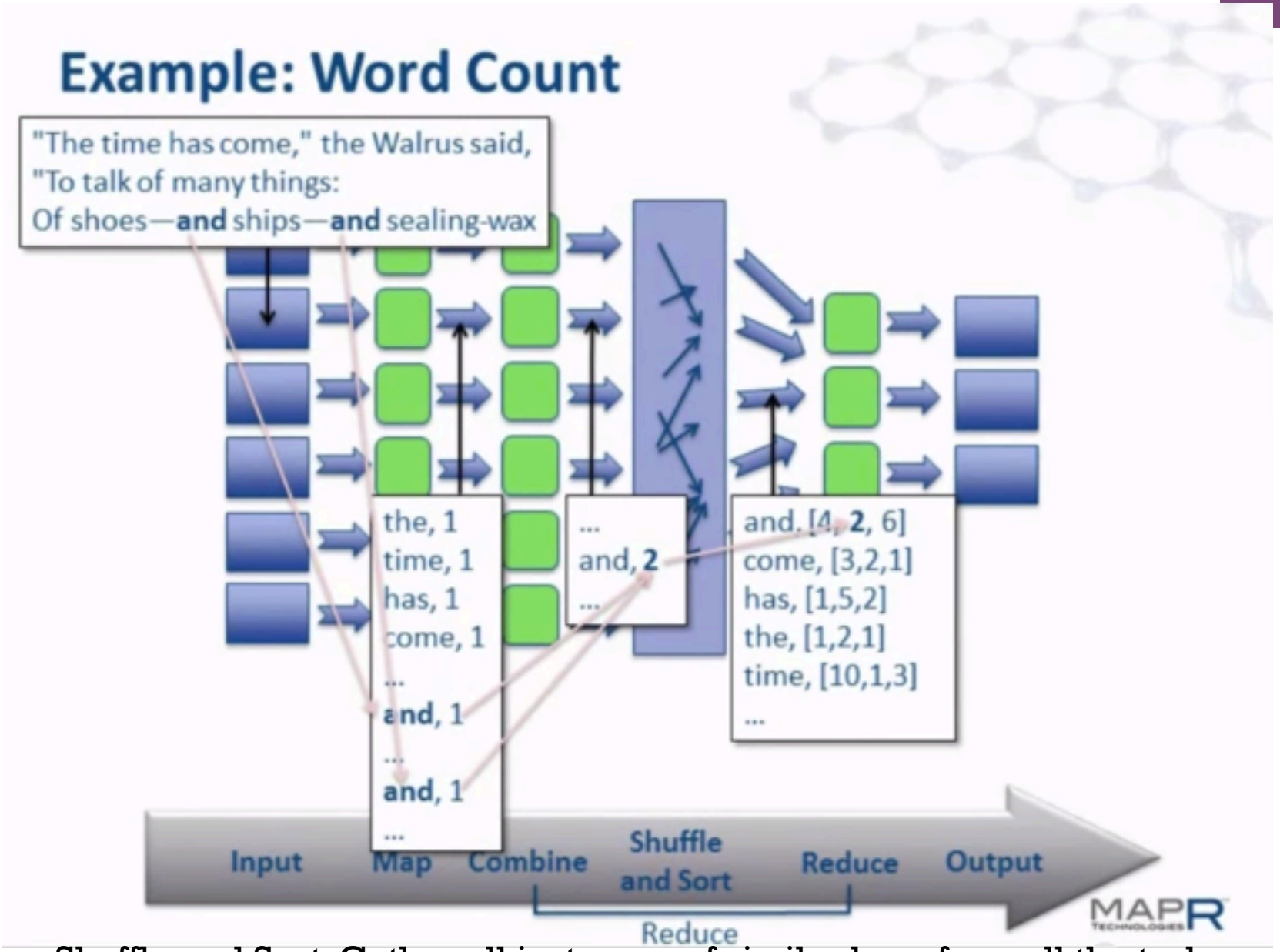


The map function tokenizes the input string and outputs key-value pair for every word. Note here "and" shows twice.

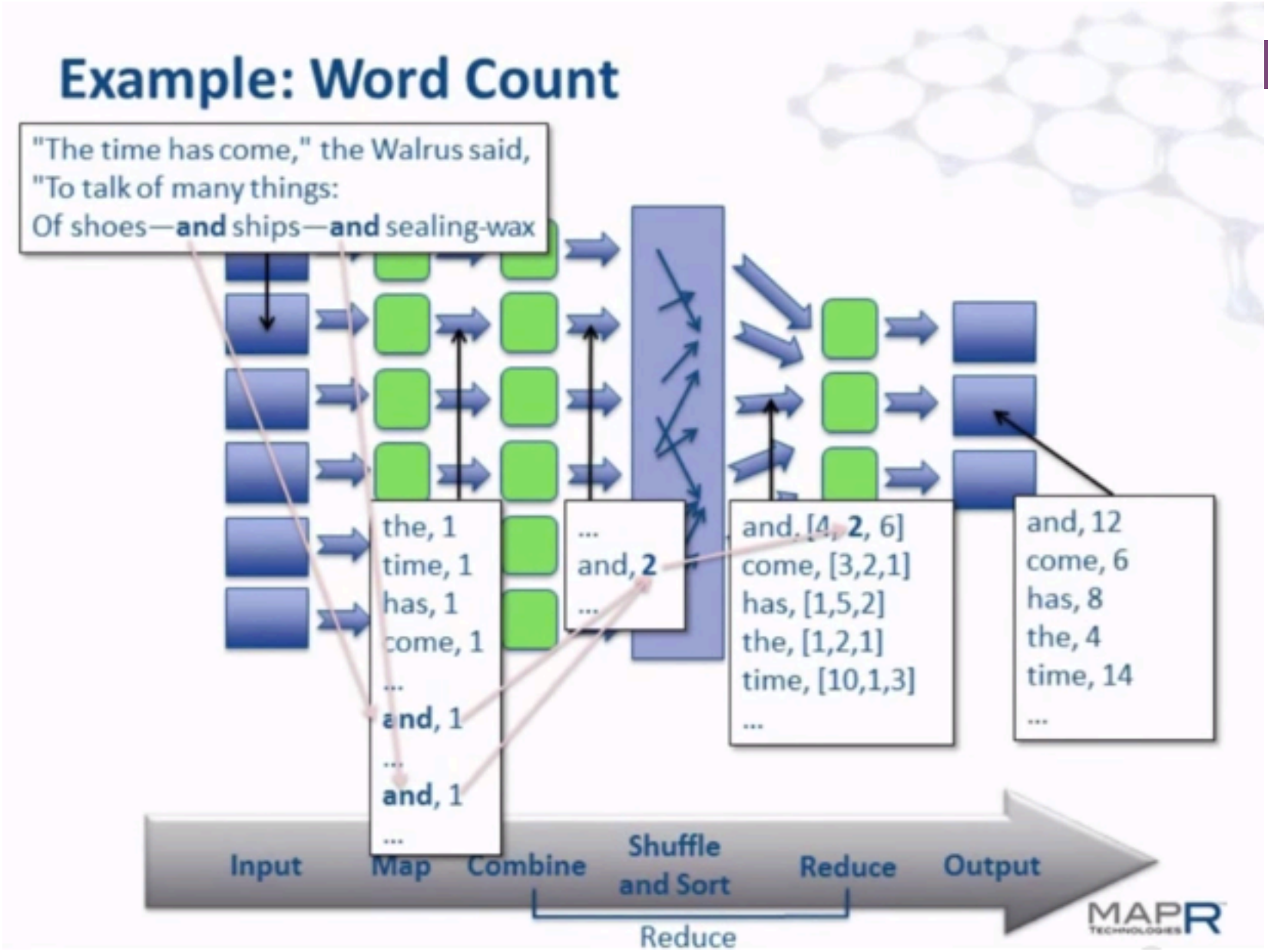

Example: Word Count
"The time has come," the Walrus said,
"To talk of many things:
Of shoes—and ships—and sealing-wax
Key-value pair
Input splits
the, 1
time, 1
has, 1
come, 1
...
and, 1
...
and, 1
...
...
and, 2
...
and, [4, 2, 6]
come, [3,2,1]
has, [1,5,2]
the, [1,2,1]
time, [10,1,3]
...
and, 12
come, 6
has, 8
the, 4
time, 14
...
Input
Map
Combine
Shuffle and Sort
Reduce
Output
Reduce
MAPR TECHNOLOGIES



Reduce phase will sum the values to create a reduced representation. Thus, multiple instances of same key are combined.



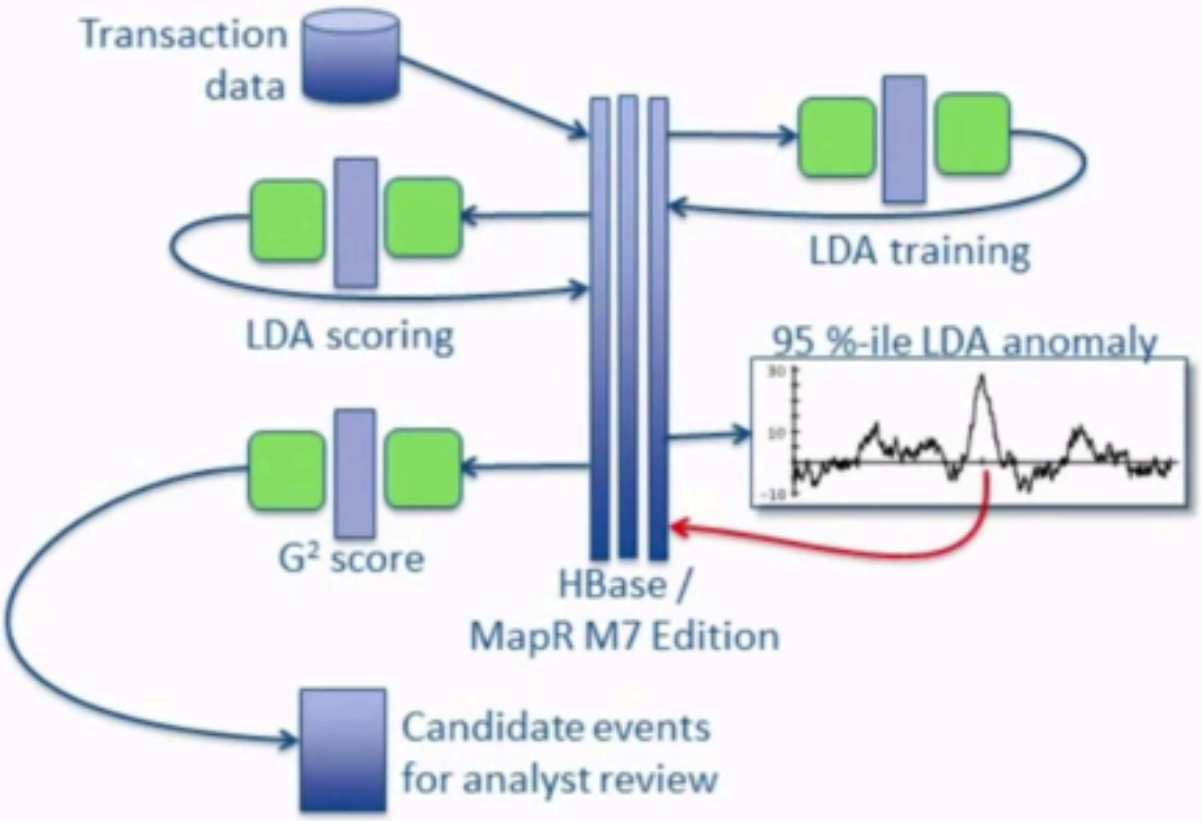
Shuffle and Sort: Gather all instances of similar keys from all the tasks
In {and, [4, **2,** 6]} the other values are from other books/tasks



Compute final result and save to disk.

Variation: Multiple MapReduces

Example: Fraud Detection in User Transactions



MapReduce - word count example

```
function map(String name, String document):
  for each word w in document:
    emit(w, 1)

function reduce(String word, Iterator partialCounts):
  totalCount = 0
  for each count in partialCounts:
    totalCount += count
  emit(word, totalCount)
```

MapReduce - Java API

- **Mapper**:

```
void map(WritableComparable key,
         Writable value,
         OutputCollector output,
         Reporter reporter)
```

- **Reducer**:

```
void reduce(WritableComparable key,
            Iterator values,
            OutputCollector output,
            Reporter reporter)
```

What about failed tasks?

- Tasks will fail
- JT will retry failed tasks up to N attempts
- After N failed attempts for a task, job fails
- Some tasks are slower than other
- Speculative execution is JT starting up multiple of the same task
- First one to complete wins, other is killed

Credit: http://www.flickr.com/photos/phobia/2308371224/

MapReduce is not good for...

- Jobs that need shared state/coordination
  - Tasks are shared-nothing
  - Shared-state requires scalable state store
- Low-latency jobs
- Jobs on small datasets
- Finding individual records

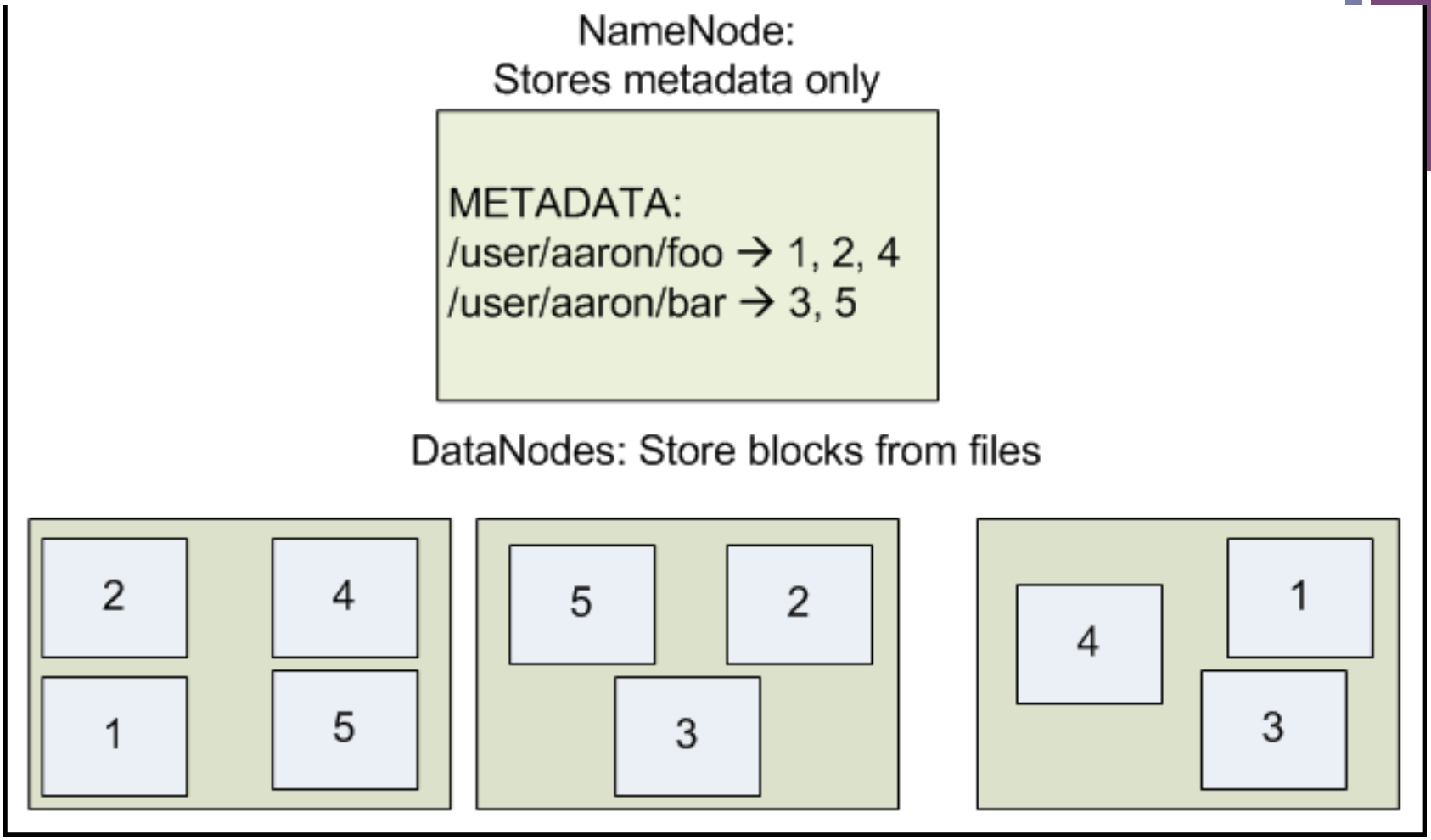
Hadoop Distributed File System

- Designed to hold large amounts of data and provide access to this data to many clients across network
- Hadoop DFS is designed to handle data spread across multiple machines
- Data redundancy: If individual machines fail, data still should be available
- Provides fast and scalable access to this information. Can add machines in the cluster while maintaining integrity of data
- Works well with Hadoop MapReduce framework

