



Intro To Hadoop

Bill Graham - @billgraham

Data Systems Engineer, Analytics Infrastructure

Info 290 - Analyzing Big Data With Twitter

UC Berkeley Information School

September 2012



This work is licensed under a Creative Commons Attribution-Noncommercial-Share Alike 3.0 United States
See <http://creativecommons.org/licenses/by-nc-sa/3.0/us/> for details

Outline

- What is Big Data?
- Hadoop
 - HDFS
 - MapReduce
- Twitter Analytics and Hadoop



What is big data?

- A bunch of data?
- An industry?
- An expertise?
- A trend?
- A cliché?



Wikipedia big data

*In information technology, big data is a loosely-defined term used to describe data sets so **large** and **complex** that they become **awkward** to work with using on-hand database management tools.*



Source: http://en.wikipedia.org/wiki/Big_data

How big is big?

- 2008: Google processes 20 PB a day
- 2009: Facebook has 2.5 PB user data + 15 TB/day
- 2009: eBay has 6.5 PB user data + 50 TB/day
- 2011: Yahoo! has 180-200 PB of data
- 2012: Facebook ingests 500 TB/day



That's a lot of data



Credit: <http://www.flickr.com/photos/19779889@N00/1367404058/>

So what?

s/data/knowledge/g



No really, what do you do with it?

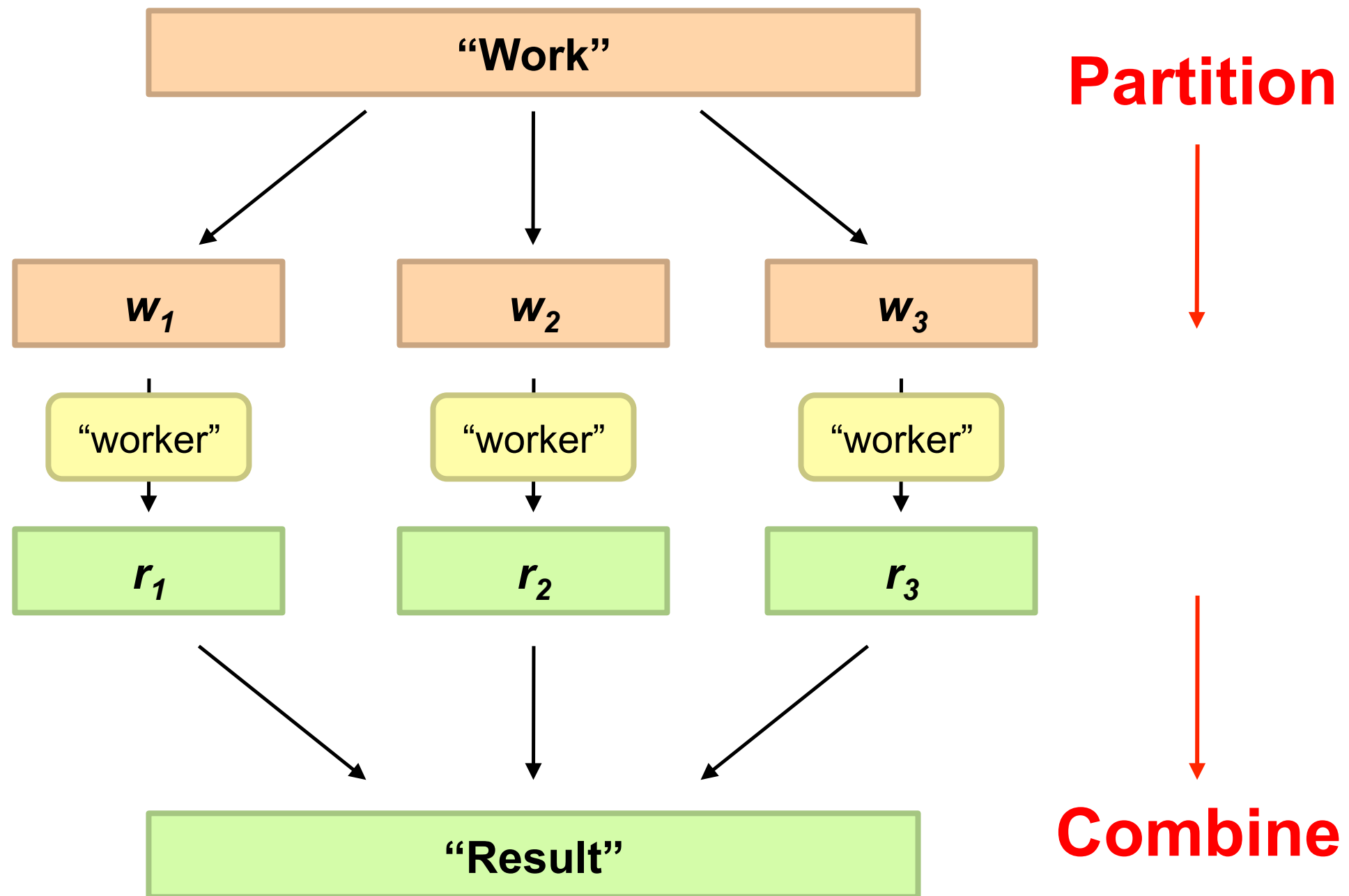
- User behavior analysis
- AB test analysis
- Ad targeting
- Trending topics
- User and topic modeling
- Recommendations
- And more...



How to scale data?



Divide and Conquer



Parallel processing is complicated

- How do we assign tasks to workers?
- What if we have more tasks than slots?
- What happens when tasks fail?
- How do you handle distributed synchronization?



Credit: <http://www.flickr.com/photos/sybrenstuvel/2468506922/>

Data storage is not trivial

- Data volumes are massive
- Reliably storing PBs of data is challenging
- Disk/hardware/network failures
- Probability of failure event increases with number of machines

For example:

1000 hosts, each with 10 disks

a disk lasts 3 year

how many failures per day?



Hadoop cluster



Cluster of machine running Hadoop at Yahoo! (credit: Yahoo!)



