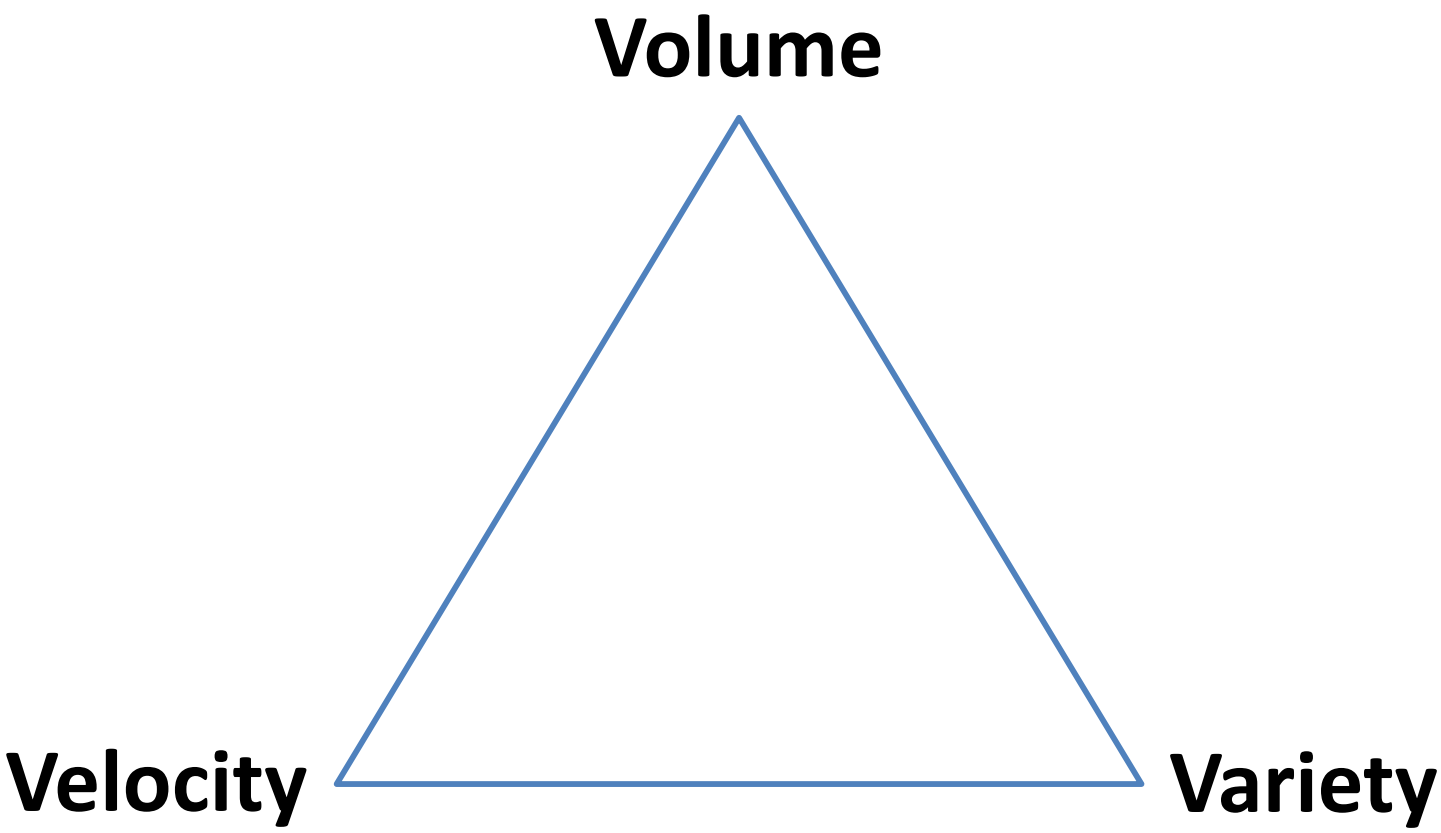
What is big data?



What is big data?

Volume

- Terabytes to Petabytes
- Challenges and opportunities

Velocity

- Growing exponentially
- Both incoming and outgoing

Variety

- Various types and sources of data
- Documents, databases
- Images, voice, videos, sensors, GSP location data, …

U.S. government commits big R&D money to 'Big Data'

Summary: *The U.S. government is investing $200 million in big data projects to help the U.S. jump ahead in the next frontier of computing.*

By Jason Hiner for Between the Lines | March 29, 2012 -- 12:50 GMT (05:50 PDT)

Follow @jasonhiner

Problem: I give you a 400 terabyte log file to process for your next homework

- Or perhaps a Web crawl:
- 20 billion Web pages x 20 Kb > 400 terabytes
- Computer can read 35Mb/sec from hard drive
  - Four months to scan the Web!
- Hundreds of hard drives needed

Source: http://static.googleusercontent.com/media/research.google.com/en/us/pubs/archive/32721.pdf

Solution:

- Buy 1,000 comput
- Have <u>each</u> computer … rabyte dataset
  - Done in < 3 !
- New issues:
  1. This requires progra
     - Computers mus
     - Load balan
     - Network and disk optimi
     - How to recover from co ter f (100 = 1 per d
     - How to optimize? Debug?
     - Data locality
  2. You need to repeat this for every pe of problem you want to solve.

Wouldn't it be great if there were a general-purpose programming language and set of core libraries that handled all of that for us?

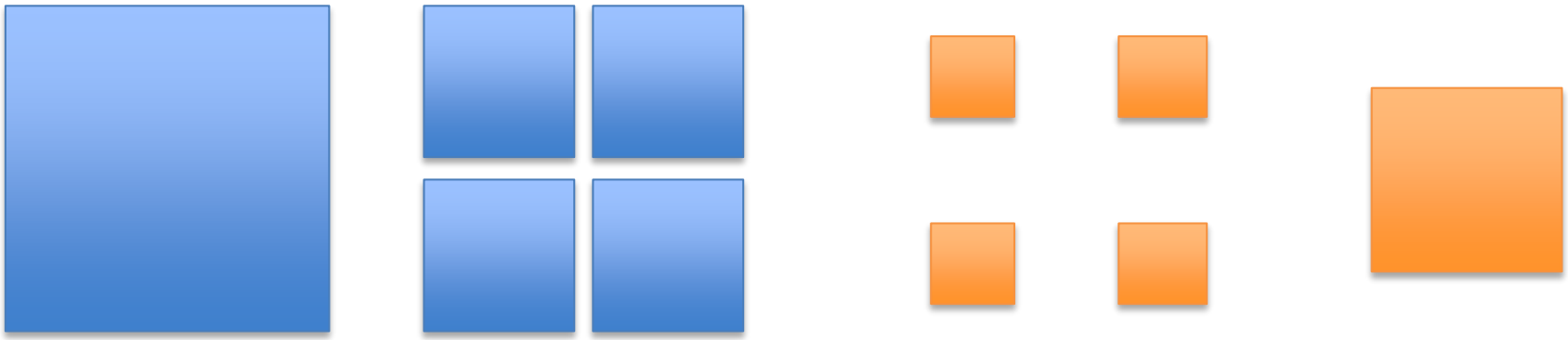
Well, actually…

Source: http://static.googleusercontent.com/media/research.google.com/en/us/pubs/archive/32721.pdf

Divide and conquer

A technique we use everyday!

- Split the task in sub-tasks
- Put resources to handle subtasks in parallel
- Combine the results
- This is a simple example of distributed computing
- We are distributing the workload across different CPUs



Divide and conquer

Word count as an example:

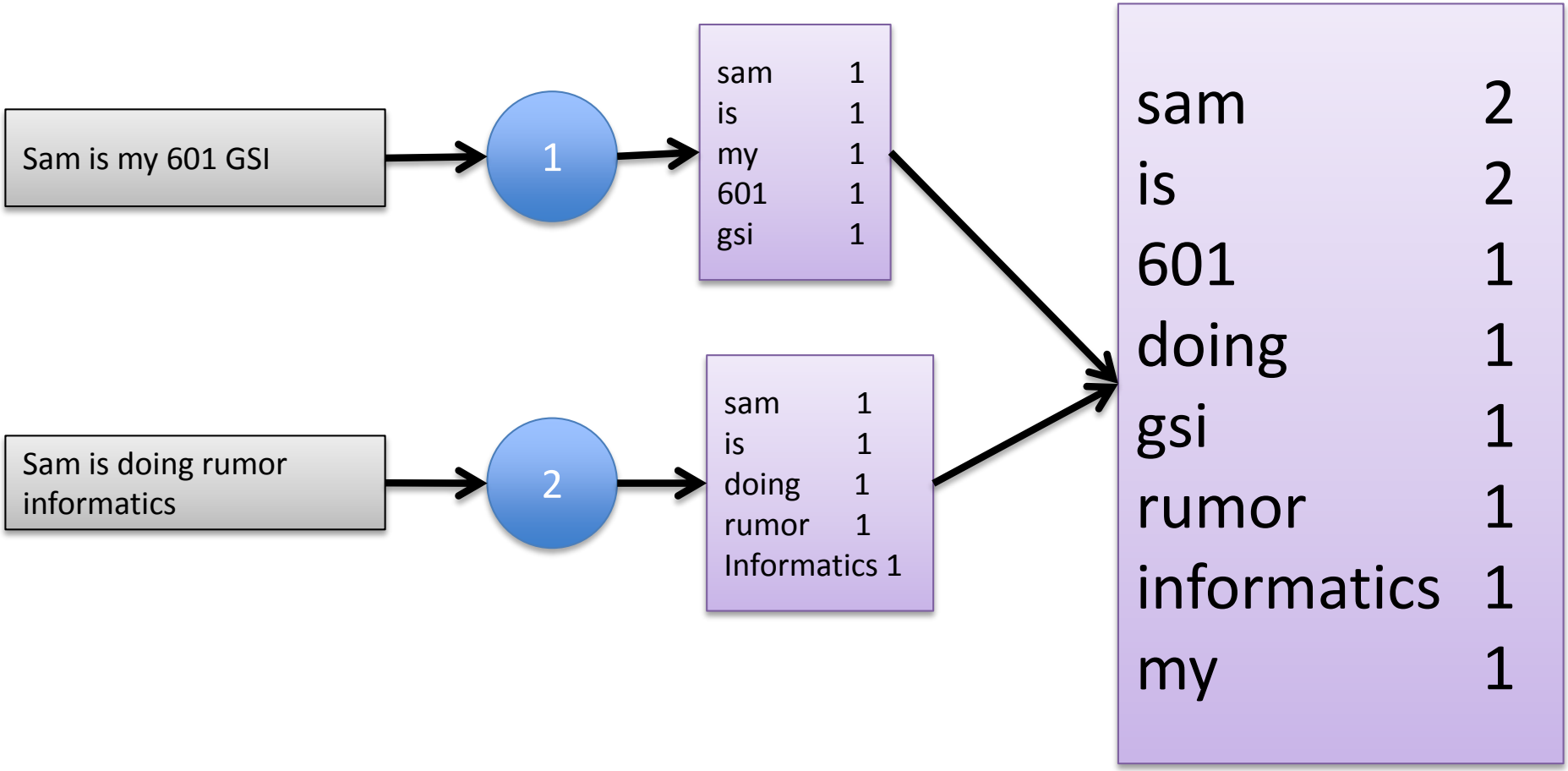
Sam is my 601 GSI

Sam is doing rumor informatics

sam	2
is	2
601	1
doing	1
gsi	1
rumor	1
informatics	1
my	1

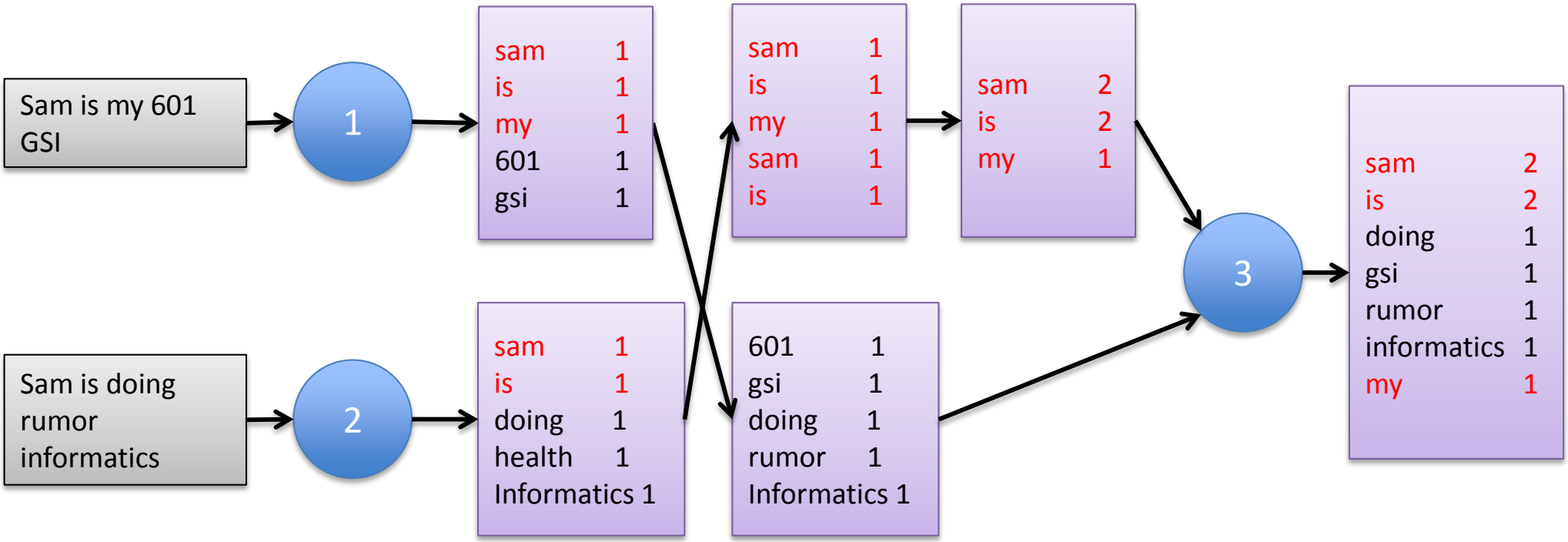
Divide and conquer

Conceptual model 1



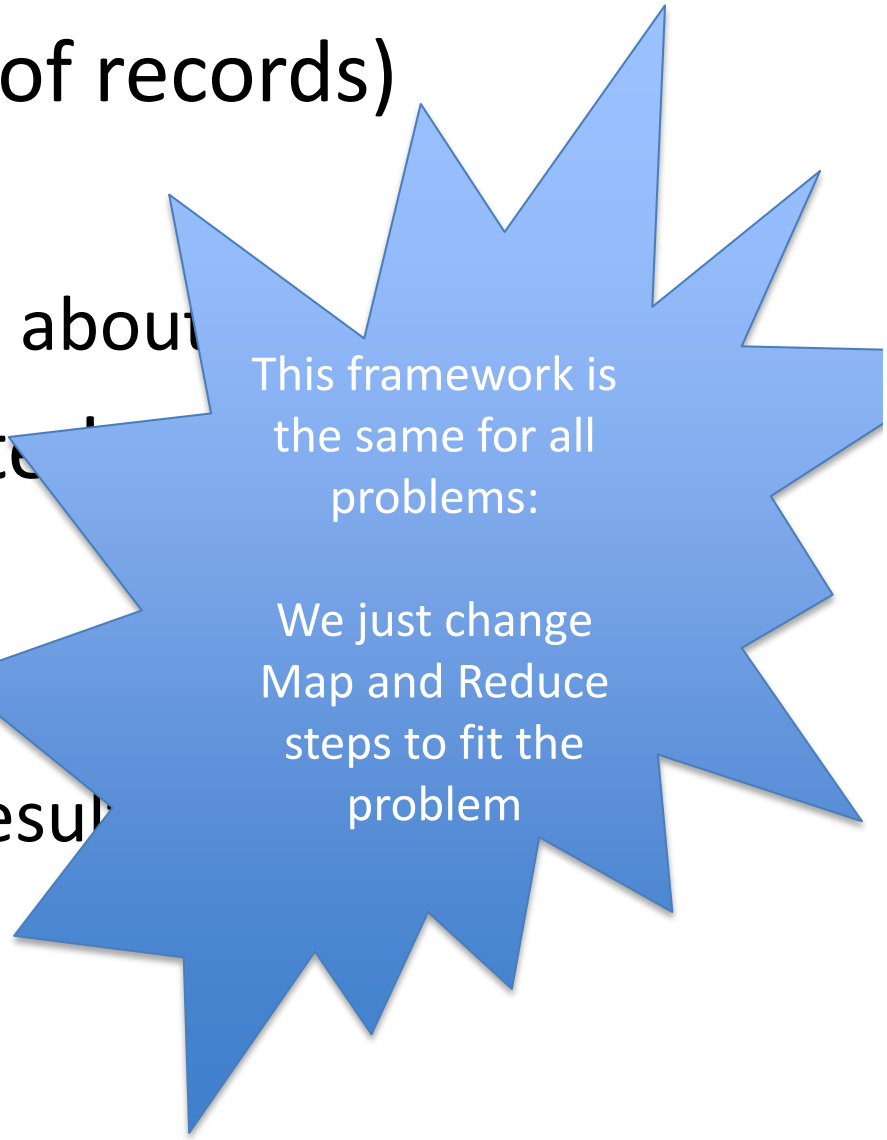
Divide and conquer

Conceptual model 2

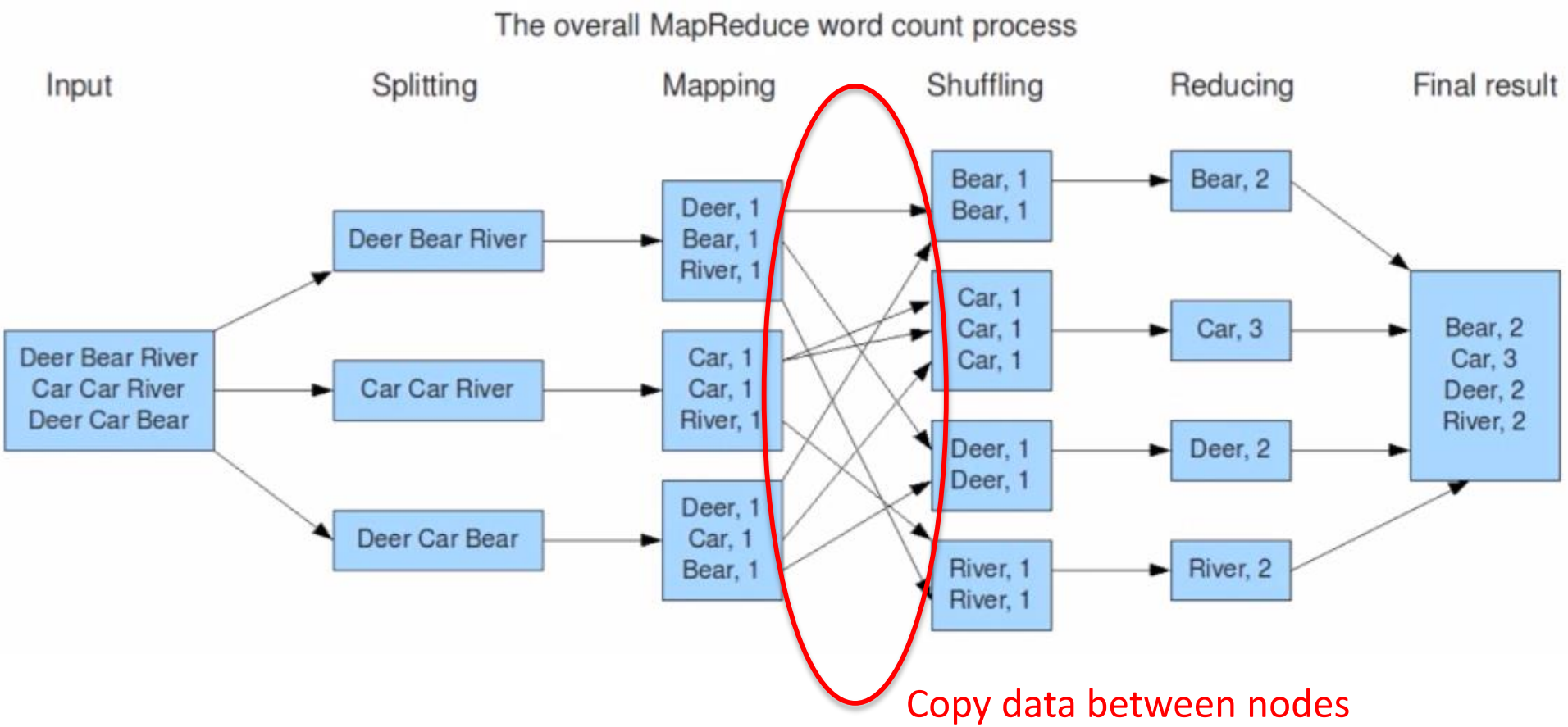


A typical MapReduce problem

- Read a bunch of data (set of records)
- Map: for each record,
  - Extract something you care about
- Sort and group the extracted
- Reduce: For all groups:
  - Summarize, filter, transform
  - Collapse the group into a result
- Write out the results



MapReduce – Word Count



Copy data between nodes based on the keys

Slide from Basic Introduction to Apache Hadoop

http://www.youtube.com/watch?v=OoEpfb6yga8

Assumptions of MapReduce

- The task can be broken into multiple pieces
- Pieces can be processed in parallel with minimal communication between pieces
- Results of each piece can be combined in the end to produce final result

Two views of MapReduce programming

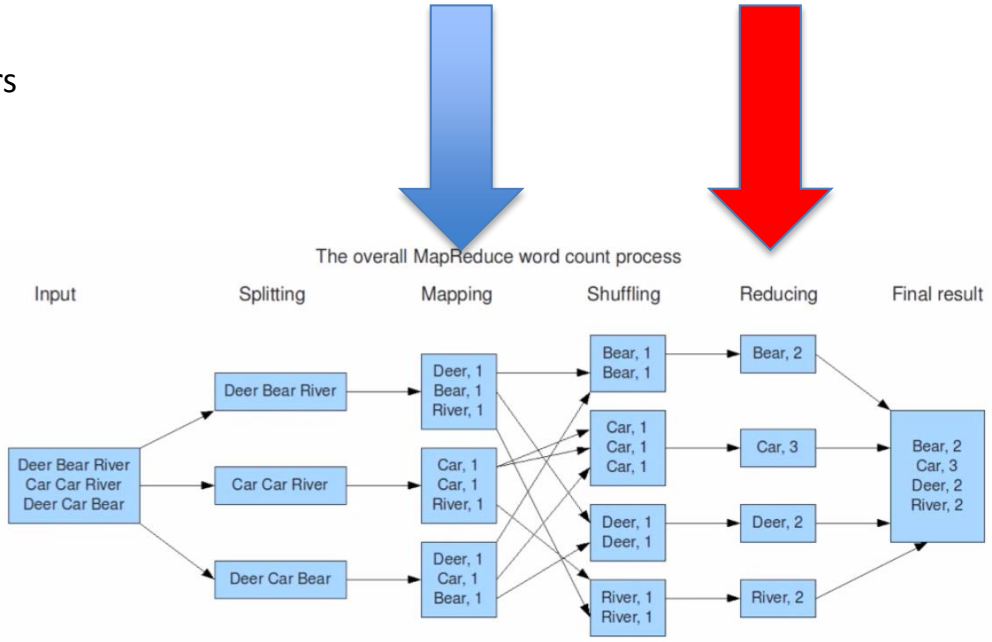
- Step-level view
  - You supply the mapper and reducer functions
  - Python MRJob (Lab 5)
- Table-level view
  - Load -> Transforms -> Dump
  - High-level table operations like SQL
    - Group records, compute aggregate function
  - Can still define custom mappers, reducers if needed
  - Hadoop + Pig programming language (Homework 5)

MRJob

- Different backends possible
  - Test on your local machine
  - Or Run on a Hadoop cluster
  - Or use Amazon Elastic MapReduce
- Base class: MRJob
