Lecture 22:

Spark
(leveraging bulk-granularity program structure)

Parallel Computer Architecture and Programming
CMU 15-418/15-618, Spring 2016

Tunes

Yeah Yeah Yeahs

Sacrilege

(Mosquito)

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

- Karen O

Review: which program performs better?

Program 1

```
void add(int n, float* A, float* B, float* C) {
    for (int i=0; i<n; i++)
       C[i] = A[i] + B[i];
}

void mul(int n, float* A, float* B, float* C) {
    for (int i=0; i<n; i++)
       C[i] = A[i] * B[i];
}


float* A, *B, *C, *D, *E, *tmp1, *tmp2;

// assume arrays are allocated here

// compute E = D + ((A + B) * C)
add(n, A, B, tmp1);
mul(n, tmp1, C, tmp2);
add(n, tmp2, D, E);
```

Two loads, one store per math op (arithmetic intensity = 1/3)

Two loads, one store per math op (arithmetic intensity = 1/3)

Overall arithmetic intensity = 1/3

Program 2

```
void fused(int n, float* A, float* B, float* C, float* D, float* E) {
    for (int i=0; i<n; i++)
       E[i] = D[i] + (A[i] + B[i]) * C[i];
}

// compute E = D + (A + B) * C
fused(n, A, B, C, D, E);
```

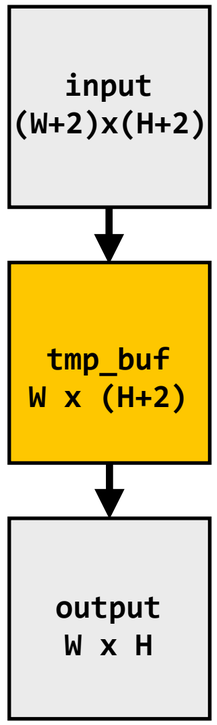
Four loads, one store per 3 math ops (arithmetic intensity = 3/5)

The transformation of the code in program 1 to the code in program 2 is called "loop fusion"

Review: why did we perform this transform?

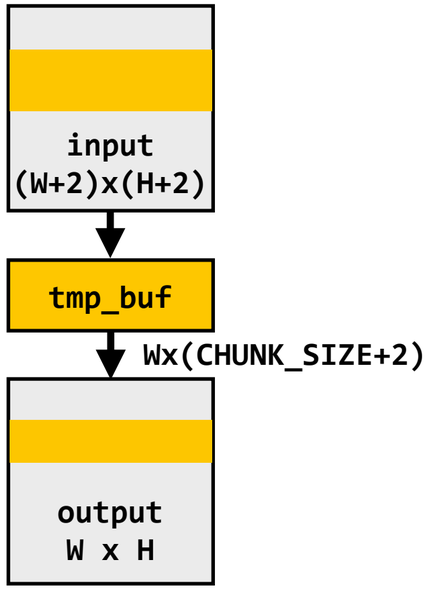
Program 1

```
int WIDTH = 1024;
int HEIGHT = 1024;
float input[(WIDTH+2) * (HEIGHT+2)];
float tmp_buf[WIDTH * (HEIGHT+2)];
float output[WIDTH * HEIGHT];

float weights[] = {1.0/3, 1.0/3, 1.0/3};

// blur image horizontally
for (int j=0; j<(HEIGHT+2); j++)
  for (int i=0; i<WIDTH; i++) {
    float tmp = 0.f;
    for (int ii=0; ii<3; ii++)
      tmp += input[j*(WIDTH+2) + i+ii] * weights[ii];
    tmp_buf[j*WIDTH + i] = tmp;
  }

// blur tmp_buf vertically
for (int j=0; j<HEIGHT; j++) {
  for (int i=0; i<WIDTH; i++) {
    float tmp = 0.f;
    for (int jj=0; jj<3; jj++)
      tmp += tmp_buf[(j+jj)*WIDTH + i] * weights[jj];
    output[j*WIDTH + i] = tmp;
  }
}
```



Program 2

```
int WIDTH = 1024;
int HEIGHT = 1024;
float input[(WIDTH+2) * (HEIGHT+2)];
float tmp_buf[WIDTH * (CHUNK_SIZE+2)];
float output[WIDTH * HEIGHT];

float weights[] = {1.0/3, 1.0/3, 1.0/3};

for (int j=0; j<HEIGHT; j+CHUNK_SIZE) {

  // blur region of image horizontally
  for (int j2=0; j2<CHUNK_SIZE+2; j2++)
    for (int i=0; i<WIDTH; i++) {
      float tmp = 0.f;
      for (int ii=0; ii<3; ii++)
        tmp += input[(j+j2)*(WIDTH+2) + i+ii] * weights[ii];
      tmp_buf[j2*WIDTH + i] = tmp;

  // blur tmp_buf vertically
  for (int j2=0; j2<CHUNK_SIZE; j2++)
    for (int i=0; i<WIDTH; i++) {
      float tmp = 0.f;
      for (int jj=0; jj<3; jj++)
        tmp += tmp_buf[(j2+jj)*WIDTH + i] * weights[jj];
      output[(j+j2)*WIDTH + i] = tmp;
    }
}
```