Hadoop



Hadoop provides

- Redundant, fault-tolerant data storage
- Parallel computation framework
- Job coordination



Joy



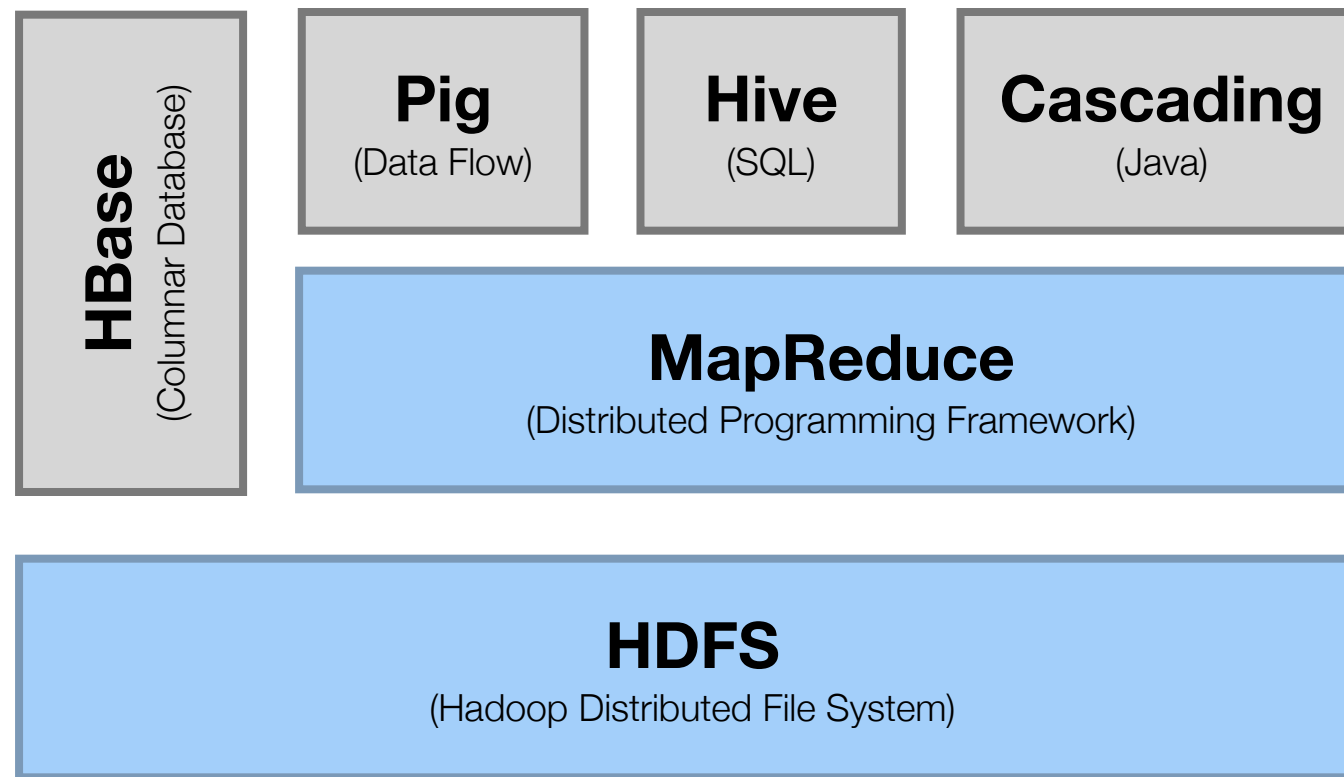
Credit: <http://www.flickr.com/photos/spyndle/3480602438/>

Hadoop origins

- Hadoop is an open-source implementation based on GFS and MapReduce from Google
- Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. (2003) The Google File System
- Jeffrey Dean and Sanjay Ghemawat. (2004) MapReduce: Simplified Data Processing on Large Clusters. OSDI 2004



Hadoop Stack



HDFS



HDFS is...

- A distributed file system
- Redundant storage
- Designed to reliably store data using commodity hardware
- Designed to expect hardware failures
- Intended for large files
- Designed for batch inserts
- The Hadoop Distributed File System



HDFS - files and blocks

- **Files** are stored as a collection of blocks
- **Blocks** are 64 MB chunks of a file (configurable)
- Blocks are replicated on 3 nodes (configurable)
- The **NameNode** (NN) manages metadata about files and blocks
- The **SecondaryNameNode** (SNN) holds a backup of the NN data
- **DataNodes** (DN) store and serve blocks