Trade-offs of HDFS

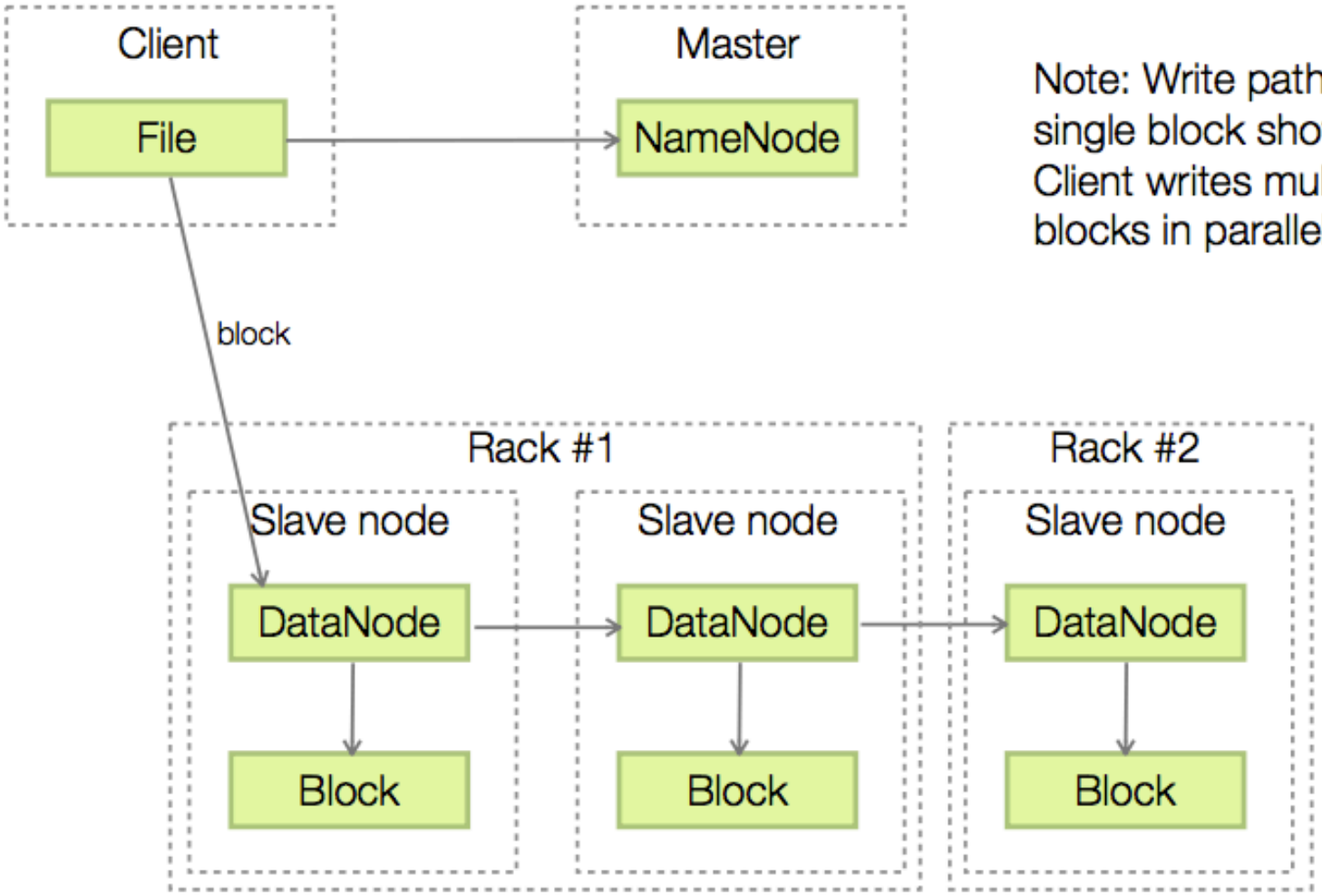
- Applications are assumed to perform long sequential streaming reads from files
- Data will be written to the HDFS once and then read several times; updates to files after they have already been closed is not supported
- Does not provide mechanism for local caching: Just re-read the data from disk
- Designed based on Google File System
- Cannot interact with files using normal Unix tools like: ls, cp, mv. It has a separate namespace
- The management information is handled by a single machine. It has redundant information to protect it from failure of that machine.

Hadoop Distributed File System



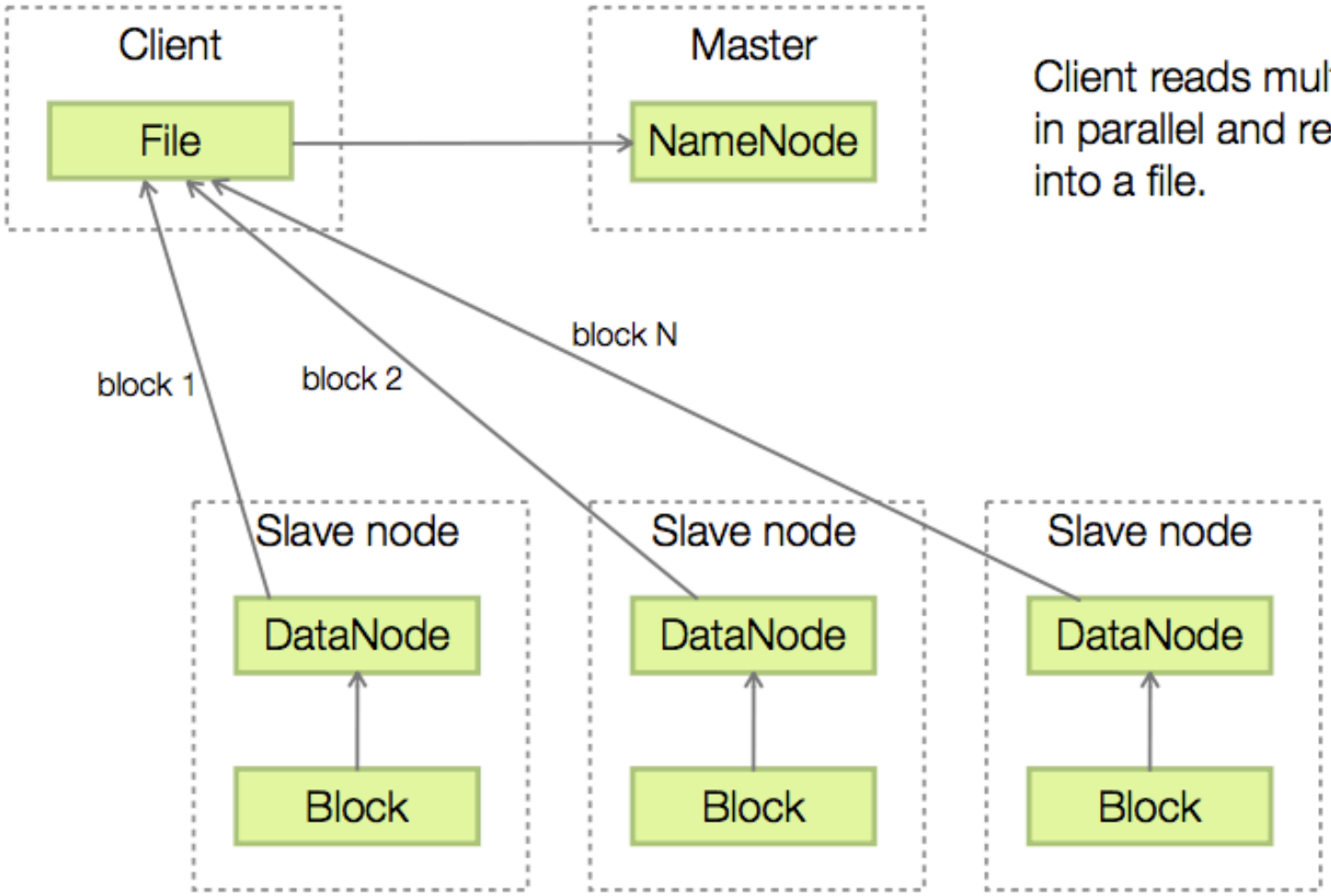
DataNodes holding blocks of multiple files with a replication factor of 2.
The NameNode maps filenames into block ids.
This redundancy in information helps when individual nodes fail

HDFS - writes



Note: Write path for a single block shown. Client writes multiple blocks in parallel.

HDFS - reads



Client reads multiple blocks in parallel and re-assembles into a file.

What about DataNode failures?

- DNs check in with the NN to report health
- Upon failure NN orders DNs to replicate under-replicated blocks

Credit: http://www.flickr.com/photos/18536761@N00/367661087/

Hadoop Map-Reduce and HDFS: Advantages

- Distribute data *and* computation
  - Computation local to data avoids network overload
- Tasks are independent
  - Easy to handle partial failures - entire nodes can fail and restart
  - Avoid crawling horrors of failure-tolerant synchronous distributed systems
  - Speculative execution to work around stragglers
- Linear scaling in the ideal case
  - Designed for cheap, commodity hardware
- Simple programming model
  - The “end-user” programmer only writes map-reduce tasks

Amazon ElasticMapReduce (EMR)

- Amazon comes to your rescue again
- Super easy to use. Generate keypair and you are good to go
- Can access API in multiple languages.
- You can start with a 10-node Hadoop cluster and scale your application
- Similar web console for launching EMR

Pig

- Pig is a platform for analyzing large dataset
- Pig lets you specify a sequence of data transformations such as merging data sets, filtering them and applying functions to records
- Purpose of Pig is to answer queries over semi-structured data such as log files
- Pig is high-level language for writing queries over this sort of data
- Programming language used to write Pig queries is called *Pig Latin*

What is Pig?

- Pig is a **scripting language**
  - No compiler
  - Rapid prototyping
  - Command line prompt (grunt shell)
- Pig is a **domain specific language**
  - No control flow (no if/then/else)
  - Specific to data flows
    - Not for writing ray tracers
    - For the distribution of a pre-existing ray tracer

Pig Latin Datatypes

- An **atom** is atomic value (e.g. "fish"). (similar to string in python)
- A **tuple** is a record of multiple values with fixed arity e.g. ("dog","sparky") (similar to tuple in python)
- A **data bag** is collection of arbitrary number of values {("dog","sparky"), ("fish","goldie")} (similar to list in python but with differences)
- A **data map** is collection with a lookup function translating to keys and values e.g. ['age': 25] (similar to dictionary in python)

Pig and MapReduce

- **MapReduce requires programmers**
  - Must think in terms of map and reduce functions
  - More than likely will require Java programmers
- **Pig provides high-level language that can be used by**
  - Analysts
  - Data Scientists
  - Statisticians
  - Etc...
- **Originally implemented at Yahoo! to allow analysts to access data**

Pig's Features

- Join Datasets
- Sort Datasets
- Filter
- Data Types
- Group By
- User Defined Functions
- Etc..

Pig's Use Cases

- **Extract Transform Load (ETL)**
  - Ex: Processing large amounts of log data
    - clean bad entries, join with other data-sets
- **Research of "raw" information**
  - Ex. User Audit Logs
  - Schema maybe unknown or inconsistent
  - Data Scientists and Analysts may like Pig's data transformation paradigm

Pig Components

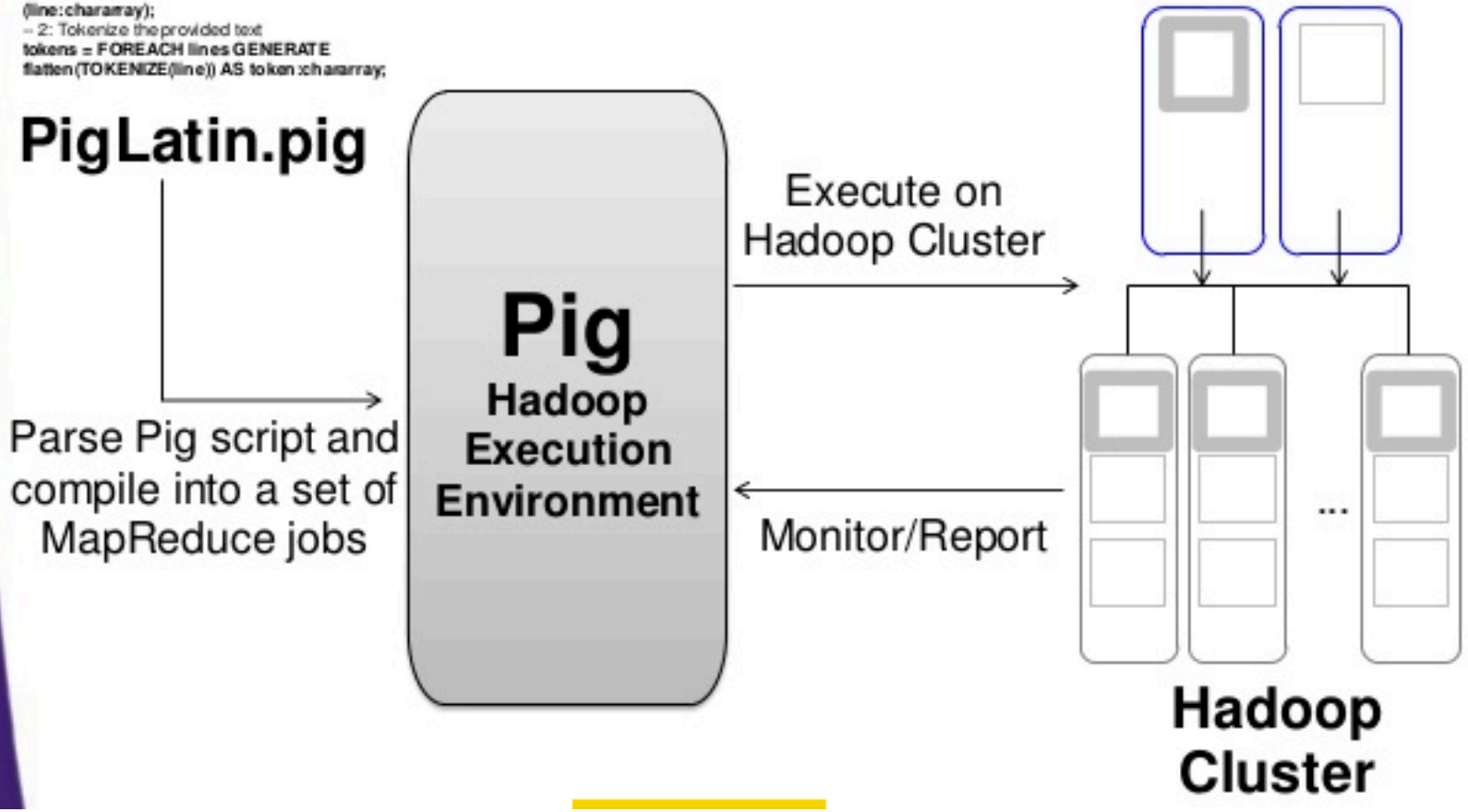
- **Pig Latin**
  - Command based language
  - Designed specifically for data transformation and flow expression
- **Execution Environment**
  - The environment in which Pig Latin commands are executed
  - Currently there is support for Local and Hadoop modes
- **Pig compiler converts Pig Latin to MapReduce**
  - Compiler strives to optimize execution
  - You automatically get optimization improvements with Pig updates

Execution Modes

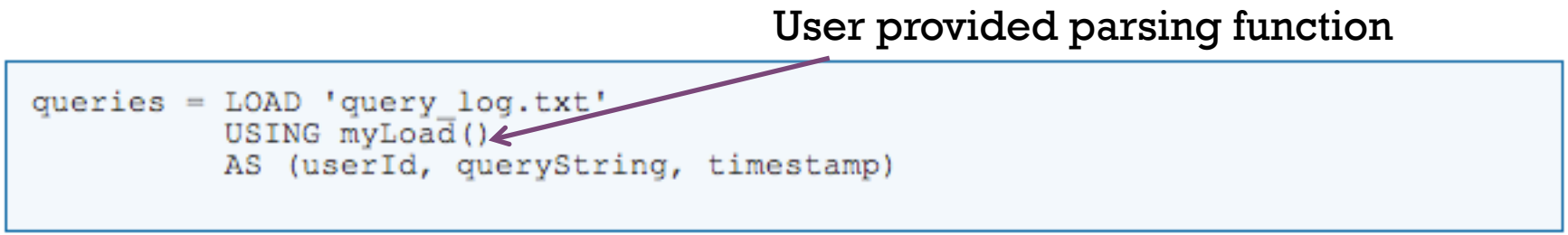
- **Local**
  - Executes in a single JVM
  - Works exclusively with local file system
  - Great for development, experimentation and prototyping
- **Hadoop Mode**
  - Also known as MapReduce mode
  - Pig renders Pig Latin into MapReduce jobs and executes them on the cluster
  - Can execute against semi-distributed or fully-distributed hadoop installation

Hadoop Mode

```
-- 1: Load text into a bag, where a row is a line of
text
lines = LOAD '/training/playArea/hamlet.txt' AS
(line:chararray);
-- 2: Tokenize the provided text
tokens = FOREACH lines GENERATE
flatten(TOKENIZE(line)) AS token:chararray;
```



Loading Data in Pig



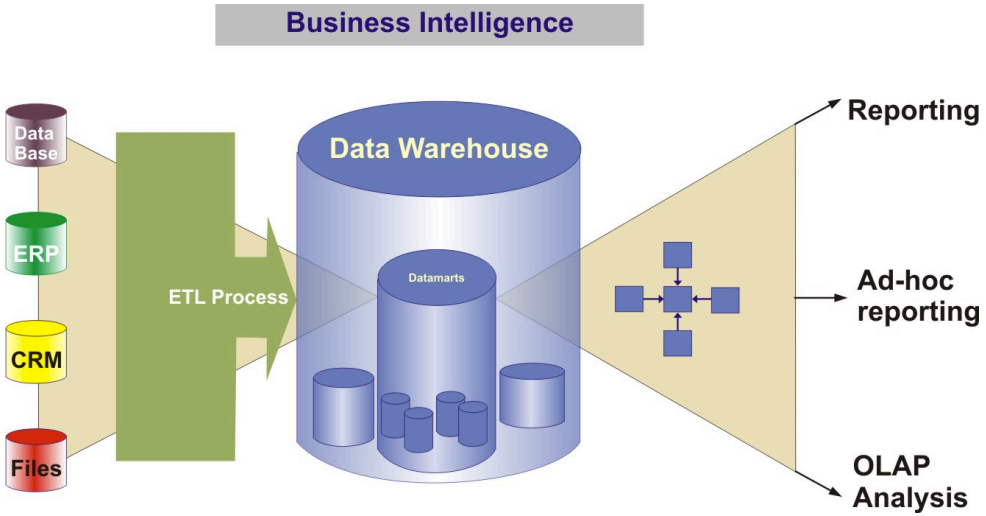
The user defined function need not be provided. A default (PigStorage()) exists, but Pig provides you an option if you want to use it.

