Lecture 25:

Spark

(leveraging bulk-granularity program structure)

Parallel Computer Architecture and Programming
CMU 15-418/15-618, Spring 2015

Tunes

Yeah Yeah Yeahs

Sacrilege

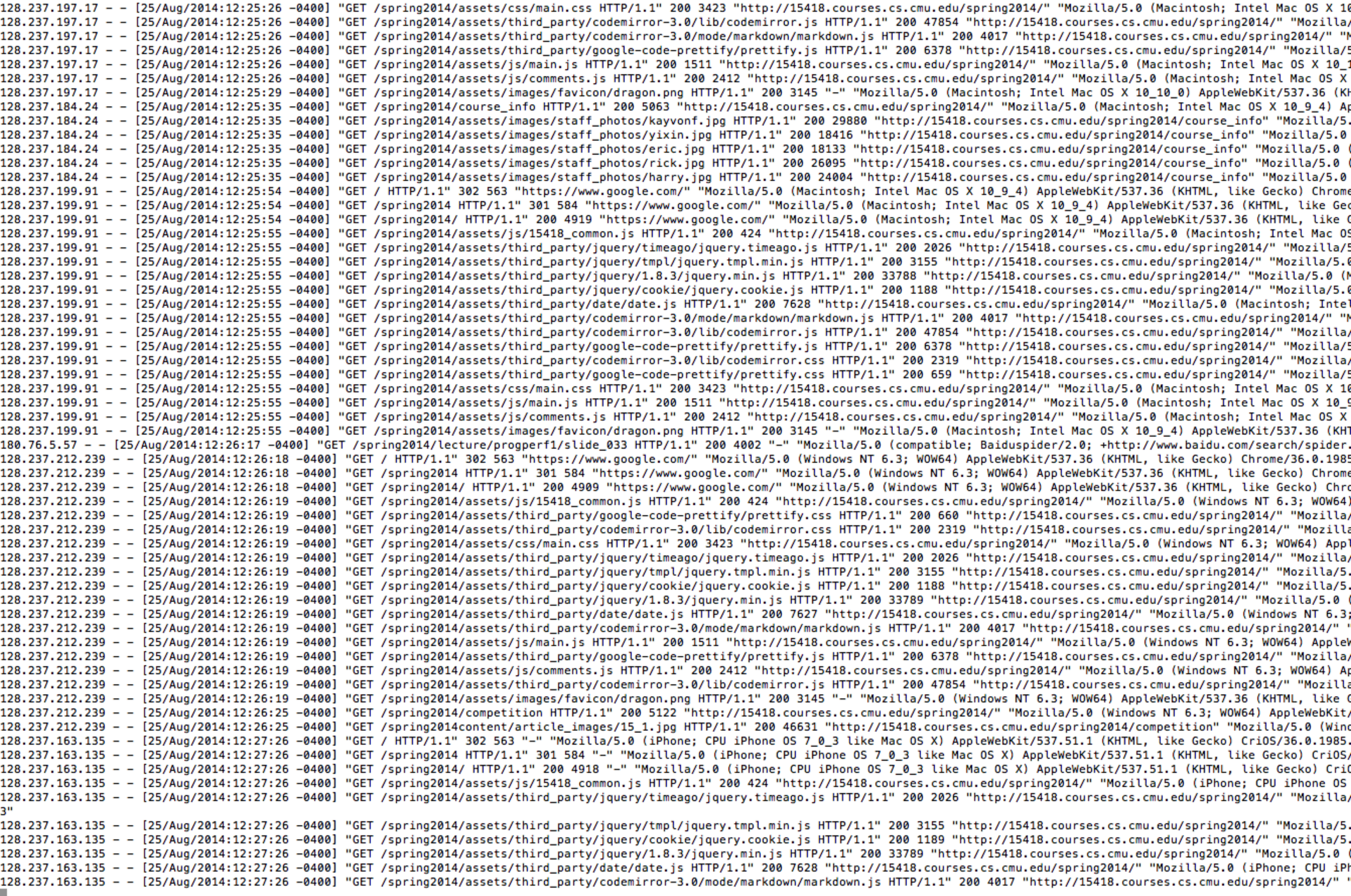
(Mosquito)

"In-memory performance and fault-tolerance across a cluster. No way!"