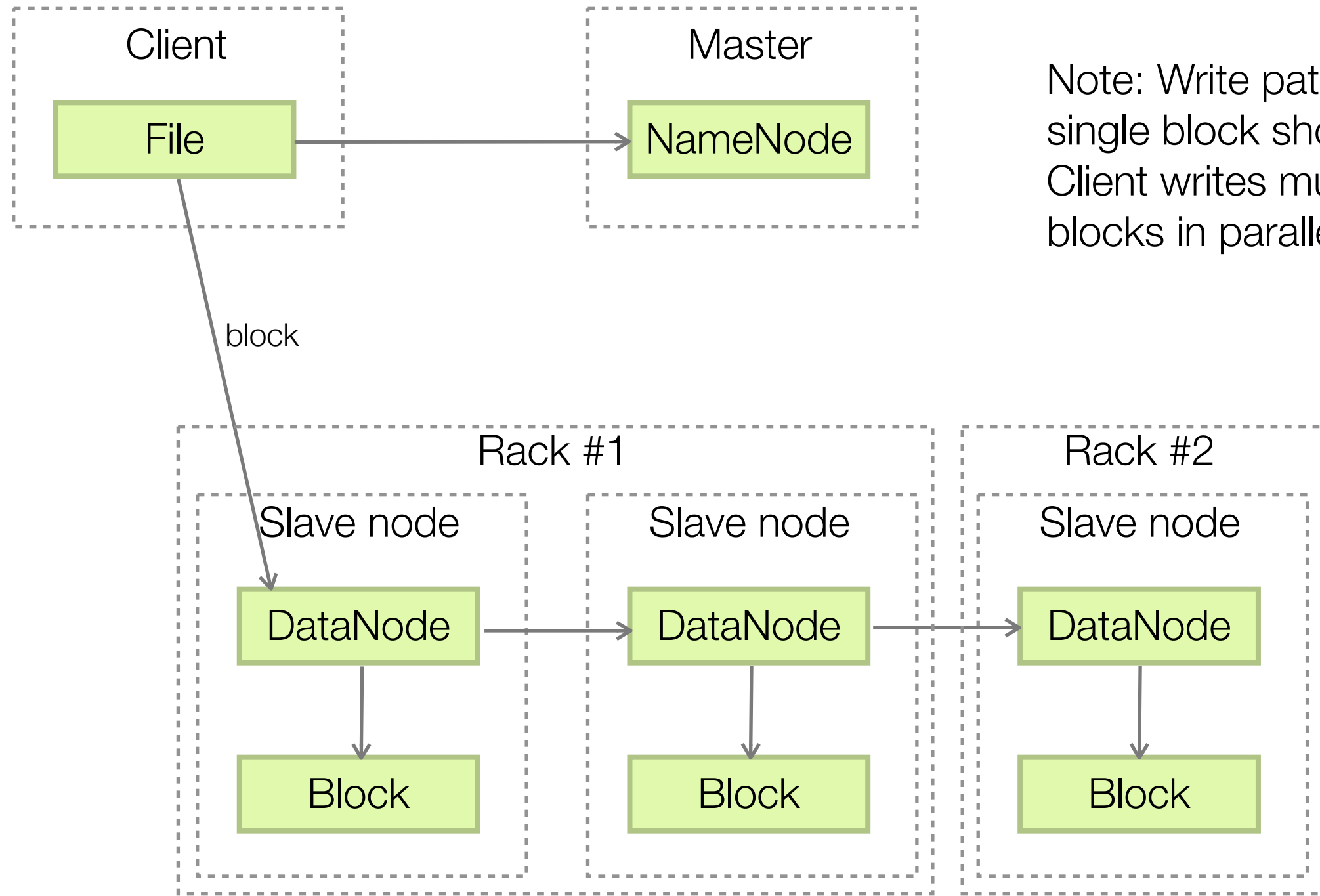


Replication

- Multiple copies of a block are stored
- Replication strategy:
 - Copy #1 on another node on same rack
 - Copy #2 on another node on different rack



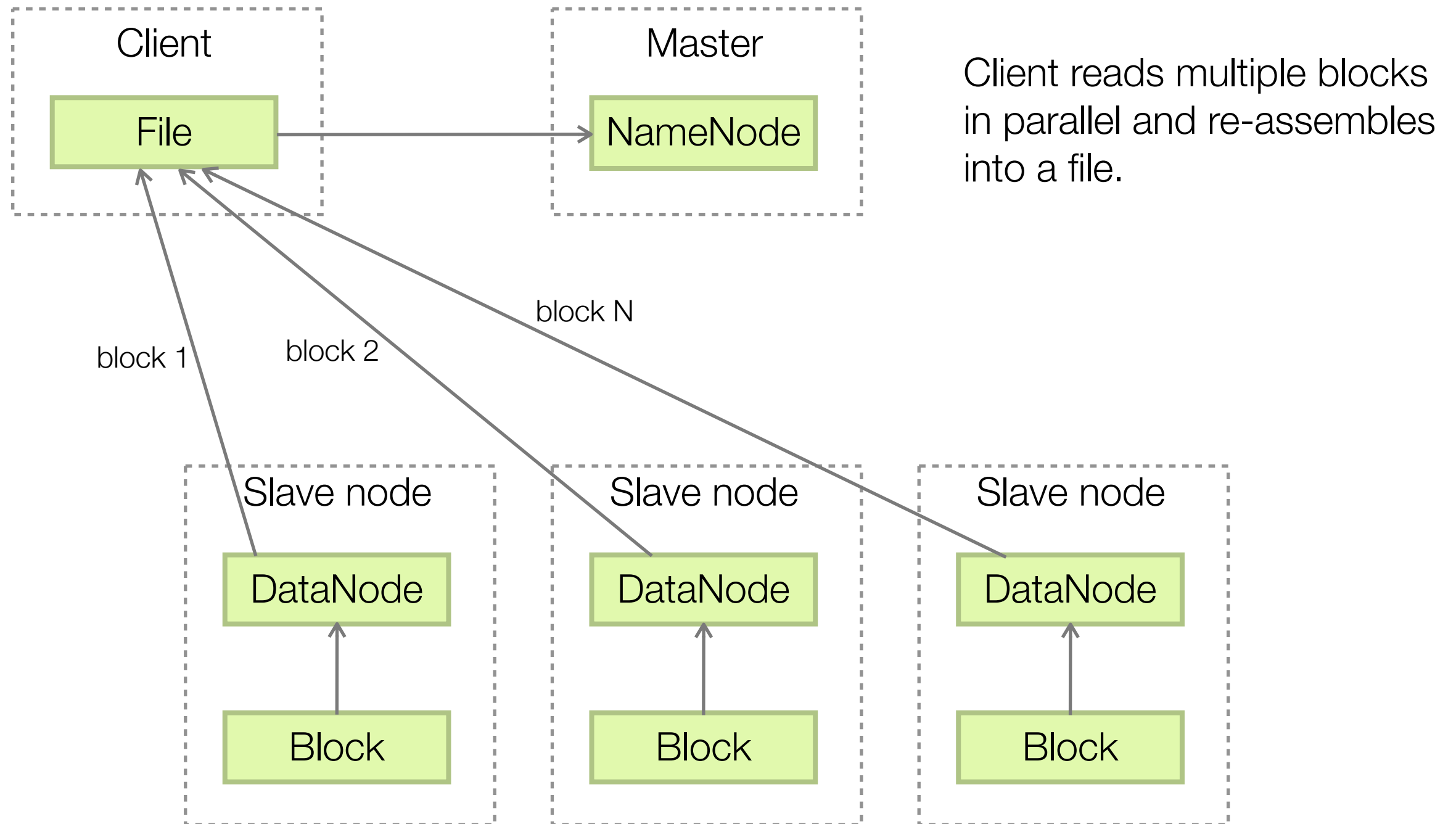
HDFS - writes



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



HDFS - reads



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/>

MapReduce

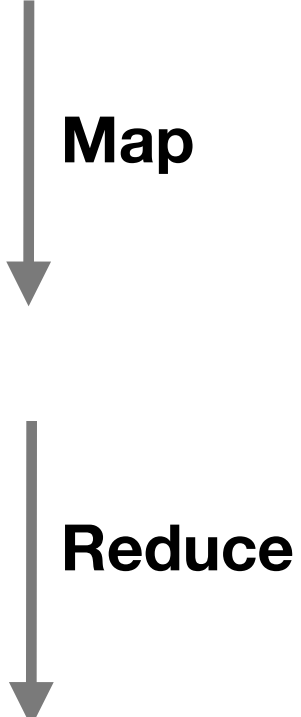


MapReduce is...