Pig Latin – Diagnostic Tools

- **Display the structure of the Bag**
  - grunt> DESCRIBE <bag_name>;
- **Display Execution Plan**
  - Produces Various reports
    - Logical Plan
    - MapReduce Plan
  - grunt> EXPLAIN <bag_name>;
- **Illustrate how Pig engine transforms the data**
  - grunt> ILLUSTRATE <bag_name>;

Business Intelligence Tools

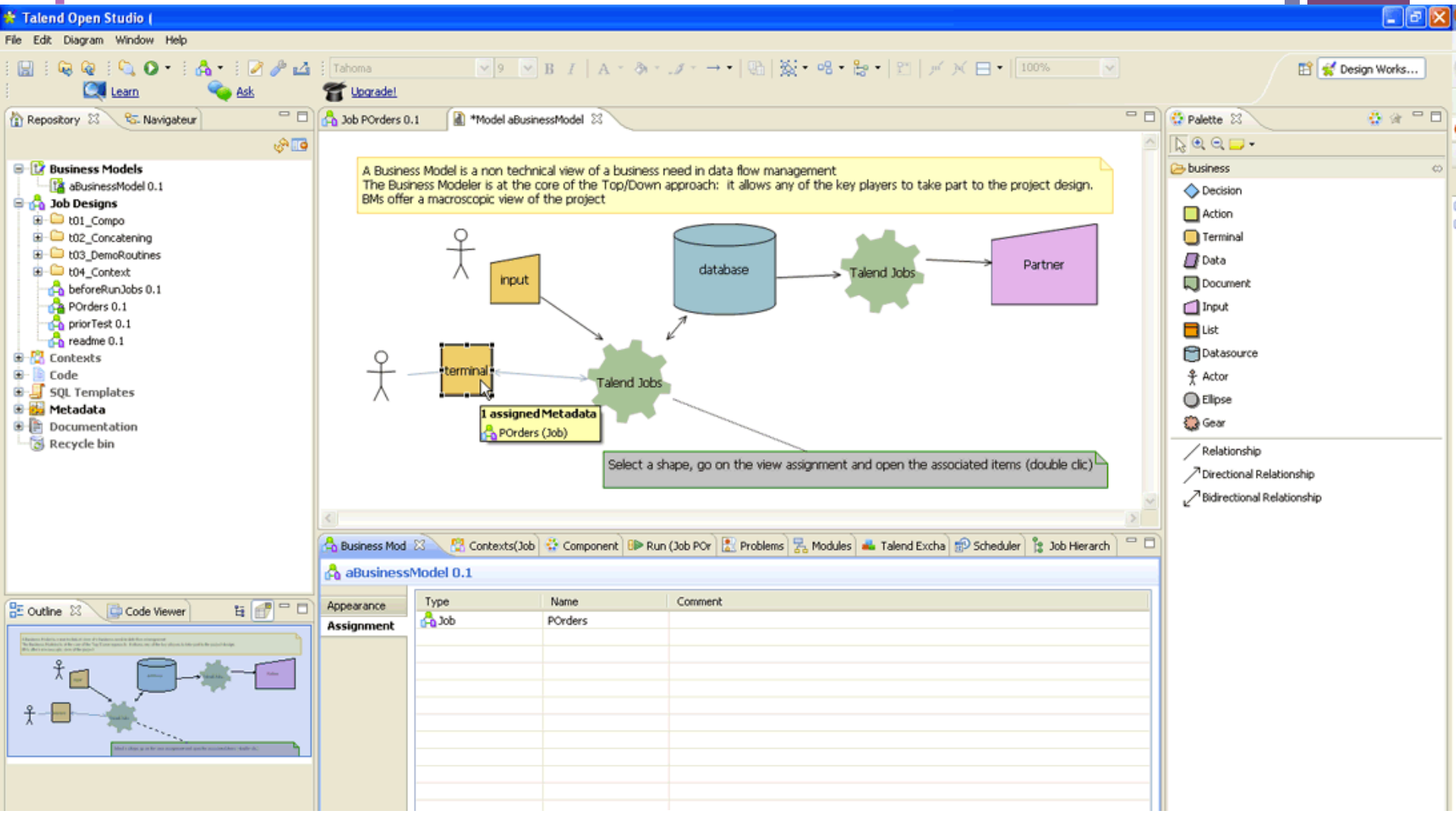
- Lots of them
  - Jaspersoft, Excel,Talend, Penthao, RapidMiner, KNIME, etc.
- Application software designed to retrieve, analyze and report data
- Mostly visual. Geared towards enterprise applications
- Many domain specific/application specific/proprietary
-



Features of Business Intelligence Tools

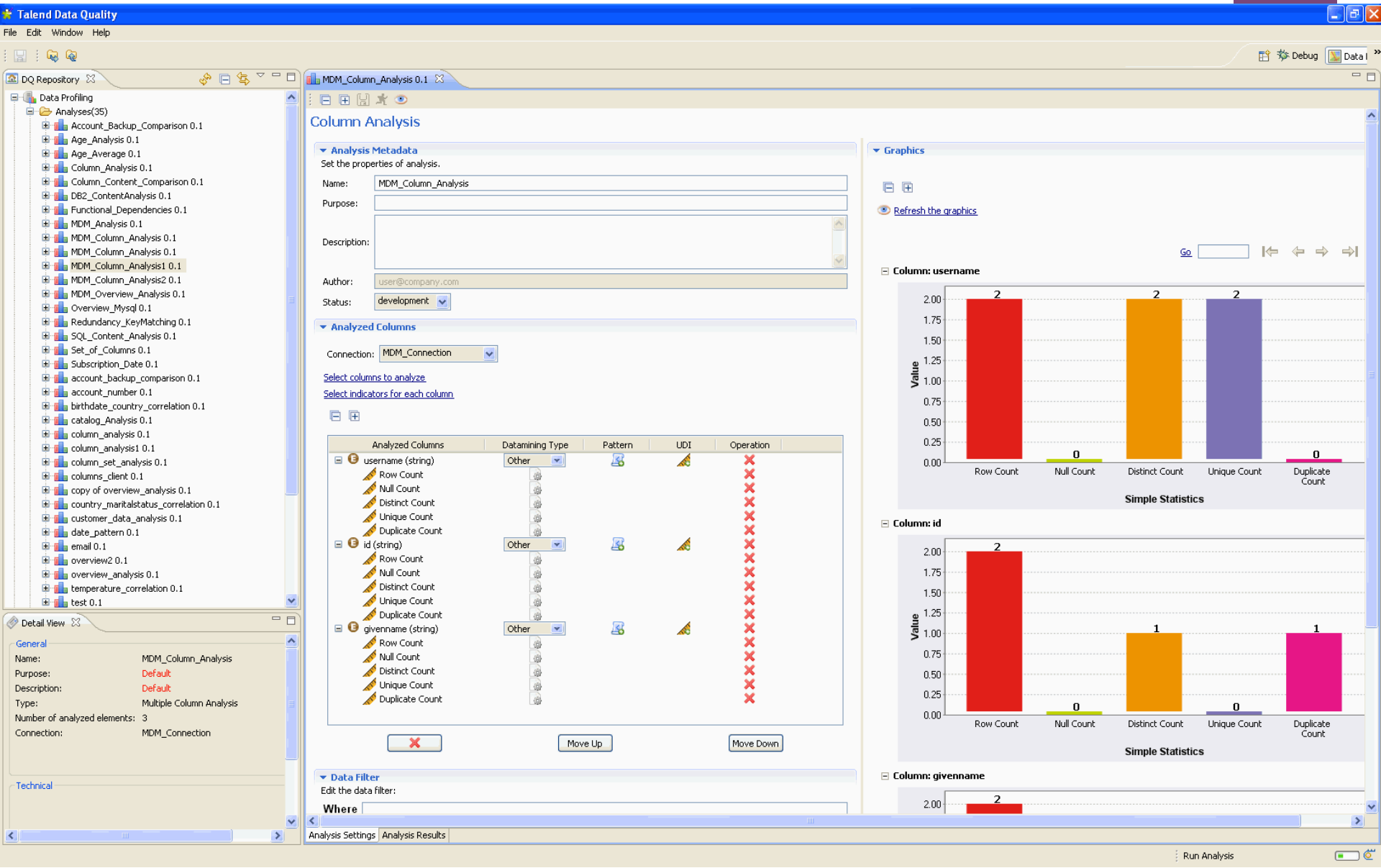
- Data management strategy
- Analytics, Reporting, scorecard and strategy management
- Highly advanced/specialized calculation engines, business user experience
- "What-if" analysis to develop applications that can forecast business performance
- Can operate of thousands of simultaneous users and terabytes of information
- GUI based interface

talend*



Allows application integration, cloud integration

SONY Allianz ebay Virgin mobile



Talend: analyze results

OBI11g SSD Win 2003 VMware Remote Console Devices

Oracle BI Interactive Dashboards - Windows Internet Explorer

http://siebelsrv:9704/analytics/saw.dll?Dashboard&PortalPath=%2Fusers%2Fweblogic%2F_portal&Page=Sales

File Edit View Favorites Tools Help

Summary of Servers - bifoun... | Oracle BI Interactive Das...

ORACLE Business Intelligence

Search All | Advanced | Administration | Help | Sign Out

My Dashboard | Home | Catalog | Dashboards | New | Open | Signed In As weblogic

Sales | Marketing | Finance

CompoundLayoutTest

Brand	LOB	LOB	Product Type	Target Quantity	Target
BizTech	Communication	Communication	Cell Phones	56,875	606,986
			Smart Phones	54,723	562,496
	Electronics	Electronics	Accessories	29,101	341,052
			Audio	72,237	751,938
FunPod	Digital	Digital	Camera	82,052	531,463
	Games	Games	Fixed	39,690	357,782
			Portable	50,620	444,068
HomeView	Services	Services	Install	7,373	68,013
			Maintenance	9,122	92,301
	TV	TV	LCD	33,456	364,720
			Plasma	37,582	379,180

CompoundLayoutTest

Brand	LOB	LOB	Product Type	Target Quantity	Target
BizTech	Communication	Communication	Cell Phones	56,875	606,986
			Smart Phones	54,723	562,496
	Electronics	Electronics	Accessories	29,101	341,052
			Audio	72,237	751,938
FunPod	Digital	Digital	Camera	82,052	531,463
	Games	Games	Fixed	39,690	357,782
			Portable	50,620	444,068
HomeView	Services	Services	Install	7,373	68,013
			Maintenance	9,122	92,301
	TV	TV	LCD	33,456	364,720
			Plasma	37,582	379,180

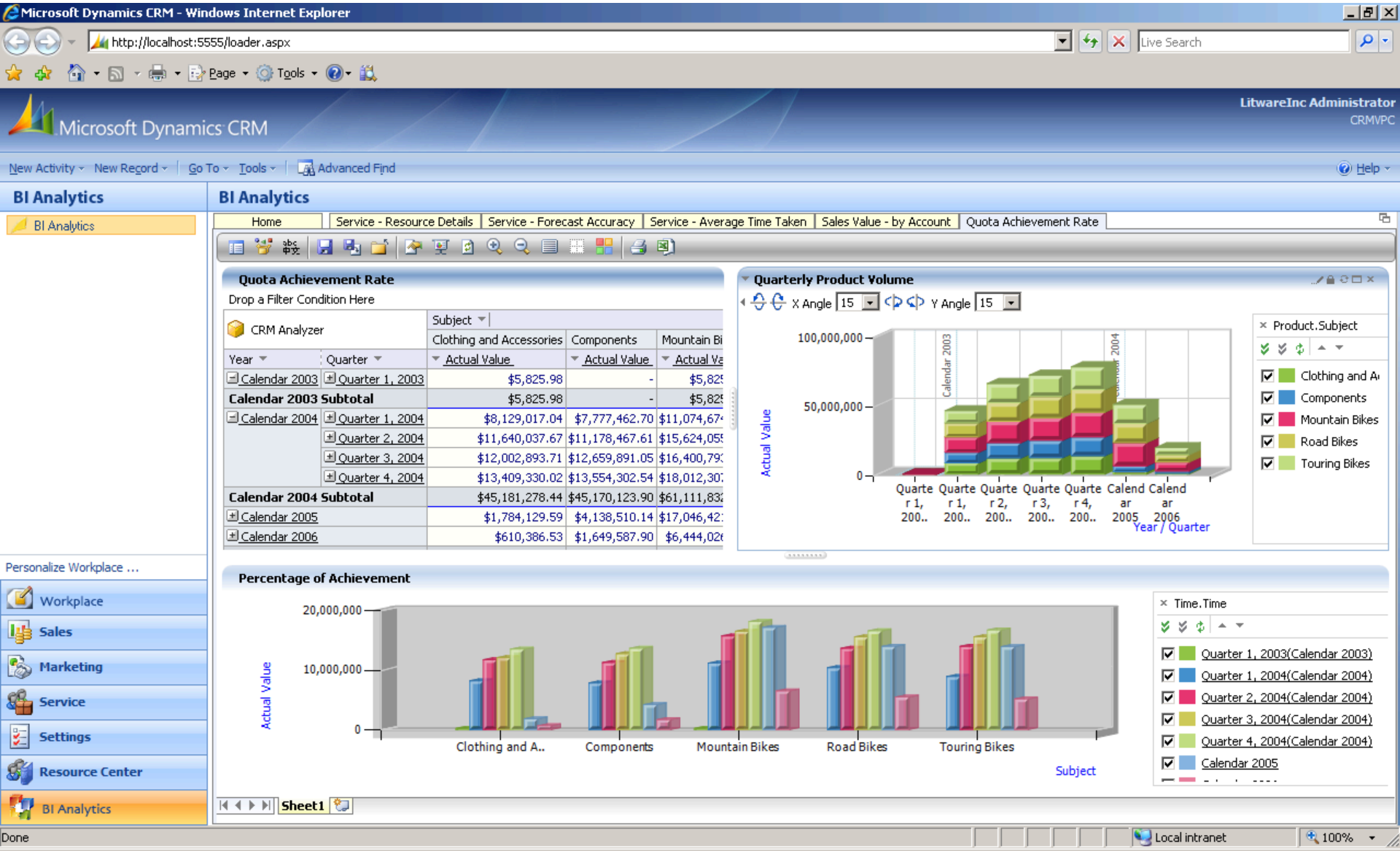
Oracle Business Intelligence Suite

Done | Trusted sites | 100%

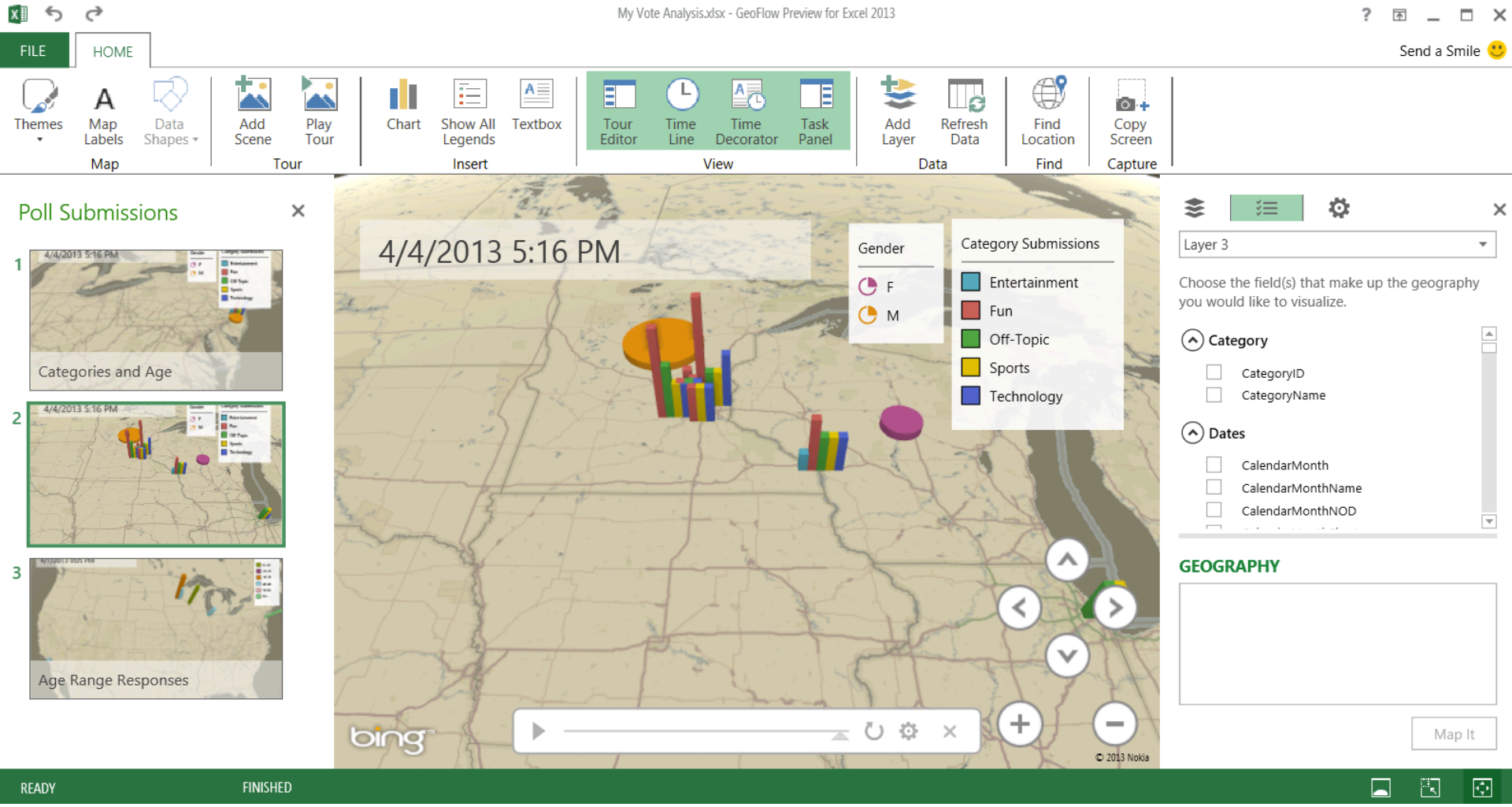
Start | Command Prompt - start... | Oracle BI Interactive ... | 3:30 AM

To direct input to this virtual machine, press Ctrl+G.