- Karen O

Analyzing site clicks

15418Log.txt



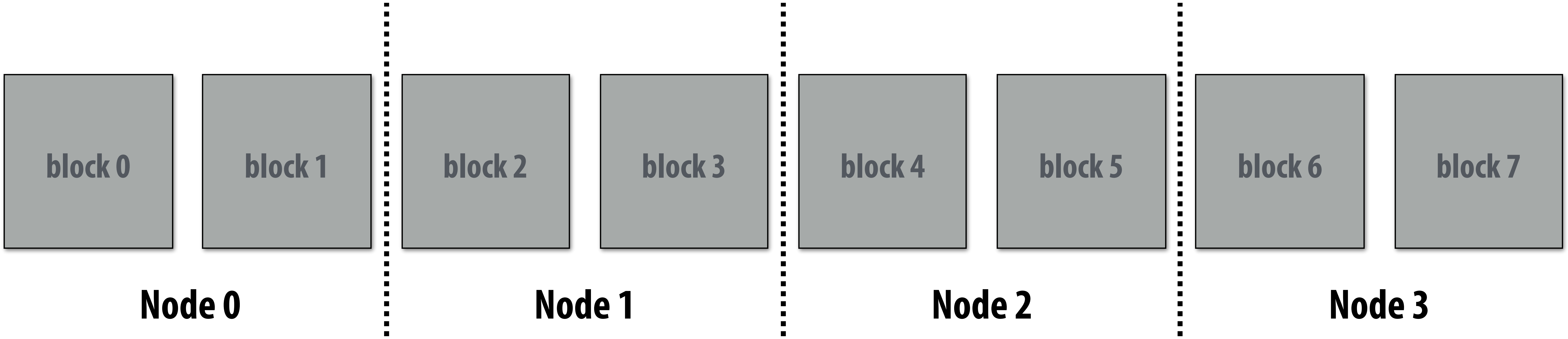
```
128.237.197.17 - - [25/Aug/2014:12:25:26 -0400] "GET /spring2014/assets/css/main.css HTTP/1.1" 200 3423 "http://15418.courses.cs.cmu.edu/spring2014/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10
128.237.197.17 - - [25/Aug/2014:12:25:26 -0400] "GET /spring2014/assets/third_party/codemirror-3.0/lib/codemirror.js HTTP/1.1" 200 47854 "http://15418.courses.cs.cmu.edu/spring2014/" "Mozilla/
128.237.197.17 - - [25/Aug/2014:12:25:26 -0400] "GET /spring2014/assets/third_party/codemirror-3.0/mode/markdown/markdown.js HTTP/1.1" 200 4017 "http://15418.courses.cs.cmu.edu/spring2014/" "M
128.237.197.17 - - [25/Aug/2014:12:25:26 -0400] "GET /spring2014/assets/third_party/google-code-prettify/prettify.js HTTP/1.1" 200 6378 "http://15418.courses.cs.cmu.edu/spring2014/" "Mozilla/5
128.237.197.17 - - [25/Aug/2014:12:25:26 -0400] "GET /spring2014/assets/js/main.js HTTP/1.1" 200 1511 "http://15418.courses.cs.cmu.edu/spring2014/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_1
128.237.197.17 - - [25/Aug/2014:12:25:26 -0400] "GET /spring2014/assets/js/comments.js HTTP/1.1" 200 2412 "http://15418.courses.cs.cmu.edu/spring2014/" "Mozilla/5.0 (Macintosh; Intel Mac OS X
128.237.197.17 - - [25/Aug/2014:12:25:29 -0400] "GET /spring2014/assets/images/favicon/dragon.png HTTP/1.1" 200 3145 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_0) AppleWebKit/537.36 (KH
128.237.184.24 - - [25/Aug/2014:12:25:35 -0400] "GET /spring2014/course_info HTTP/1.1" 200 5063 "http://15418.courses.cs.cmu.edu/spring2014/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) Ap
128.237.184.24 - - [25/Aug/2014:12:25:35 -0400] "GET /spring2014/assets/images/staff_photos/kayvonf.jpg HTTP/1.1" 200 29880 "http://15418.courses.cs.cmu.edu/spring2014/course_info" "Mozilla/5.
128.237.184.24 - - [25/Aug/2014:12:25:35 -0400] "GET /spring2014/assets/images/staff_photos/yixin.jpg HTTP/1.1" 200 18416 "http://15418.courses.cs.cmu.edu/spring2014/course_info" "Mozilla/5.0
128.237.184.24 - - [25/Aug/2014:12:25:35 -0400] "GET /spring2014/assets/images/staff_photos/eric.jpg HTTP/1.1" 200 18133 "http://15418.courses.cs.cmu.edu/spring2014/course_info" "Mozilla/5.0 (
128.237.184.24 - - [25/Aug/2014:12:25:35 -0400] "GET /spring2014/assets/images/staff_photos/rick.jpg HTTP/1.1" 200 26095 "http://15418.courses.cs.cmu.edu/spring2014/course_info" "Mozilla/5.0 (
128.237.184.24 - - [25/Aug/2014:12:25:35 -0400] "GET /spring2014/assets/images/staff_photos/harry.jpg HTTP/1.1" 200 24004 "http://15418.courses.cs.cmu.edu/spring2014/course_info" "Mozilla/5.0
128.237.199.91 - - [25/Aug/2014:12:25:54 -0400] "GET / HTTP/1.1" 302 563 "https://www.google.com/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome
128.237.199.91 - - [25/Aug/2014:12:25:54 -0400] "GET /spring2014 HTTP/1.1" 301 584 "https://www.google.com/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.36 (KHTML, like Gec
128.237.199.91 - - [25/Aug/2014:12:25:54 -0400] "GET /spring2014/ HTTP/1.1" 200 4919 "https://www.google.com/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.36 (KHTML, like G
128.237.199.91 - - [25/Aug/2014:12:25:55 -0400] "GET /spring2014/assets/js/15418_common.js HTTP/1.1" 200 424 "http://15418.courses.cs.cmu.edu/spring2014/" "Mozilla/5.0 (Macintosh; Intel Mac OS
128.237.199.91 - - [25/Aug/2014:12:25:55 -0400] "GET /spring2014/assets/third_party/jquery/timeago/jquery.timeago.js HTTP/1.1" 200 2026 "http://15418.courses.cs.cmu.edu/spring2014/" "Mozilla/5
128.237.199.91 - - [25/Aug/2014:12:25:55 -0400] "GET /spring2014/assets/third_party/jquery/tmpl/jquery.tmpl.min.js HTTP/1.1" 200 3155 "http://15418.courses.cs.cmu.edu/spring2014/" "Mozilla/5.0
128.237.199.91 - - [25/Aug/2014:12:25:55 -0400] "GET /spring2014/assets/third_party/jquery/1.8.3/jquery.min.js HTTP/1.1" 200 33788 "http://15418.courses.cs.cmu.edu/spring2014/" "Mozilla/5.0 (M
128.237.199.91 - - [25/Aug/2014:12:25:55 -0400] "GET /spring2014/assets/third_party/jquery/cookie/jquery.cookie.js HTTP/1.1" 200 1188 "http://15418.courses.cs.cmu.edu/spring2014/" "Mozilla/5.0
128.237.199.91 - - [25/Aug/2014:12:25:55 -0400] "GET /spring2014/assets/third_party/date/date.js HTTP/1.1" 200 7628 "http://15418.courses.cs.cmu.edu/spring2014/" "Mozilla/5.0 (Macintosh; Intel
128.237.199.91 - - [25/Aug/2014:12:25:55 -0400] "GET /spring2014/assets/third_party/codemirror-3.0/mode/markdown/markdown.js HTTP/1.1" 200 4017 "http://15418.courses.cs.cmu.edu/spring2014/" "M
128.237.199.91 - - [25/Aug/2014:12:25:55 -0400] "GET /spring2014/assets/third_party/codemirror-3.0/lib/codemirror.js HTTP/1.1" 200 47854 "http://15418.courses.cs.cmu.edu/spring2014/" "Mozilla/
128.237.199.91 - - [25/Aug/2014:12:25:55 -0400] "GET /spring2014/assets/third_party/google-code-prettify/prettify.js HTTP/1.1" 200 6378 "http://15418.courses.cs.cmu.edu/spring2014/" "Mozilla/5
128.237.199.91 - - [25/Aug/2014:12:25:55 -0400] "GET /spring2014/assets/third_party/codemirror-3.0/lib/codemirror.css HTTP/1.1" 200 2319 "http://15418.courses.cs.cmu.edu/spring2014/" "Mozilla/
128.237.199.91 - - [25/Aug/2014:12:25:55 -0400] "GET /spring2014/assets/third_party/google-code-prettify/prettify.css HTTP/1.1" 200 659 "http://15418.courses.cs.cmu.edu/spring2014/" "Mozilla/5
128.237.199.91 - - [25/Aug/2014:12:25:55 -0400] "GET /spring2014/assets/css/main.css HTTP/1.1" 200 3423 "http://15418.courses.cs.cmu.edu/spring2014/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10
128.237.199.91 - - [25/Aug/2014:12:25:55 -0400] "GET /spring2014/assets/js/main.js HTTP/1.1" 200 1511 "http://15418.courses.cs.cmu.edu/spring2014/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9
128.237.199.91 - - [25/Aug/2014:12:25:55 -0400] "GET /spring2014/assets/js/comments.js HTTP/1.1" 200 2412 "http://15418.courses.cs.cmu.edu/spring2014/" "Mozilla/5.0 (Macintosh; Intel Mac OS X
128.237.199.91 - - [25/Aug/2014:12:25:55 -0400] "GET /spring2014/assets/images/favicon/dragon.png HTTP/1.1" 200 3145 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.36 (KHT
180.76.5.57 - - [25/Aug/2014:12:26:17 -0400] "GET /spring2014/lecture/progperf1/slide_033 HTTP/1.1" 200 4002 "-" "Mozilla/5.0 (compatible; Baiduspider/2.0; +http://www.baidu.com/search/spider.
128.237.212.239 - - [25/Aug/2014:12:26:18 -0400] "GET / HTTP/1.1" 302 563 "https://www.google.com/" "Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1985
128.237.212.239 - - [25/Aug/2014:12:26:18 -0400] "GET /spring2014 HTTP/1.1" 301 584 "https://www.google.com/" "Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome
128.237.212.239 - - [25/Aug/2014:12:26:18 -0400] "GET /spring2014/ HTTP/1.1" 200 4909 "https://www.google.com/" "Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chro
128.237.212.239 - - [25/Aug/2014:12:26:19 -0400] "GET /spring2014/assets/js/15418_common.js HTTP/1.1" 200 424 "http://15418.courses.cs.cmu.edu/spring2014/" "Mozilla/5.0 (Windows NT 6.3; WOW64)
128.237.212.239 - - [25/Aug/2014:12:26:19 -0400] "GET /spring2014/assets/third_party/google-code-prettify/prettify.css HTTP/1.1" 200 660 "http://15418.courses.cs.cmu.edu/spring2014/" "Mozilla/
128.237.212.239 - - [25/Aug/2014:12:26:19 -0400] "GET /spring2014/assets/third_party/codemirror-3.0/lib/codemirror.css HTTP/1.1" 200 2319 "http://15418.courses.cs.cmu.edu/spring2014/" "Mozilla
128.237.212.239 - - [25/Aug/2014:12:26:19 -0400] "GET /spring2014/assets/css/main.css HTTP/1.1" 200 3423 "http://15418.courses.cs.cmu.edu/spring2014/" "Mozilla/5.0 (Windows NT 6.3; WOW64) Appl
128.237.212.239 - - [25/Aug/2014:12:26:19 -0400] "GET /spring2014/assets/third_party/jquery/timeago/jquery.timeago.js HTTP/1.1" 200 2026 "http://15418.courses.cs.cmu.edu/spring2014/" "Mozilla/
128.237.212.239 - - [25/Aug/2014:12:26:19 -0400] "GET /spring2014/assets/third_party/jquery/tmpl/jquery.tmpl.min.js HTTP/1.1" 200 3155 "http://15418.courses.cs.cmu.edu/spring2014/" "Mozilla/5.
128.237.212.239 - - [25/Aug/2014:12:26:19 -0400] "GET /spring2014/assets/third_party/jquery/cookie/jquery.cookie.js HTTP/1.1" 200 1188 "http://15418.courses.cs.cmu.edu/spring2014/" "Mozilla/5.
128.237.212.239 - - [25/Aug/2014:12:26:19 -0400] "GET /spring2014/assets/third_party/jquery/1.8.3/jquery.min.js HTTP/1.1" 200 33789 "http://15418.courses.cs.cmu.edu/spring2014/" "Mozilla/5.0 (
128.237.212.239 - - [25/Aug/2014:12:26:19 -0400] "GET /spring2014/assets/third_party/date/date.js HTTP/1.1" 200 7627 "http://15418.courses.cs.cmu.edu/spring2014/" "Mozilla/5.0 (Windows NT 6.3;
128.237.212.239 - - [25/Aug/2014:12:26:19 -0400] "GET /spring2014/assets/third_party/codemirror-3.0/mode/markdown/markdown.js HTTP/1.1" 200 4017 "http://15418.courses.cs.cmu.edu/spring2014/" "
128.237.212.239 - - [25/Aug/2014:12:26:19 -0400] "GET /spring2014/assets/js/main.js HTTP/1.1" 200 1511 "http://15418.courses.cs.cmu.edu/spring2014/" "Mozilla/5.0 (Windows NT 6.3; WOW64) AppleW
128.237.212.239 - - [25/Aug/2014:12:26:19 -0400] "GET /spring2014/assets/third_party/google-code-prettify/prettify.js HTTP/1.1" 200 6378 "http://15418.courses.cs.cmu.edu/spring2014/" "Mozilla/
128.237.212.239 - - [25/Aug/2014:12:26:19 -0400] "GET /spring2014/assets/js/comments.js HTTP/1.1" 200 2412 "http://15418.courses.cs.cmu.edu/spring2014/" "Mozilla/5.0 (Windows NT 6.3; WOW64) Ap
128.237.212.239 - - [25/Aug/2014:12:26:19 -0400] "GET /spring2014/assets/third_party/codemirror-3.0/lib/codemirror.js HTTP/1.1" 200 47854 "http://15418.courses.cs.cmu.edu/spring2014/" "Mozilla
128.237.212.239 - - [25/Aug/2014:12:26:19 -0400] "GET /spring2014/assets/images/favicon/dragon.png HTTP/1.1" 200 3145 "-" "Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like G
128.237.212.239 - - [25/Aug/2014:12:26:25 -0400] "GET /spring2014/competition HTTP/1.1" 200 5122 "http://15418.courses.cs.cmu.edu/spring2014/" "Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/
128.237.212.239 - - [25/Aug/2014:12:26:25 -0400] "GET /spring2014content/article_images/15_1.jpg HTTP/1.1" 200 46631 "http://15418.courses.cs.cmu.edu/spring2014/competition" "Mozilla/5.0 (Wind
128.237.163.135 - - [25/Aug/2014:12:27:26 -0400] "GET / HTTP/1.1" 302 563 "-" "Mozilla/5.0 (iPhone; CPU iPhone OS 7_0_3 like Mac OS X) AppleWebKit/537.51.1 (KHTML, like Gecko) CriOS/36.0.1985.
128.237.163.135 - - [25/Aug/2014:12:27:26 -0400] "GET /spring2014 HTTP/1.1" 301 584 "-" "Mozilla/5.0 (iPhone; CPU iPhone OS 7_0_3 like Mac OS X) AppleWebKit/537.51.1 (KHTML, like Gecko) CriOS/
128.237.163.135 - - [25/Aug/2014:12:27:26 -0400] "GET /spring2014/ HTTP/1.1" 200 4918 "-" "Mozilla/5.0 (iPhone; CPU iPhone OS 7_0_3 like Mac OS X) AppleWebKit/537.51.1 (KHTML, like Gecko) CriO
128.237.163.135 - - [25/Aug/2014:12:27:26 -0400] "GET /spring2014/assets/js/15418_common.js HTTP/1.1" 200 424 "http://15418.courses.cs.cmu.edu/spring2014/" "Mozilla/5.0 (iPhone; CPU iPhone OS
128.237.163.135 - - [25/Aug/2014:12:27:26 -0400] "GET /spring2014/assets/third_party/jquery/timeago/jquery.timeago.js HTTP/1.1" 200 2026 "http://15418.courses.cs.cmu.edu/spring2014/" "Mozilla/
3"
128.237.163.135 - - [25/Aug/2014:12:27:26 -0400] "GET /spring2014/assets/third_party/jquery/tmpl/jquery.tmpl.min.js HTTP/1.1" 200 3155 "http://15418.courses.cs.cmu.edu/spring2014/" "Mozilla/5.
128.237.163.135 - - [25/Aug/2014:12:27:26 -0400] "GET /spring2014/assets/third_party/jquery/cookie/jquery.cookie.js HTTP/1.1" 200 1189 "http://15418.courses.cs.cmu.edu/spring2014/" "Mozilla/5.
128.237.163.135 - - [25/Aug/2014:12:27:26 -0400] "GET /spring2014/assets/third_party/jquery/1.8.3/jquery.min.js HTTP/1.1" 200 33789 "http://15418.courses.cs.cmu.edu/spring2014/" "Mozilla/5.0 (
128.237.163.135 - - [25/Aug/2014:12:27:26 -0400] "GET /spring2014/assets/third_party/date/date.js HTTP/1.1" 200 7628 "http://15418.courses.cs.cmu.edu/spring2014/" "Mozilla/5.0 (iPhone; CPU iPh
128.237.163.135 - - [25/Aug/2014:12:27:26 -0400] "GET /spring2014/assets/third_party/codemirror-3.0/mode/markdown/markdown.js HTTP/1.1" 200 4017 "http://15418.courses.cs.cmu.edu/spring2014/" "
```

A simple programming model

```
// called once per line in file
void mapper(string line, map<string,string> results) {
   string useragent = parse_requester_useragent(line);
   if (is_mobile_client(useragent))
     results.add(useragent, 1);
}

// called once per unique key in results
void reducer(string key, list<string> values, int& result) {
    int sum = 0;
    for (v in values)
       sum += v;
    result = sum;
}

LineByLineReader input("hdfs://15418log.txt");
Writer output("hdfs://…");
runMapReduceJob(mapper, reducer, input, output);
```

Example:
Count number of page views made from each type of mobile client.