  - You create your own subclass inheriting from MRJob with your desired mapper and reducer methods
  - You must define at least one of:
    - mapper, reducer, combiner
- Install package "mrjob"
- Documentation: http://pythonhosted.org/mrjob/

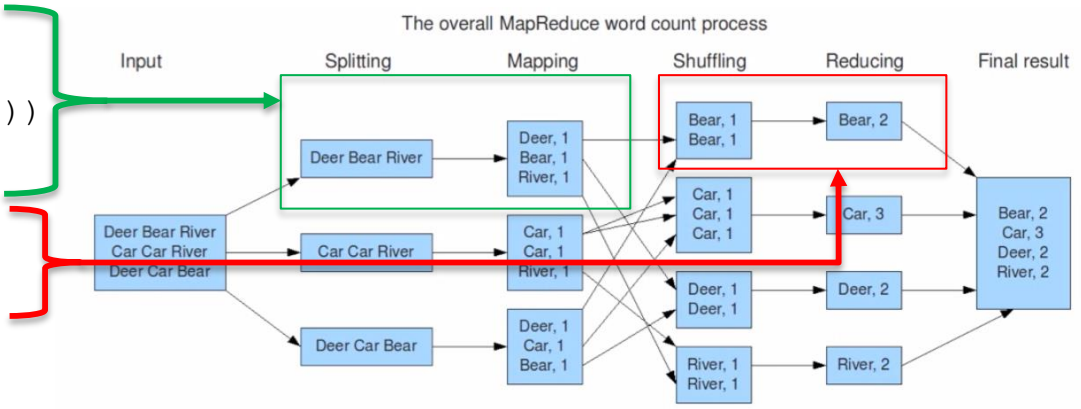
You plug in a mapper and reducer: MRJob framework does the rest

- Mapper
  - Goal: Break input line into a set of key, value pairs
  - Input: single line from text file
  - Output:
    zero or more (key, value) pairs
- Reducer
  - Goal: Take all (key, value) pairs with the same k
    compute an aggregate function over those valu
  - Input: a key and a list of all values seen for that
  - Output:
    zero or more (key, value) pairs
  - Most typically, the values are from an aggregat
    function over value_list, e.g. sum(value-list)
- The MRJob framework takes care of the rest:
  - Sorting the mapper output. Invoking reduce tasks
  - Assembling reduce outputs into final result
  - Scheduling, monitoring all tasks, re-starting failed tasks



Example: MRJob program for counting characters, words, and lines

```
#!/usr/bin/python
from mrjob.job import MRJob

class MRWordFrequencyCount(MRJob):
    # mapper: key _ is always None and ignored
    def mapper(self, _, line):
        yield "chars", len(line)
        yield "words", len(line.split())
        yield "lines", 1

    def reducer(self, key, values):
        yield key, sum(values)


if __name__ == '__main__':
    MRWordFrequencyCount.run()
```



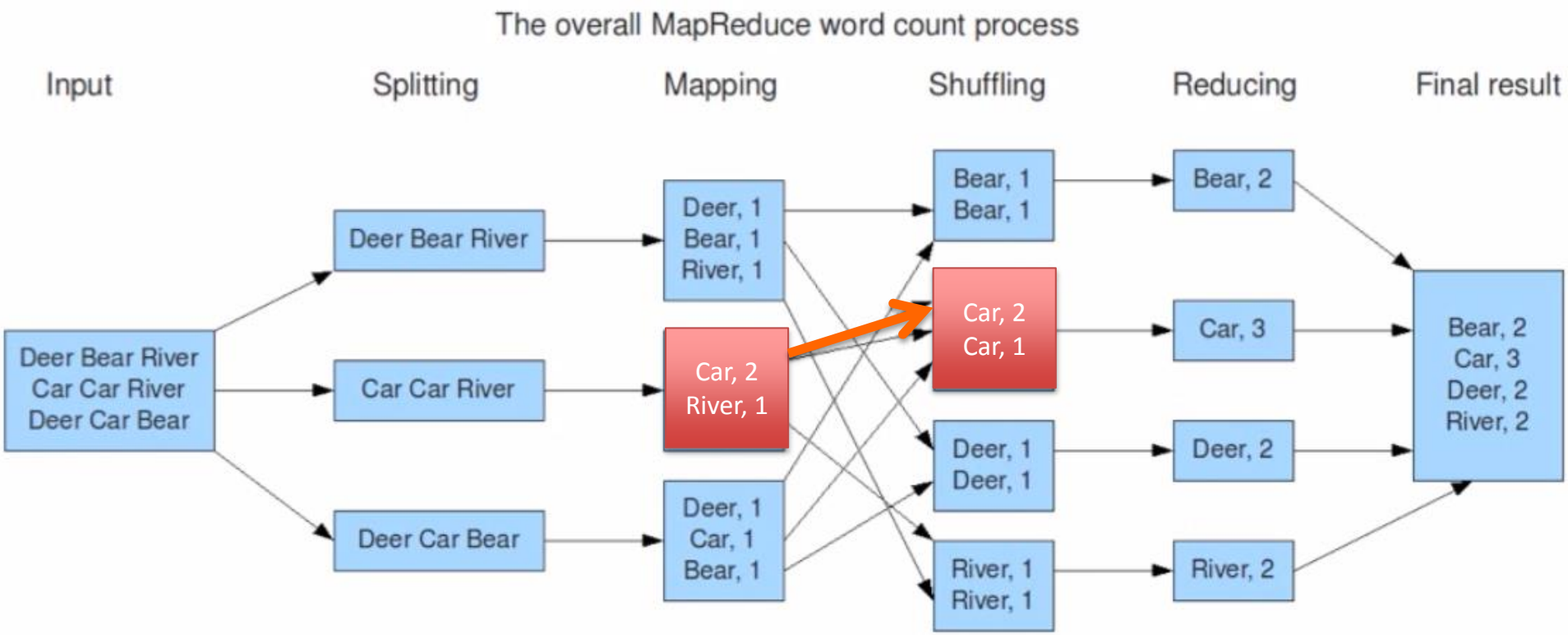
$ python word_count.py my_file.txt
[…a bunch of log output…]
"chars" 3654
"lines" 123
"words" 417