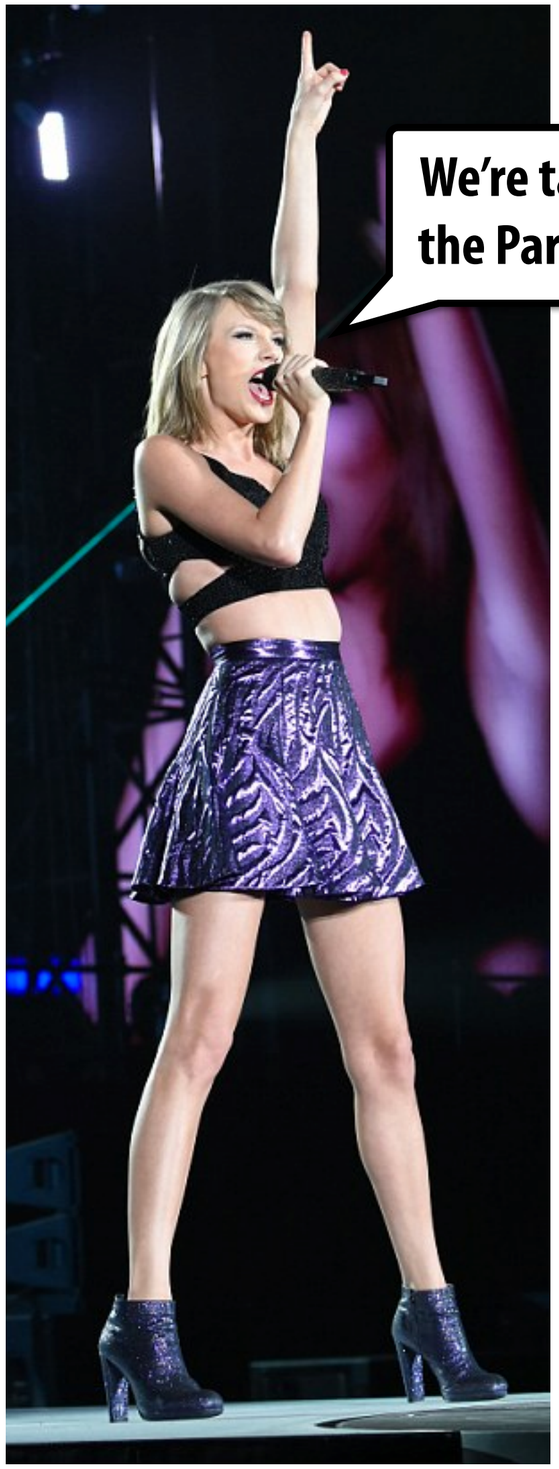
Both of the previous examples involved globally restructuring the order of computation to improve produce-consumer locality

(improve arithmetic intensity of program)