Assume: 15418Log.txt is a large file, stored in a distributed file system, like HDFS



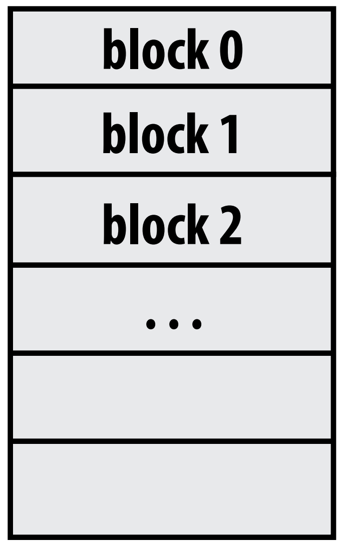
Let’s design an implementation of runMapReduceJob

Step 1: running the mapper function

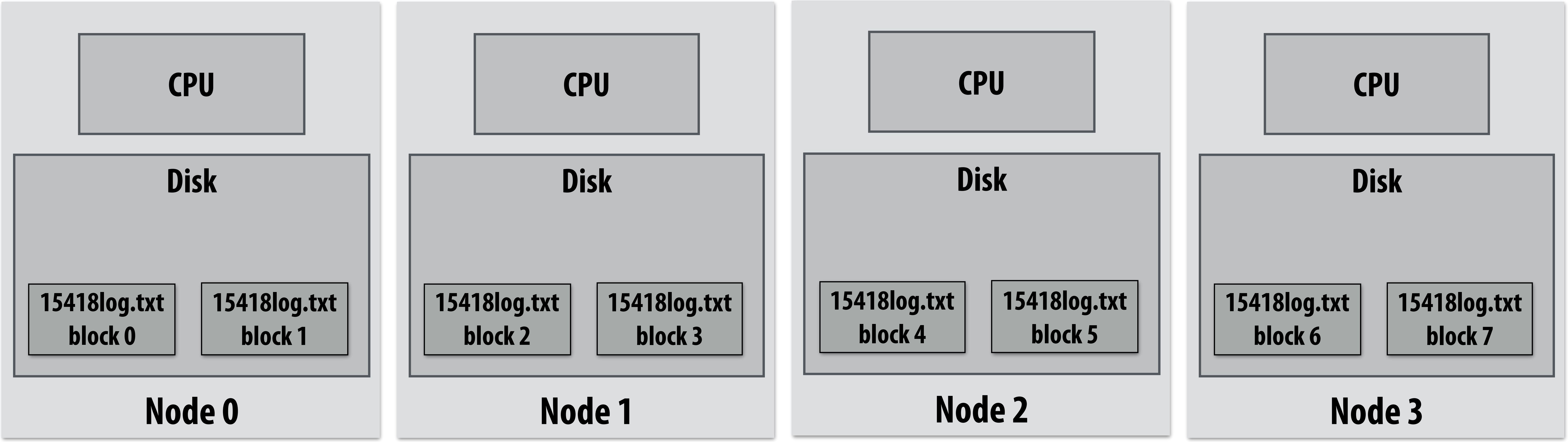
```
// called once per line in file
void mapper(string line, map<string,string> results) {
   string useragent = parse_requester_useragent(line);
   if (is_mobile_client(useragent))
     results.add(useragent, 1);
}

// called once per unique key in results
void reducer(string key, list<string> values, int& result) {
    int sum = 0;
    for (v in values)
       sum += v;
    result = sum;
}

LineByLineReader input(“hdfs://15418log.txt”);
Writer output(“hdfs://…”);
runMapReduceJob(mapper, reducer, input, output);
```

Step 1: run mapper function on all lines of file
Question: How to assign work to nodes?

Idea 1: use work queue for list of input blocks to process
Dynamic assignment: free node takes next available block



Idea 2: data distribution based assignment: Each node processes lines in blocks of input file that are stored locally.



Steps 2 and 3: Gathering data, running the reducer

```
// called once per line in file
void mapper(string line, map<string,string> results) {
   string useragent = parse_requester_useragent(line);
   if (is_mobile_client(useragent))
     results.add(useragent, 1);
}

// called once per unique key in results
void reducer(string key, list<string> values, int& result) {
    int sum = 0;
    for (v in values)
       sum += v;
    result = sum;
}

LineByLineReader input(“hdfs://15418log.txt”);
Writer output(“hdfs://…”);
runMapReduceJob(mapper, reducer, input, output);
```

Step 2: Prepare intermediate data for reducer.

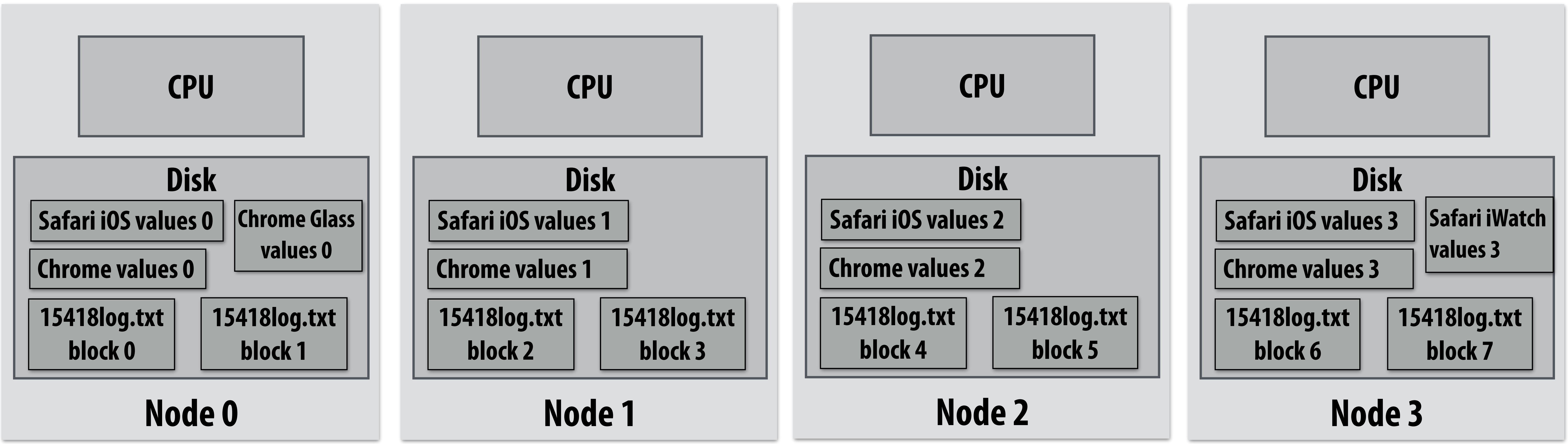
Step 3: Run reducer function on all keys.

Question: how to assign reducer tasks?

Question: how to get all data for key onto the correct worker node?

Keys to reduce:
(generated by mapper):

Safari iOS
Chrome
Safari iWatch
Chrome Glass



Steps 2 and 3: Gathering data, running the reducer

```
// gather all input data for key, then execute reducer
// to produce final result
void runReducer(string key, reducer, result) {
   list<string> inputs;
   for (n in nodes) {
        filename = get_filename(key, n);
        read lines of filename, append into inputs;
   }
   reducer(key, inputs, result);
}
```

Step 2: Prepare intermediate data for reducer.

Step 3: Run reducer function on all keys.

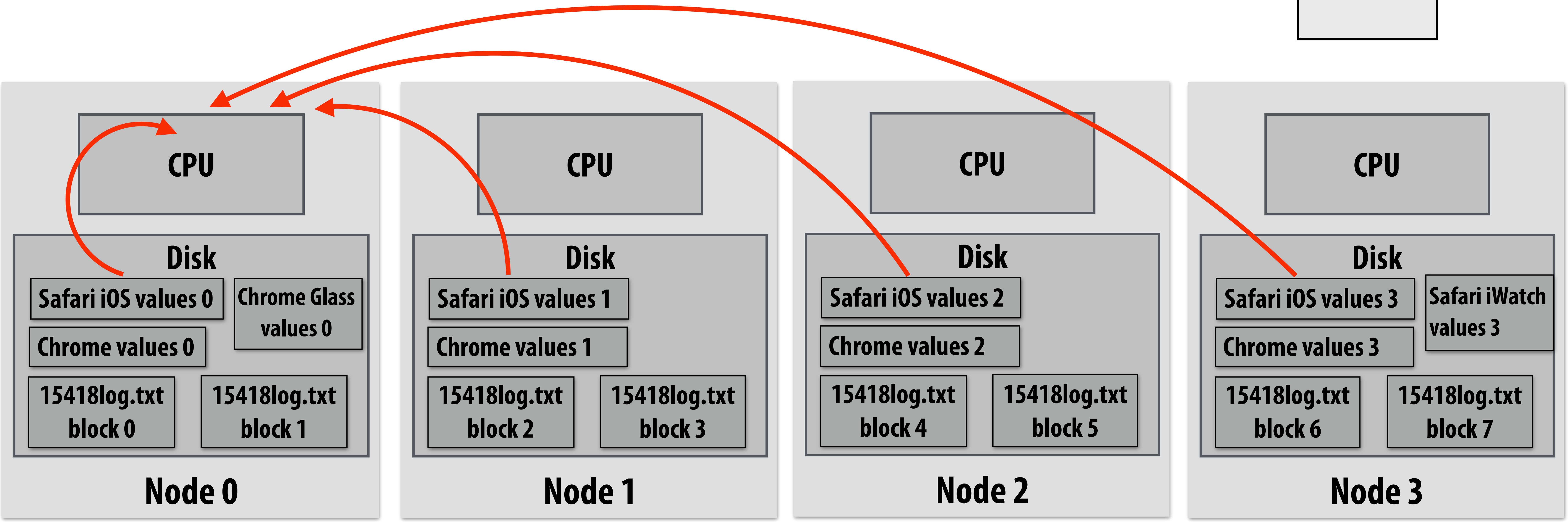
Question: how to assign reducer tasks?

Question: how to get all data for key onto the correct worker node?

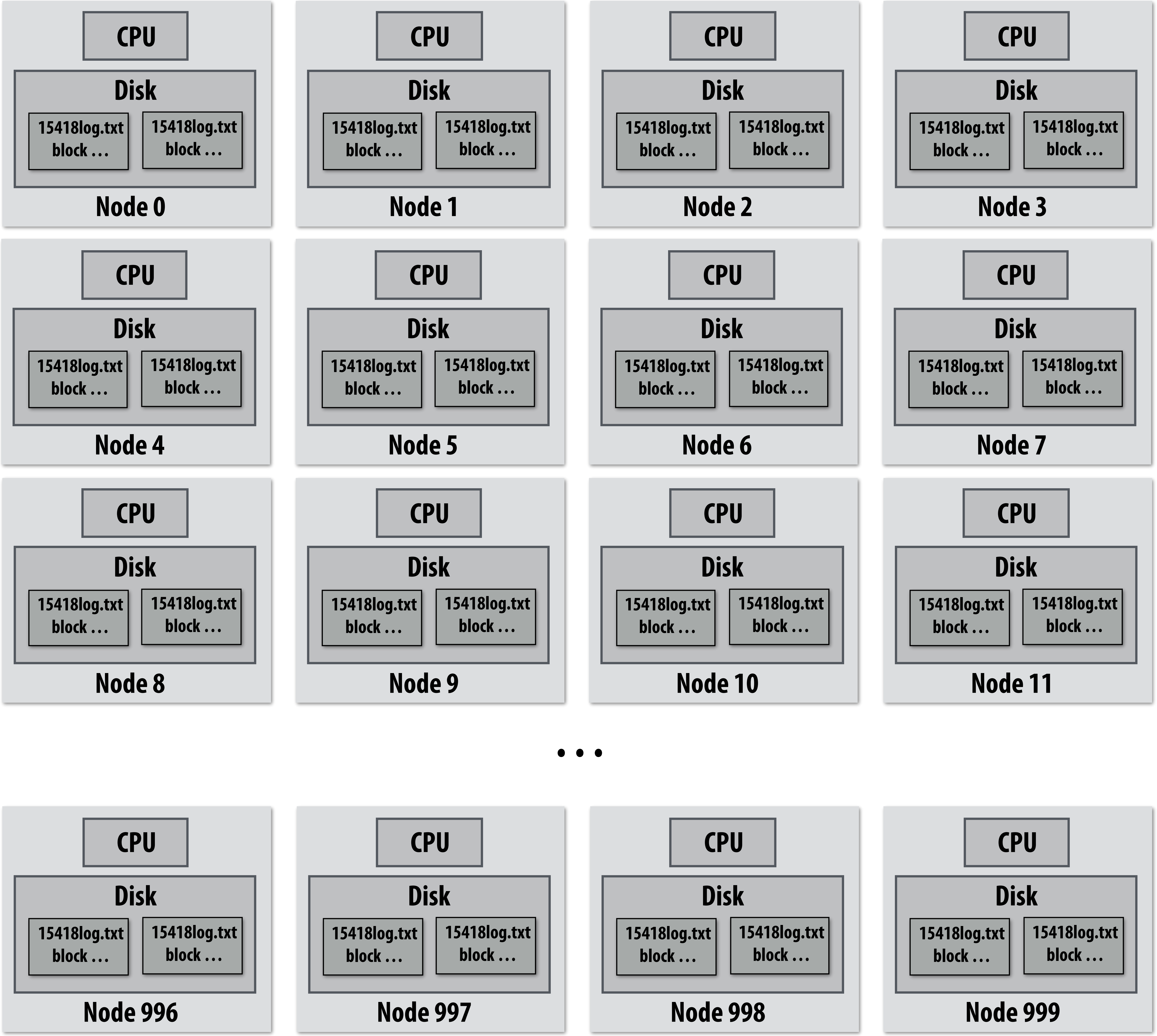
Keys to reduce:
(generated by mapper):

Safari iOS
Chrome
Safari iWatch
Chrome Glass

Example:
Assign Safari iOS to Node 0



Additional implementation challenges at scale



Node failures during program execution

Slow running nodes

Job scheduler responsibilities

- **Exploit data locality: “move computation to the data”**
  - Run mapper jobs on nodes that contain input files
  - Run reducer jobs on nodes that already have most of data for a certain key

- **Handling node failures:**
  - Scheduler detects job failures and reruns job on new machines
    - Possible since inputs reside in persistent storage (distributed file system)
  - Scheduler duplicates jobs on multiple machines (reduce overall processing latency incurred by node failures)

- **Handling slow machines:**
  - Scheduler duplicates jobs on multiple machines

runMapReduceJob problems?