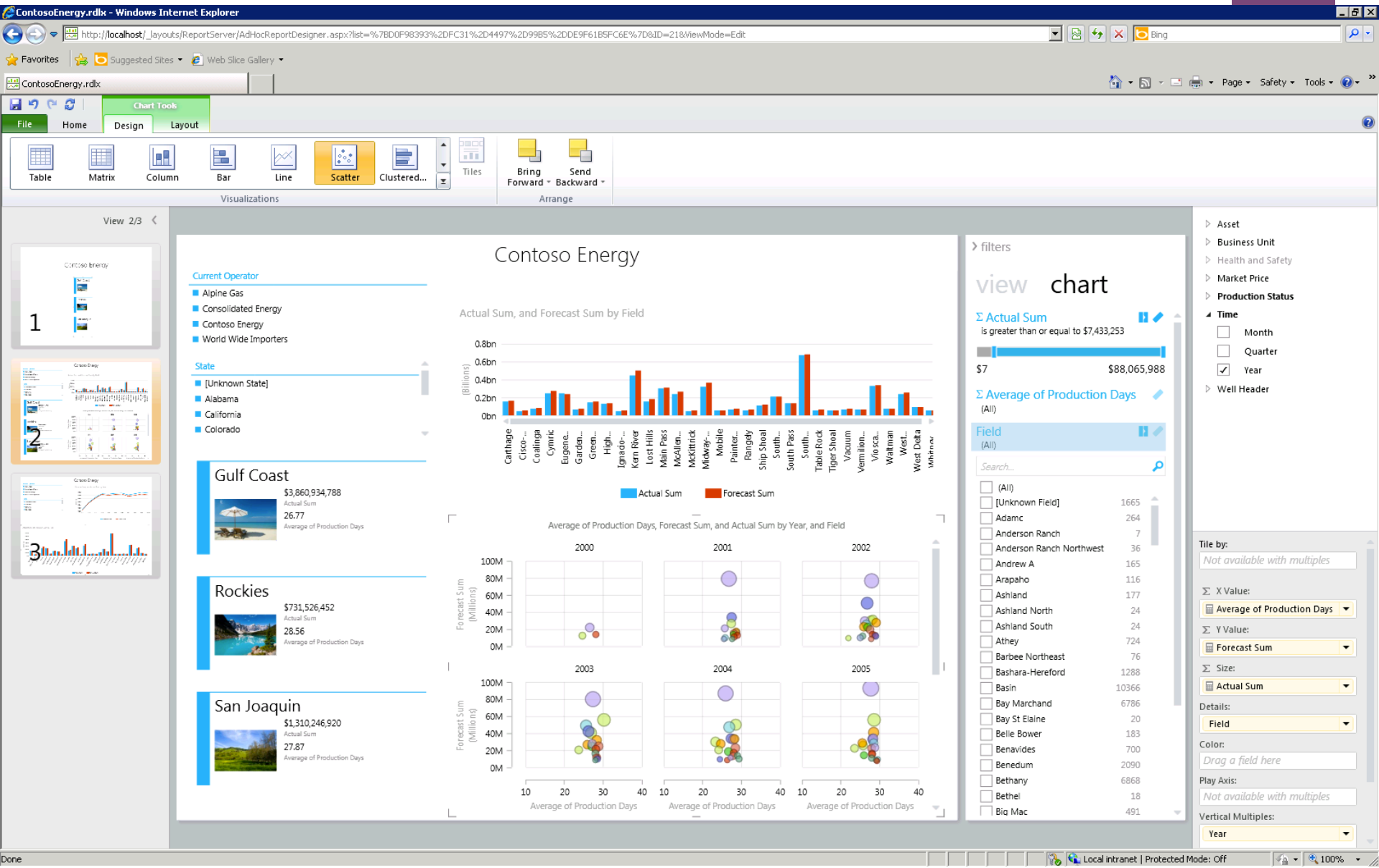
TR 14:31 17.10.2010



Microsoft Business Intelligence



Microsoft Power Maps

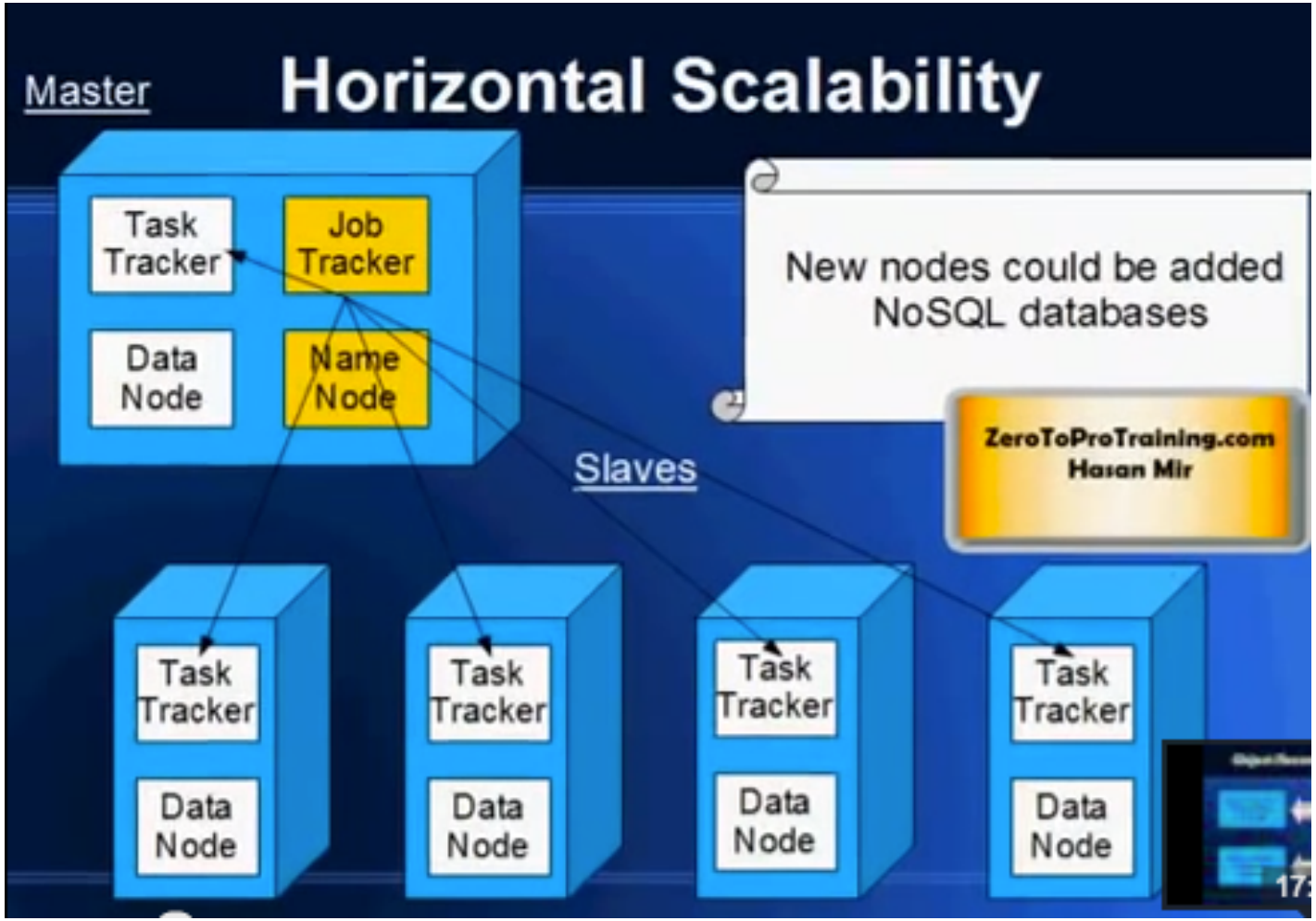


Microsoft Power View

Storing Documents …

Document Store

- Document oriented database is a computer program designed for storing, retrieving and managing document oriented information
- E.g. formats: XML, YAML, JSON, BSON
- Allows structured queries and retrieval. Makes storing and retrieving data easier.
- Lots of packages available for reading such documents
- **Keys:** Documents are addressed in the database via a unique key that represents that document. Database retains an index on the key to speed up document retrieval
- **Retrieval**: Database offers an API or query language
- E.g. Cassandra, CouchDB, MongoDB, Lotus Notes etc.
- RDBMS/Flat File Systems cannot handle big data. Cannot handle horizontal scalability. Hence, NoSQL databases



Keep on adding more computers as you need more compute power i,e.
Performance is linearly proportional to no. of computers.
Relational Databases cannot handle horizontal scalability

Structured Documents

personal.xml

```xml
<personnel>
    <person id="Big.Boss">
        <name>
            <family>Boss</family>
            <given>Big</given>
        </name>
        <email>chief@oxygenxml.com</email>
        <link subordinates="one.worker"/>
    </person>
    <person id="one.worker">
        <name>
            <family>Worker</family>
            <given>One</given>
        </name>
        <email>one@oxygenxml.com</email>
        <link manager="Big.Boss"/>
    </person>
    <person id="two.worker">
        <name>
```

personal.json

```json
{"personnel": {"person": [
    {
        "id": "Big.Boss",
        "name": {
            "family": "Boss",
            "given": "Big"
        },
        "email": "chief@oxygenxml.com",
        "link": {"subordinates": "one.worker"}
    },
    {
        "id": "one.worker",
        "name": {
            "family": "Worker",
            "given": "One"
        },
        "email": "one@oxygenxml.com",
        "link": {"manager": "Big.Boss"}
    },
```

XML
Extensible Markup Language

JSON
JavaScript Object Notation

MongoDB

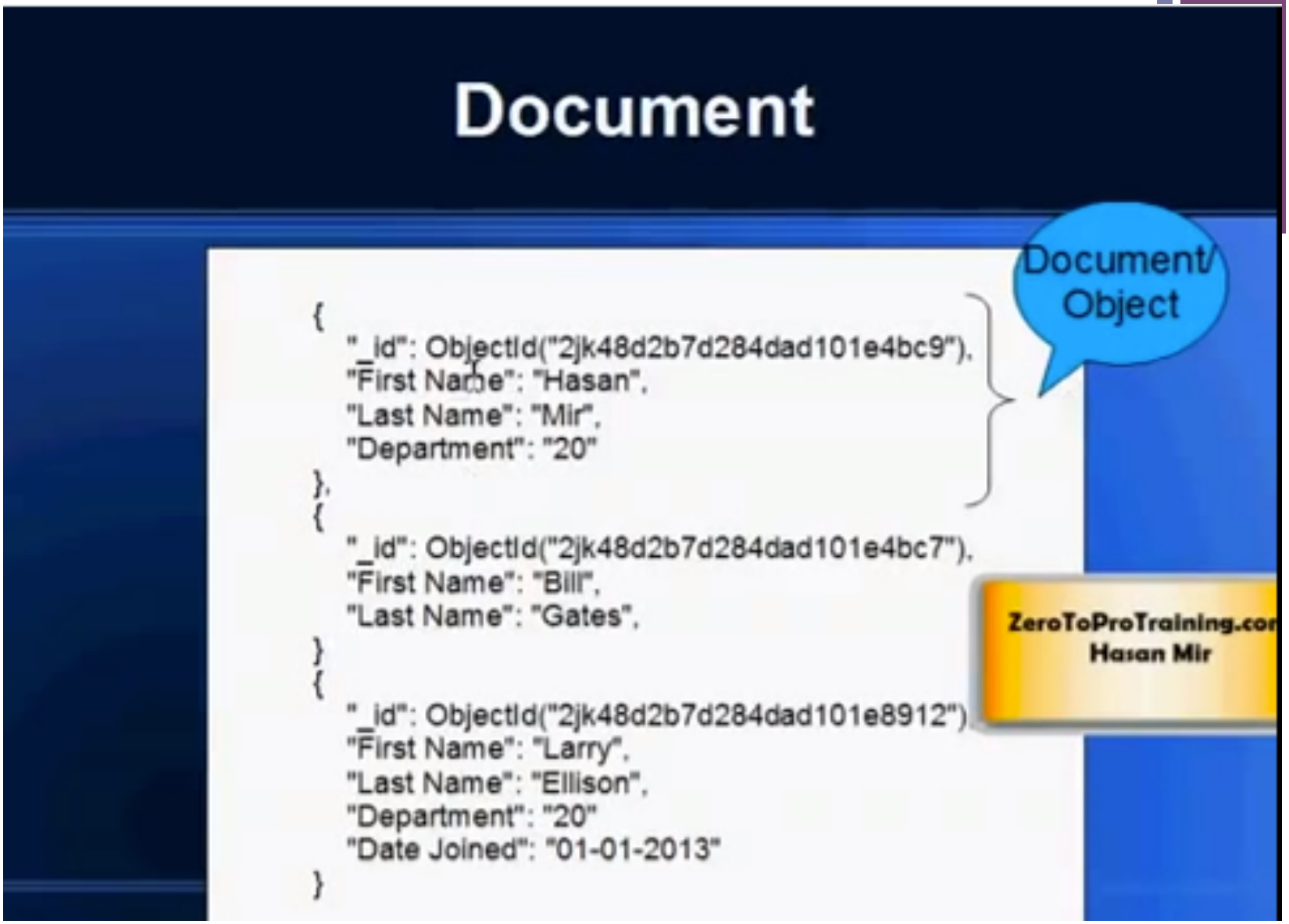
- Humongous
- Document oriented database using JSON document syntax
- Features:
  - Flexibilty
  - Power
  - Scaling
  - Ease of Use
  - Built-in Javascript
- Clientele: Craigslist, eBay, Foursquare, SourceForge, and The New York Times.

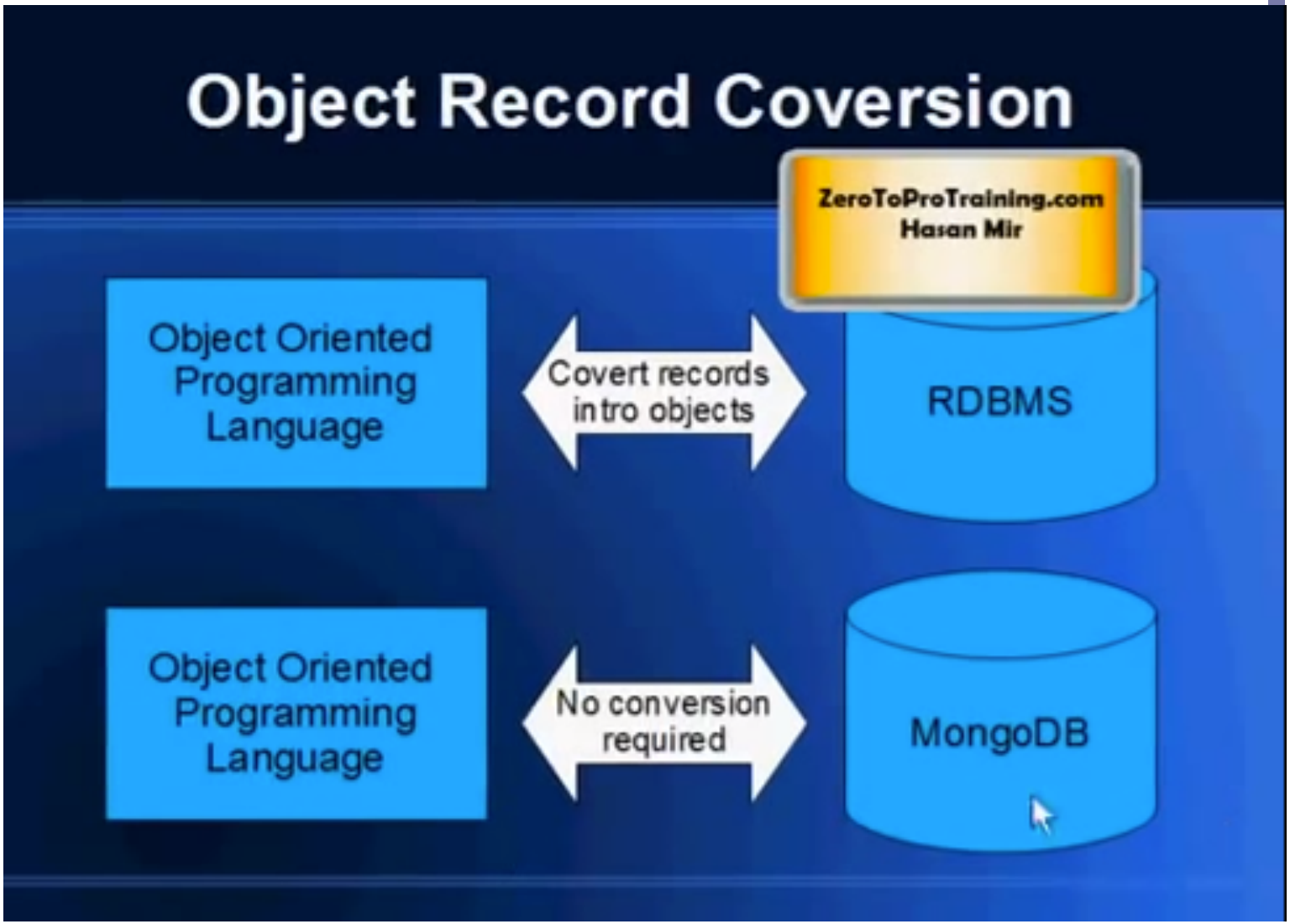
MongoDB

- A record in MongoDB is a document, which is a data structure composed of field and value pairs
- MongoDB documents are similar to JSON objects. It is a NoSQL database
- Advantages:
  - Documents (i.e. objects) correspond to native data-types in many programming languages
  - Embedded documents and arrays reduce need for expensive joins
  - Allows Map-Reduce programming model. Written in C++ and open-source. Uses replication to maintain data consistency/ availability

```
{
  name: "sue",                          <-- field: value
  age: 26,                              <-- field: value
  status: "A",                          <-- field: value
  groups: [ "news", "sports" ]          <-- field: value
}
```

Document


{
"_id": ObjectId("2jk48d2b7d284dad101e4bc9"),
"First Name": "Hasan",
"Last Name": "Mir",
"Department": "20"
},
{
"_id": ObjectId("2jk48d2b7d284dad101e4bc7"),
"First Name": "Bill",
"Last Name": "Gates",
}
{
"_id": ObjectId("2jk48d2b7d284dad101e8912"),
"First Name": "Larry",
"Last Name": "Ellison",
"Department": "20"
"Date Joined": "01-01-2013"
}
Document/
Object
ZeroToProTraining.com
Hasan Mir



Works with Java, JavaScript, Python, Ruby, C#, PHP, C++ etc.