A log of page views on the 418 web site

```
- [05/Apr/2016:22:44:10 -0400] "GET /spring2016content/lectures/16_synchronization/thumbs/slide_012.jpg HTTP/1.1" 200 20186 "http://15418.courses.cs.cmu.edu/spring2016/lecture/synchronization" "Mozilla/5.0 (Macintosh; Intel Ma
- [05/Apr/2016:22:44:10 -0400] "GET /spring2016content/lectures/16_synchronization/thumbs/slide_029.jpg HTTP/1.1" 200 31979 "http://15418.courses.cs.cmu.edu/spring2016/lecture/synchronization" "Mozilla/5.0 (Macintosh; Intel Ma
- [05/Apr/2016:22:44:10 -0400] "GET /spring2016content/lectures/16_synchronization/thumbs/slide_031.jpg HTTP/1.1" 200 8425 "http://15418.courses.cs.cmu.edu/spring2016/lecture/synchronization" "Mozilla/5.0 (Macintosh; Intel Mac
- [05/Apr/2016:22:44:10 -0400] "GET /spring2016content/lectures/16_synchronization/thumbs/slide_035.jpg HTTP/1.1" 200 29266 "http://15418.courses.cs.cmu.edu/spring2016/lecture/synchronization" "Mozilla/5.0 (Macintosh; Intel Ma
- [05/Apr/2016:22:44:10 -0400] "GET /spring2016content/lectures/16_synchronization/thumbs/slide_041.jpg HTTP/1.1" 200 32678 "http://15418.courses.cs.cmu.edu/spring2016/lecture/synchronization" "Mozilla/5.0 (Macintosh; Intel Ma
- [05/Apr/2016:22:44:10 -0400] "GET /spring2016content/lectures/16_synchronization/thumbs/slide_042.jpg HTTP/1.1" 200 32585 "http://15418.courses.cs.cmu.edu/spring2016/lecture/synchronization" "Mozilla/5.0 (Macintosh; Intel Ma
- - [05/Apr/2016:22:44:15 -0400] "GET /spring2016/lecture/snoopimpl/slide_042 HTTP/1.1" 200 3689 "http://15418.courses.cs.cmu.edu/spring2016/lecture/snoopimpl/slide_041" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_3) AppleWeb
- - [05/Apr/2016:22:44:15 -0400] "GET /spring2016content/lectures/12_snoopimpl/images/slide_042.jpg HTTP/1.1" 200 161338 "http://15418.courses.cs.cmu.edu/spring2016/lecture/snoopimpl/slide_042" "Mozilla/5.0 (Macintosh; Intel Ma
- - [05/Apr/2016:22:44:17 -0400] "GET /spring2016/lecture/snoopimpl/slide_041 HTTP/1.1" 200 3093 "http://15418.courses.cs.cmu.edu/spring2016/lecture/snoopimpl/slide_042" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_3) AppleWeb
- [05/Apr/2016:22:44:17 -0400] "GET /spring2016/lecture/synchronization/slide_020 HTTP/1.1" 200 3180 "http://15418.courses.cs.cmu.edu/spring2016/lecture/synchronization" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_4) AppleWe
- - [05/Apr/2016:22:44:18 -0400] "GET /spring2016/keep_alive HTTP/1.1" 200 957 "http://15418.courses.cs.cmu.edu/spring2016/lecture/basicarch/slide_073" "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko
- [05/Apr/2016:22:44:18 -0400] "GET /spring2016content/lectures/16_synchronization/images/slide_020.jpg HTTP/1.1" 200 174283 "http://15418.courses.cs.cmu.edu/spring2016/lecture/synchronization/slide_020" "Mozilla/5.0 (Macintos

- [05/Apr/2016:22:44:18 -0400] "GET /spring2016content/profile_pictures/sidwad.jpg HTTP/1.1" 200 34712 "http://15418.courses.cs.cmu.edu/spring2016/lecture/synchronization/slide_020" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_
- [05/Apr/2016:22:44:18 -0400] "GET /spring2016content/profile_pictures/TomoA.jpg HTTP/1.1" 200 40709 "http://15418.courses.cs.cmu.edu/spring2016/lecture/synchronization/slide_020" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11
- [05/Apr/2016:22:44:18 -0400] "GET /spring2016content/profile_pictures/eknight7.jpg HTTP/1.1" 200 3132 "http://15418.courses.cs.cmu.edu/spring2016/lecture/synchronization/slide_020" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_
- [05/Apr/2016:22:44:18 -0400] "GET /spring2016content/profile_pictures/thomasts.jpg HTTP/1.1" 200 42369 "http://15418.courses.cs.cmu.edu/spring2016/lecture/synchronization/slide_020" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10
- - [05/Apr/2016:22:44:18 -0400] "GET /spring2016/lecture/snoopimpl/slide_040 HTTP/1.1" 200 4985 "http://15418.courses.cs.cmu.edu/spring2016/lecture/snoopimpl/slide_041" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_3) AppleWeb
- - [05/Apr/2016:22:44:19 -0400] "GET /spring2016/lecture/snoopimpl/slide_039 HTTP/1.1" 200 3447 "http://15418.courses.cs.cmu.edu/spring2016/lecture/snoopimpl/slide_040" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_3) AppleWeb
- - [05/Apr/2016:22:44:19 -0400] "GET /spring2016/lecture/snoopimpl/slide_040 HTTP/1.1" 200 4985 "http://15418.courses.cs.cmu.edu/spring2016/lecture/snoopimpl/slide_039" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_3) AppleWeb
- [05/Apr/2016:22:44:21 -0400] "GET /spring2016/users/login HTTP/1.1" 200 2302 "http://15418.courses.cs.cmu.edu/spring2016/lecture/synchronization/slide_020" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_4) AppleWebKit/601.5.1
- [05/Apr/2016:22:44:26 -0400] "POST /spring2016/users/do_login HTTP/1.1" 302 1061 "http://15418.courses.cs.cmu.edu/spring2016/users/login" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_4) AppleWebKit/601.5.17 (KHTML, like Gec
- [05/Apr/2016:22:44:26 -0400] "GET /spring2016/ HTTP/1.1" 200 4767 "http://15418.courses.cs.cmu.edu/spring2016/users/login" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_4) AppleWebKit/601.5.17 (KHTML, like Gecko) Version/9.1
- [05/Apr/2016:22:44:26 -0400] "GET /spring2016content/profile_pictures/cmusam.jpg HTTP/1.1" 200 42983 "http://15418.courses.cs.cmu.edu/spring2016/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_4) AppleWebKit/601.5.17 (KHTML,
- [05/Apr/2016:22:44:30 -0400] "GET /spring2016/lectures HTTP/1.1" 200 6322 "http://15418.courses.cs.cmu.edu/spring2016/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_4) AppleWebKit/601.5.17 (KHTML, like Gecko) Version/9.1 Sa
- [05/Apr/2016:22:44:33 -0400] "GET /spring2016/lecture/synchronization HTTP/1.1" 200 2871 "http://15418.courses.cs.cmu.edu/spring2016/lectures" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_4) AppleWebKit/601.5.17 (KHTML, lik
- [05/Apr/2016:22:44:35 -0400] "GET /spring2013content/lectures/03_progmodels/images/slide_032.png HTTP/1.1" 304 189 "-" "Mozilla/5.0 (compatible; YandexImages/3.0; +http://yandex.com/bots)"
- [05/Apr/2016:22:44:38 -0400] "GET /spring2016/lecture/synchronization/slide_020 HTTP/1.1" 200 3852 "http://15418.courses.cs.cmu.edu/spring2016/lecture/synchronization" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_4) AppleWe
- [05/Apr/2016:22:45:00 -0400] "GET /spring2013/article/26 HTTP/1.1" 200 5900 "http://www.google.co.in/search?ie=UTF-8&q=split+transaction+bus&revid=112973050&sa=X&ved=0ahUKEwio0PfG_vjLAhVinIMKHQO5AdYQ1QIIBQ" "UCWEB/2.0 (Java;
- [05/Apr/2016:22:45:01 -0400] "GET /spring2013/assets/js/15418_common.js HTTP/1.1" 200 425 "http://15418.courses.cs.cmu.edu/spring2013/article/26" "UCWEB/2.0 (Java; U; MIDP-2.0; Nokia203/20.37) U2/1.0.0 UCBrowser/8.7.0.218 U2/
- [05/Apr/2016:22:45:01 -0400] "GET /spring2013/assets/third_party/jquery/timeago/jquery.timeago.js HTTP/1.1" 200 2026 "http://15418.courses.cs.cmu.edu/spring2013/article/26" "UCWEB/2.0 (Java; U; MIDP-2.0; Nokia203/20.37) U2/1.
- [05/Apr/2016:22:45:01 -0400] "GET /spring2013/assets/third_party/jquery/cookie/jquery.cookie.js HTTP/1.1" 200 1189 "http://15418.courses.cs.cmu.edu/spring2013/article/26" "UCWEB/2.0 (Java; U; MIDP-2.0; Nokia203/20.37) U2/1.0.
- [05/Apr/2016:22:45:01 -0400] "GET /spring2013/assets/third_party/date/date.js HTTP/1.1" 200 7628 "http://15418.courses.cs.cmu.edu/spring2013/article/26" "UCWEB/2.0 (Java; U; MIDP-2.0; Nokia203/20.37) U2/1.0.0 UCBrowser/8.7.0.
- [05/Apr/2016:22:45:01 -0400] "GET /spring2013/assets/third_party/codemirror-3.0/mode/markdown/markdown.js HTTP/1.1" 200 4018 "http://15418.courses.cs.cmu.edu/spring2013/article/26" "UCWEB/2.0 (Java; U; MIDP-2.0; Nokia203/20.
- [05/Apr/2016:22:45:01 -0400] "GET /spring2013/assets/third_party/google-code-prettify/prettify.js HTTP/1.1" 200 6379 "http://15418.courses.cs.cmu.edu/spring2013/article/26" "UCWEB/2.0 (Java; U; MIDP-2.0; Nokia203/20.37) U2/1.