- A programming model for expressing distributed computations at a massive scale
- An execution framework for organizing and performing such computations
- An open-source implementation called Hadoop



Typical large-data problem

- Iterate over a large number of records
 - Extract something of interest from each
 - Shuffle and sort intermediate results
 - Aggregate intermediate results
 - Generate final output
- 
- The diagram illustrates the MapReduce process. It consists of two vertical arrows pointing downwards. The top arrow is labeled 'Map' and the bottom arrow is labeled 'Reduce'. The 'Map' arrow starts at the first two bullet points and ends at the third. The 'Reduce' arrow starts at the third bullet point and ends at the fifth.



MapReduce paradigm

- Implement two functions:

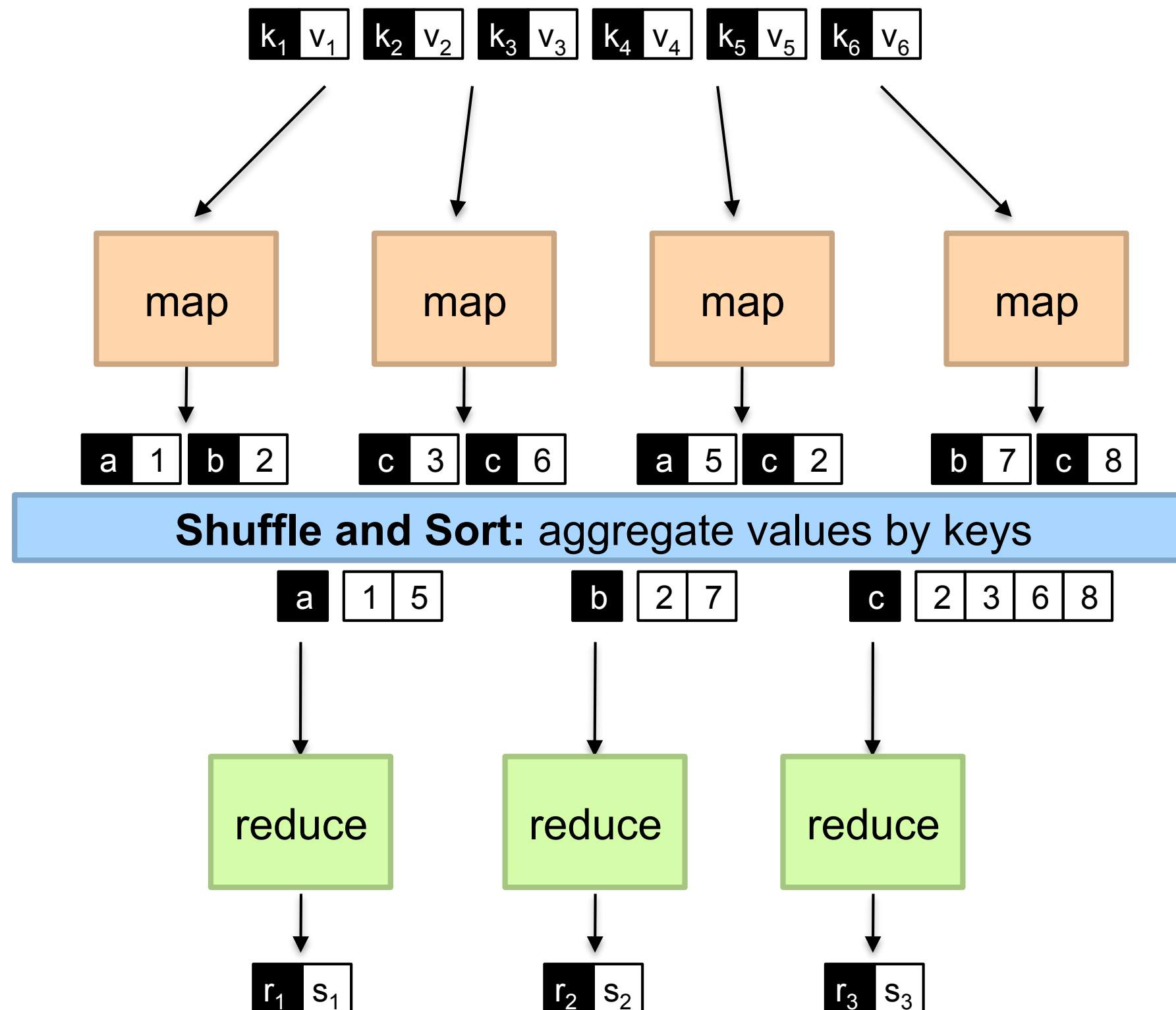
`Map(k1, v1) -> list(k2, v2)`

`Reduce(k2, list(v2)) -> list(v3)`

- Framework handles everything else*
- Value with same key go to same reducer



MapReduce Flow



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)
```

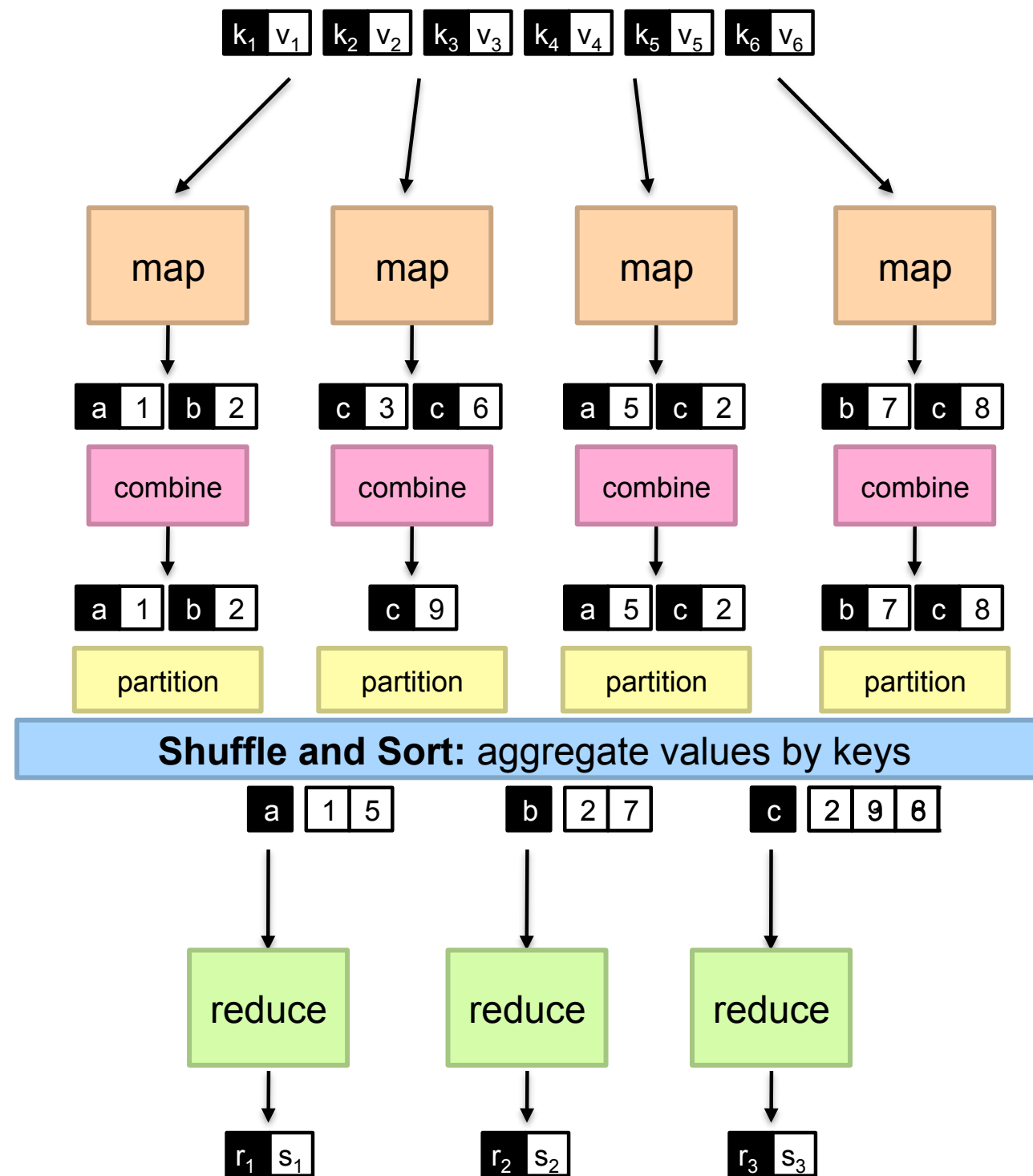


MapReduce paradigm - part 2

- There's more!
- **Partitioners** decide what key goes to what reducer
 - `partition(k', numPartitions) -> partNumber`