MongoDB Operations

codecentric

MongoDB - Cheat Sheet
Version 1.0 / 27th December 2012 / Page 1 of 4

More information can be found from the MongoDB Tutorial
@ http://blog.codecentric.de/en/2012/12/mongodb-tutorial/

About this Cheat Sheet

The idea behind this is to have all (well, most) information from the above mentioned Tutorial immediately available in a very compact format. All commands can be used on a small data basis created in the insert-section. All information in this sheet comes **without the slightest warranty for correctness**. Use at your own risk. Have fun ☺!

Basic Information

Download MongoDB	http://www.mongodb.org/downloads
JSON Specification	http://www.json.org/
BSON Specification	http://bsonspec.org/
Java Tutorial	http://www.mongodb.org/display/DOCS/Java+Tutorial

Inserting Documents

```
db.ships.insert({name:'USS Enterprise-D',operator:'Starfleet',type:'Explorer',class:'Galaxy',crew:750,codes:[10,11,12]})
db.ships.insert({name:'USS Prometheus',operator:'Starfleet',class:'Prometheus',crew:4,codes:[1,14,17]})
db.ships.insert({name:'USS Defiant',operator:'Starfleet',class:'Defiant',crew:50,codes:[10,17,19]})
db.ships.insert({name:'IKS Buruk',operator:' Klingon Empire',class:'Warship',crew:40,codes:[100,110,120]})
db.ships.insert({name:'IKS Somraw',operator:' Klingon Empire',class:'Raptor',crew:50,codes:[101,111,120]})
db.ships.insert({name:'Scimitar',operator:'Romulan Star Empire',type:'Warbird',class:'Warbird',crew:25,codes:[201,211,220]})
db.ships.insert({name:'Narada',operator:'Romulan Star Empire',type:'Warbird',class:'Warbird',crew:65,codes:[251,251,220]})
```

Finding Documents

`db.ships.findOne()`	Finds one arbitrary document
`db.ships.find().prettyPrint()`	Finds all documents and using nice formatting
`db.ships.find({}, {name:true, _id:false})`	Shows only the names of the ships
`db.ships.findOne({'name':'USS Defiant'})`	Finds one document by attribute

Basic Concepts & Shell Commands

`db.ships.<command>`	`db` – implicit handle to the used database `ships` – name of the used collection
`use <database>`	Switch to another database
`show collections`	Lists the available collections
`help`	Prints available commands and help

MongoDB Operations

codecentric

MongoDB - Cheat Sheet
Version 1.0 / 27th December 2012 / Page 2 of 4

More information can be found from the MongoDB Tutorial
@ http://blog.codecentric.de/en/2012/12/mongodb-tutorial/

Updating Documents

`db.ships.update({name : 'USS Prometheus'}, {name : 'USS Something'})`	Replaces the whole document
`db.ships.update({name : 'USS Something'}, {$set : {operator : 'Starfleet', class : 'Prometheus'}})`	sets / changes certain attributes of a given document
`db.ships.update({name : 'USS Something'}, {$unset : {operator : 1}})`	removes an attribute from a given document

Removing Documents

`db.ships.remove({name : 'USS Prometheus'})`	removes the document
`db.ships.remove({name:{$regex:'^USS\\sE'}})`	removes using operator

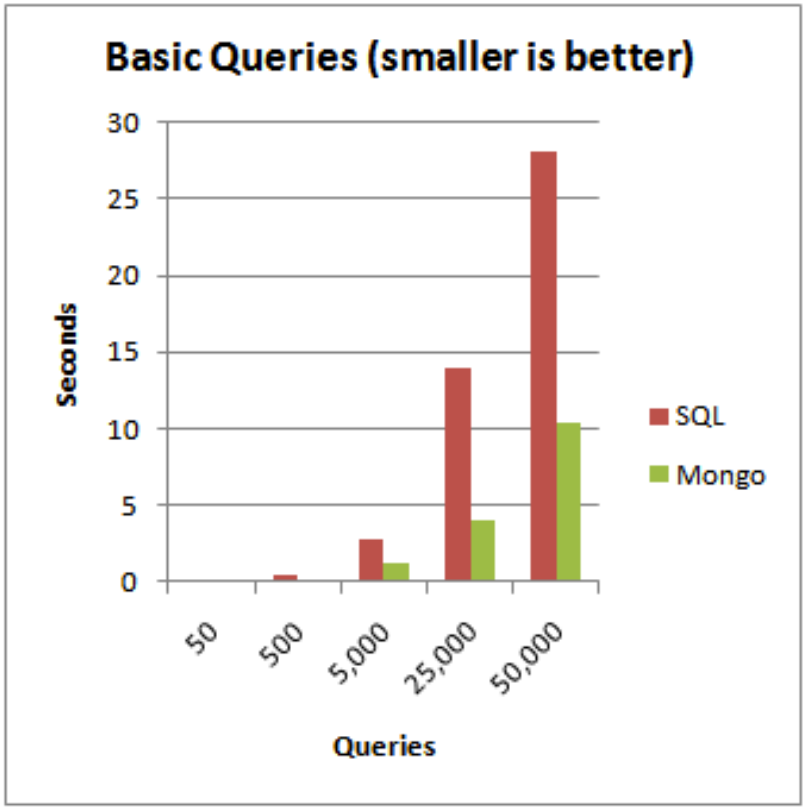
Each individual document removal is atomic with respect to a concurrent reader or writer. No client will see a document half removed.

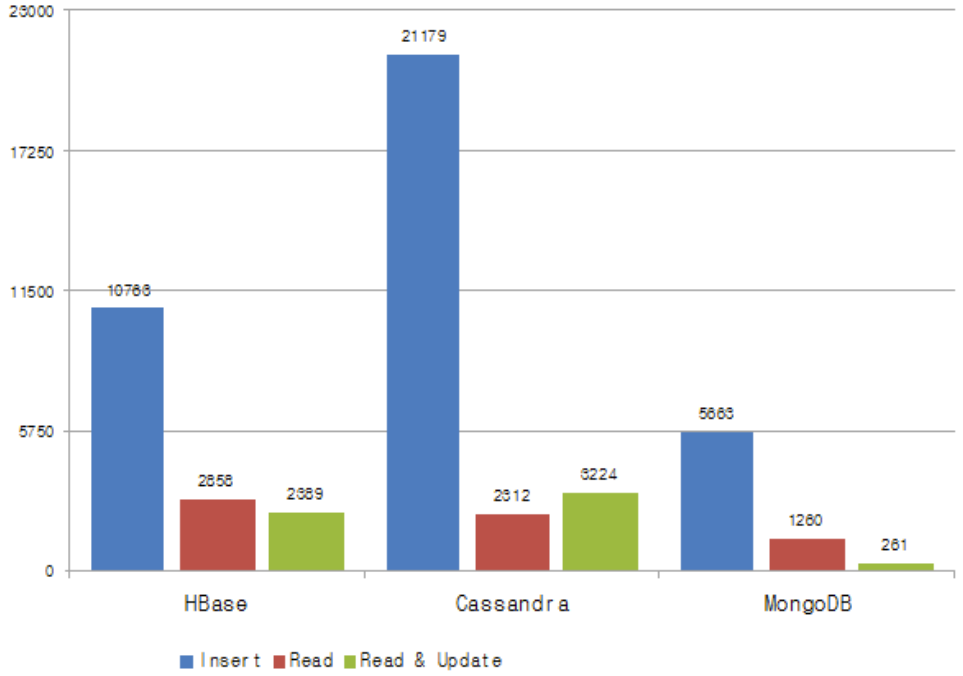
G+ Community Page:
https://plus.google.com/u/0/communities/115421122548465808444

Working with Indexes

Creating an index	`db.ships.ensureIndex({name : 1})`
Dropping an index	`db.ships.dropIndex({name : 1})`
Creating a compound index	`db.ships.ensureIndex({name : 1, operator : 1, class : 0})`
Dropping a compound index	`db.ships.dropIndex({name : 1, operator : 1, class : 0})`
Creating a unique compound index	`db.ships.ensureIndex({name : 1, operator : 1, class : 0}, {unique : true})`

Its is Fast..!


Basic Queries (smaller is better)
Seconds
30
25
20
15
10
5
0
50
500
5,000
25,000
50,000
Queries
SQL
Mongo


23000
17250
11500
5750
0
10766
2856
2389
21179
2312
3224
5666
1260
261
HBase
Cassandra
MongoDB
Insert
Read
Read & Update

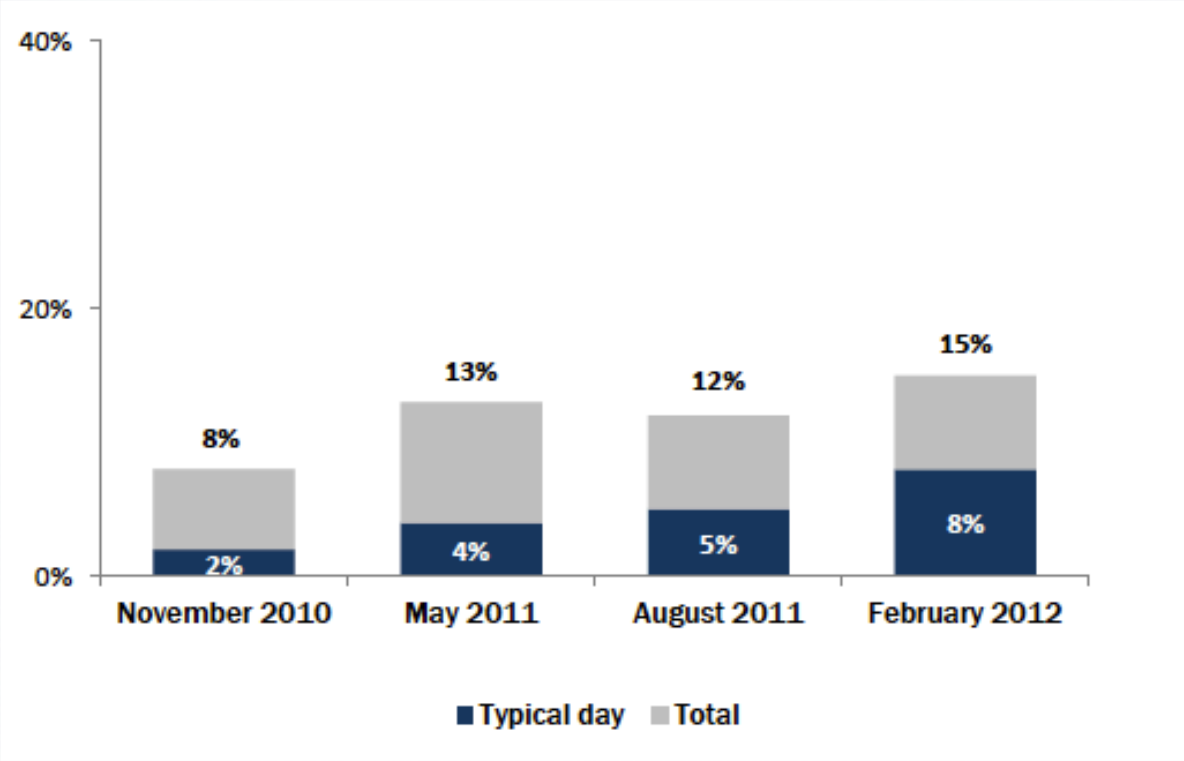
Twitter Analytics and Hadoop

WHAT IS TWITTER?

TWITTER STATS

Twitter usage over time

% of internet users who use Twitter



Source: Pew Research Center's Internet & American Life Project Winter 2012 Tracking Survey, January 20-February 19, 2012. N=2,253 adults age 18 and older, including 901 cell phone interviews. Interviews conducted in English and Spanish. Margin of error is +/-2.7 percentage points for internet users (n=1,729).

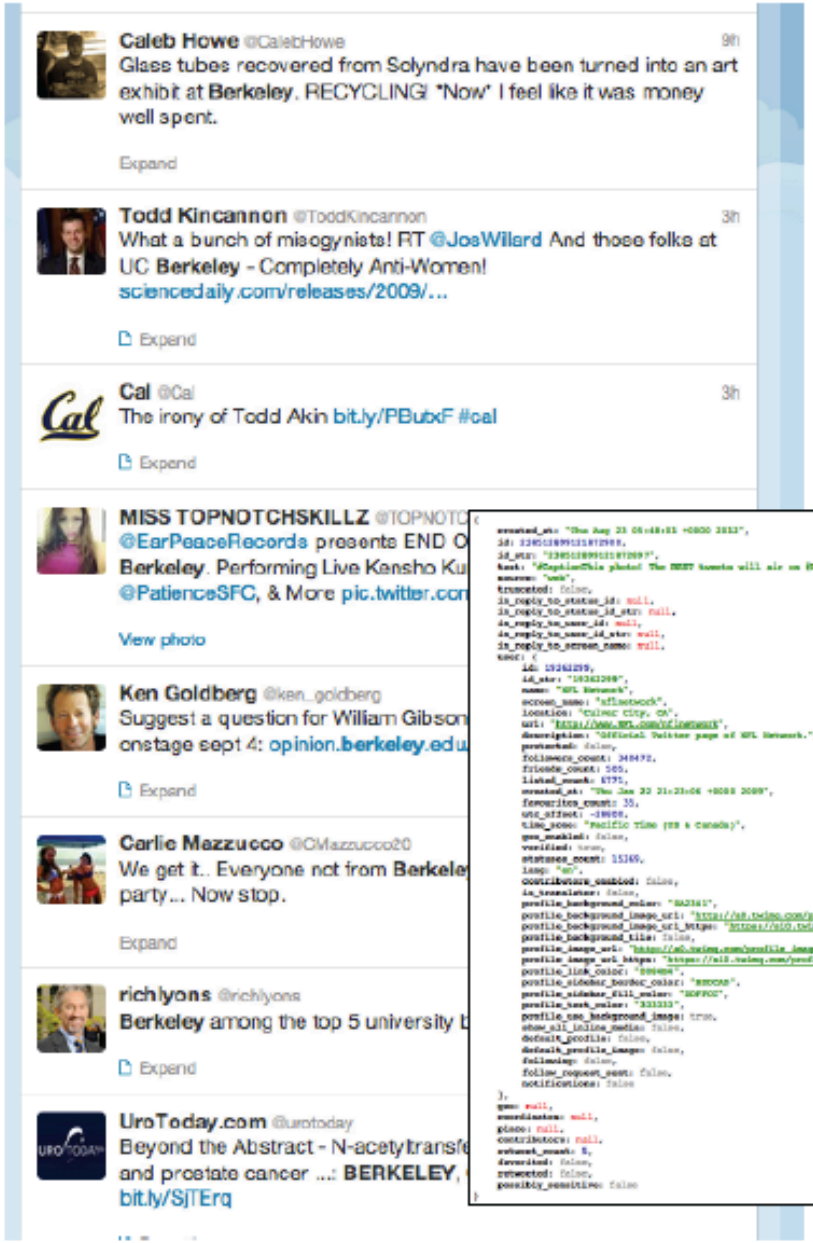
About Twitter

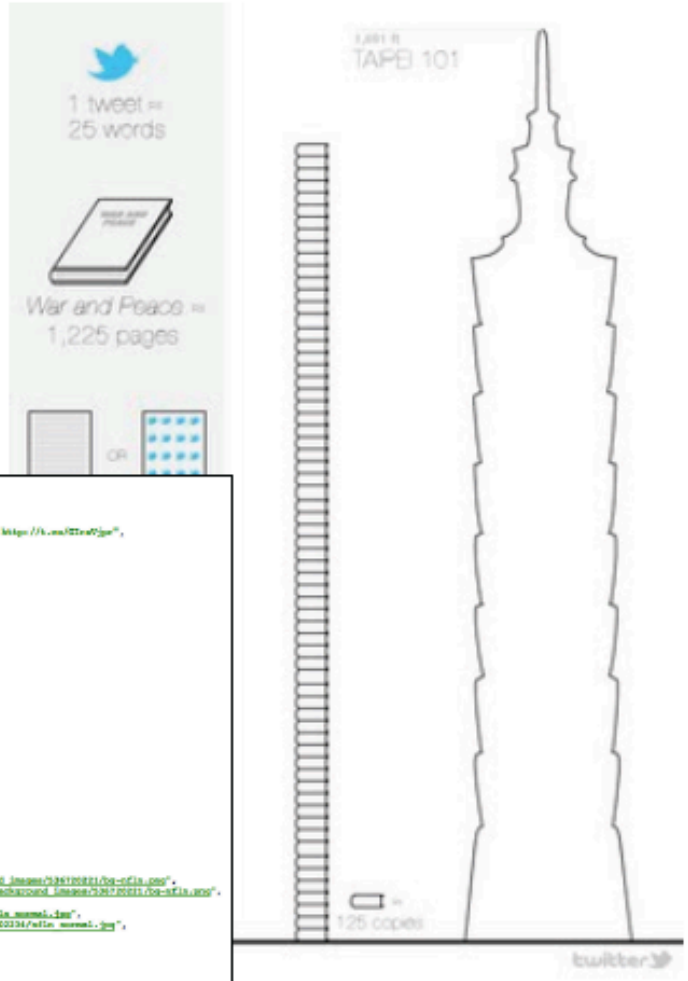
- The fastest, simplest way to communicate