- [05/Apr/2016:22:45:01 -0400] "GET /spring2013/assets/third_party/jquery/tmpl/jquery.tmpl.min.js HTTP/1.1" 200 3155 "http://15418.courses.cs.cmu.edu/spring2013/article/26" "UCWEB/2.0 (Java; U; MIDP-2.0; Nokia203/20.37) U2/1.0.
- [05/Apr/2016:22:45:01 -0400] "GET /spring2013/assets/css/main.css HTTP/1.1" 200 3368 "http://15418.courses.cs.cmu.edu/spring2013/article/26" "UCWEB/2.0 (Java; U; MIDP-2.0; Nokia203/20.37) U2/1.0.0 UCBrowser/8.7.0.218 U2/1.0.0
- [05/Apr/2016:22:45:01 -0400] "GET /spring2013/assets/third_party/jquery/1.8.3/jquery.min.js HTTP/1.1" 200 33789 "http://15418.courses.cs.cmu.edu/spring2013/article/26" "UCWEB/2.0 (Java; U; MIDP-2.0; Nokia203/20.37) U2/1.0.0 U
- [05/Apr/2016:22:45:01 -0400] "GET /spring2013/assets/third_party/codemirror-3.0/lib/codemirror.css HTTP/1.1" 200 2319 "http://15418.courses.cs.cmu.edu/spring2013/article/26" "UCWEB/2.0 (Java; U; MIDP-2.0; Nokia203/20.37) U2/1
- [05/Apr/2016:22:45:01 -0400] "GET /spring2013/assets/third_party/google-code-prettify/prettify.css HTTP/1.1" 200 660 "http://15418.courses.cs.cmu.edu/spring2013/article/26" "UCWEB/2.0 (Java; U; MIDP-2.0; Nokia203/20.37) U2/1.
- [05/Apr/2016:22:45:01 -0400] "GET /spring2013/assets/js/main.js HTTP/1.1" 200 1512 "http://15418.courses.cs.cmu.edu/spring2013/article/26" "UCWEB/2.0 (Java; U; MIDP-2.0; Nokia203/20.37) U2/1.0.0 UCBrowser/8.7.0.218 U2/1.0.0 M
- [05/Apr/2016:22:45:01 -0400] "GET /spring2013/assets/third_party/codemirror-3.0/lib/codemirror.js HTTP/1.1" 200 47855 "http://15418.courses.cs.cmu.edu/spring2013/article/26" "UCWEB/2.0 (Java; U; MIDP-2.0; Nokia203/20.37) U2/1
- [05/Apr/2016:22:45:01 -0400] "GET /spring2013/assets/js/comments.js HTTP/1.1" 200 2413 "http://15418.courses.cs.cmu.edu/spring2013/article/26" "UCWEB/2.0 (Java; U; MIDP-2.0; Nokia203/20.37) U2/1.0.0 UCBrowser/8.7.0.218 U2/1.0
- [05/Apr/2016:22:45:01 -0400] "GET /spring2013//assets/images/favicon/dragon.png HTTP/1.1" 200 3145 "-" "UCWEB/2.0 (Java; U; MIDP-2.0; Nokia203/20.37) U2/1.0.0 UCBrowser/8.7.0.218 U2/1.0.0 Mobile"
- [05/Apr/2016:22:45:01 -0400] "GET /spring2013content/article_images/26_3.jpg HTTP/1.1" 200 28441 "http://15418.courses.cs.cmu.edu/spring2013/article/26" "UCWEB/2.0 (Java; U; MIDP-2.0; Nokia203/20.37) U2/1.0.0 UCBrowser/8.7.0.
- [05/Apr/2016:22:45:01 -0400] "GET /spring2013content/article_images/26_2.jpg HTTP/1.1" 200 25683 "http://15418.courses.cs.cmu.edu/spring2013/article/26" "UCWEB/2.0 (Java; U; MIDP-2.0; Nokia203/20.37) U2/1.0.0 UCBrowser/8.7.0.
- [05/Apr/2016:22:45:01 -0400] "GET /spring2013content/article_images/26_4.jpg HTTP/1.1" 200 38414 "http://15418.courses.cs.cmu.edu/spring2013/article/26" "UCWEB/2.0 (Java; U; MIDP-2.0; Nokia203/20.37) U2/1.0.0 UCBrowser/8.7.0.
- [05/Apr/2016:22:45:01 -0400] "GET /spring2013content/profile_pictures/lazyplus.jpg HTTP/1.1" 200 40708 "http://15418.courses.cs.cmu.edu/spring2013/article/26" "UCWEB/2.0 (Java; U; MIDP-2.0; Nokia203/20.37) U2/1.0.0 UCBrowser/
- - [05/Apr/2016:22:45:10 -0400] "GET /spring2016/keep_alive HTTP/1.1" 200 957 "http://15418.courses.cs.cmu.edu/spring2016/article/9" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/49.0.2623
- - [05/Apr/2016:22:46:31 -0400] "GET / HTTP/1.1" 302 564 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/49.0.2623.110 Safari/537.36"
- - [05/Apr/2016:22:46:31 -0400] "GET /spring2016 HTTP/1.1" 301 584 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/49.0.2623.110 Safari/537.36"
- - [05/Apr/2016:22:46:31 -0400] "GET /spring2016/ HTTP/1.1" 200 5254 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/49.0.2623.110 Safari/537.36"
- [05/Apr/2016:22:46:40 -0400] "GET /spring2016/lecture/lockfree/slide_028 HTTP/1.1" 200 4008 "-" "Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)"
- - [05/Apr/2016:22:46:41 -0400] "GET /spring2016/keep_alive HTTP/1.1" 200 968 "http://15418.courses.cs.cmu.edu/spring2016/article/14" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_3) AppleWebKit/537.36 (KHTML, like Gecko) Chro
- [05/Apr/2016:22:46:51 -0400] "GET /spring2016/keep_alive HTTP/1.1" 200 999 "http://15418.courses.cs.cmu.edu/spring2016/article/14" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/49.0.2623.
- [05/Apr/2016:22:47:00 -0400] "GET /spring2014content/lectures/24_memory/images/slide_014.png HTTP/1.1" 304 190 "-" "Mozilla/5.0 (compatible; YandexImages/3.0; +http://yandex.com/bots)"
- [05/Apr/2016:22:47:25 -0400] "GET /spring2015/article/11 HTTP/1.1" 200 6180 "https://www.google.com/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/49.0.2623.110 Safari/537.36"
- [05/Apr/2016:22:47:26 -0400] "GET /spring2015/assets/third_party/jquery/timeago/jquery.timeago.js HTTP/1.1" 200 2026 "http://15418.courses.cs.cmu.edu/spring2015/article/11" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_4) App
- [05/Apr/2016:22:47:26 -0400] "GET /spring2015/assets/third_party/jquery/1.8.3/jquery.min.js HTTP/1.1" 200 33788 "http://15418.courses.cs.cmu.edu/spring2015/article/11" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_4) AppleWeb
- [05/Apr/2016:22:47:26 -0400] "GET /spring2015/assets/third_party/codemirror-3.0/lib/codemirror.css HTTP/1.1" 200 2319 "http://15418.courses.cs.cmu.edu/spring2015/article/11" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_4) Ap
- [05/Apr/2016:22:47:26 -0400] "GET /spring2015/assets/third_party/google-code-prettify/prettify.css HTTP/1.1" 200 660 "http://15418.courses.cs.cmu.edu/spring2015/article/11" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_4) App
- [05/Apr/2016:22:47:26 -0400] "GET /spring2015/assets/css/main.css HTTP/1.1" 200 4011 "http://15418.courses.cs.cmu.edu/spring2015/article/11" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_4) AppleWebKit/537.36 (KHTML, like Gec
- [05/Apr/2016:22:47:26 -0400] "GET /spring2015/assets/third_party/jquery/cookie/jquery.cookie.js HTTP/1.1" 200 1189 "http://15418.courses.cs.cmu.edu/spring2015/article/11" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_4) Apple
- [05/Apr/2016:22:47:26 -0400] "GET /spring2015/assets/third_party/codemirror-3.0/lib/codemirror.js HTTP/1.1" 200 47854 "http://15418.courses.cs.cmu.edu/spring2015/article/11" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_4) Ap
- [05/Apr/2016:22:47:26 -0400] "GET /spring2015/assets/third_party/codemirror-3.0/mode/markdown/markdown.js HTTP/1.1" 200 4017 "http://15418.courses.cs.cmu.edu/spring2015/article/11" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_1
- [05/Apr/2016:22:47:26 -0400] "GET /spring2015/assets/third_party/google-code-prettify/prettify.js HTTP/1.1" 200 6378 "http://15418.courses.cs.cmu.edu/spring2015/article/11" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_4) App
- [05/Apr/2016:22:47:26 -0400] "GET /spring2015/assets/js/main.js HTTP/1.1" 200 806 "http://15418.courses.cs.cmu.edu/spring2015/article/11" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_4) AppleWebKit/537.36 (KHTML, like Gecko)
- [05/Apr/2016:22:47:26 -0400] "GET /spring2015/assets/js/comments.js HTTP/1.1" 200 3779 "http://15418.courses.cs.cmu.edu/spring2015/article/11" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_4) AppleWebKit/537.36 (KHTML, like G
- [05/Apr/2016:22:47:29 -0400] "GET /spring2015/assets/images/favicon/dragon.png HTTP/1.1" 200 3145 "http://15418.courses.cs.cmu.edu/spring2015/article/11" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_4) AppleWebKit/537.36 (KH
- - [05/Apr/2016:22:47:41 -0400] "GET /spring2015content/lectures/21_heterogeneity/thumbs/slide_037.png HTTP/1.1" 304 133 "-" "Mozilla/4.0 (compatible;)"
- [05/Apr/2016:22:48:50 -0400] "GET /spring2016content/article_images/3_.jpg HTTP/1.1" 200 65173 "https://www.google.com/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/49.0.262
- [05/Apr/2016:22:49:09 -0400] "GET /spring2016/keep_alive HTTP/1.1" 200 1012 "http://15418.courses.cs.cmu.edu/spring2016/article/15" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_3) AppleWebKit/537.36 (KHTML, like Gecko) Chro
```