 - Divides key space into parallel reducers chunks
 - Default is hash-based
- **Combiners** can combine Mapper output before sending to reducer
 - `Reduce(k2, list(v2)) -> list(v3)`



MapReduce flow



MapReduce additional details

- Reduce starts after all mappers complete
- Mapper output gets written to disk
- Intermediate data can be copied sooner
- Reducer gets keys in sorted order
- Keys not sorted across reducers
- Global sort requires 1 reducer or smart partitioning

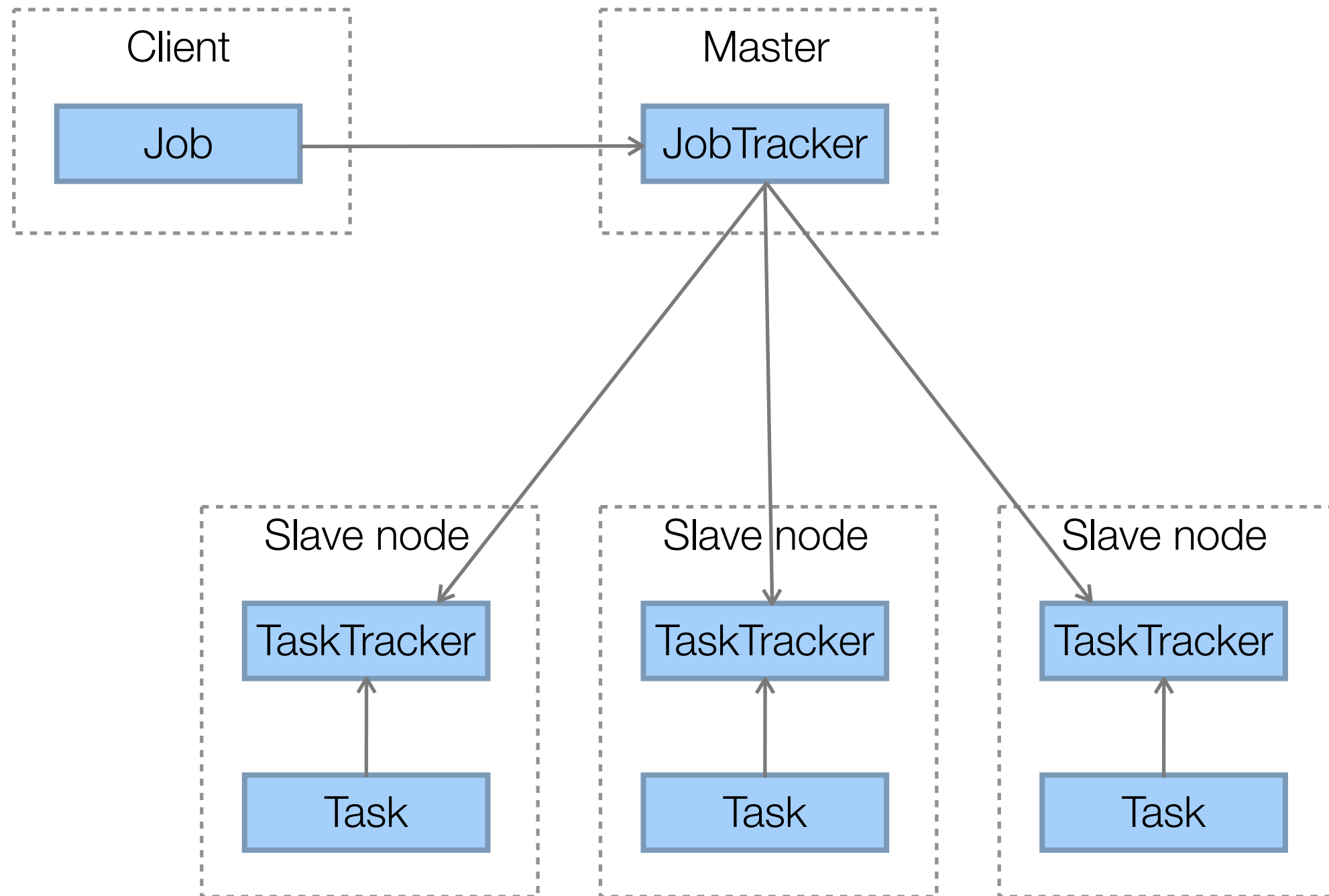


MapReduce - jobs and tasks

- **Job**: a user-submitted map and reduce implementation to apply to a data set
- **Task**: a single mapper or reducer task
 - Failed tasks get retried automatically
 - Tasks run local to their data, ideally
- **JobTracker** (JT) manages job submission and task delegation
- **TaskTrackers** (TT) ask for work and execute tasks



MapReduce architecture



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 data locality

- Move computation to the data
- Moving data between nodes has a cost
- MapReduce tries to schedule tasks on nodes with the data
- When not possible TT has to fetch data from DN



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)
```