Can define custom **protocols** for input, output, transfer
Default runner: InlineMRJobRunner will do all steps locally

Quiz

- Is using a reducer always necessary?
  - No, if there are no sorting or grouping tasks
  - Example: change all words into upper case
- Shuffle seems like it passes a lot of data around! How can we reduce that?
  - Add a combiner after mapping

MapReduce – Word Count

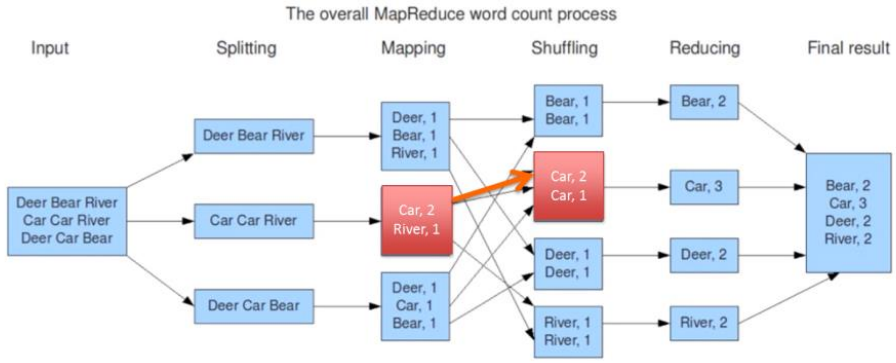


Slide from Basic Introduction to Apache Hadoop

http://www.youtube.com/watch?v=OoEpfb6yga8

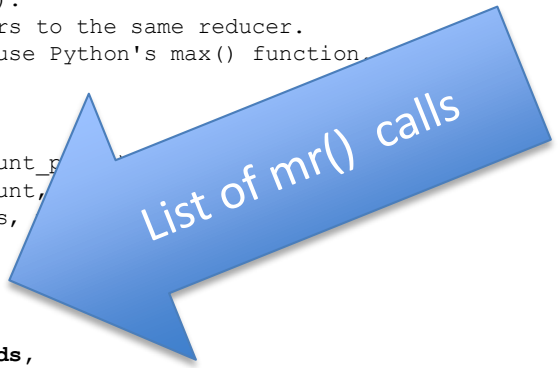
Adding a combiner can increase efficiency

- Run immediately after each mapper
- Used to decrease total data transfer
- Input: key, and a subset of values for that key
- Output: zero or more (key, value) pairs
- Example:
  - *mapper*: splits line into words
    "the", 1
    "wheels", 1
    "of", 1
    "the ", 1 ...
  - *combiner*: sums word counts over mapper output
    "the", 2
    "wheels", ..
  - *reducer*: sums words counts over combiner outputs



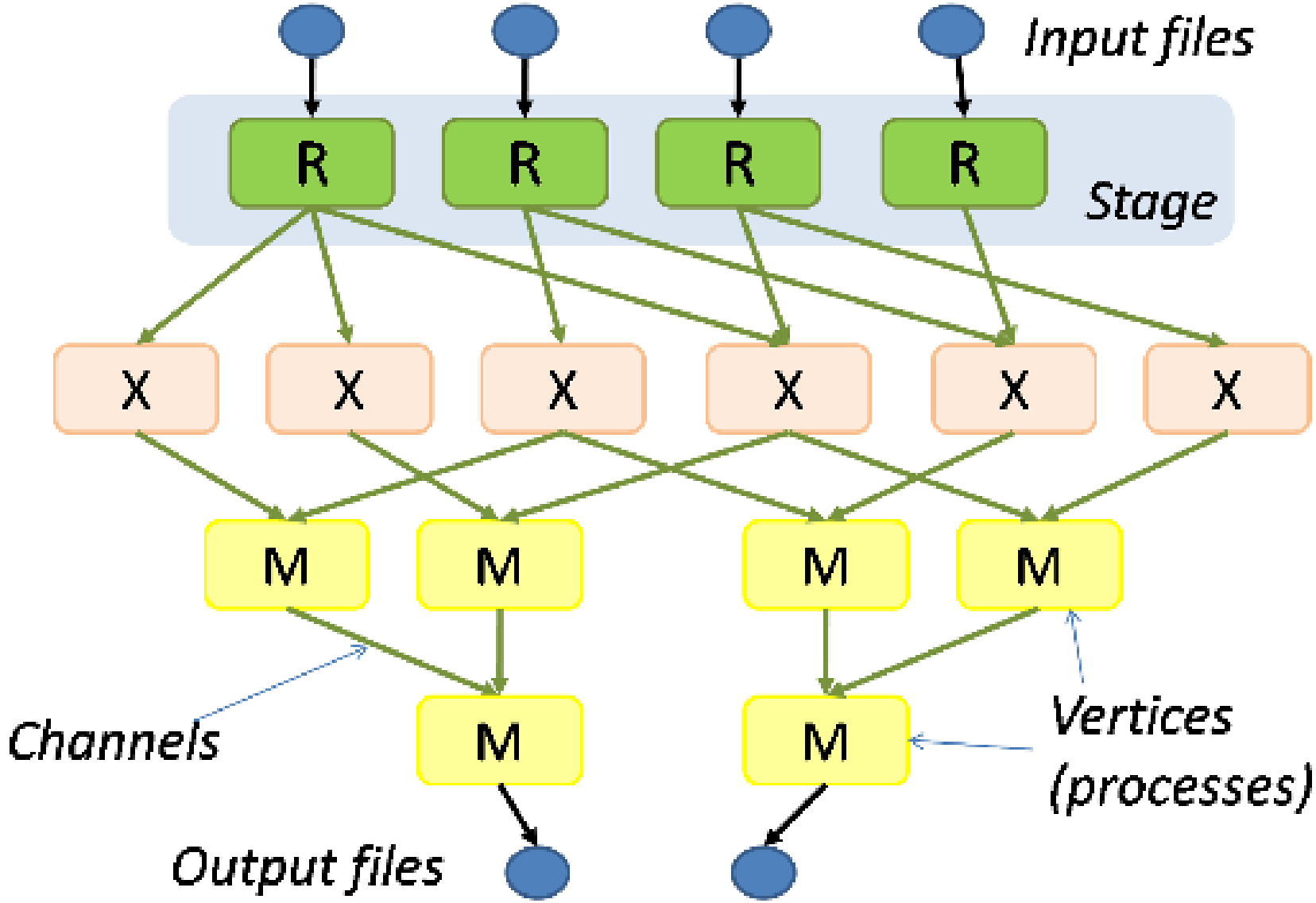
MRJob: Running multiple steps by providing a custom steps() method

```
from mrjob.job import MRJob
import re

WORD_RE = re.compile(r"[\w']+")


class MRMostUsedWord(MRJob):

    def mapper_get_words(self, _, line):
        # yield each word in the line
        for word in WORD_RE.findall(line):
            yield (word.lower(), 1)

    def combiner_count_words(self, word, counts):
        # optimization: sum the words we've seen so far
        yield (word, sum(counts))

    def reducer_count_words(self, word, counts):
        # send all (num_occurrences, word) pairs to the same reducer.
        # num_occurrences is so we can easily use Python's max() function.
        yield None, (sum(counts), word)

    # discard the key; it is just None
    def reducer_find_max_word(self, _, word_count_p
        # each item of word_count_pairs is (count,
        # so yielding one results in key=counts,
        yield max(word_count_pairs)

    def steps(self):
        return [
            self.mr(mapper=self.mapper_get_words,
                    combiner=self.combiner_count_words,
                    reducer=self.reducer_count_words),
            self.mr(reducer=self.reducer_find_max_word)
        ]


if __name__ == '__main__':
    MRMostUsedWord.run()
```



Beyond MapReduce:
Other distributed computing models

A more sophisticated computation flow (Microsoft Dryad)



When should we use MapReduce?