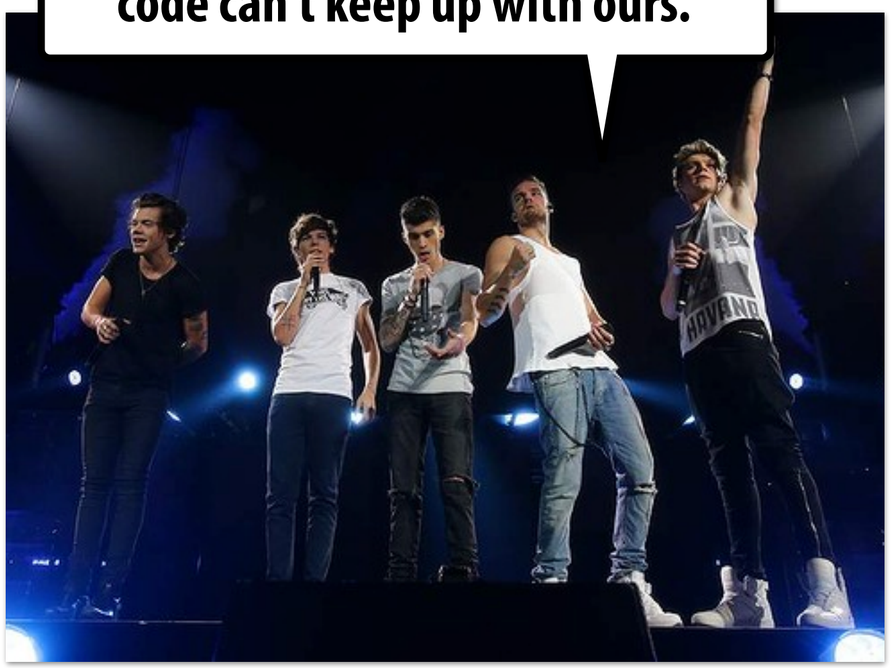
Let's say 418 gets very popular…



We're taking the 1989 Tour to the Parallelism Computation!