- **Emits only a very simple program structure**
  - Programs must be structured as: map, followed by reduce by key
  - Generalize structure to DAGs (see DryadLINQ)

- **Iterative algorithms must load from disk each iteration**
  - Recall the lecture on graph processing:

```
void pagerank_mapper(graphnode n, map<string,string> results) {
   float val = compute update value for n
   for (dst in outgoing links from n)
     results.add(dst.node, val);
}

void pagerank_reducer(graphnode n, list<float> values, float& result) {
    float sum = 0.0;
    for (v in values)
       sum += v;
    result = sum;
}

for (i = 0 to NUM_ITERATIONS) {
   input = load graph from last iteration
   output = file for this iteration output
   runMapReduceJob(pagerank_mapper, pagerank_reducer, result[i-1], result[i]);
}
```

Spark

in-memory, fault-tolerant distributed computing

Goals

- **Programming model for cluster-scale computations where there is significant reuse of intermediate datasets**
  - Iterative machine learning and graph algorithms
  - Interactive data mining: load large dataset into aggregate memory of cluster and then perform ad-hoc queries

- **Don't want incur inefficiency of writing intermediates to persistent distributed file system (want to keep it in memory)**
  - Challenge: efficiently implementing fault tolerance for large-scale distributed in-memory computations.

Fault tolerance for in-memory calculations

- **Replicate all computations**
  - Expensive solution: decreases peak throughput
- **Checkpoint and rollback**
  - Periodically save state of program to persistent storage
  - Restart from last checkpoint on node failure
- **Maintain log of updates (commands and data)**
  - High overhead for maintaining logs

Recall Map-reduce solutions:

- Checkpoints after each map/reduce step by writing results to file system
- Scheduler's list of outstanding (but not yet complete) jobs is a log
- Functional structure of programs allows for restart at granularity of a single mapper or reducer invocation (don't have to restart entire program)

Resilient distributed dataset (RDD)

Spark's key programming abstraction:

- Read-only collection of records (immutable)
- RDDs can only be created by deterministic ***transformations*** on data in persistent storage or on existing RDDs
- ***Actions*** on RDDs return data to application

```
// create RDD from file system data
var lines = spark.textFile("hdfs://15418log.txt");

// create RDD using filter() transformation on lines
var mobileViews = lines.filter((x: String) => isMobileClient(x));

// instruct Spark runtime to try to keep mobileViews in memory
mobileViews.persist();

// create a new RDD by filtering mobileViews
// then count number of elements in new RDD via count() action
var numViews = mobileViews.filter(_.contains("Safari")).count();

// 1. create new RDD by filtering only Chrome views
// 2. for each element, split string and take first element (forming new RDD)
// 3. convert RDD To a scalar sequence (collect() action)
var ip = mobileView.filter(_.contains("Chrome"))
                   .map(_.split(" ")(0))
                   .collect();
```

Repeating the map-reduce example

```
// create RDD from file system data
var lines = spark.textFile(“hdfs://15418log.txt”);

// 1. create RDD with only lines from mobile clients
// 2. create RDD with elements of type (String,Int) from line string
// 3. group elements by key
// 4. call provided reduction function on all keys to count views
var perAgentCounts = lines.filter(x => isMobileClient(x))
                          .map(x => (parseUserAgent(x),1));
                          .reduceByKey((x,y) => x+y)
                          .collect();
```

RDD Transformations and Actions

Transformations: (data parallel operators taking an input RDD to a new RDD)

$\mathit{map}(f : \mathrm{T} \Rightarrow \mathrm{U})$	$\mathrm{RDD[T]} \Rightarrow \mathrm{RDD[U]}$
$\mathit{filter}(f : \mathrm{T} \Rightarrow \mathrm{Bool})$	$\mathrm{RDD[T]} \Rightarrow \mathrm{RDD[T]}$
$\mathit{flatMap}(f : \mathrm{T} \Rightarrow \mathrm{Seq[U]})$	$\mathrm{RDD[T]} \Rightarrow \mathrm{RDD[U]}$
$\mathit{sample}(\mathit{fraction} : \mathrm{Float})$	$\mathrm{RDD[T]} \Rightarrow \mathrm{RDD[T]}$ (Deterministic sampling)
$\mathit{groupByKey}()$	$\mathrm{RDD[(K, V)]} \Rightarrow \mathrm{RDD[(K, Seq[V])]}$
$\mathit{reduceByKey}(f : (\mathrm{V}, \mathrm{V}) \Rightarrow \mathrm{V})$	$\mathrm{RDD[(K, V)]} \Rightarrow \mathrm{RDD[(K, V)]}$
$\mathit{union}()$	$(\mathrm{RDD[T]}, \mathrm{RDD[T]}) \Rightarrow \mathrm{RDD[T]}$
$\mathit{join}()$	$(\mathrm{RDD[(K, V)]}, \mathrm{RDD[(K, W)]}) \Rightarrow \mathrm{RDD[(K, (V, W))]}$
$\mathit{cogroup}()$	$(\mathrm{RDD[(K, V)]}, \mathrm{RDD[(K, W)]}) \Rightarrow \mathrm{RDD[(K, (Seq[V], Seq[W]))]}$
$\mathit{crossProduct}()$	$(\mathrm{RDD[T]}, \mathrm{RDD[U]}) \Rightarrow \mathrm{RDD[(T, U)]}$
$\mathit{mapValues}(f : \mathrm{V} \Rightarrow \mathrm{W})$	$\mathrm{RDD[(K, V)]} \Rightarrow \mathrm{RDD[(K, W)]}$ (Preserves partitioning)
$\mathit{sort}(c : \mathrm{Comparator[K]})$	$\mathrm{RDD[(K, V)]} \Rightarrow \mathrm{RDD[(K, V)]}$
$\mathit{partitionBy}(p : \mathrm{Partitioner[K]})$	$\mathrm{RDD[(K, V)]} \Rightarrow \mathrm{RDD[(K, V)]}$

Actions: (provide data back to driver application)

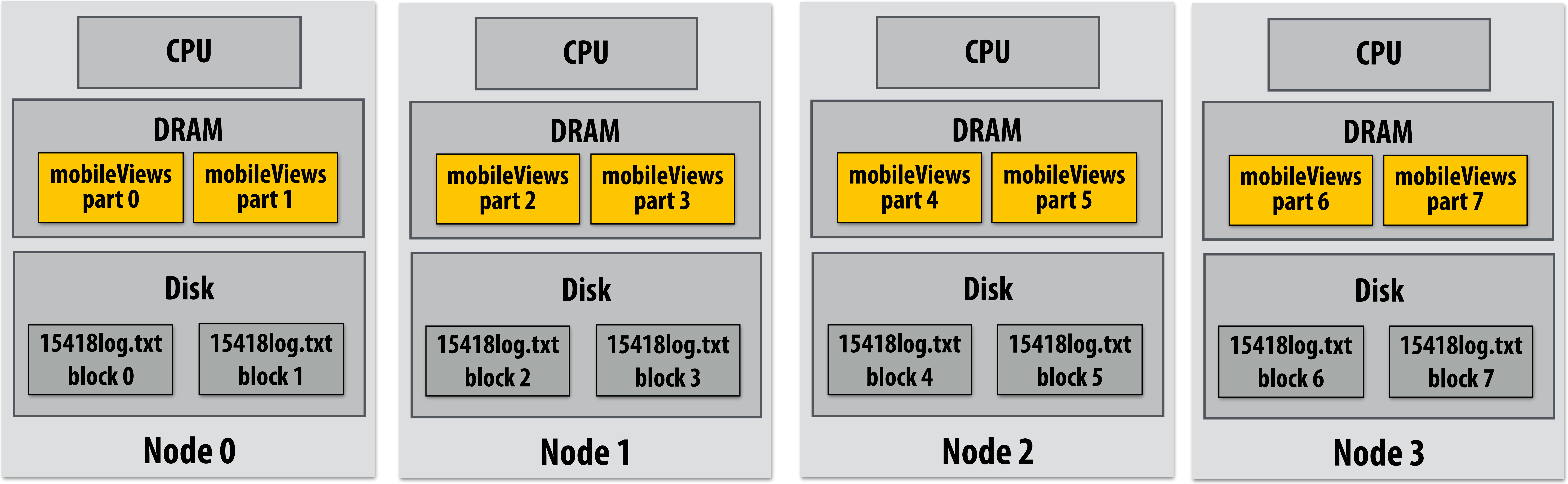
$\mathit{count}()$	$\mathrm{RDD[T]} \Rightarrow \mathrm{Long}$
$\mathit{collect}()$	$\mathrm{RDD[T]} \Rightarrow \mathrm{Seq[T]}$
$\mathit{reduce}(f : (\mathrm{T}, \mathrm{T}) \Rightarrow \mathrm{T})$	$\mathrm{RDD[T]} \Rightarrow \mathrm{T}$
$\mathit{lookup}(k : \mathrm{K})$	$\mathrm{RDD[(K, V)]} \Rightarrow \mathrm{Seq[V]}$ (On hash/range partitioned RDDs)
$\mathit{save}(\mathit{path} : \mathrm{String})$	Outputs RDD to a storage system, *e.g.,* HDFS

RDDs are distributed objects

- Implementation of RDDs…
  - May distribute contents of an RDD across nodes
  - May materialize RDD's contents in memory (or disk)

```
// create RDD from file system data
var lines = spark.textFile("hdfs://15418log.txt");

// create RDD using filter() transformation on lines
var mobileViews = lines.filter(x => isMobileClient(x));
```