MapReduce - Java API

- `Writable`
 - Hadoop wrapper interface
 - `Text`, `IntWritable`, `LongWritable`, etc
- `WritableComparable`
 - `Writable` classes implement `WritableComparable`
- `OutputCollector`
 - Class that collects keys and values
- `Reporter`
 - Reports progress, updates counters
- `InputFormat`
 - Reads data and provide `InputSplits`
 - Examples: `TextInputFormat`, `KeyValueTextInputFormat`
- `OutputFormat`
 - Writes data
 - Examples: `TextOutputFormat`, `SequenceFileOutputFormat`



MapReduce - Counters are...

- A distributed count of events during a job
- A way to indicate job metrics without logging
- Your friend

- Bad:

```
System.out.println("Couldn't parse value");
```

- Good:

```
reporter.incrCounter(BadParseEnum, 1L);
```



MapReduce - word count mapper

```
public static class Map 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 tokenizer = new StringTokenizer(line);
        while (tokenizer.hasMoreTokens()) {
            word.set(tokenizer.nextToken());
            output.collect(word, one);
        }
    }
}
```



MapReduce - word count reducer

```
public static 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));
    }
}
```



MapReduce - word count main

```
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);  
}
```



MapReduce - running a job

- To run word count, add files to HDFS and do:

```
$ bin/hadoop jar wordcount.jar  
org.myorg.WordCount input_dir output_dir
```



MapReduce is good for...

- Embarrassingly parallel algorithms
- Summing, grouping, filtering, joining
- Off-line batch jobs on massive data sets
- Analyzing an entire large dataset



MapReduce is ok for...

- Iterative jobs (i.e., graph algorithms)
 - Each iteration must read/write data to disk
 - IO and latency cost of an iteration is high