You kids might want to learn to handle at least two directions before you take a stab at 16 cores. Count me in.

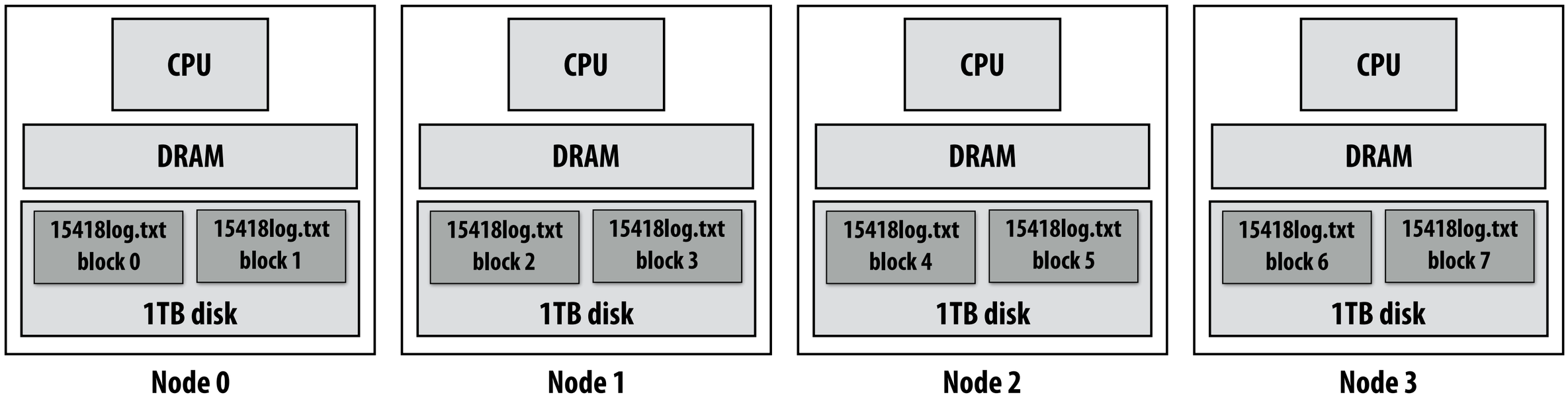
Your name might be Swift, but your code can't keep up with ours.



The log of page views gets quite large…

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

Below: cluster of 4 nodes, each node with a 1 TB disk
Contents of 15418log.txt are distributed evenly in blocks across the cluster



The respective agents of the new Parallelism Competition entrants want to know a bit more about the fans tracking the competition on the 418 site…

For example:
"What type of mobile phone are all my client's fans using?"

A simple programming model

```
// called once per line in input file by runtime
// input: string (line of input file)
// output: adds (user_agent, 1) entry to list
void mapper(string line, multimap<string,string>& results) {
   string user_agent = parse_requester_user_agent(line);
   if (is_mobile_client(user_agent))
     results.add(user_agent, 1);
}

// called once per unique key (user_agent) in results
// values is a list of values associated with the given key
void reducer(string key, list<string> values, int& result) {
    int sum = 0;
    for (v in values)
       sum += v;
    result = sum;
}


// iterator over lines of text file
LineByLineReader input("hdfs://15418log.txt");

// stores output
Writer output("hdfs://…");

// do stuff
runMapReduceJob(mapper, reducer, input, output);
```

(The code above computes the count of page views by each type of mobile phone.)

Let's design an implementation of runMapReduceJob

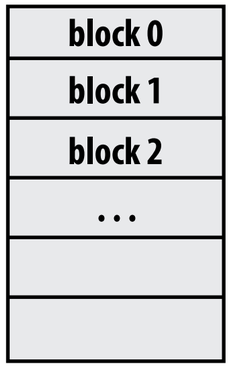
Step 1: running the mapper function

```
// called once per line in file
void mapper(string line, multimap<string,string>& results) {
   string user_agent = parse_requester_user_agent(line);
   if (is_mobile_client(user_agent))
     results.add(user_agent, 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);
```

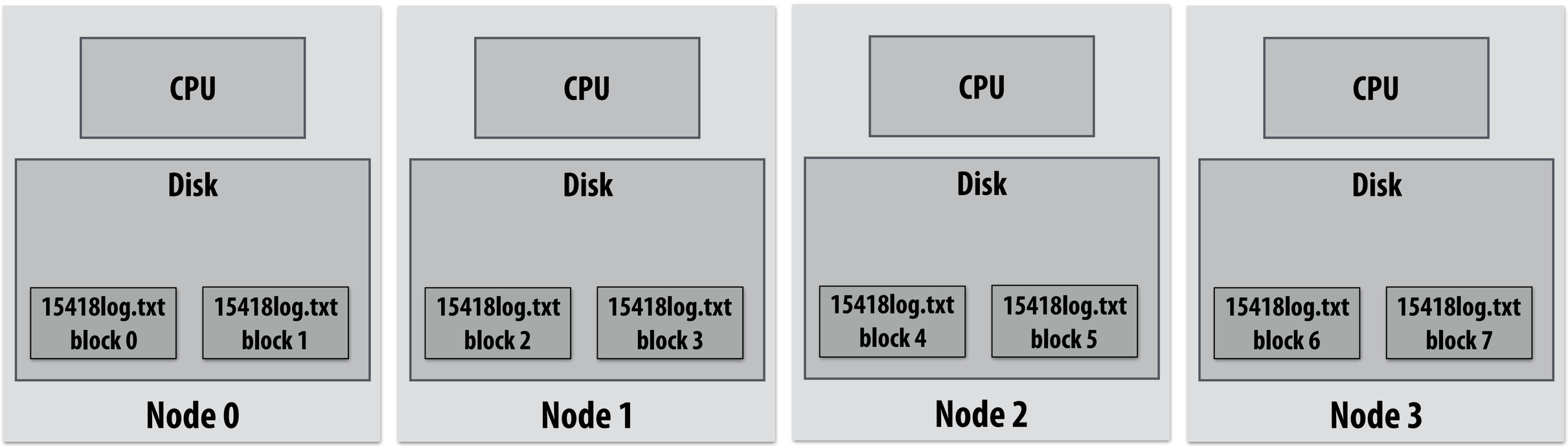
Step 1: run mapper function on all lines of file