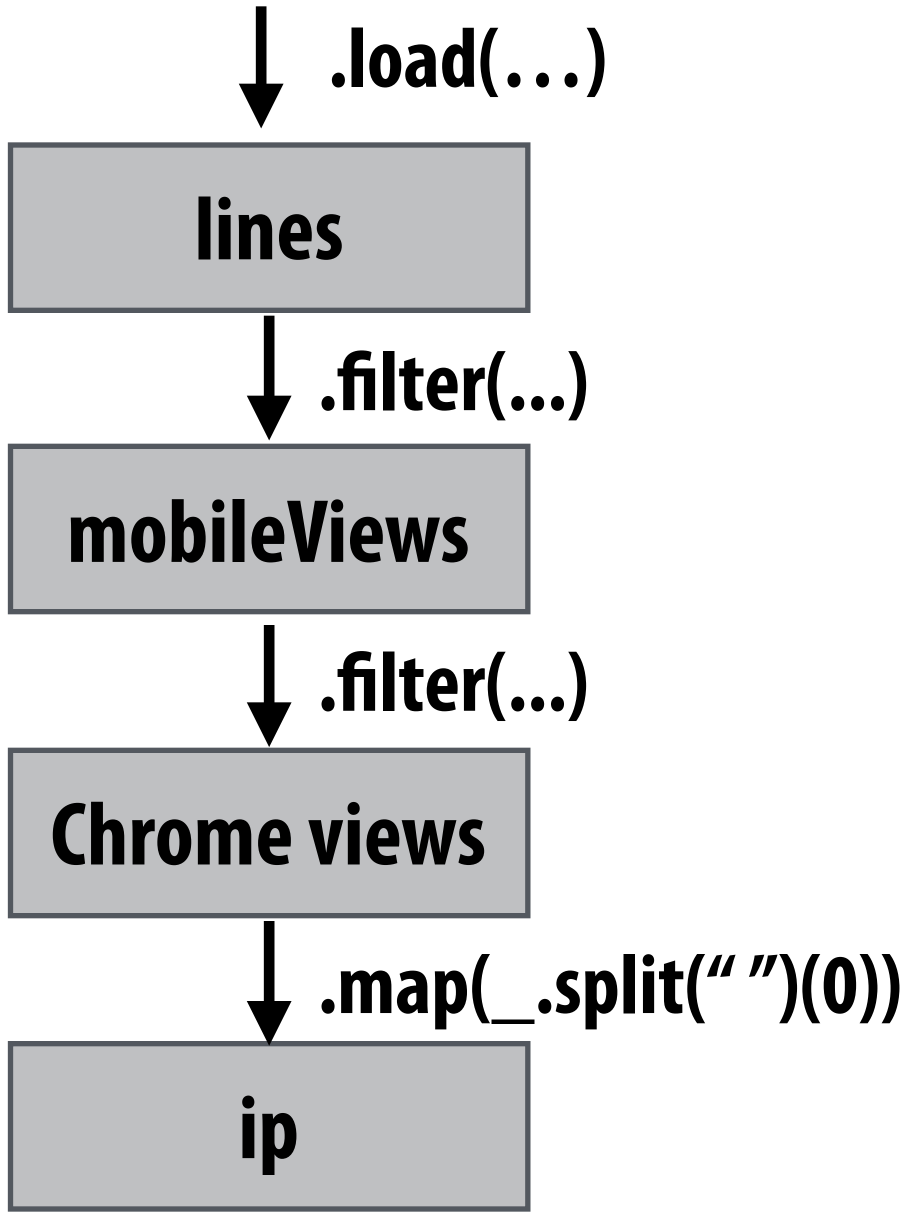
This example:
loading RDD from storage yields one partition per filesystem block. RDD created by filter() takes on same partitions as source.



Implementing resilience via lineage

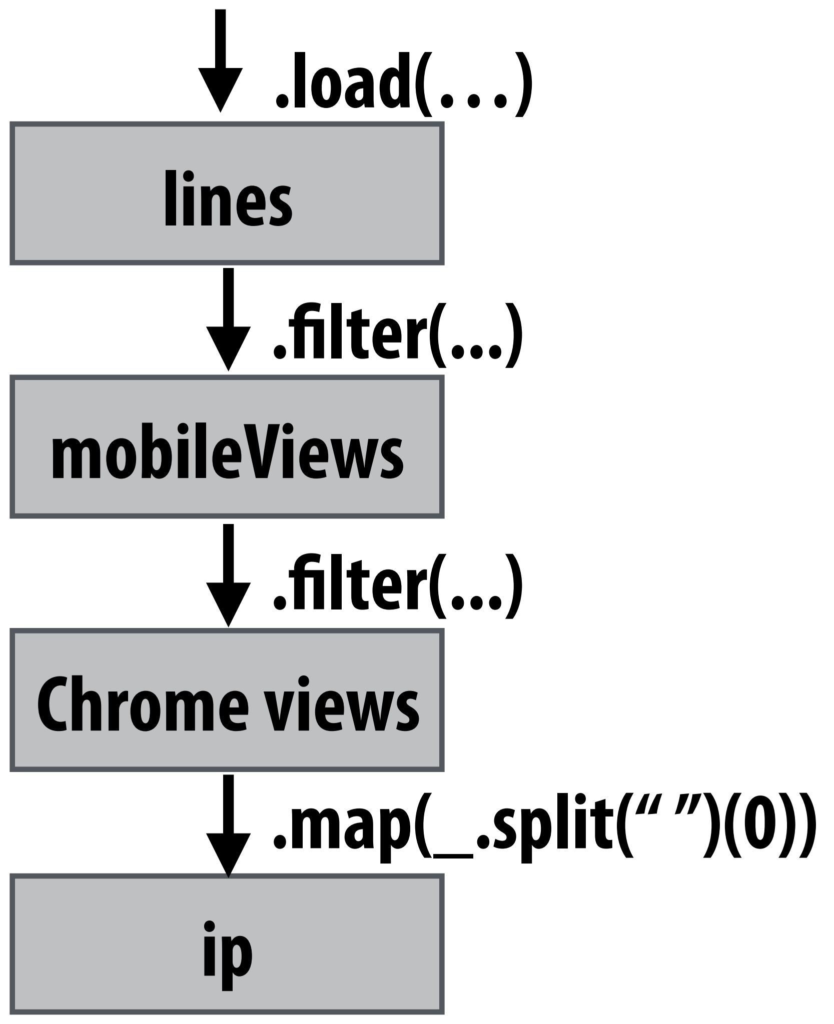
- RDD transformations are, bulk, deterministic, and functional
  - Implication: runtime can always reconstruct contents of RDD from its lineage (the sequence of transformations used to create it)
  - Lineage is a log of transformations
  - Efficient: since log records bulk data-parallel operations, overhead of logging is low (compared to logging fine-grained operations, like in a database)

```
// create RDD from file system data
var lines = spark.textFile(“hdfs://15418log.txt”);

// create RDD using filter() transformation on lines
var mobileViews = lines.filter((x: String) => isMobileClient(x));

// 1. create new RDD by filtering only Chrome views
// 2. for each element, split string and take ip (first element)
// 3. convert RDD To a scalar sequence (collect() action)
var ip = mobileView.filter(_.contains(“Chrome”))
                   .map(_.split(“ ”)(0));
```

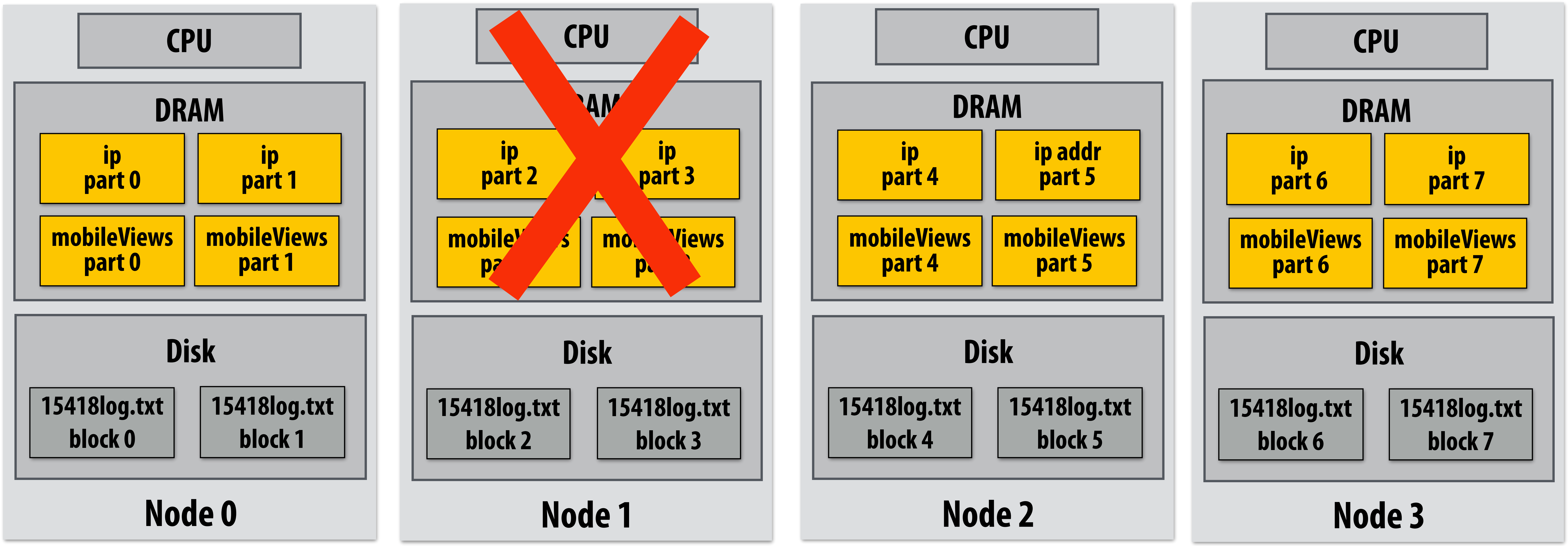


Upon failure: recompute RDD partitions from lineage

```
var lines = spark.textFile(“hdfs://15418log.txt”);
var mobileViews = lines.filter((x: String) => isMobileClient(x));
var ip = mobileView.filter(_.contains(“Chrome”))
                   .map(_.split(“ ”)(0));
```



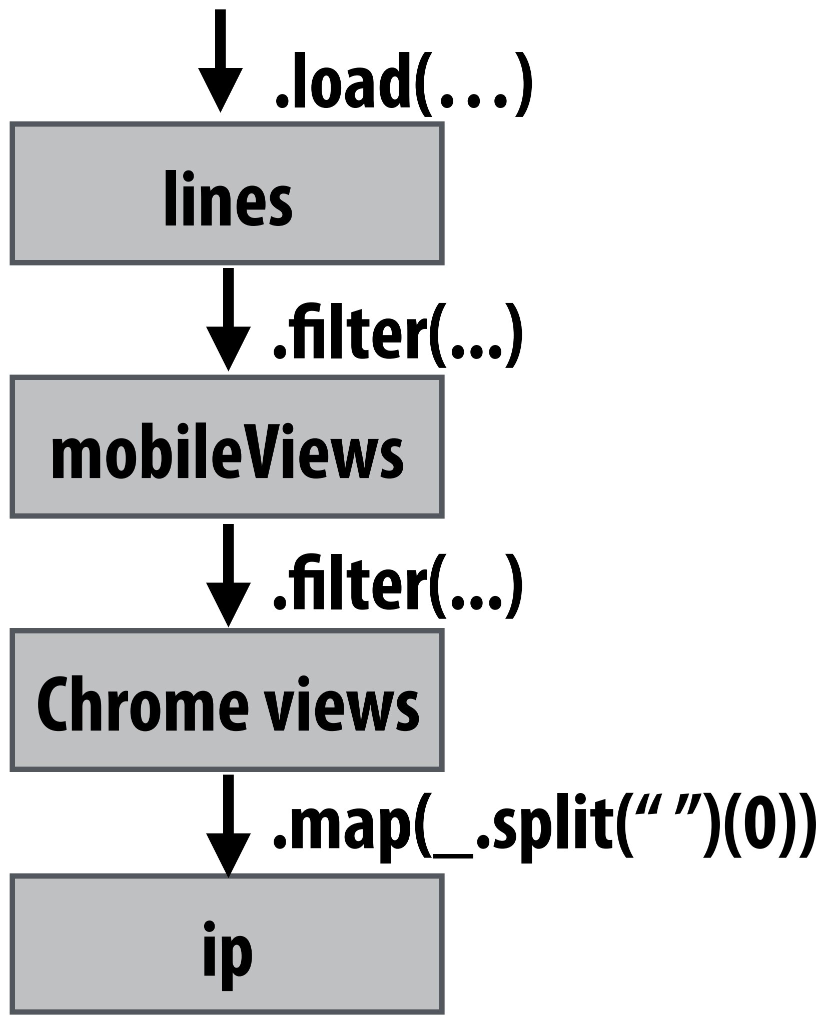
Must reload required subset of data from disk and recompute entire sequence of operations given by lineage to regenerate partitions 2 and 3 of RDD ip.