- More than 140M active users
  - Majority (also) mobile; 60% out of U.S.
- More than 400M twitter.com visitors
- More than 400M tweets/day (peak: 25K/sec)

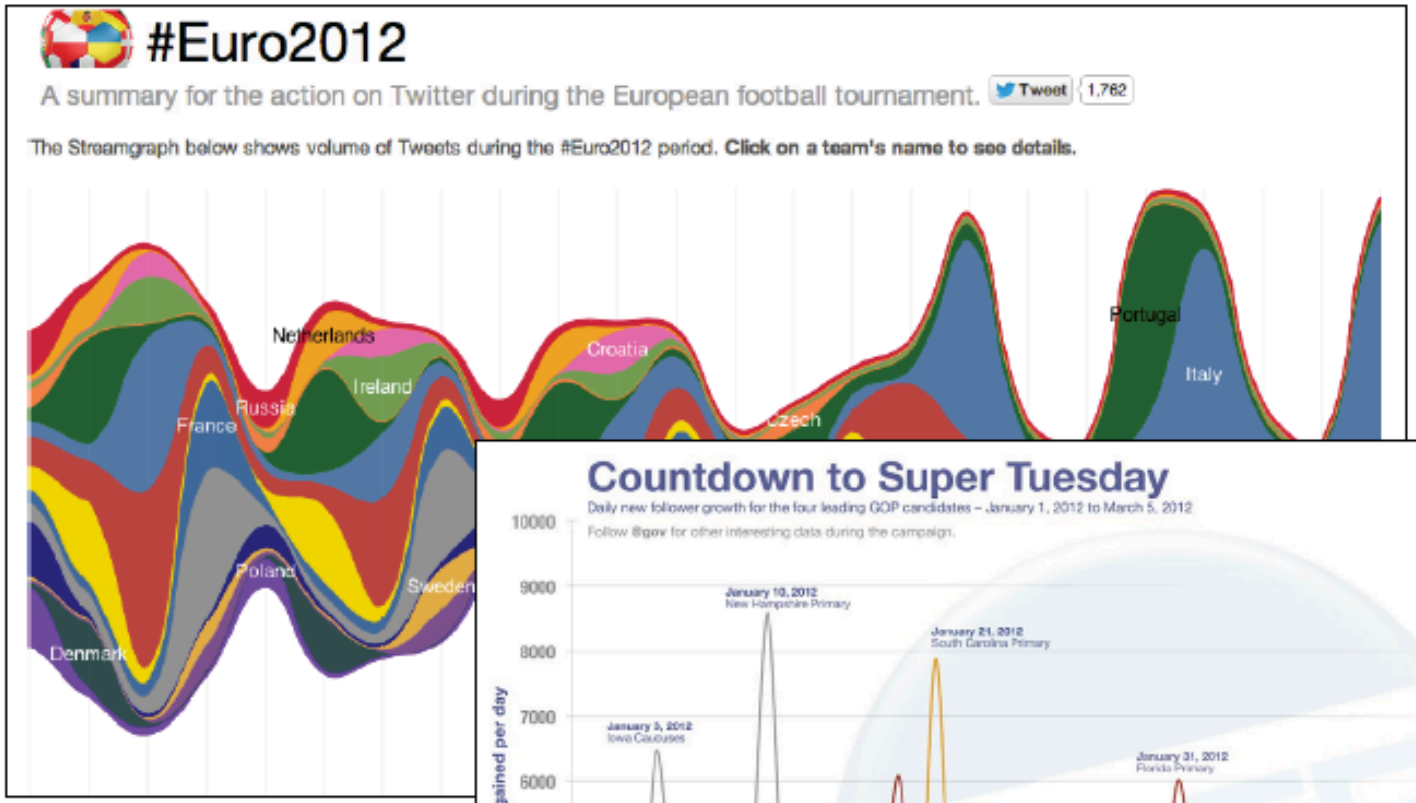
- 1,000 employees (majority in San Francisco)
  - 50% engineers