- Good MapReduce scenarios?
  - Data can be trivially partitioned in parallel
  - Few/no dependencies between the pieces
  - Results can be trivially recombined
  - Have lots of parallel CPUs w/ good bandwidth
  - Processing speed matters
  - e.g. feature extraction
- Bad MapReduce scenarios?
  - Lots of dependencies between data elements
    - e.g. need similarity between every pair of tweets
  - Instead, use graph (network)-based computation:
    - See GraphLab and other graph-based frameworks
    - http://graphlab.com/products/create/technology.html
    - Order of magnitude speedup over MapReduce in such cases

Hadoop: architecture

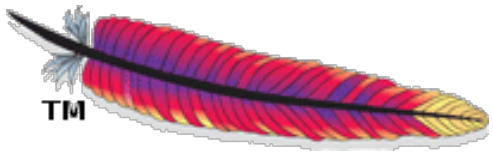
- The Apache™ Hadoop® project develops open-source software for reliable, scalable, distributed computing.

Leading framework: Hadoop

- A software framework for distributed computing
- Take 100 'commodity' machines that don't share memory or disk storage
- Turns commodity machines into a cluster
  - ✔ *Redundant and Reliable*
  - ✔ *Powerful and Scalable*
  - ✔ *Cost-effective*
- Java-based APIs to Hadoop services
  - But calling these directly is tedious and error-prone so people use programming languages like pig to perform Hadoop jobs
- Batch mode

Many key ideas behind Hadoop were originally developed at Google to handle huge data volumes

Google



Google File System (GFS)

Hadoop Distributed File System (HDFS)

http://research.google.com/archive/gfs.html

MapReduce

Hadoop MapReduce

http://research.google.com/archive/mapreduce.html

BigTable

Hadoop HBase

http://research.google.com/archive/bigtable.html

Originally developed at Google. See paper links.

Leading framework: Hadoop

Major components

1. MapReduce (algorithm)
   - A programming model for large-scale data processing
2. Hadoop Distributed File System (data storage)
   - Stores and aggregates data on cluster machines
3. Hardware Architecture
   - Networked machines

Cluster of machines running Hadoop at Yahoo! (Source: Yahoo!) via
http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/

Hadoop Distributed File System (HDFS)

We need a mechanism to support the map-reduce process at the data level.

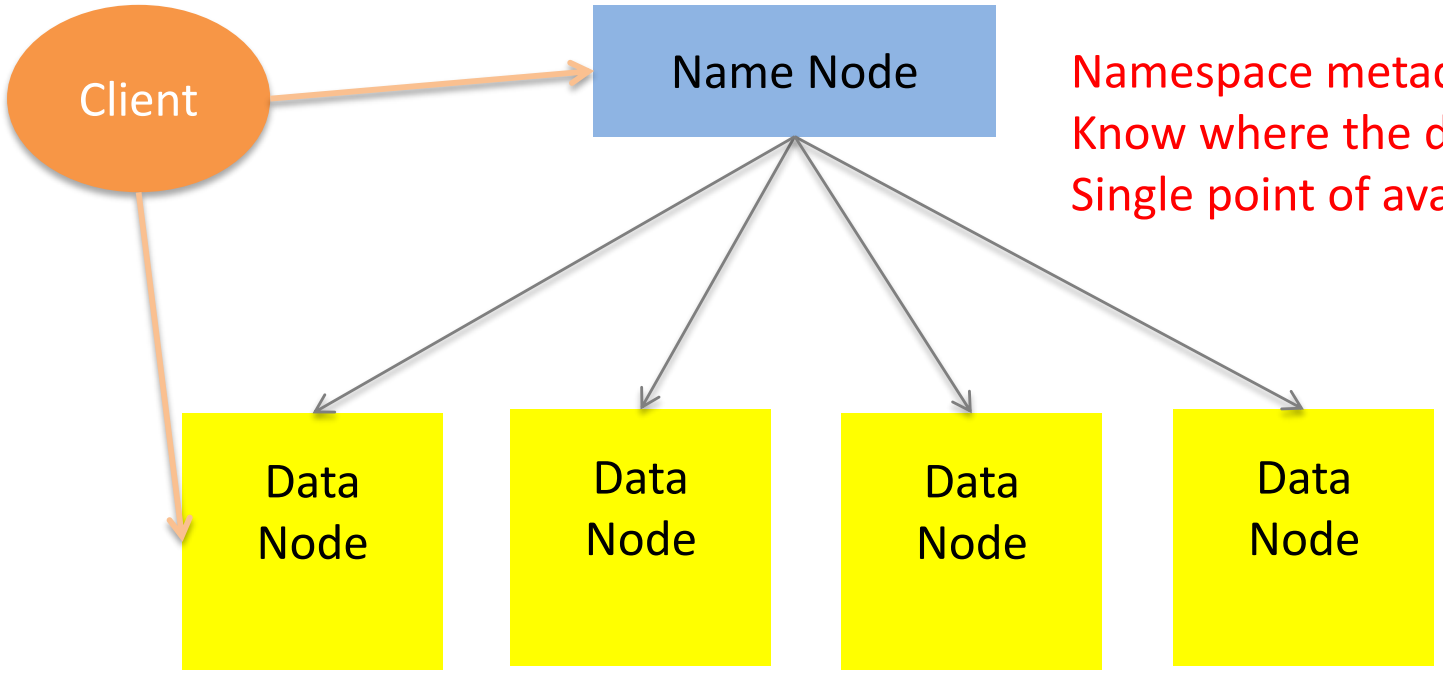
HDFS is designed to be…

- Scalable in storage and I/O bandwidth
- Highly fault-tolerant (check periodically)
- Optimized for commodity machines

Typical settings:

- Save a file into blocks (128MB)
- Replicate 3 times

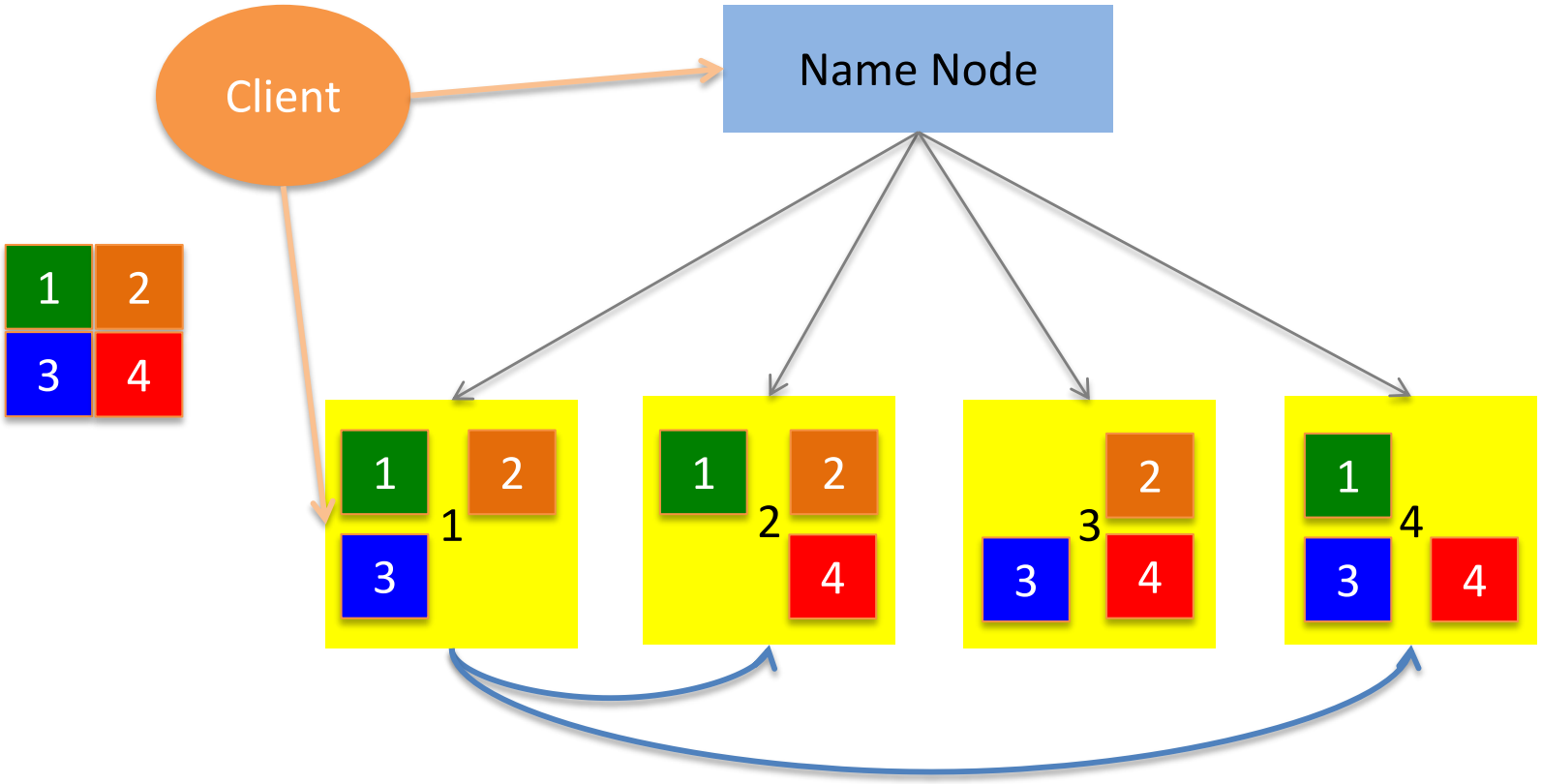
Hadoop Distributed File System (HDFS)



Namespace metadata
Know where the data pieces are
Single point of availability failure