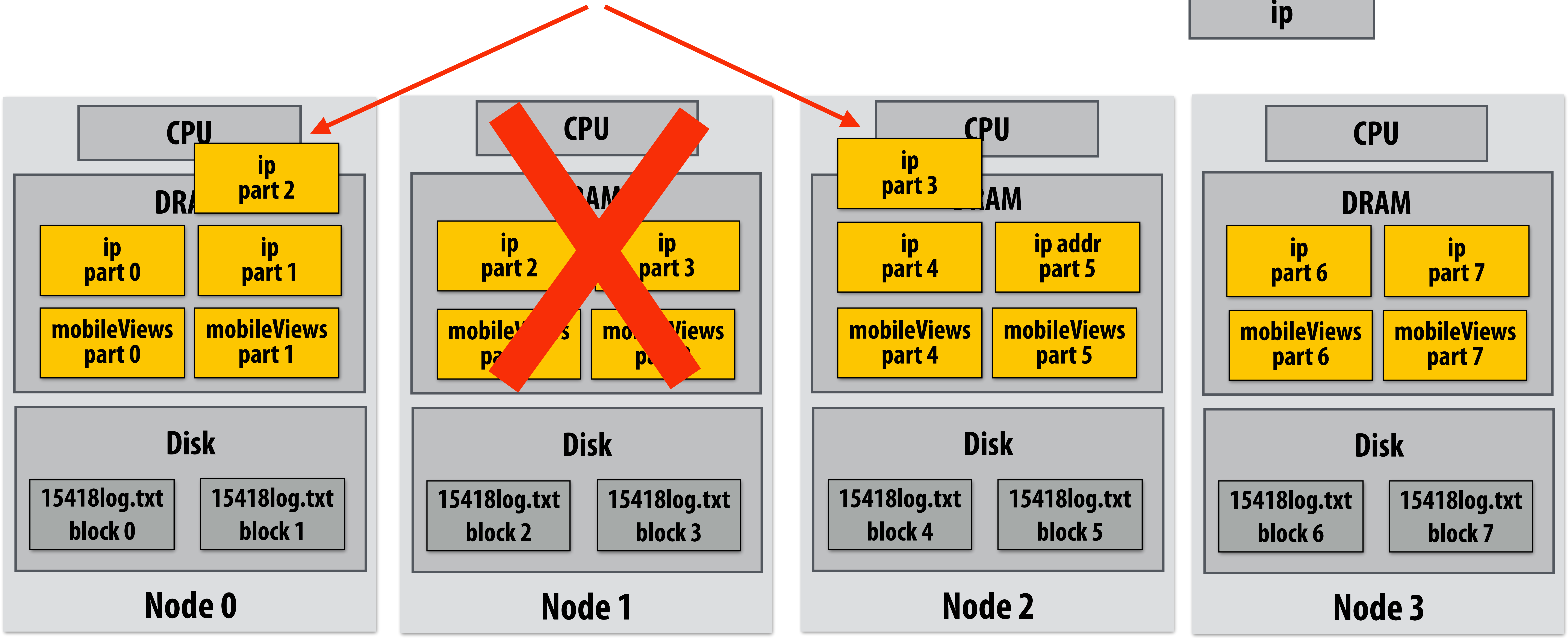
Note: (not shown): file system data is replicated so assume blocks 2 and 3 remain accessible to all nodes

Upon failure: recompute RDD partitions from lineage

```
var lines = spark.textFile("hdfs://15418log.txt");
var mobileViews = lines.filter((x: String) => isMobileClient(x));
var ip = mobileView.filter(_.contains("Chrome"))
                   .map(_.split(" ")(0));
```



Must reload required subset of data from disk and recompute entire sequence of operations given by lineage to regenerate partitions 2 and 3 of RDD ip.



Note: (not shown): file system data is replicated so assume blocks 2 and 3 remain accessible to all nodes

RDD implementation

- **Internal interface for RDD objects**

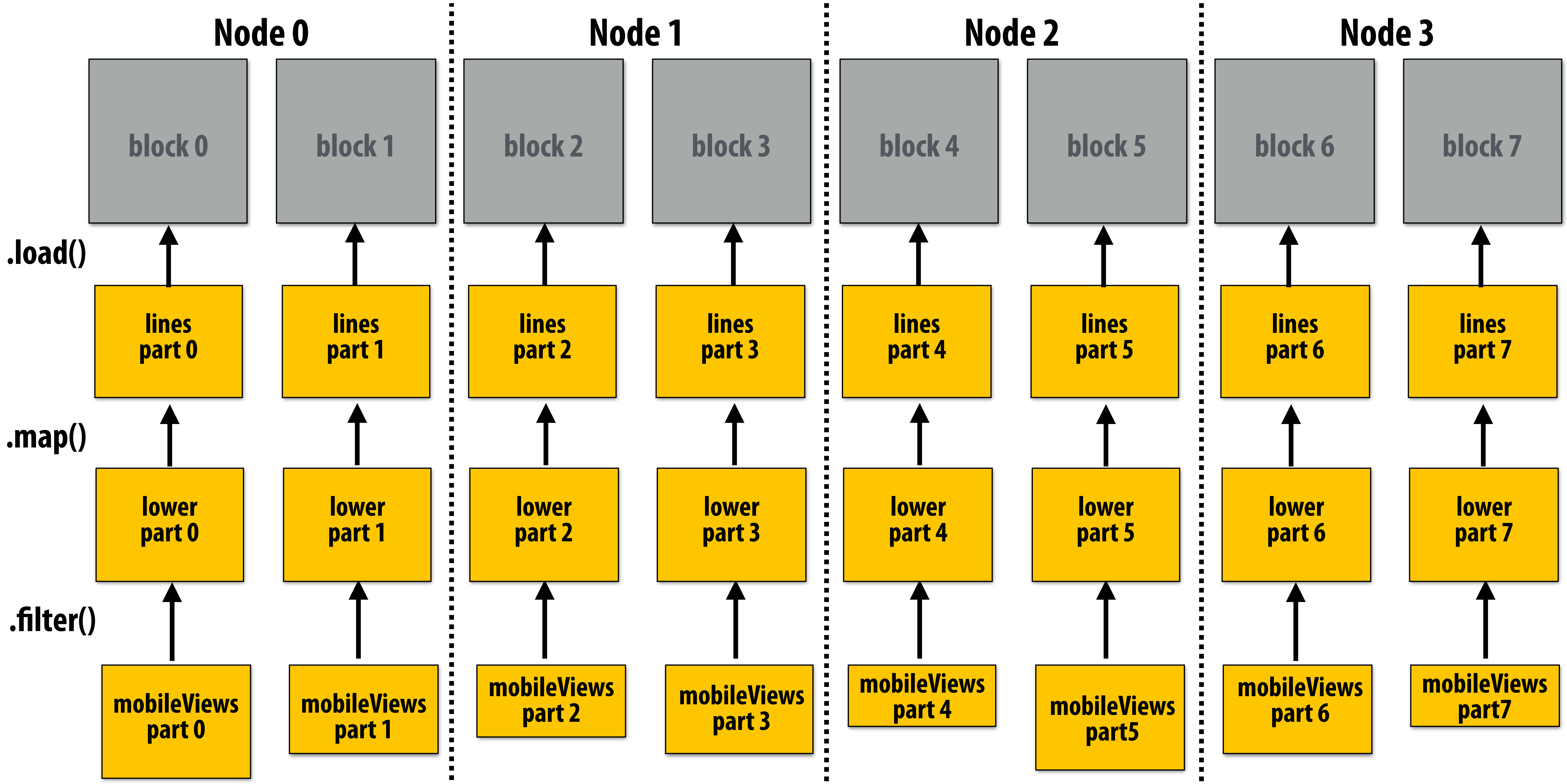
`partitions()`	**Return a list of partitions in the RDD**
`preferredLocations(p)`	**Given partition *p*, return nodes that can access *p* efficiently due to locality (e.g., node that held associated block on disk, node that computed *p*)**
`dependencies()`	**List of parent RDDs**
`iterator(p, parentIters)`	**Iterator for all elements in a partition p. Requires iterators to parent RDD iterators in order to source input data**
`partitioner()`	**Return information about partitioning function being used. (e.g., range partitioner, hash partitioner)**

Example: implementing iterator.next() for map(func) RDD transformation:

```
void next() {
    return func(parentIter().next())
}
```

Partitioning and dependencies

```
var lines = spark.textFile(“hdfs://15418log.txt”);
var lower = lines.map(_.toLower());
var mobileViews = lower.filter(x => isMobileClient(x));
```



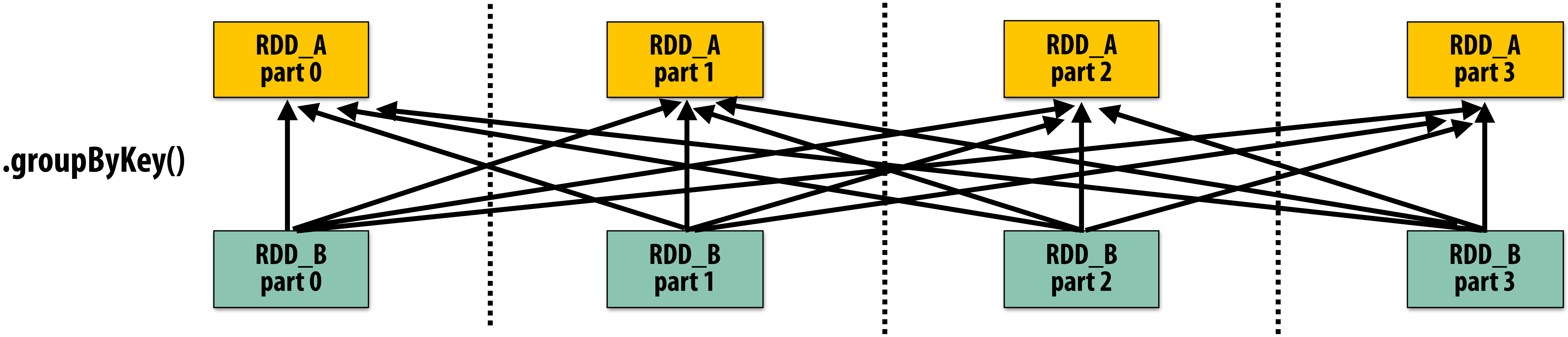
Black lines show dependencies between RDD positions.