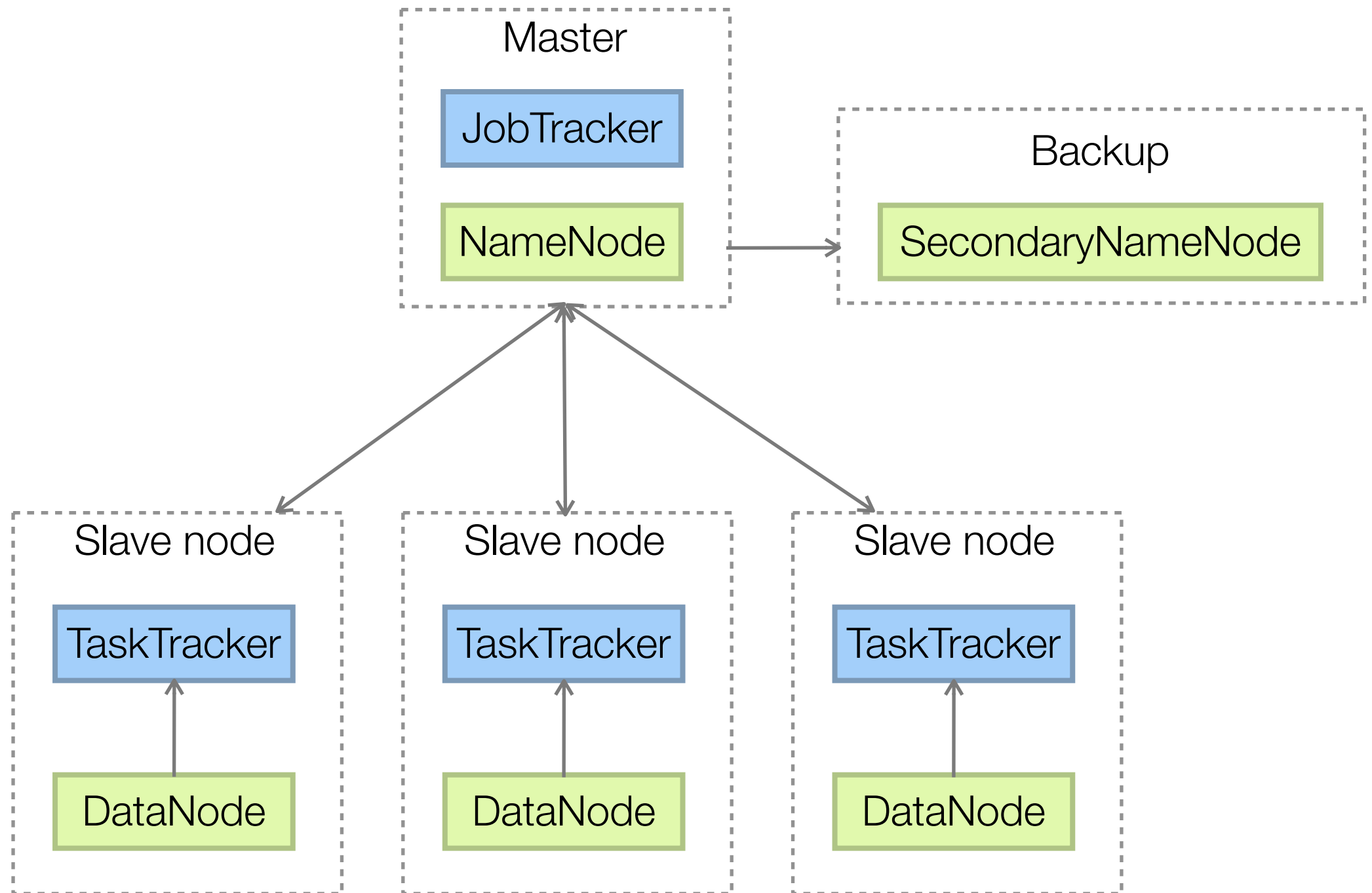


MapReduce is not good for...

- 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 combined architecture



NameNode UI

- Tool for browsing HDFS

[Browse the filesystem](#)

[Namenode Logs](#)

Cluster Summary

1232833 files and directories, 1223577 blocks = 2456410 total. Heap Size is 63.98 GB / 63.98 GB (100%)

Configured Capacity	:	860.48 TB
DFS Used	:	180.96 TB
Non DFS Used	:	0 KB
DFS Remaining	:	679.52 TB
DFS Used%	:	21.03 %
DFS Remaining%	:	78.97 %
Live Nodes	:	42
Dead Nodes	:	0
Decommissioning Nodes	:	0
Number of Under-Replicated Blocks	:	0



JobTracker UI

- Tool to see running/completed/failed jobs

Cluster Summary (Heap Size is 63.99 GB/63.99 GB)

Running Map Tasks	Running Reduce Tasks	Total Submissions	Nodes	Occupied Map Slots	Occupied Reduce Slots	Reserved Map Slots	Reserved Reduce Slots	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/No
0	0	5137	41	0	0	0	0	410	164	14.00

Scheduling Information

Queue Name	State	Scheduling Information
default	running	N/A

Filter (Jobid, Priority, User, Name)

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

Running Jobs

none

Completed Jobs

Jobid	Priority	User	Name	Map % Complete	Map Total	Maps Completed	Reduce % Complete	Reduce Total	Reduces Completed
job_201206241643_5131	NORMAL	hudson	word count	100.00% <div></div>	1	1	100.00% <div></div>	1	1
job_201206241643_5132	NORMAL	hudson	word count	100.00% <div></div>	1	1	100.00% <div></div>	1	1
job_201206241643_5133	NORMAL	hudson	word count	100.00% <div></div>	1	1	100.00% <div></div>	1	1



Running Hadoop

- Multiple options
- On your local machine (standalone or pseudo-distributed)
- Local with a virtual machine
- On the cloud (i.e. Amazon EC2)
- In your own datacenter



Cloudera VM

- Virtual machine with Hadoop and related technologies pre-loaded
- Great tool for learning Hadoop
- Eases the pain of downloading/installing
- Pre-loaded with sample data and jobs
- Documented tutorials
- VM: <https://ccp.cloudera.com/display/SUPPORT/Cloudera%27s+Hadoop+Demo+VM>
- Tutorial: <https://ccp.cloudera.com/display/SUPPORT/Hadoop+Tutorial>