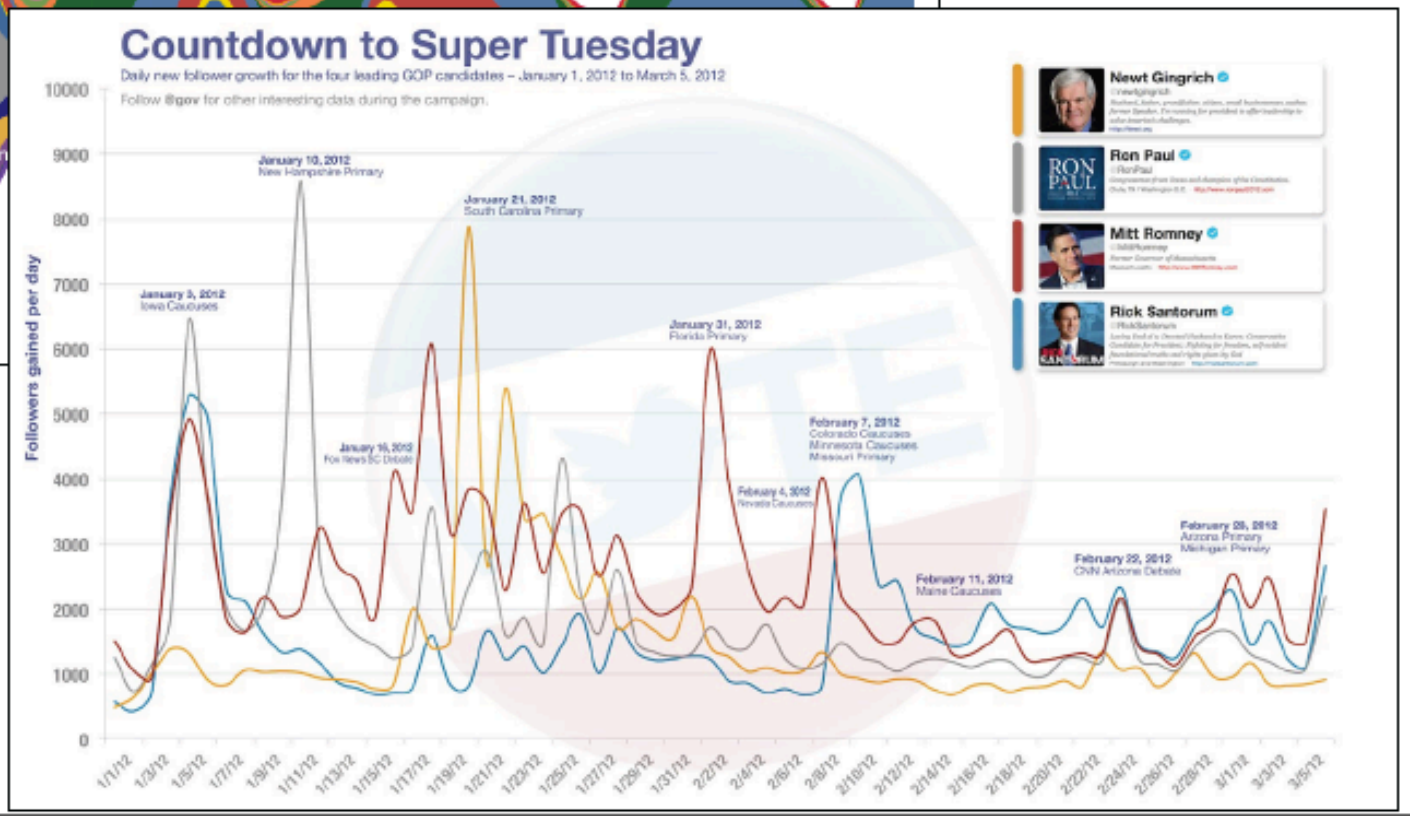
Twitter data: text



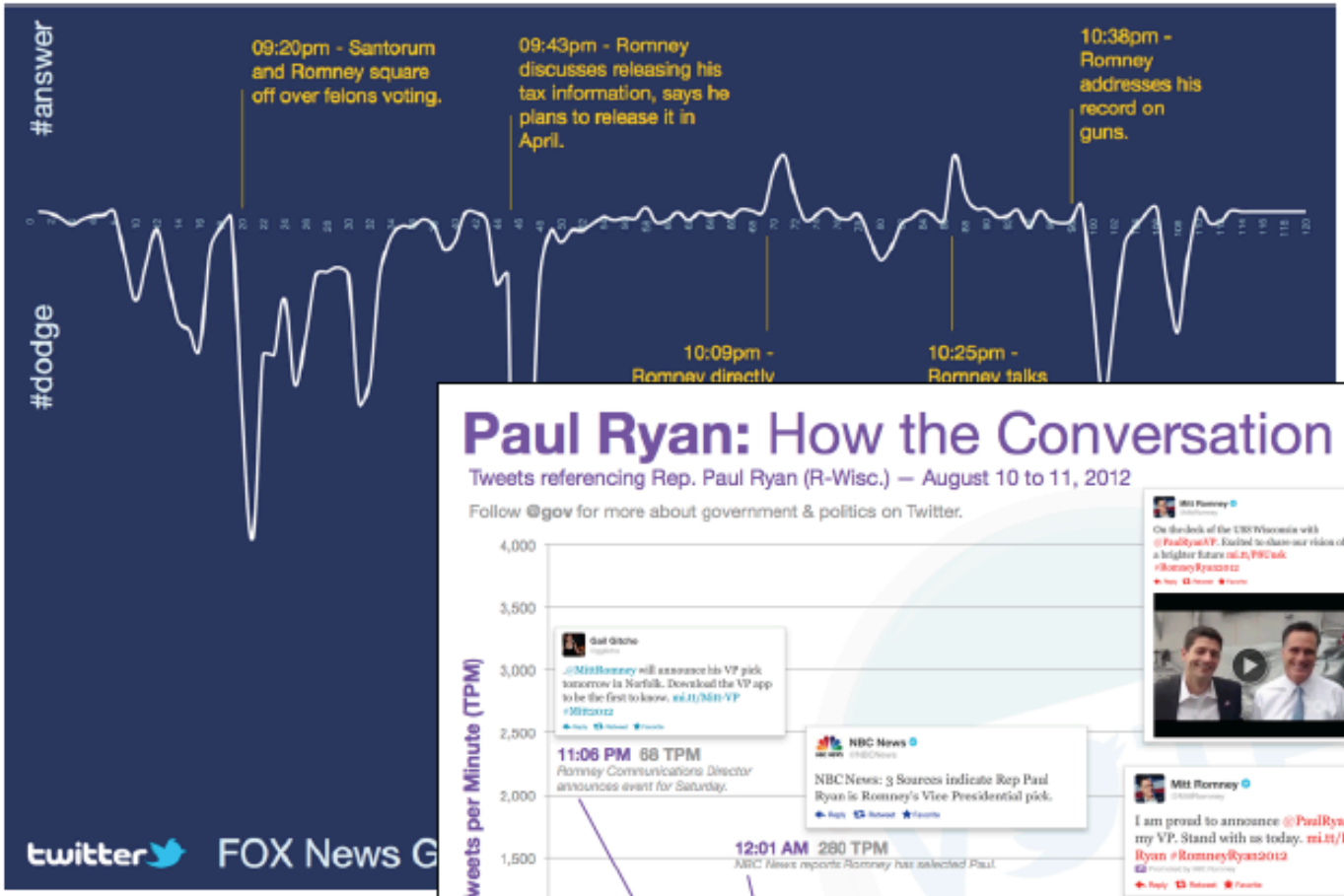


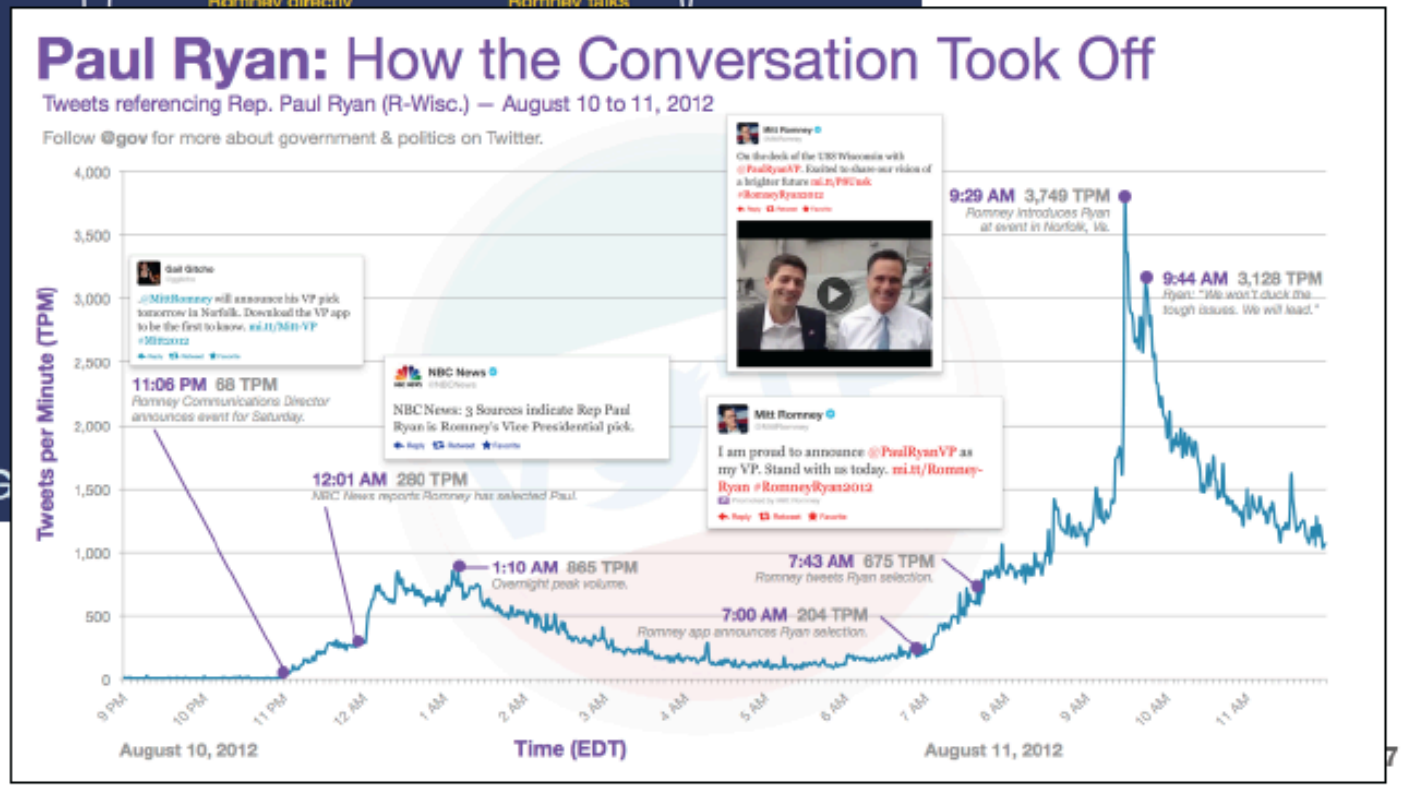
Twitter data: time series





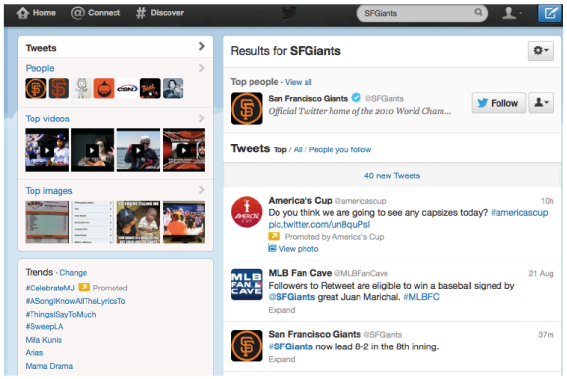
Combined: the pulse of the world



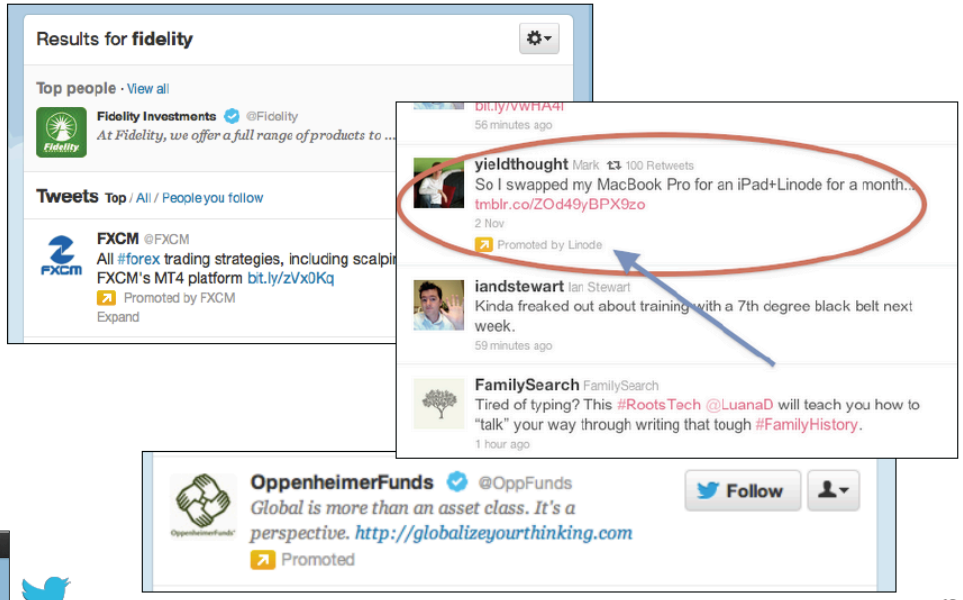


+ Examples of Analytic Tasks

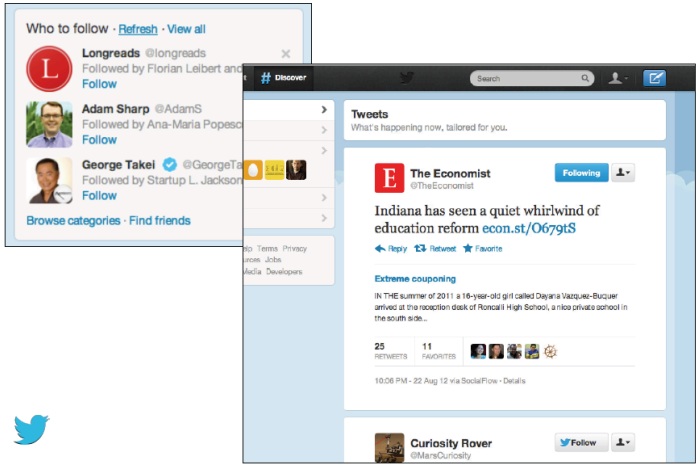
Search



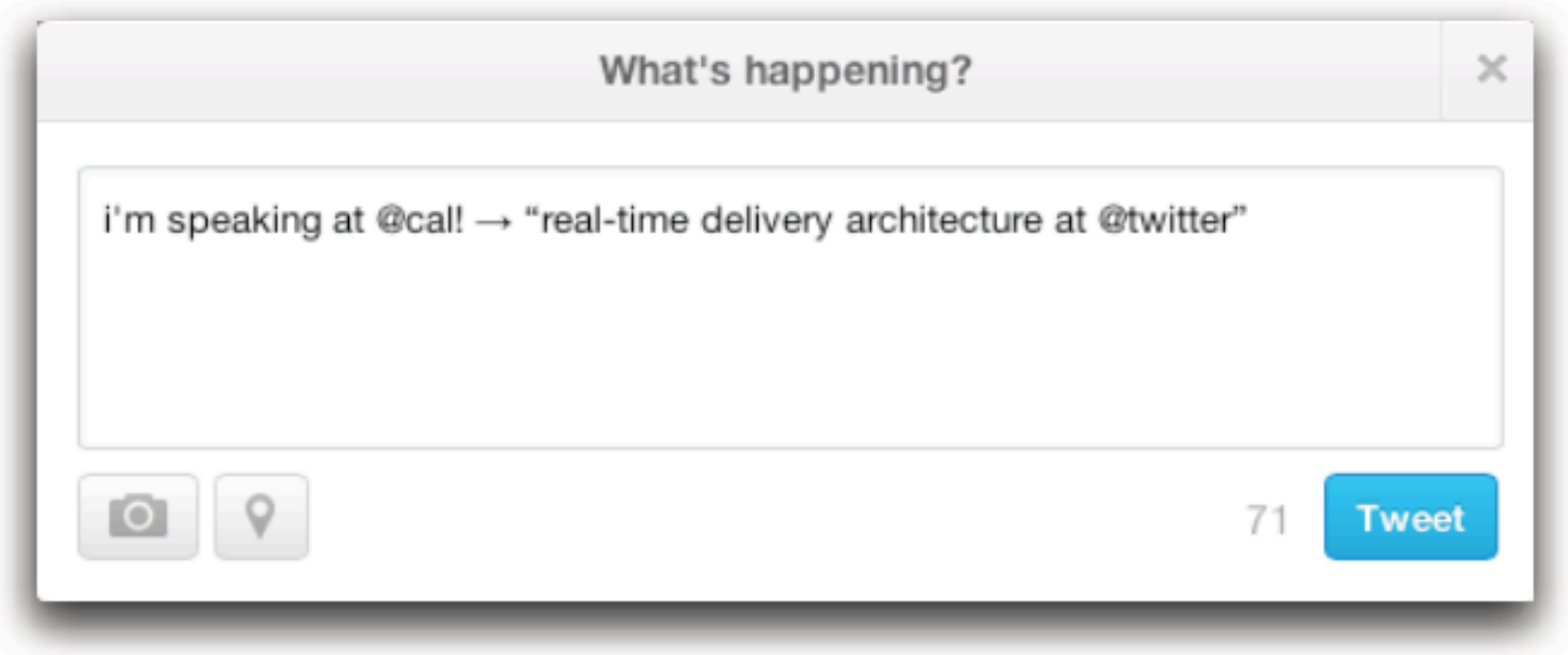
Ads



Recommendations

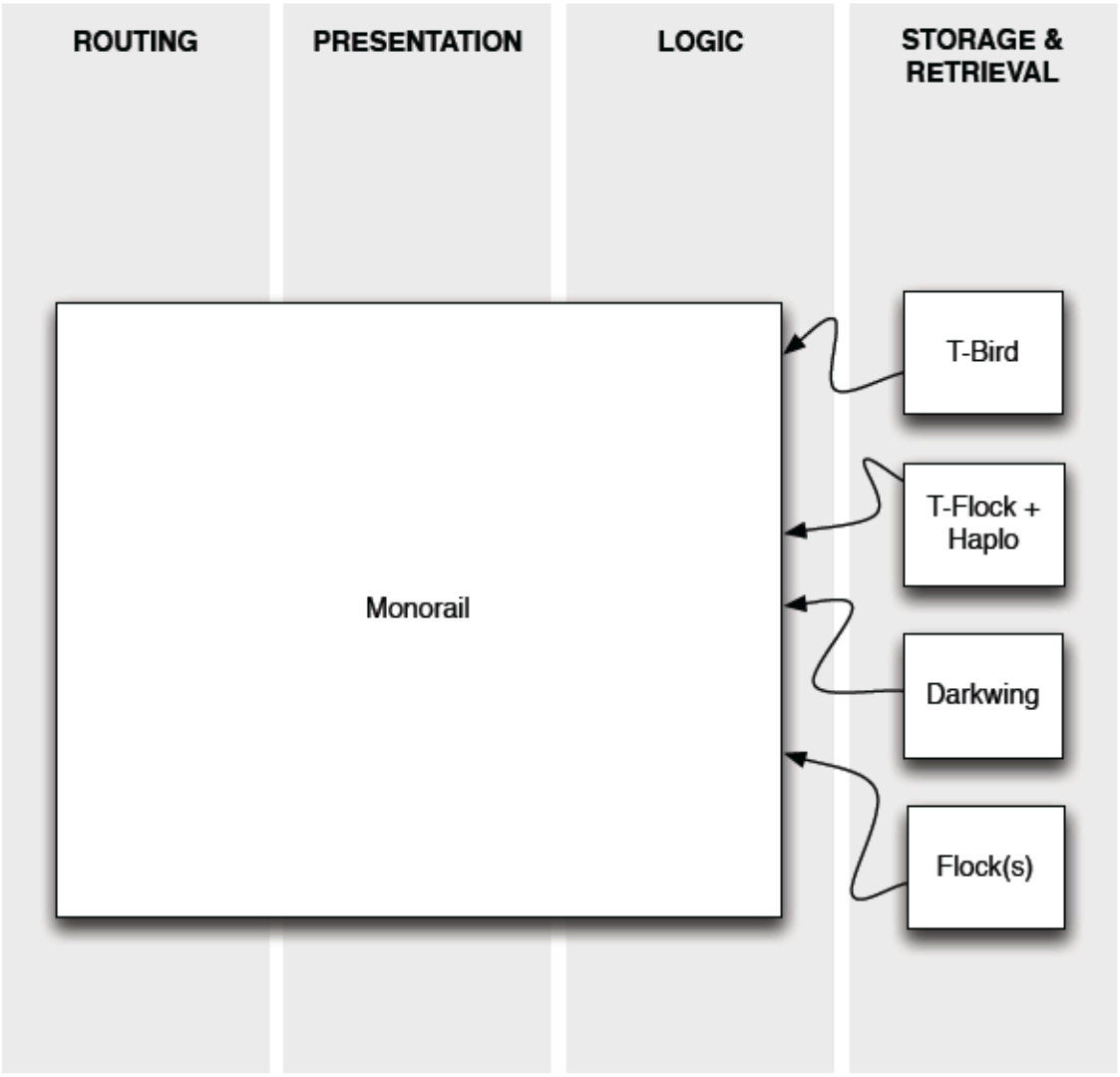


Anti-Spam



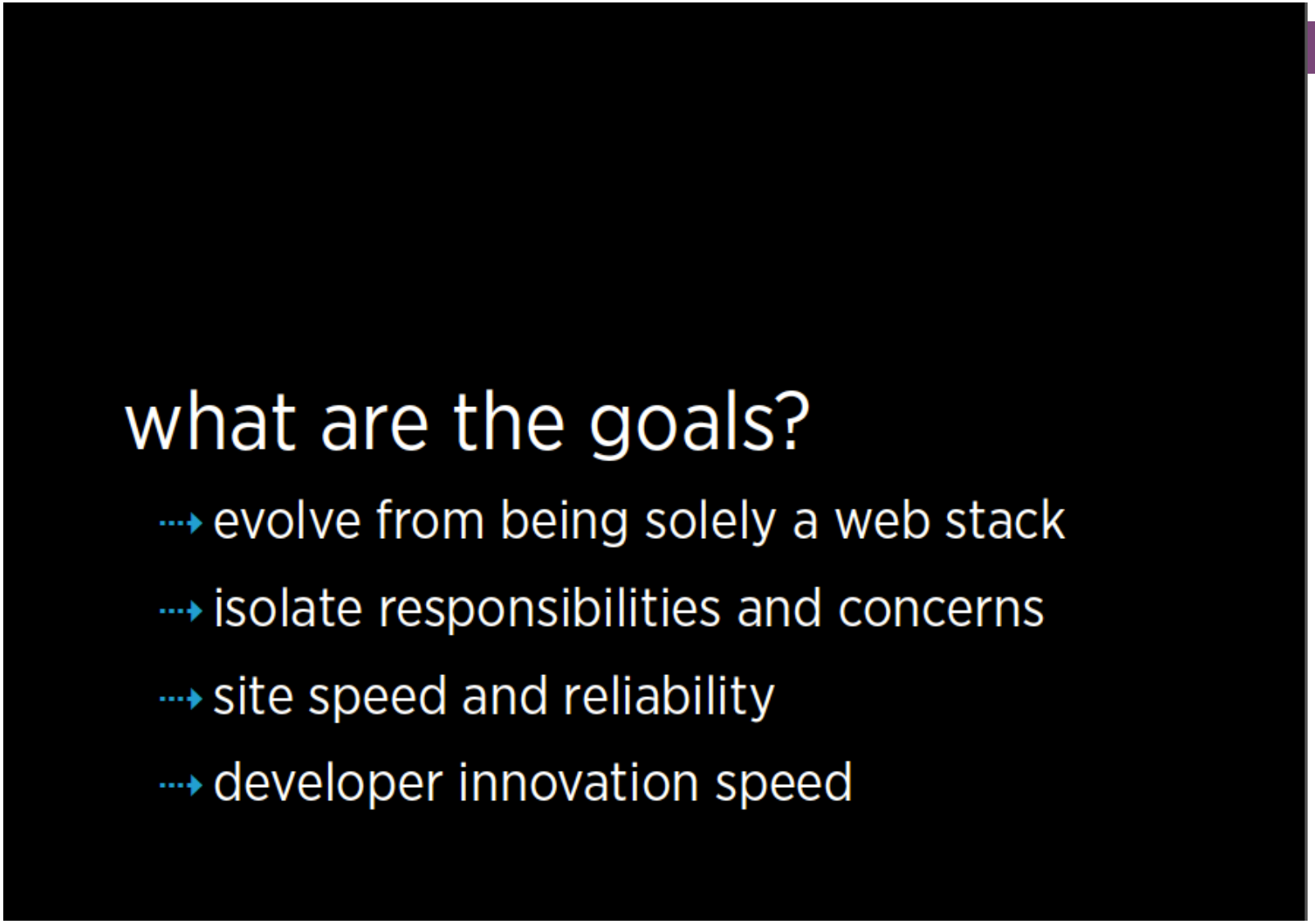
What happens when you "Tweet" this message?

2009

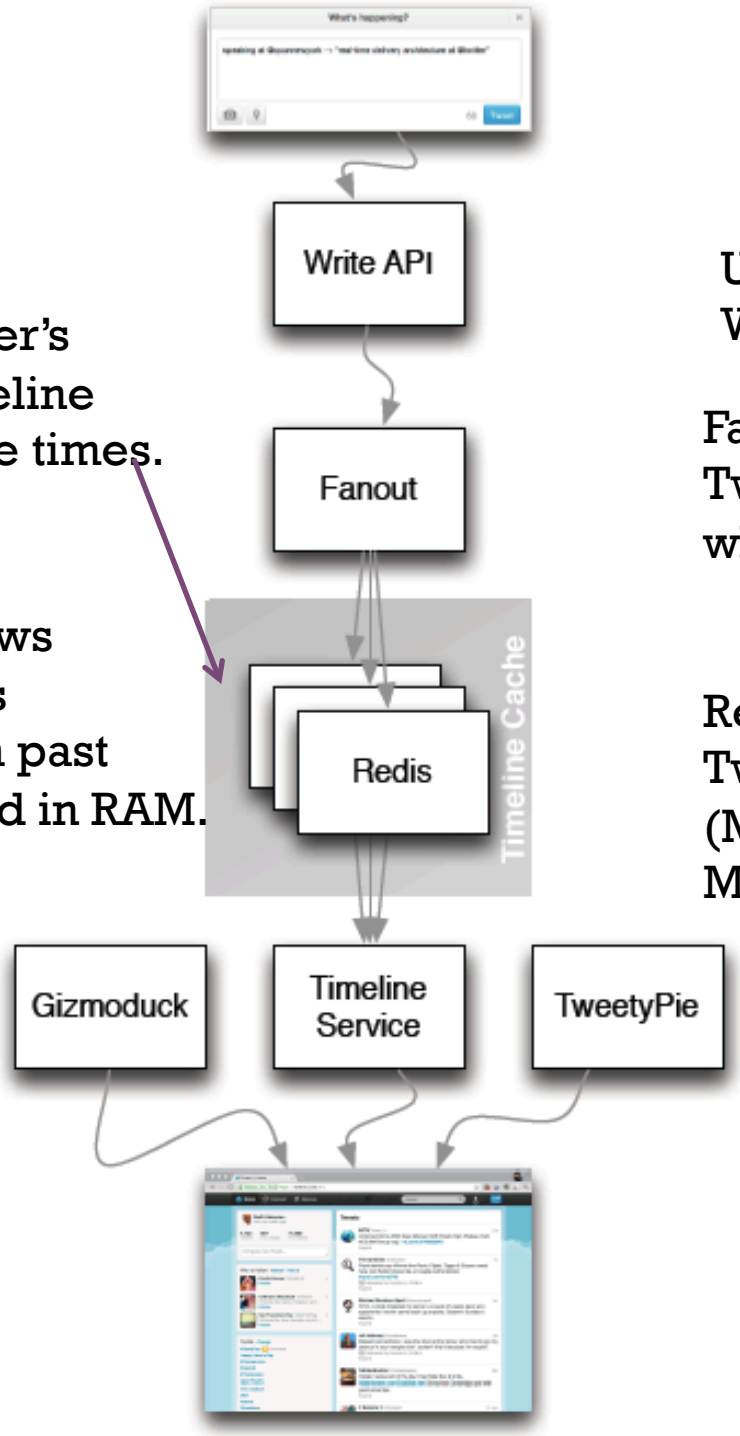


Older Twitter Model:

Based on Ruby on Rails. Everything was being written on a big monolithic stack. Doesn't scale, 400 engineers work on same code base, no independence to team, too much time spent in co-ordination



Everything has to happen in Real-Time.
Event driven programming model to understand when a Tweet was posted, when someone replied etc.



This is needed to Push tweets in user's timeline. The timeline is replicated three times.