“Narrow dependencies” = each partition of parent RDD referenced by at most one child RDD partition

Advantages: allows for fusing of operations (here: can apply map and then filter all at once on input element)

Partitioning and dependencies

groupByKey: RDD[(K,V)], RDD[(K,W)] → RDD[(K,Seq[V])]

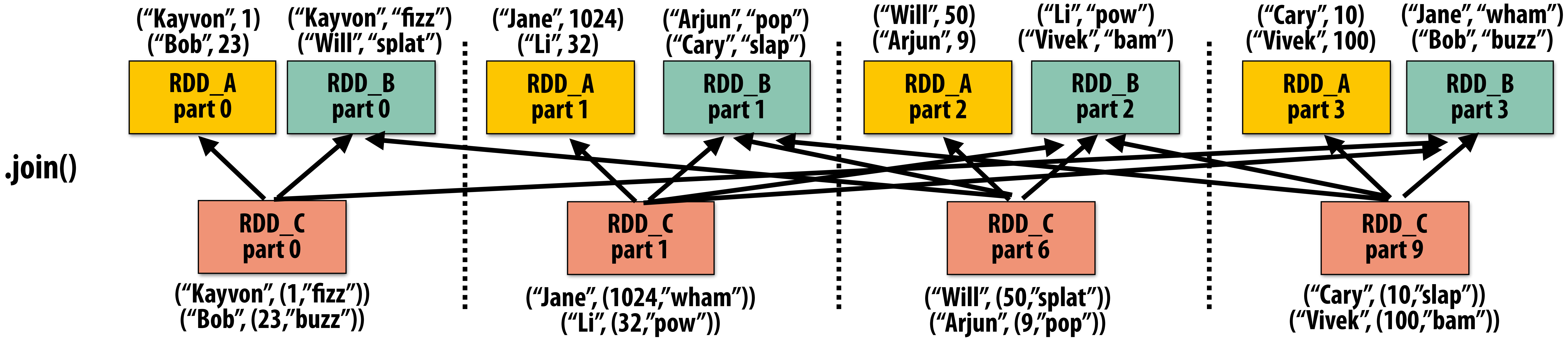


- Wide dependencies = each partition of parent RDD referenced by multiple child RDD partitions
- Challenges:
  - Must compute all of RDD_A before computing RDD_B (example: groupByKey may induce all-to-all communication)
  - May trigger significant recompilation of ancestor lineage upon node failure

Partitioning and dependencies

join: RDD[(K,V)], RDD[(K,W)] → RDD[(K,(V,W))]

RDD_A and RDD_B have different hash partitions: join creates wide dependencies



.join()

("Kayvon", 1)
("Bob", 23)
RDD_A part 0

("Kayvon", "fizz")
("Will", "splat")
RDD_B part 0

("Jane", 1024)
("Li", 32)
RDD_A part 1

("Arjun", "pop")
("Cary", "slap")
RDD_B part 1

("Will", 50)
("Arjun", 9)
RDD_A part 2

("Li", "pow")
("Vivek", "bam")
RDD_B part 2

("Cary", 10)
("Vivek", 100)
RDD_A part 3

("Jane", "wham")
("Bob", "buzz")
RDD_B part 3

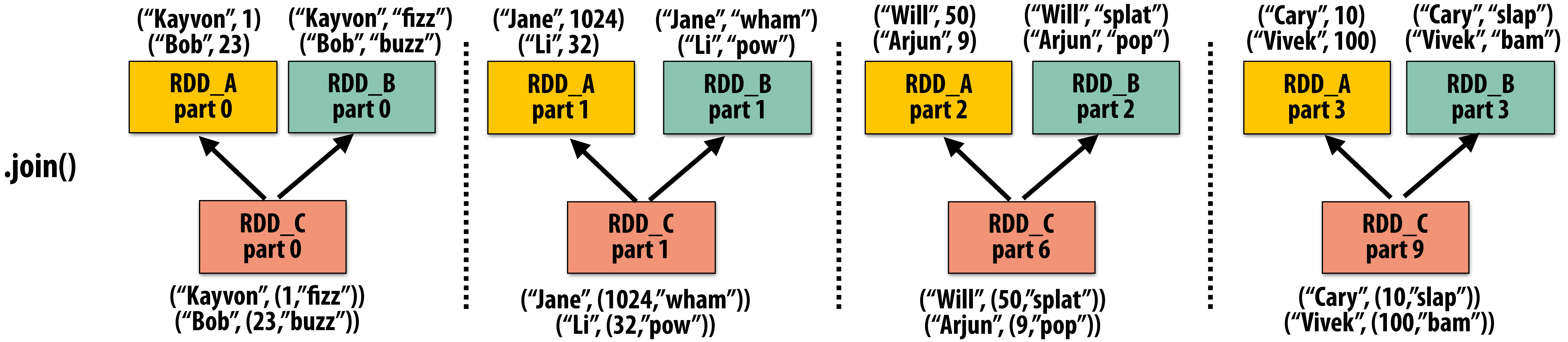
RDD_C part 0
("Kayvon", (1,"fizz"))
("Bob", (23,"buzz"))

RDD_C part 1
("Jane", (1024,"wham"))
("Li", (32,"pow"))

RDD_C part 6
("Will", (50,"splat"))
("Arjun", (9,"pop"))

RDD_C part 9
("Cary", (10,"slap"))
("Vivek", (100,"bam"))

RDD_A and RDD_B have same hash partition: join only create narrow dependencies



.join()

("Kayvon", 1)
("Bob", 23)
RDD_A part 0

("Kayvon", "fizz")
("Bob", "buzz")
RDD_B part 0

("Jane", 1024)
("Li", 32)
RDD_A part 1

("Jane", "wham")
("Li", "pow")
RDD_B part 1

("Will", 50)
("Arjun", 9)
RDD_A part 2

("Will", "splat")
("Arjun", "pop")
RDD_B part 2

("Cary", 10)
("Vivek", 100)
RDD_A part 3

("Cary", "slap")
("Vivek", "bam")
RDD_B part 3

RDD_C part 0
("Kayvon", (1,"fizz"))
("Bob", (23,"buzz"))

RDD_C part 1
("Jane", (1024,"wham"))
("Li", (32,"pow"))

RDD_C part 6
("Will", (50,"splat"))
("Arjun", (9,"pop"))

RDD_C part 9
("Cary", (10,"slap"))
("Vivek", (100,"bam"))

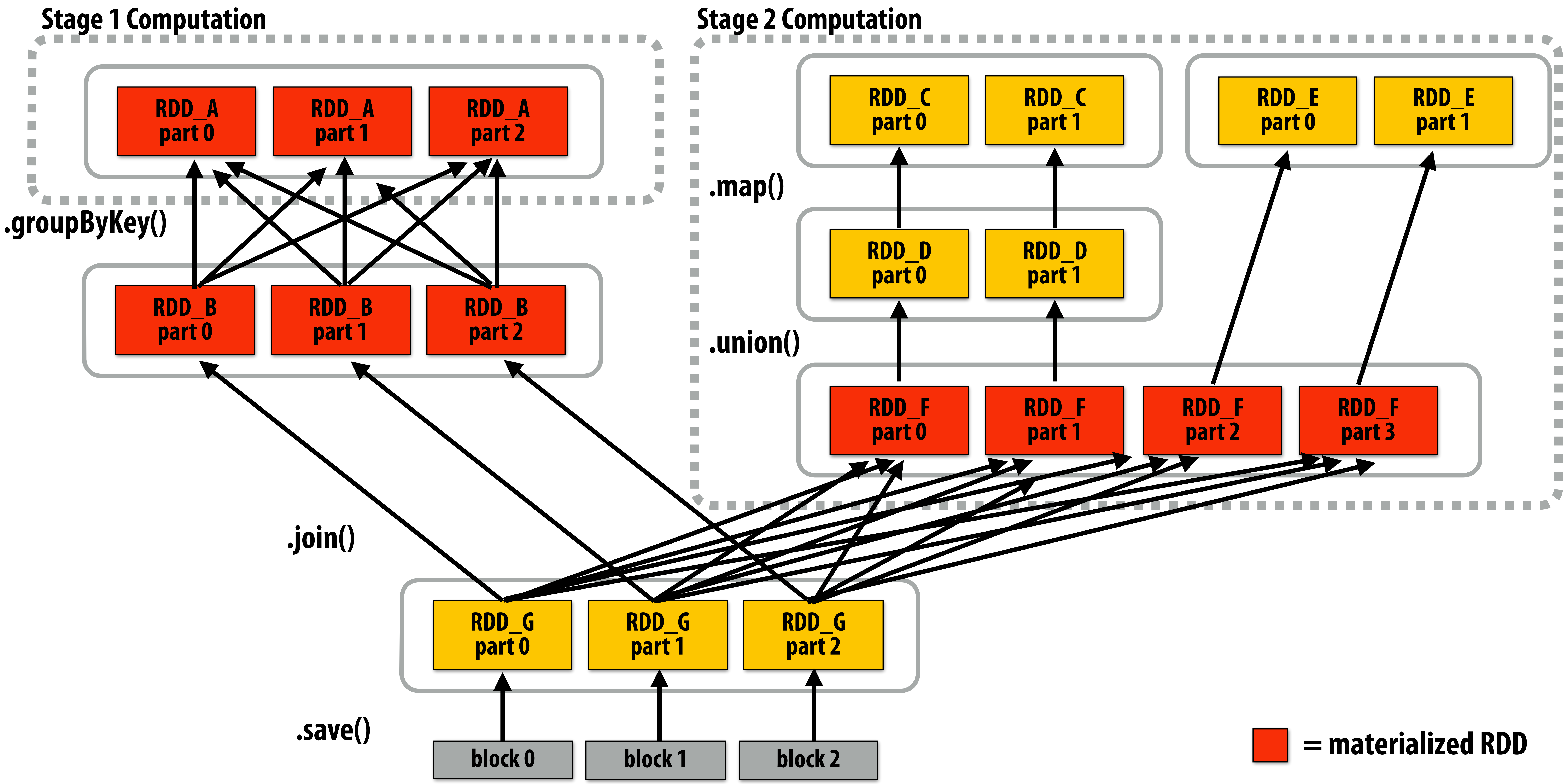
PartitionBy() transformation

- Inform Spark on how to partition an RDD
  - e.g., HashPartitioner, RangePartitioner

```
// create RDD from file system data
var lines = spark.textFile(“hdfs://15418log.txt”);
var clientInfo = spark.textFile(“hdfs://clientssupported.txt”); // (useragent, “yes”/“no”)

// create RDD using filter() transformation on lines
var mobileViews = lines.filter(x => isMobileClient(x)).map(x => parseUserAgent(x));

// HashPartitioner maps keys to integers
var partitioner = spark.HashPartitioner(100);

var mobileViewPartitioned = mobileViews.partitionBy(partitioner)
                                       .persist();
var clientInfoPartitioned = clientInfo.partitionBy(partitioner)
                                      .persist();

// join agents with whether they are supported or not supported
// Note: this join only creates narrow dependencies
void joined = mobileViewPartitioned.join(clientInfoPartitioned);
```

- .persist():
  - Inform Spark this RDD materialized contents should be retained in memory
  - .persist(RELIABLE) = store contents in durable storage (like a checkpoint)