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 user_agent = parse_requester_user_agent(line);
   if (is_mobile_client(user_agent))
     results.add(user_agent, 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

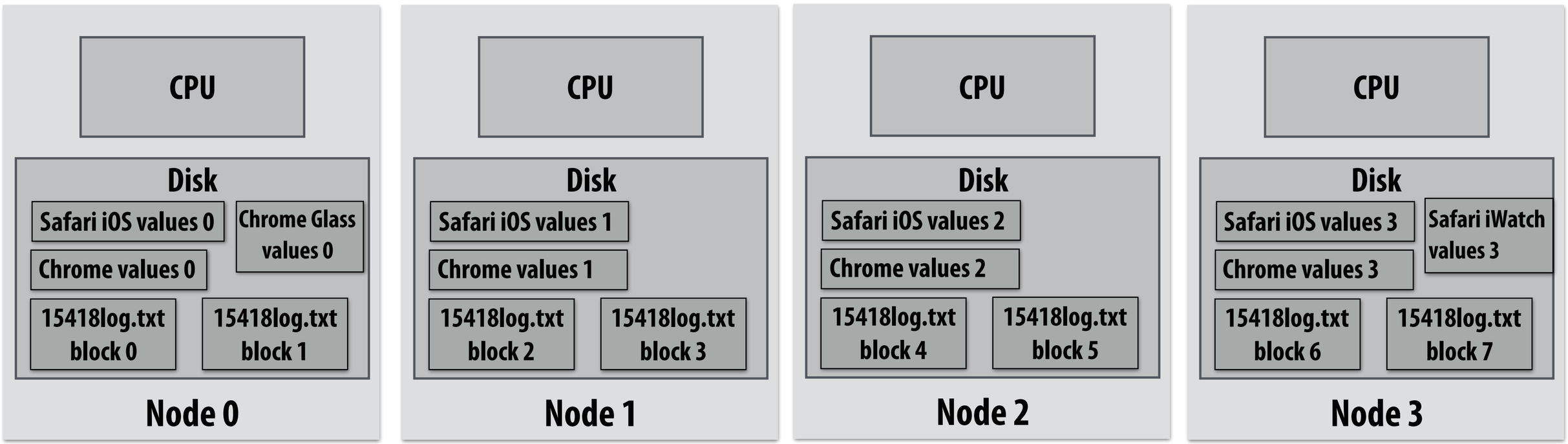
Step 3: Run reducer function on all keys