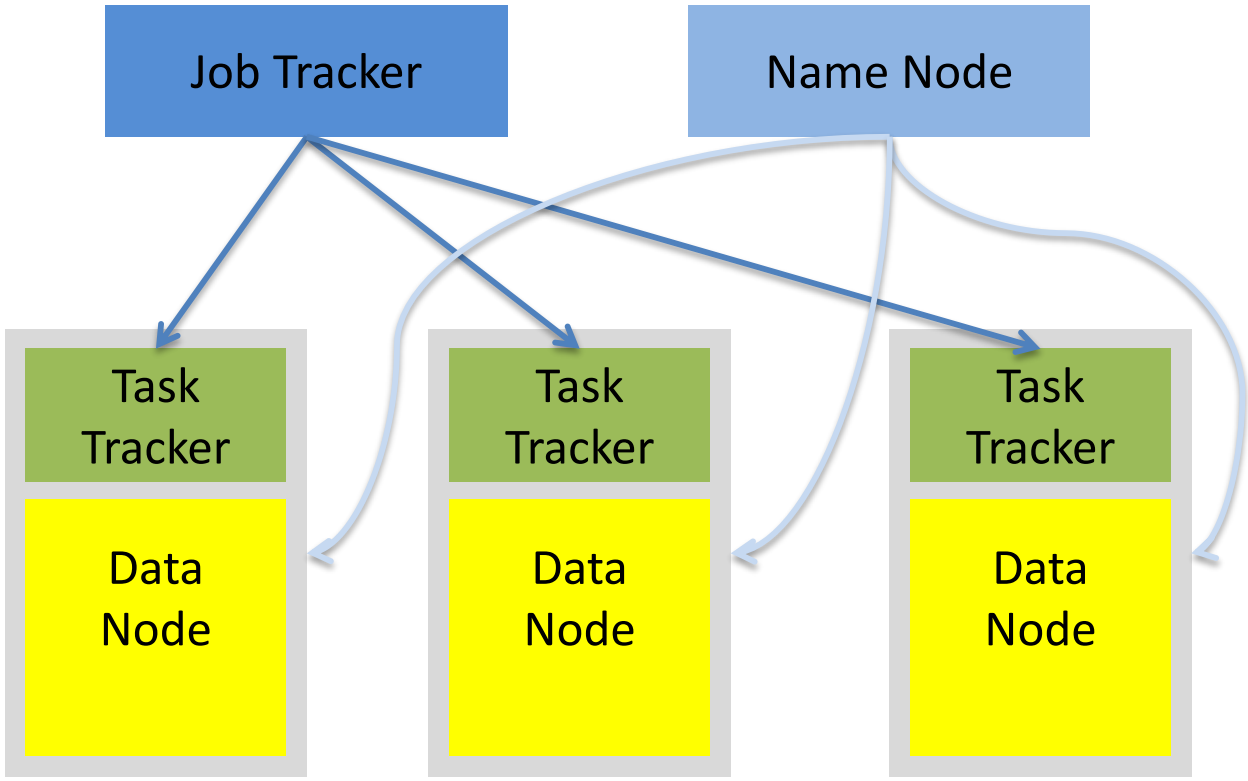
Responsible for actual read and write

Hadoop Distributed File System (HDFS)

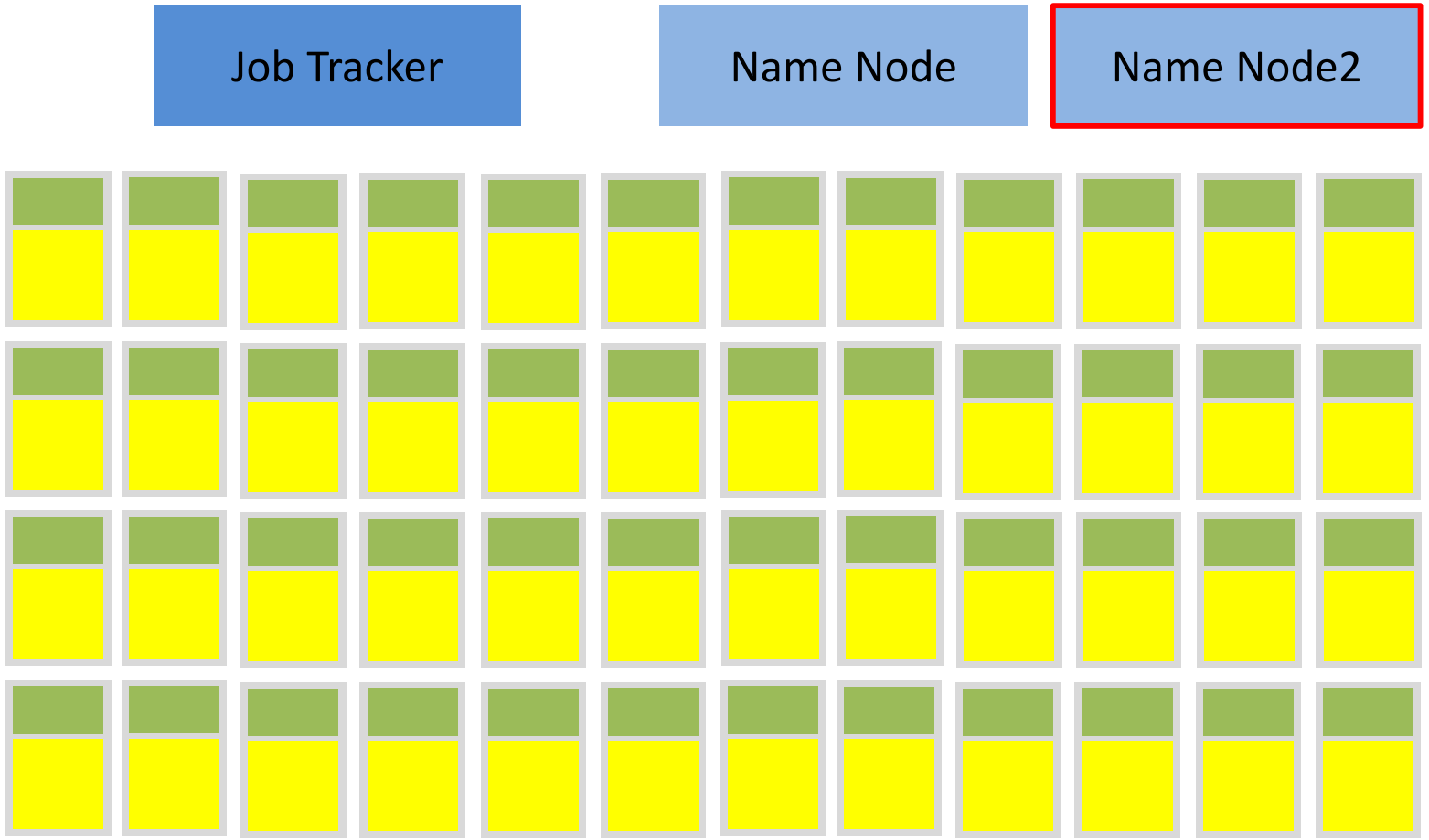


Hadoop Tutorial: Intro to HDFS
http://www.youtube.com/watch?v=ziqx2hJY8Hg

Hadoop Hardware Architecture



Hadoop Hardware Architecture



Hadoop: Pig programming

How do people actually use the power of a Hadoop cluster?

- Pig is a programming language for describing data manipulation jobs
  - Think of your data like a big table
  - You can LOAD, FILTER, GROUP, etc rows in your table
  - Semi-structured data e.g. log files
  - http://pig.apache.org/

- Hive enables Hadoop to operate as a data warehouse.
  - It superimposes structure on data in HDFS
  - Good for static structured data
  - Permits SQL-like queries over the data
  - Like pig, can write custom modules

Pig programming stages

- LOAD data
- Series of transformations
- DUMP (view) or STORE (save) results

- Guide: http://pig.apache.org/docs/r0.12.1/start.html

Pig programming example

```
A = LOAD 'student' USING PigStorage()
     AS (name: chararray, age: int, gpa:float);
B = FOREACH A GENERATE name;
DUMP B;
(John)
(Mary)
(Bill)
(Joe)
```

'student'

name	age	gpa
John	42	3.5
Mary	25	4.0
Bill	31	3.0
Joe	29	3.7

Think of pig like SQL

Loads data from the file system.

LOAD 'data' [USING function] [AS schema];

records = load 'student.txt'

as (**name:chararray, age:int, gpa:double**);

Generates data transformations based on columns of data.

alias = **FOREACH** { gen_blk | nested_gen_blk } [AS schema];

Sometimes we want to eliminate nesting. This can be accomplished via the FLATTEN keyword.

words = foreach lines GENERATE

FLATTEN(TOKENIZE((chararray)$0)) as word;

Source: http://salsahpc.indiana.edu/ScienceCloud/pig_word_count_tutorial.htm

Think of pig like SQL

The GROUP operator groups together tuples that have the same group key (key field).

alias = **GROUP** alias { ALL | BY expression}

[, alias ALL | BY expression …] [USING 'collected'];

word_groups = **GROUP** words by word;

Use the COUNT function to compute the number of elements in a bag.