Not saved to disk..! Stored in RAM, Allows fast recovery: 45 ms Only active users in past 28 days (LRU) stored in RAM. Rest goes on Disk.

800 tweets per home timeline. Rest is stored on Data-centers

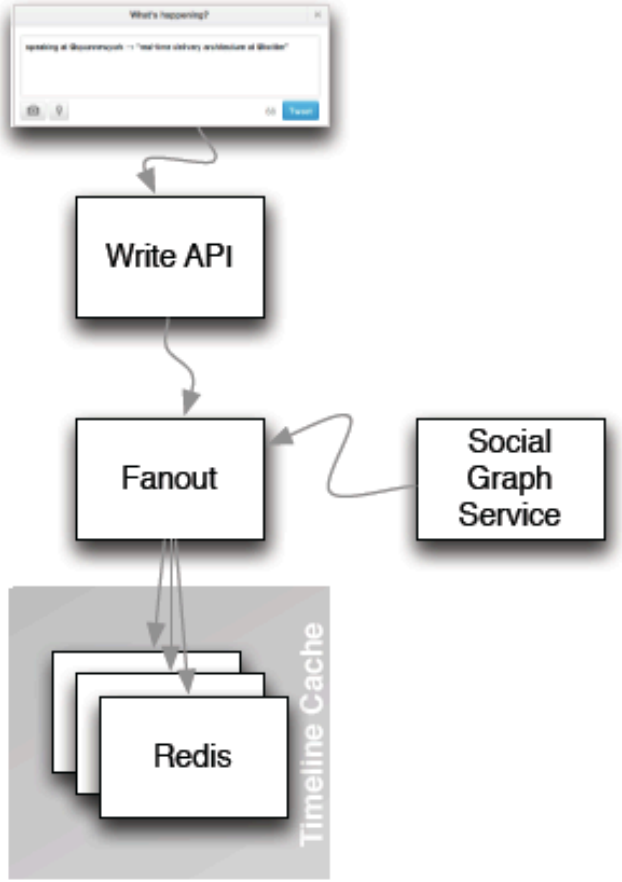
Use Write API to Write tweet in DB

Fan out is delivering the Tweet to every single person who is following that person

Redis Cluster: user-id = key, Tweet = value…! (Map part of) Map Reduce Programming

Timeline Service figures out where person's Timeline lies in Redis cluster (Reduce part of Map-Reduce

The goal is avoid hitting the disk as much as Possible..!

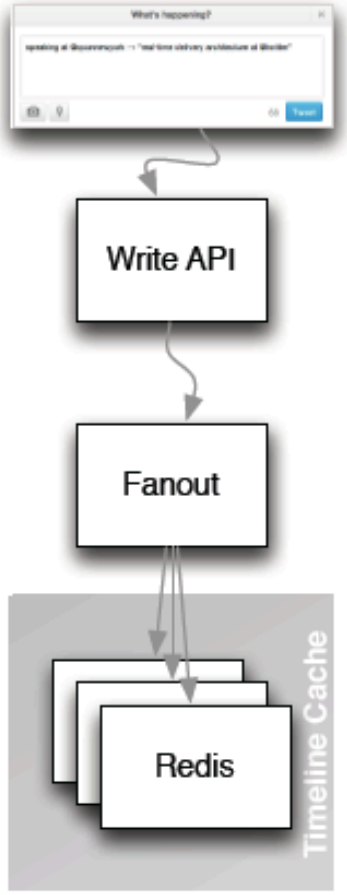


insert

- keyed off "recipient"
- pipelined 4k "destinations" at a time
- replicated

using redis

- native list structure
- RPUSHX to only add to cached timelines

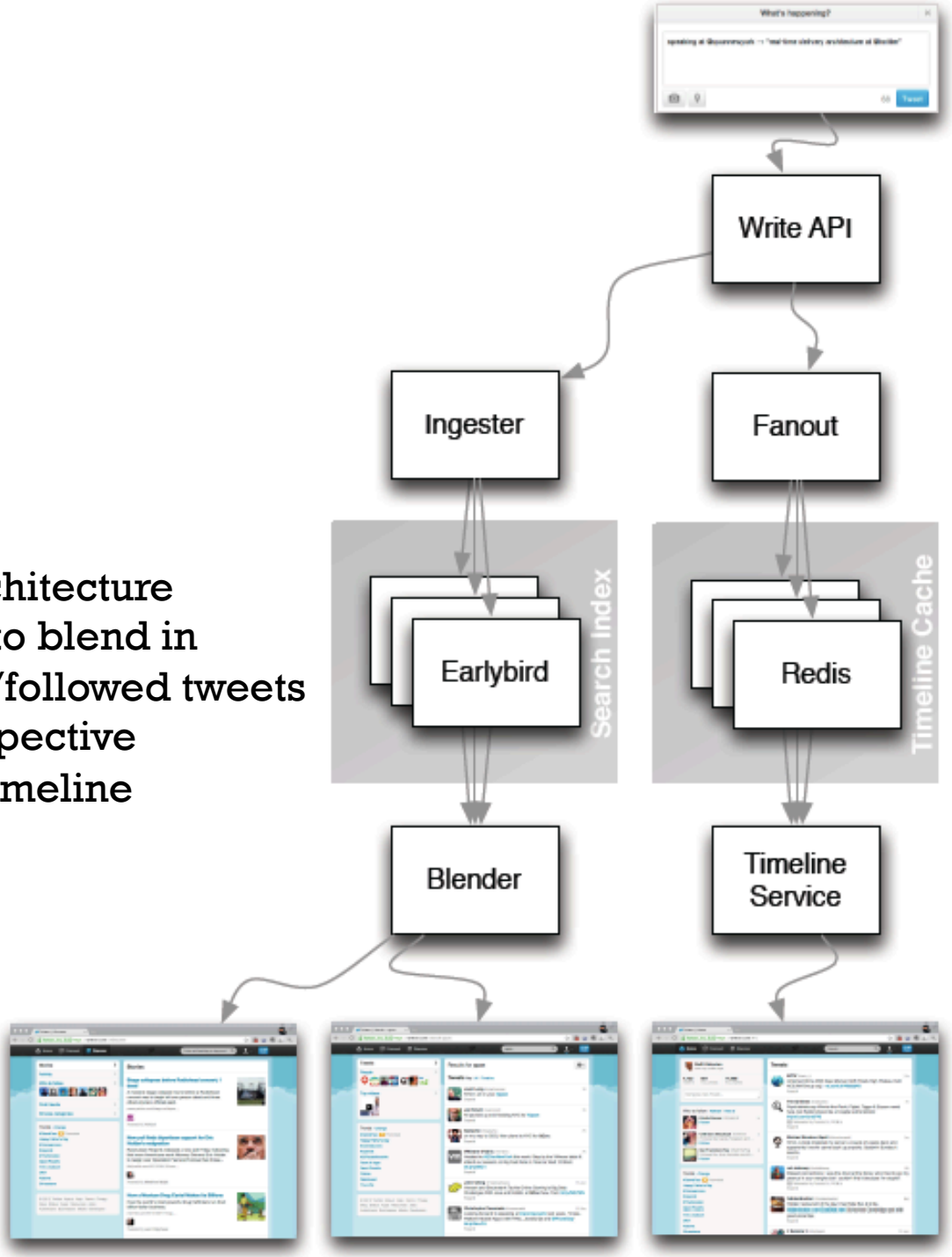


Tweet ID	User ID	Bits	
Tweet ID	User ID	Bits	Tweet ID
Tweet ID	User ID	Bits	
Tweet ID	User ID	Bits	
Tweet ID	User ID	Bits	Tweet ID
Tweet ID	User ID	Bits	
Tweet ID	User ID	Bits	
Tweet ID	User ID	Bits	Tweet ID
Tweet ID	User ID	Bits	
Tweet ID	User ID	Bits	

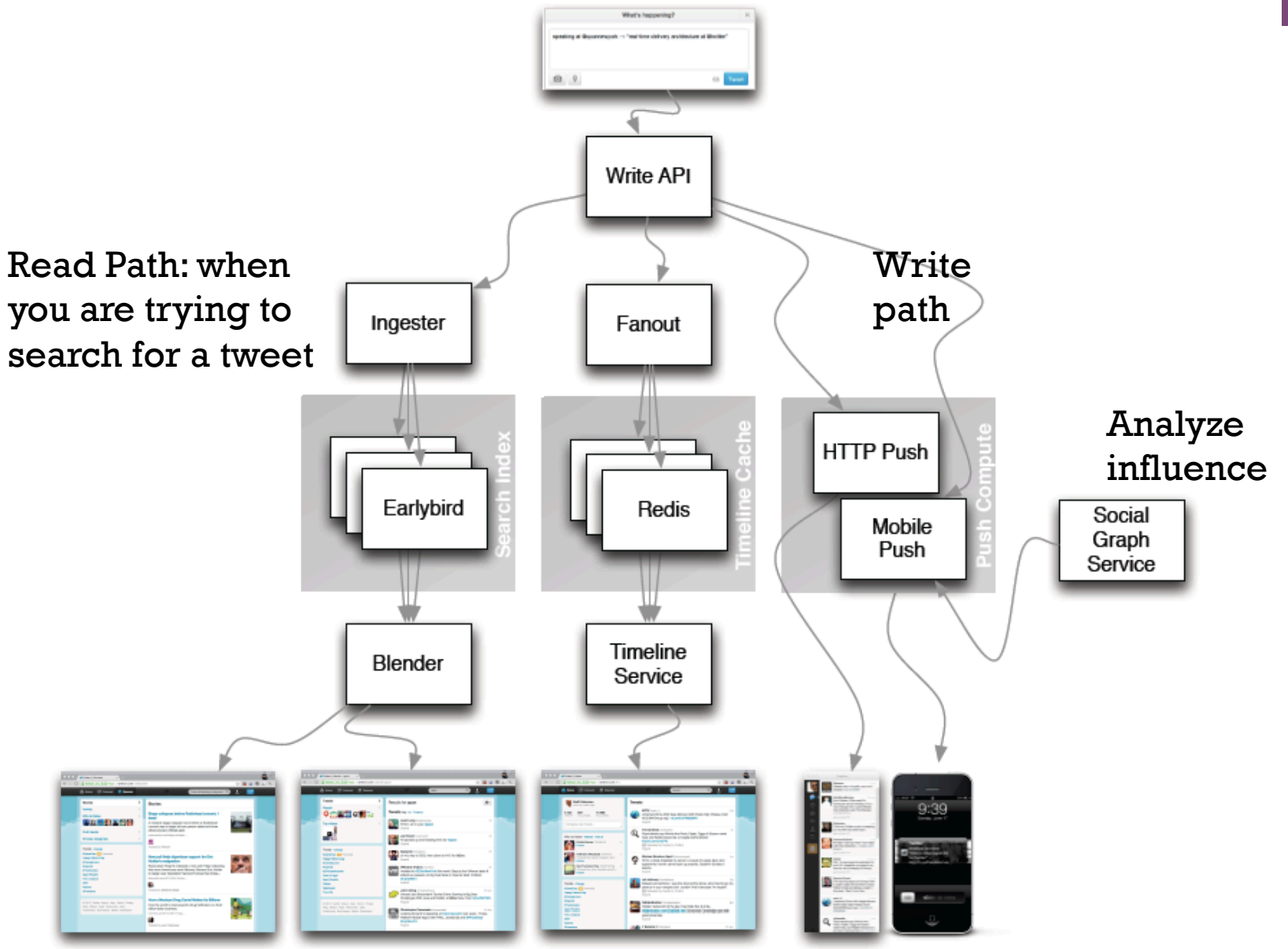
+ Anatomy of a User


```
id: 6253282,
id_str: "6253282",
name: "Twitter API",
screen_name: "twitterapi",
location: "San Francisco, CA",
url: "http://dev.twitter.com",
description: "The Real Twitter API. I tweet about API changes, service issues and happily answer questions about Twitter
protected: false,
followers_count: 1217031,
friends_count: 31,
listed_count: 10784,
created_at: "Wed May 23 06:01:13 +0000 2007",
favourites_count: 25,
utc_offset: -28800,
time_zone: "Pacific Time (US & Canada)",
geo_enabled: true,
verified: true,
statuses_count: 3336,
lang: "en",
status: {
    created_at: "Thu Sep 06 17:55:54 +0000 2012",
contributors_enabled: true,
is_translator: false,
profile_background_color: "C0DEED",
profile_background_image_url: "http://a0.twimg.com/images/themes/theme1/bg.png",
profile_background_image_url_https: "https://si0.twimg.com/images/themes/theme1/bg.png",
profile_background_tile: false,
profile_image_url: "http://a0.twimg.com/profile_images/2284174872/7df3h38zabcvjylnyfe3_normal.png",
profile_image_url_https: "https://si0.twimg.com/profile_images/2284174872/7df3h38zabcvjylnyfe3_normal.png",
profile_banner_url: "https://si0.twimg.com/profile_banners/6253282/1347053495",
profile_link_color: "0084B4",
profile_sidebar_border_color: "C0DEED",
profile_sidebar_fill_color: "DDEEF6",
profile_text_color: "333333",
profile_use_background_image: true,
show_all_inline_media: false,
default_profile: true,
default_profile_image: false,
following: null,
follow_request_sent: null,
notifications: null
```


This architecture
Allows to blend in
copied/followed tweets
into respective
wsers timeline

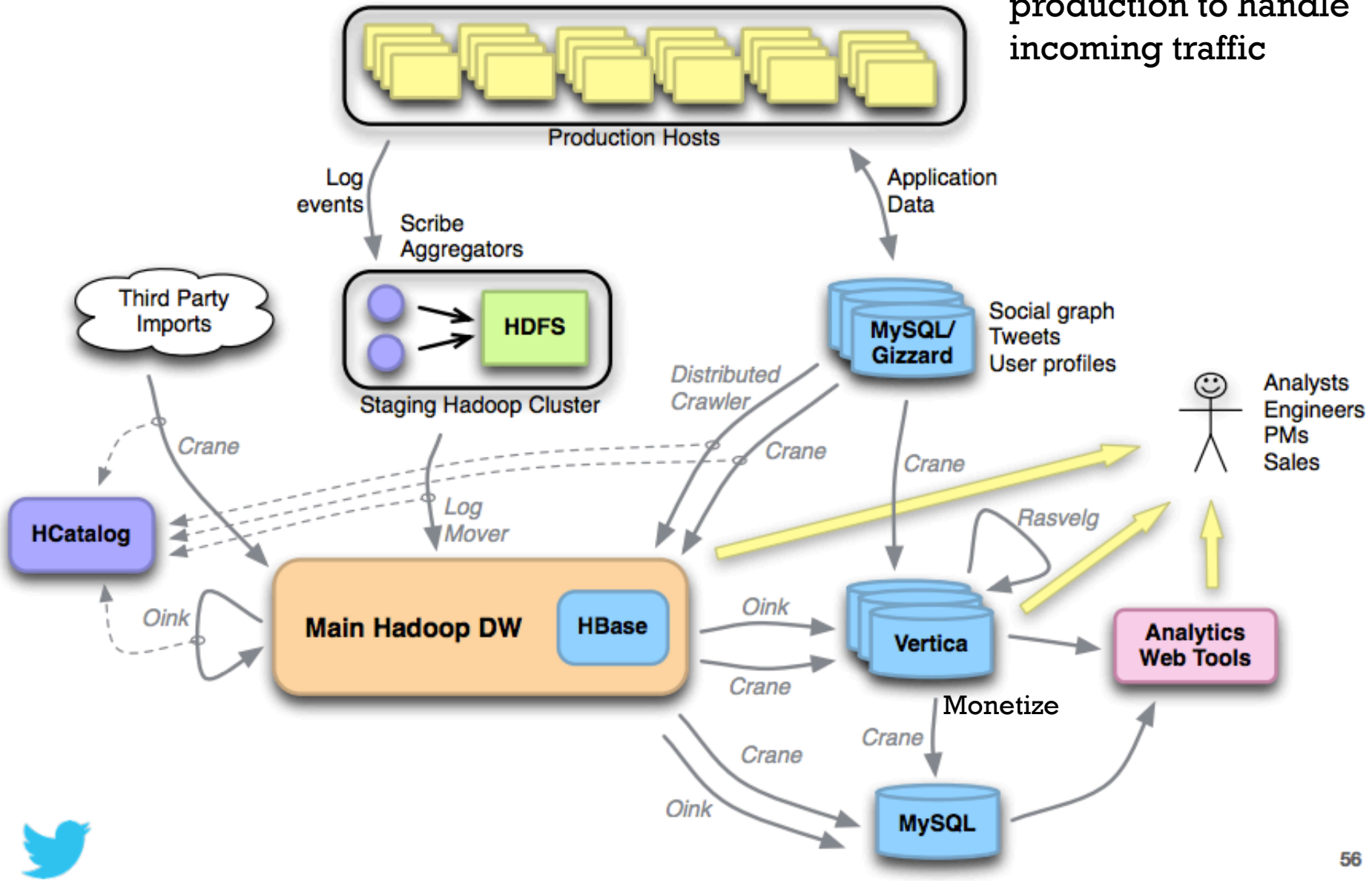


200-700ms


Read Path: when you are trying to search for a tweet
Write path
Analyze influence
Write API
Ingester
Fanout
Search Index
Earlybird
Timeline Cache
Redis
Push Compute
HTTP Push
Mobile Push
Social Graph Service
Blender
Timeline Service
9:39
Search Operation
Regular tweet
Access via web/mobile

Twitter Analytics data flow

Servers in production to handle incoming traffic



Analyzing Machine Generated Data

- Searching, monitoring and analyzing machine generated big data via web interface
- Allows real-time response model when servers/clusters fail
- Allows trend detection/understanding unpredicted events
- Widely used in web-analytics