Scheduling Spark computations



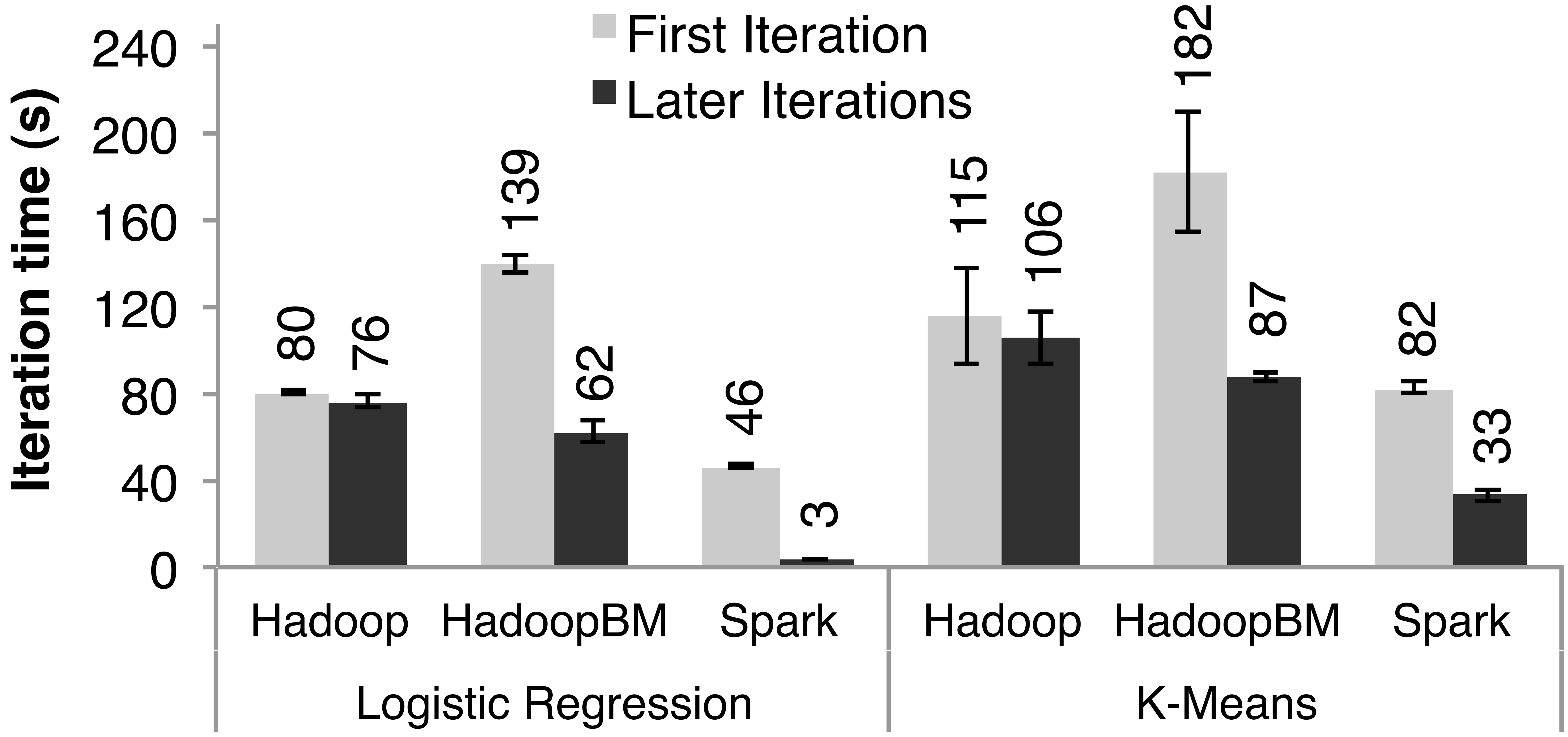
Actions (e.g., save()) trigger evaluation of Spark lineage graph.

Stage 1 Computation: do nothing since input already materialized in memory

Stage 2 Computation: evaluate map, only actually materialize RDD F

Stage 3 Computation: execute join (could stream the operation to disk, do not need to materialize)

Performance



HadoopBM = Hadoop Binary In-Memory (convert text input to binary, store in in-memory version of HDFS)

Q. Wait, the baseline parses text input in each iteration of an iterative algorithm? A. Yes.

Anything else puzzling here?

HadoopBM's first iteration is slow because it runs an extra Hadoop job to copy binary form of input data to in memory HDFS

Accessing data from HDFS, even if in memory has high overhead:

- **Multiple mem copies in file system + a checksum**
- **Conversion from serialized form to Java object**

Spark summary

- Introduces opaque collection abstraction (RDD) to encapsulate intermediates of cluster computations (previously… frameworks stored intermediates in the file system)
    - Observation: "files are a poor abstraction for intermediate variables in large scale data-parallel programs"
    - RDDs are read-only, and created by deterministic data-parallel operators
    - Lineage tracked and used for locality-aware scheduling and fault-tolerance (allows recomputation of partitions of RDD on failure, rather than restore from checkpoint *)
        - Bulk operations allow overhead of lineage tracking (logging) to be low.

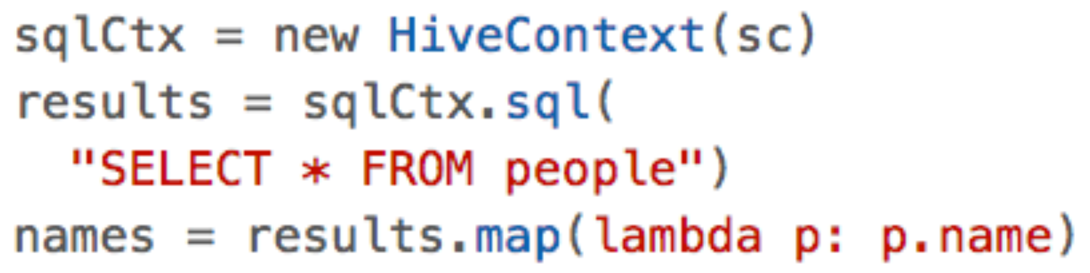
- Simple, versatile abstraction upon which many domain-specific distributed computing frameworks are being implemented.
    - See Apache Spark project: spark.apache.org

* Note that .persist(RELIABLE) allows programmer to request checkpointing in long lineage situations.

Modern Spark ecosystem

Compelling feature: enables integration/composition of multiple domain-specific frameworks (since all collections implemented under the hood with RDDs and scheduled using Spark scheduler)

Spark SQL

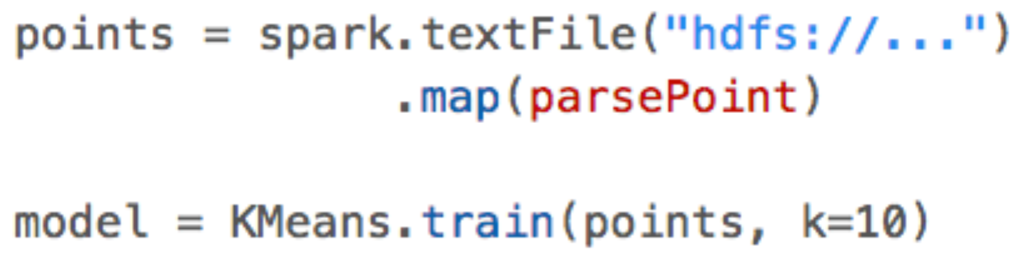


```
sqlCtx = new HiveContext(sc)
results = sqlCtx.sql(
  "SELECT * FROM people")
names = results.map(lambda p: p.name)
```

Interleave computation and database query
Can apply transformations to RDDs produced by SQL queries

Spark MLlib

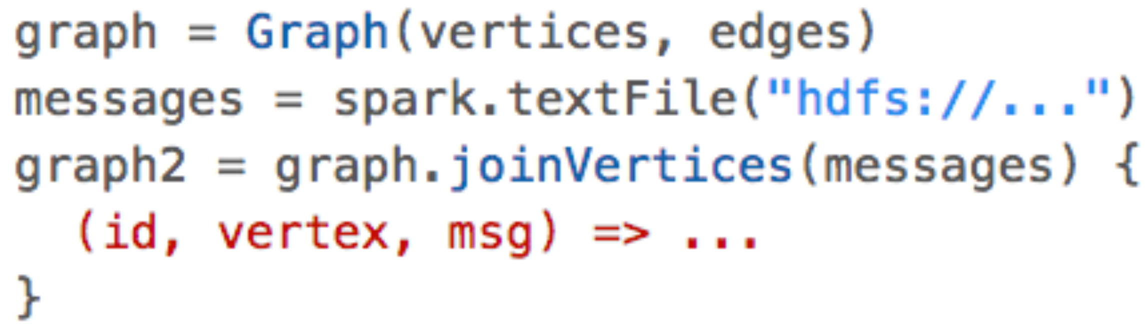
Machine learning library build on top of Spark abstractions.



```
points = spark.textFile("hdfs://...")
              .map(parsePoint)

model = KMeans.train(points, k=10)
```

Spark GraphX

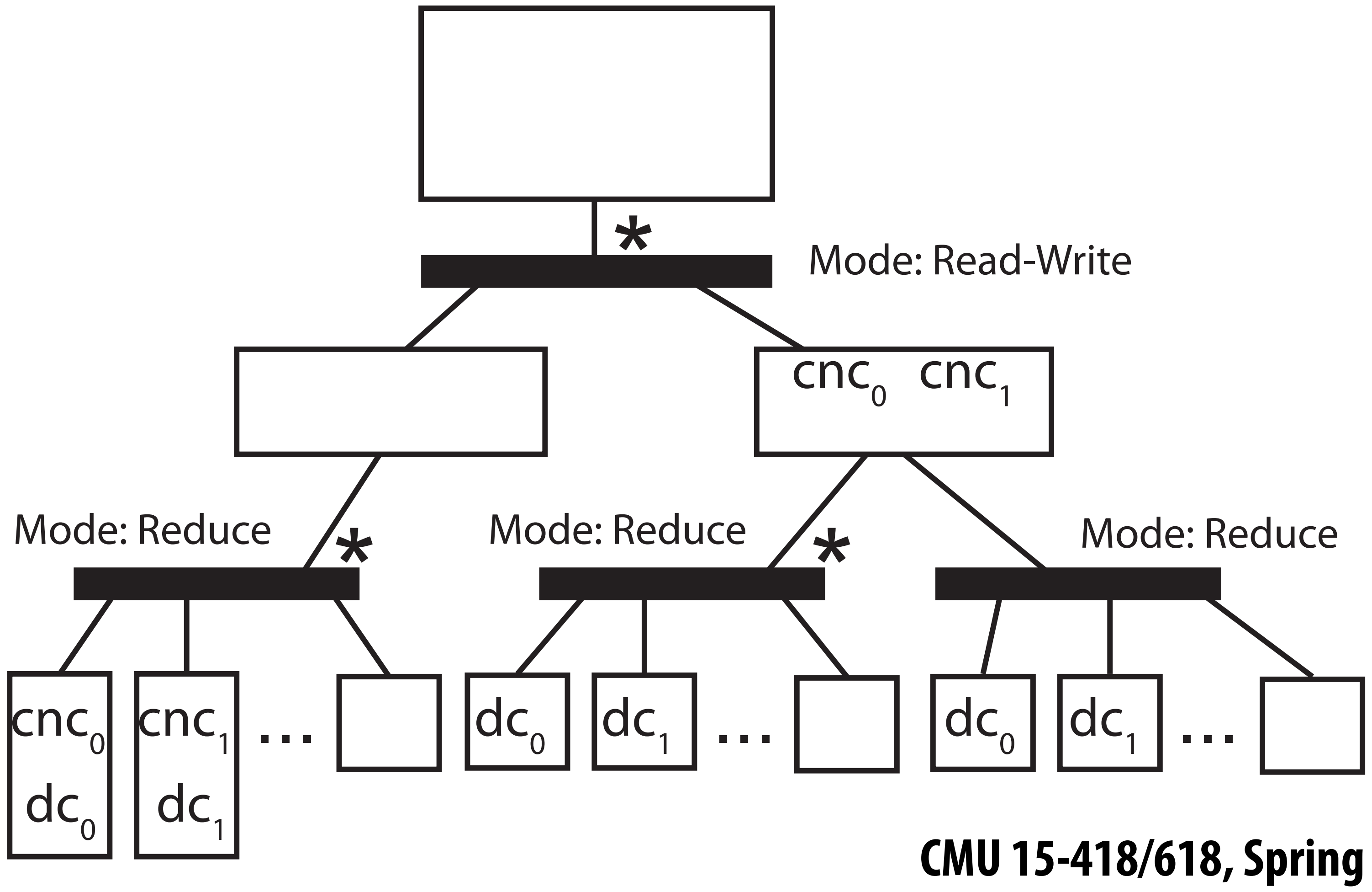
GraphLab-like library built on top of Spark abstractions.



```
graph = Graph(vertices, edges)
messages = spark.textFile("hdfs://...")
graph2 = graph.joinVertices(messages) {
  (id, vertex, msg) => ...
}
```

In you enjoyed today's topic

- I recommend looking at Legion
  - legion.stanford.edu
- Designed from Supercomputing perspective (not distributed computing, like Spark was)
- Key idea: programming via logical regions
  - Operators for hierarchically partitioning contents of regions
  - Tasks operate on regions with certain privileges (read/write/etc.)
  - Scheduler schedules tasks based on privileges to avoid race conditions
- Another of example of bulk-granularity functional programming
  - Overheads are amortized over very large data-parallel operations


*
Mode: Read-Write
cnc0 cnc1
Mode: Reduce
*
Mode: Reduce
*
Mode: Reduce
cnc0 dc0
cnc1 dc1
...
dc0
dc1
...
dc0
dc1
...