COUNT(expression)

D = foreach C generate **COUNT**(B), group;

Source: http://salsahpc.indiana.edu/ScienceCloud/pig_word_count_tutorial.htm

Word counting in Pig

```
-- 1. source = (sent)
source = LOAD 'input.txt' AS (sent:chararray);

-- 2. words = (word)
words = FOREACH source GENERATE
FLATTEN(TOKENIZE(sent)) as word;

-- 3. wordgroup = (group, words)
-- word is renamed to group
wordgroup = GROUP words BY word;

-- 4. results = (word_cnt, group)
results = FOREACH wordgroup GENERATE
COUNT(words) AS word_cnt, group;

-- 5. order by
results_sort = ORDER results by word_cnt DESC,
group;

-- 6. output
STORE results_sort INTO 'wordcount';
```

source =

sent
Sam is is the GSI
He is the best GSI ever

words =

word
Sam
is
is
the
GSI
He
is
the
best
GSI
ever

wordgroup =

group	words
Sam	[Sam]
is	[is, is, is]
my	[my]
GSI	[GSI, GSI]
He	[He]
the	[the,the]
best	[best]
ever	[ever]

Word counting in Pig

```
-- 1. source = (sent)
source = LOAD 'input.txt' AS (sent:chararray);

-- 2. words = (word)
words = FOREACH source GENERATE
FLATTEN(TOKENIZE(sent)) as word;

-- 3. wordgroup = (group, words)

wordgroup = GROUP words BY word;

-- 4. results = (word_cnt, group)
results = FOREACH wordgroup GENERATE
COUNT(words) AS word_cnt, group;

-- 5. order by
results_sort = ORDER results by word_cnt DESC,
group;

-- 6. output
STORE results_sort INTO 'wordcount';
```

wordgroup =

group	words
Sam	[Sam]
is	[is, is, is]
my	[my]
GSI	[GSI, GSI]
He	[He]
the	[the,the]
best	[best]
ever	[ever]

results =

word_cnt	group
1	Sam
3	is
1	my
2	GSI
.. etc.	.. etc..

results_sort =

word_cnt	group
2	is
1	Sam
1	GSI
1	my
.. etc..	..etc..

Working with data in Pig

- FILTER: work with tuples or rows
- FOREACH: work with columns
- GROUP: group data in single relation
- COGROUP and JOIN: group data w/ multi-relations
- UNION: merge results
- SPLIT: partition results

Usage details

- Pig is case-sensitive
- Single line comments: --
- Multi-line comments: /* … */

Using Pig to mine Twitter rumors

- Example courtesy of
  Zhe Zhao, ForeCeer group (Prof. Qiaozhu Mei)

Given 40,000,000 Tweets (10%)

STXherry Sherry
RT @AlinskyDefeater: Obama IS inspiring. Look how many people he's inspiring to vote Republican. #ocra #tcot
18 minutes ago

pleasuredaniel Daniel Lopes Mendes
as pessoas gostam de ir no playcenter e no hopi hari comigo pra ficarem rindo pra sempre dos meus sustos, tipo o lucas! D:
19 minutes ago

BWsocial Britty Wagner
How a Silicon Valley Investor Views a Post-Facebook World http://tinyurl.com/2al8e4m
19 minutes ago

tweetadderman LD Bland
It requires less character to discover the faults of others, than to tolerate them. ~J. Petit Senn
19 minutes ago Favorite Retweet Reply

CollChris Chris Collins
RT @twomaris Bagel thins look like lost hope.
19 minutes ago

charlesyeo Charles Yeo
Report: Relatively few people use cellphone apps http://bit.ly/bgUbJ8
19 minutes ago

taxcuts4all Dean A. Smith
RT @Conservativeind RT @ superlaura : RT @ iowahawkblog : #WhyImVotingDemocrat I'm too lazy to do my own stealing.
19 minutes ago

elwiciado Elwiz
Ola rs (@_leobaffe live on http://twitcam.com/2a4ko)
19 minutes ago

. .

Given 40,000,000 Tweets (10%)

STXherry Sherry
RT @AlinskyDefeater: Obama IS inspiring. Look how many people he's inspiring to vote Republican. #ocra #tcot
18 minutes ago

pleasuredaniel Daniel Lopes Mendes
as pessoas gostam de ir no playcenter e no hopi hari comigo pra ficarem rindo pra sempre dos meus sustos, tipo o lucas! D:
19 minutes ago

BWsocial Britty Wagner
How a Silicon Valley Investor Views a Post-Facebook World http://tinyurl.com/2al8e4m
19 minutes ago

tweetadderman LD Bland
It requires less character to discover the faults of others, than to tolerate them. ~J. Petit Senn
19 minutes ago Favorite Retweet Reply

CollChris Chris Collins
RT @twomaris Bagel thins look like lost hope.
19 minutes ago

charlesyeo Charles Yeo
Report: Relatively few people use cellphone apps http://bit.ly/bgUbJ8
19 minutes ago

taxcuts4all Dean A. Smith
RT @Conservativeind RT @ superlaura : RT @ iowahawkblog : #WhyImVotingDemocrat I'm too lazy to do my own stealing.
19 minutes ago

elwiciado Elwiz
Ola rs (@_leobaffe live on http://twitcam.com/2a4ko)
19 minutes ago

. . . . . .

Given 40,000,000 Tweets (10%)

. . . . . 20mins

Given 40,000,000 Tweets (10%)

. . . . . 20mins to locate a tweet using one core

Given 40,000,000 Tweets (10%)

. . . . . 20mins
to locate a tweet using one core

. 10secs
using MapReduce with 6 nodes, 66 cores

Given all tweets...

Given all tweets...

STXherry Sherry
RT @AlinskyDefeater: Obama IS inspiring. Look how many people he's inspiring to vote Republican. #ocra #tcot
18 minutes ago

pleasuredaniel Daniel Lopes Mendes
as pessoas gostam de ir no playcenter e no hopi hari comigo pra ficarem rindo pra sempre dos meus sustos, tipo o lucas! D:
19 minutes ago

BWsocial Britty Wagner
How a Silicon Valley Investor Views a Post-Facebook World http://tinyurl.com/2al8e4m
19 minutes ago

tweetadderman LD Bland
It requires less character to discover the faults of others, than to tolerate them. ~J. Petit Senn
19 minutes ago ☆ Favorite ↻ Retweet ↩ Reply

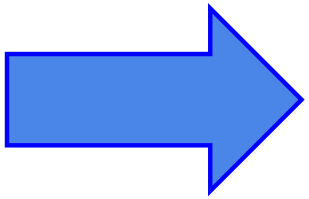
CollChris Chris Collins
RT @twomaris Bagel thins look like lost hope.
19 minutes ago

charlesyeo Charles Yeo
Report: Relatively few people use cellphone apps http://bit.ly/bgUbJ8
19 minutes ago

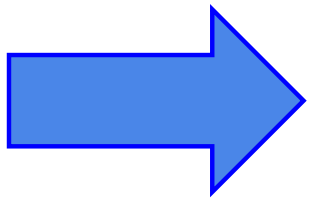
taxcuts4all Dean A. Smith
RT @Conservativeind RT @ superlaura : RT @ iowahawkblog : #WhyImVotingDemocrat I'm too lazy to do my own stealing.
19 minutes ago

elwiciado Elwiz
Ola rs (@_leobaffe live on http://twitcam.com/2a4ko)
19 minutes ago

Given all tweets...



Given all tweets...



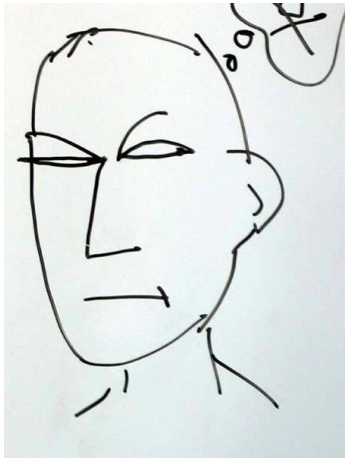
Mission Impossible for single core
~10 hrs using thousands of nodes,
tens of thousands of cores

A Simple Solution for Rumor Detection

@AP: Breaking: Two Explosions in the White House and Barack Obama is injured

A Simple Solution for Rumor Detection

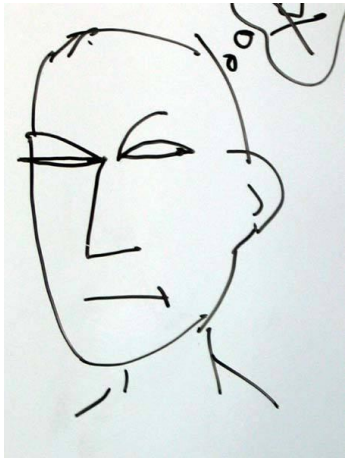
@AP: Breaking: Two Explosions in the White House and Barack Obama is injured



Is this true!! What!! "@AP: Breaking: Two Explosions in the White House and Barack Obama is injured"

A Simple Solution for Rumor Detection

@AP: Breaking: Two Explosions in the White House and Barack Obama is injured



Is this true!! What!! "@AP: Breaking: Two Explosions in the White House and Barack Obama is injured"

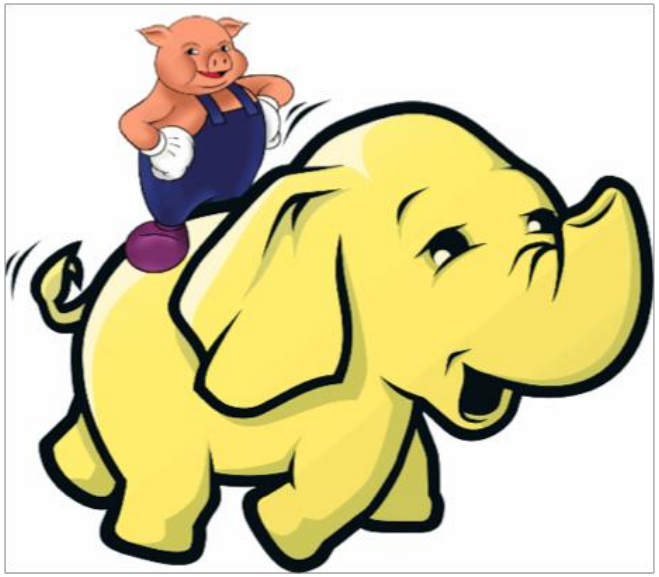
RT @Sipho_Tshabalal: **Is this a joke?** @Bhintsintsi: What?!! @AP: Breaking: Two Explosions in the White ... **http://t.co/PtgZQex1DW**