Twitter Analytics and Hadoop

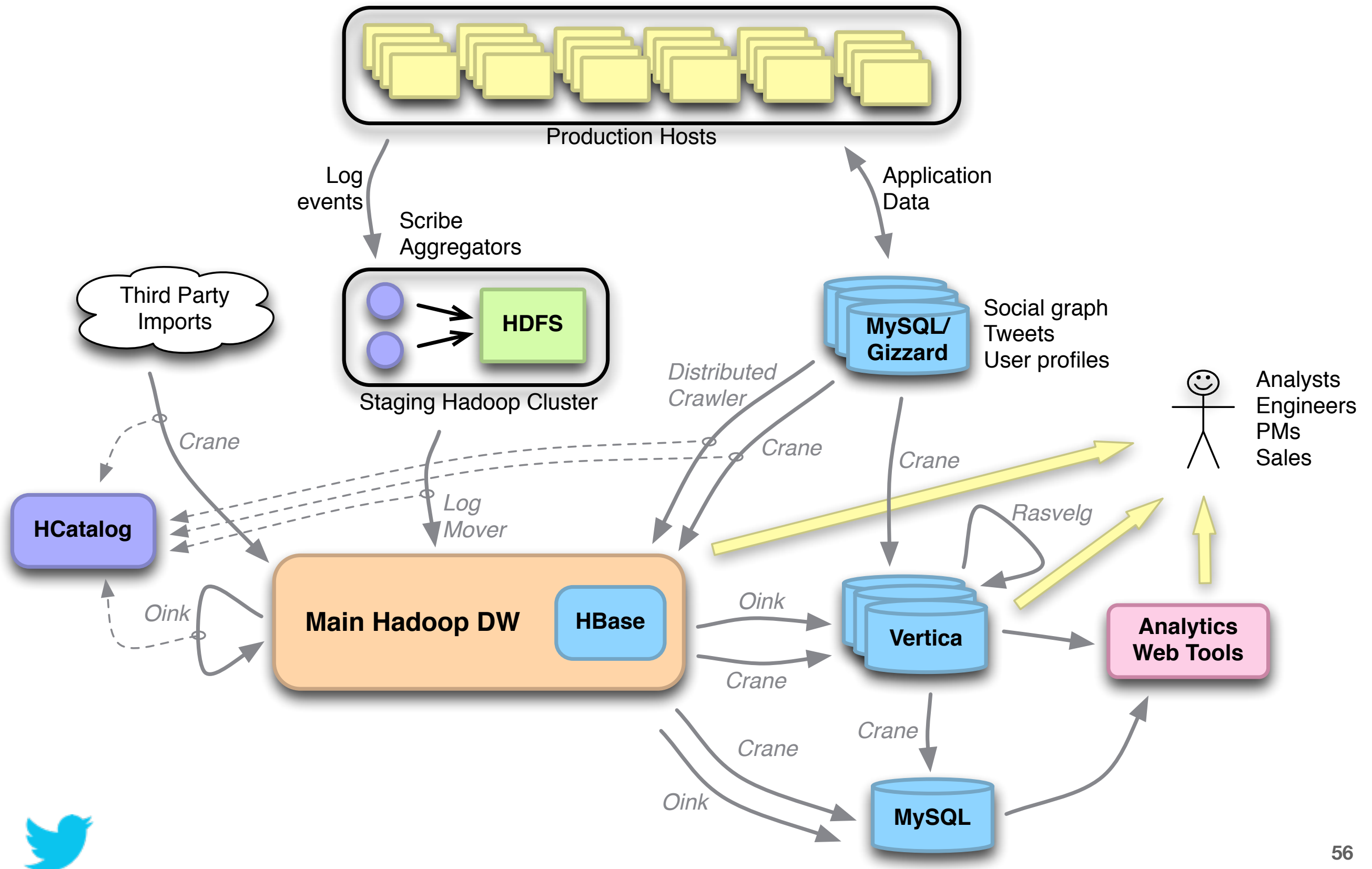


Multiple teams use Hadoop

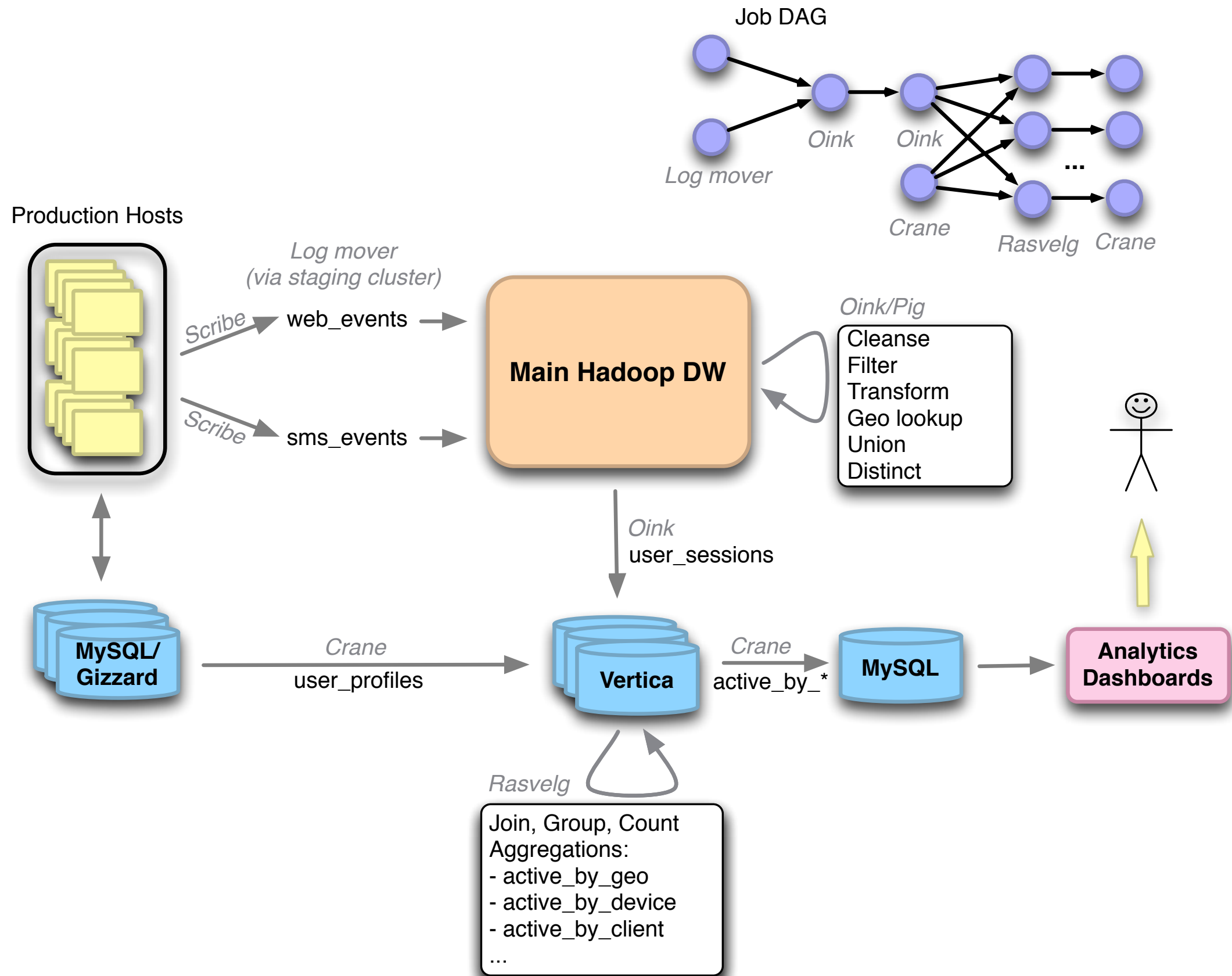
- Analytics
- Revenue
- Personalization & Recommendations
- Growth



Twitter Analytics data flow



Example: active users



Credits

- Data-Intensive Information Processing Applications — Session #1, Jimmy Lin, University of Maryland



Questions?

Bill Graham - @billgraham