A Simple Solution for Rumor Detection

1. Read Tweets from file system
2. Find English tweets
3. Find English tweets that express suspicious attitude(contain pattern ‘is it true’)
4. Extract URL from tweets returned by step #3.
5. Count occurrences of the URL
6. Rank URLs based on occurrence count

1. Read Tweets from File System

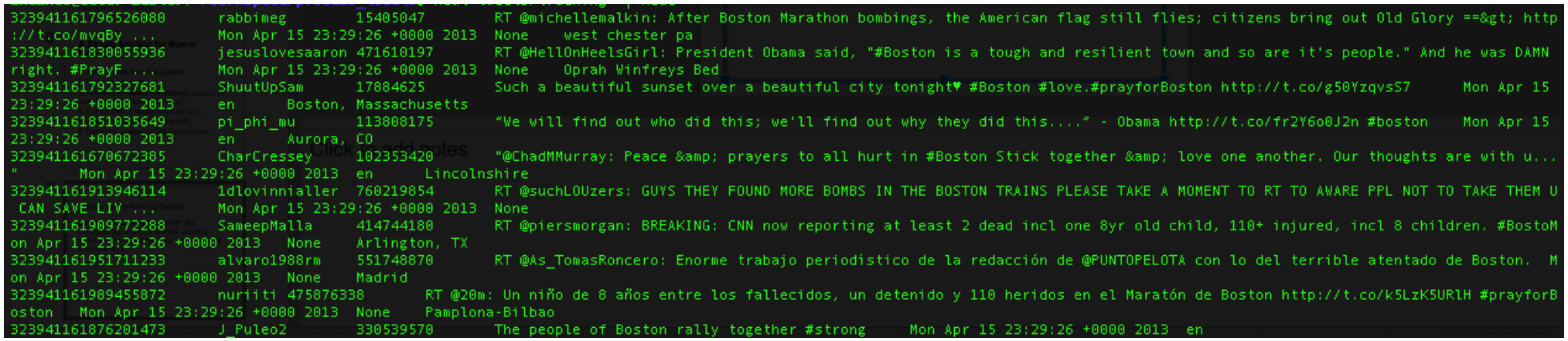
```
tweets = LOAD 'tweets/tracking*' AS
    (tid:chararray, uname:chararray, uid:long,
    text:chararray, date:chararray, lang:chararray,
    loc:chararray);
```



1. Read Tweets from File System

```
tweets = LOAD 'tweets/tracking*' AS
    (tid:chararray, uname:chararray, uid:long,
    text:chararray, date:chararray, lang:chararray,
    loc:chararray);
```

Schema of Input File



2. Find English tweets

```
tweets = FILTER tweets BY lang=='en';
```

3. Find English tweets that express suspicious attitude (e.g. contain pattern ‘is it true’)

```
tweets = FOREACH tweets GENERATE tid,
   uname, uid, text, date, lang, loc,
  REGEX_EXTRACT(LOWER(text),
        'is (this|it) true', 0) AS match;

tweets = FILTER tweets BY match is not null;
```

4. Extract URL from tweets returned by step 3.

```
Register '/home/users/rumorudf.py' using jython as eg_udfs;
urls = FOREACH tweets
            GENERATE FLATTEN( eg_udfs.extractURL(text) ) AS url;
```

User Defined Function(UDF): eg_udfs.py

```
import re
@outputSchema("urls:bag{(url:chararray)}")
def extractURL(text):
    return re.findall("(?P<url>https?://[^\s]+)", text)
```

5.Count the occurrence of the URL

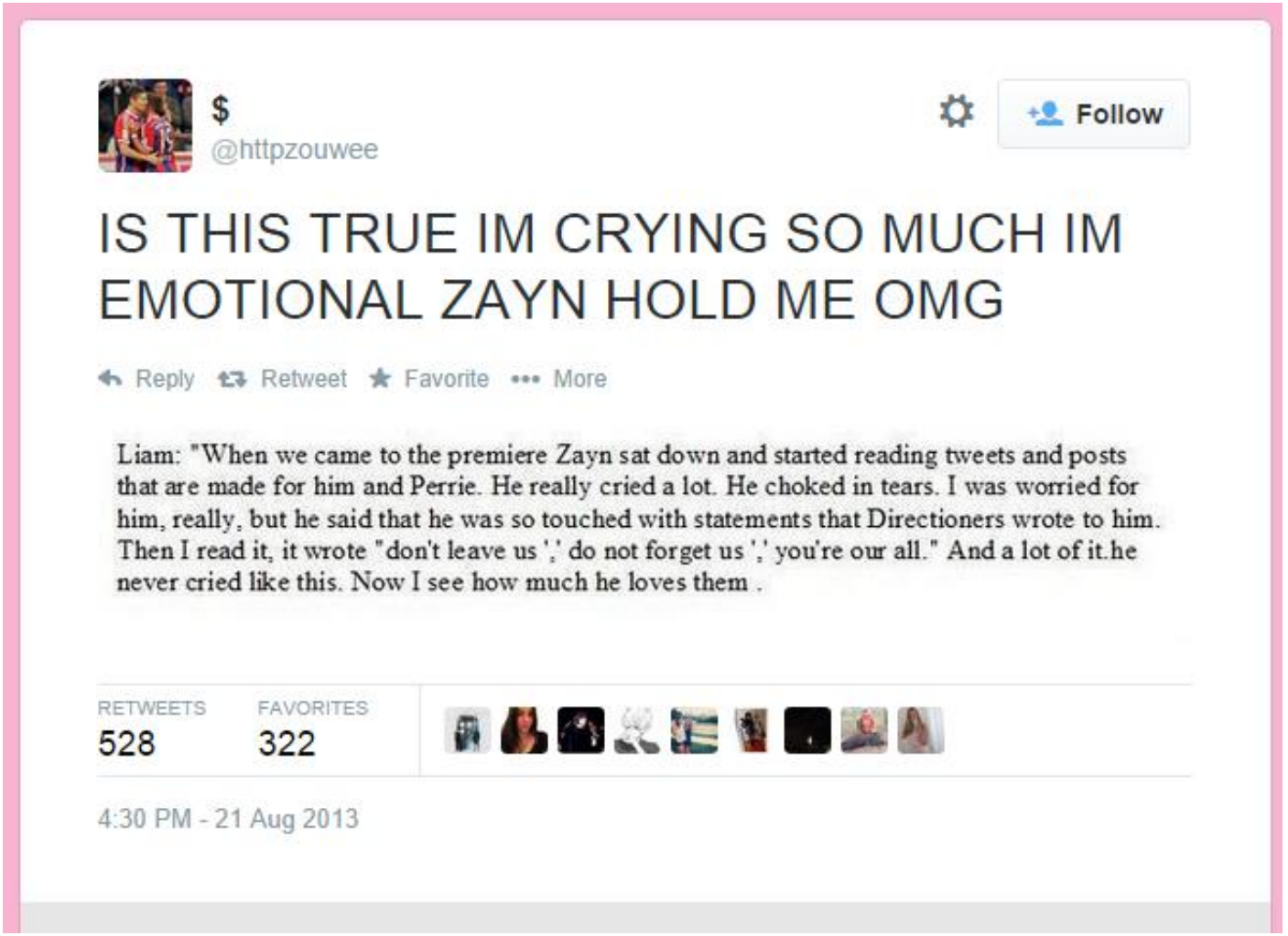
```
urls = FOREACH ( GROUP urls BY url ) GENERATE
   group AS url, COUNT(urls) AS count;
```

6. Rank URLs based on the occurrences.

```
urls = ORDER urls BY count DESC;

STORE urls INTO 'results/SI601/tracking/urls';
```

Output of running script on Hadoop cluster

50	http://t.co/2ERL5C7D4h
39	http://t.co/WCznFT1UnD
25	http://t.co/EV3n8hWVBC
17	http://t.co/SfQ30k4Sez
14	http://t.co/mylsFETAui
13	http://t.co/N10QQY3HNI
11	http://t.co/la14Vi2h7D
11	http://t.co/2zw4ie3K37
10	http://t.co/RRWMMKTU1V
10	http://t.co/EdmcVeCvoU
10	http://t.co/5q1ICMZUpq
9	http://t.co/rknho9v87A
9	http://t.co/PgBfec5EGq
9	http://t.co/IITpOS4lIj
8	http://t.co/rn6iFP3p1n
8	http://t.co/oXyy4vXKl7
8	http://t.co/o5SfLbQ7Qu
8	http://t.co/Jo9lisN40h
8	http://t.co/BKl9PbHh5R
8	http://t.co/7eRITgNT87
7	http://t.co/vKxaO5mVov
7	http://t.co/mdM8Tn3D5E
7	http://t.co/YWy0935ECA
7	http://t.co/VDLCJQaUJD
7	http://t.co/EI4SJj7p2z
6	http://t.co/yAayTU3k4k
6	http://t.co/tjXLtnChiV
6	http://t.co/gjbaSXpkZi
6	http://t.co/XsrgZJGGpk
6	http://t.co/WZ2K0omqnc
6	http://t.co/T6of5HI0fO
6	http://t.co/SoeaXJ1slY
6	http://t.co/QbA5cJHp9B
6	http://t.co/MHtzZzGHNe



Summary

- When to use Mapreduce:
  - Big data
  - Examine/Extract/Modify properties of each record, e.g., extract urls from each tweet
- When not to use Mapreduce:
  - More complex computation flow
  - Finding associations in record pairs, sets, e.g., calculate similarity between every two tweets.