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

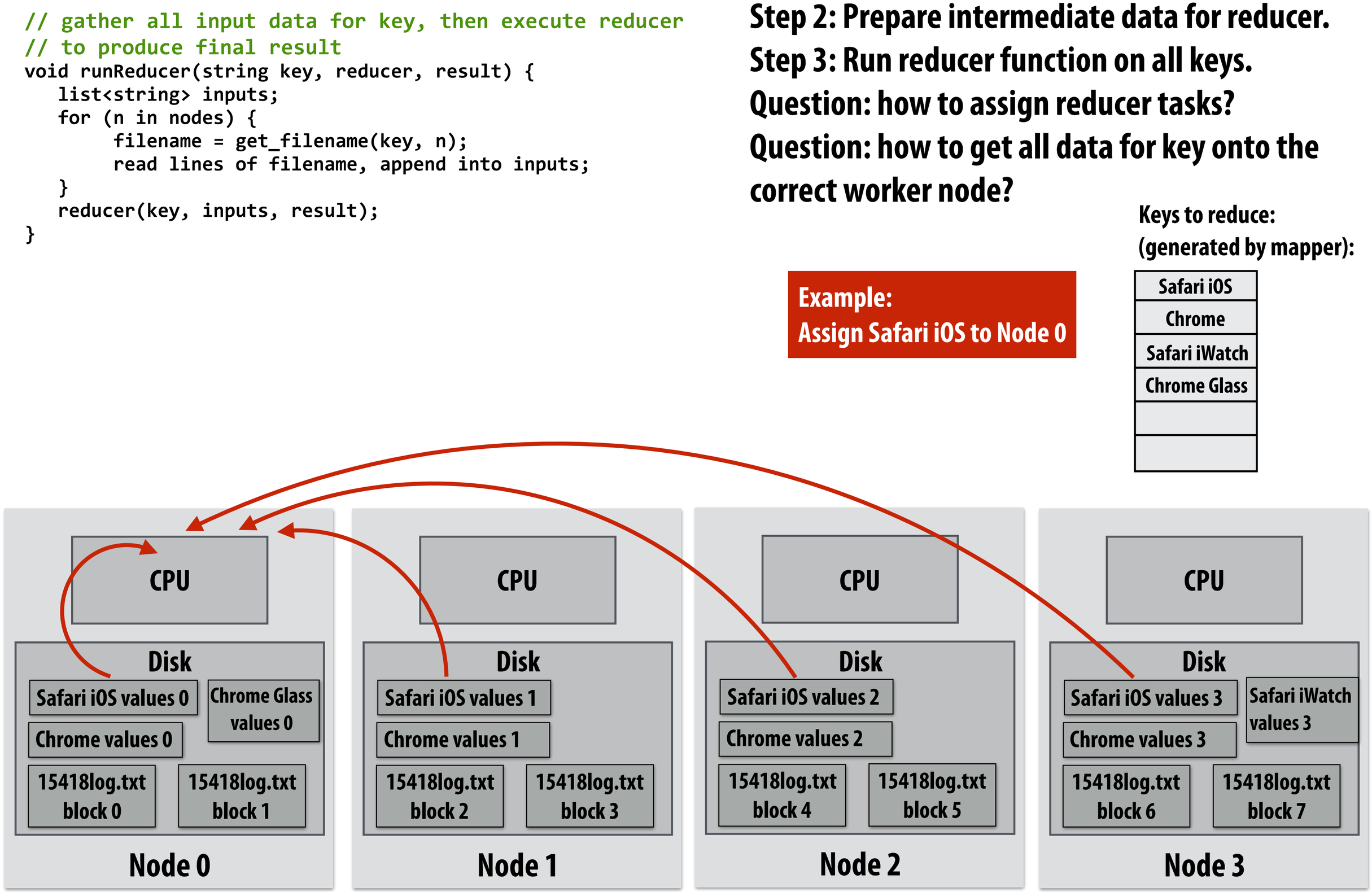
```
// gather all input data for key, then execute reducer
// to produce final result
void runReducer(string key, reducer, result) {
   list<string> inputs;
   for (n in nodes) {
        filename = get_filename(key, n);
        read lines of filename, append into inputs;
   }
   reducer(key, inputs,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

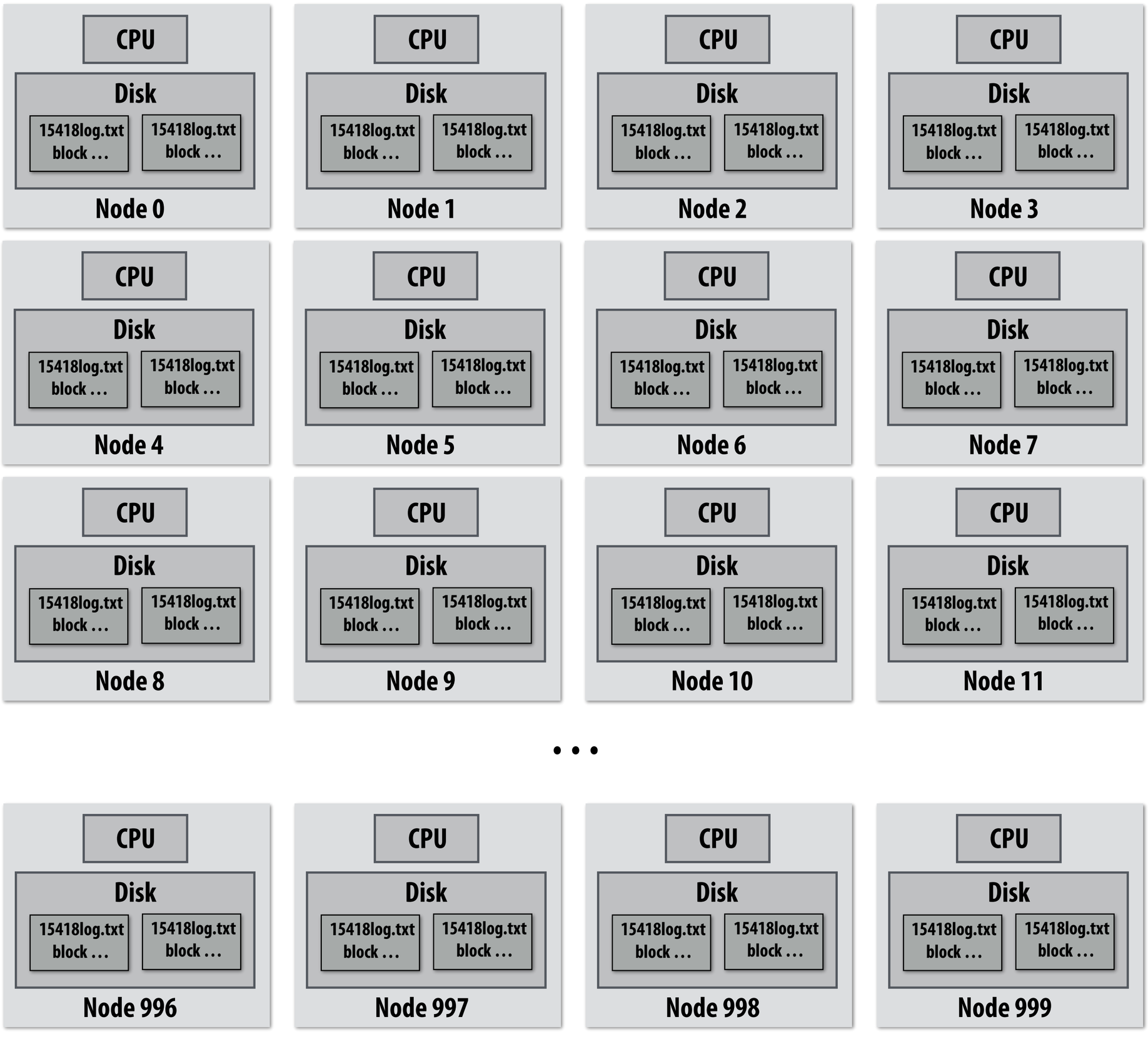
Example:
Assign Safari iOS to Node 0

Keys to reduce:
(generated by mapper):

Safari iOS
Chrome
Safari iWatch
Chrome Glass



Additional implementation challenges at scale



Nodes may fail during program execution

Some nodes may run slower than others (due to different amounts of work, heterogeneity in the cluster, etc..)

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
    - This is 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?

- **Permits 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 last 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]);
}
```

in-memory, fault-tolerant distributed computing

http://spark.apache.org/

[Zaharia et al. NSDI 2012]

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 multiple 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)

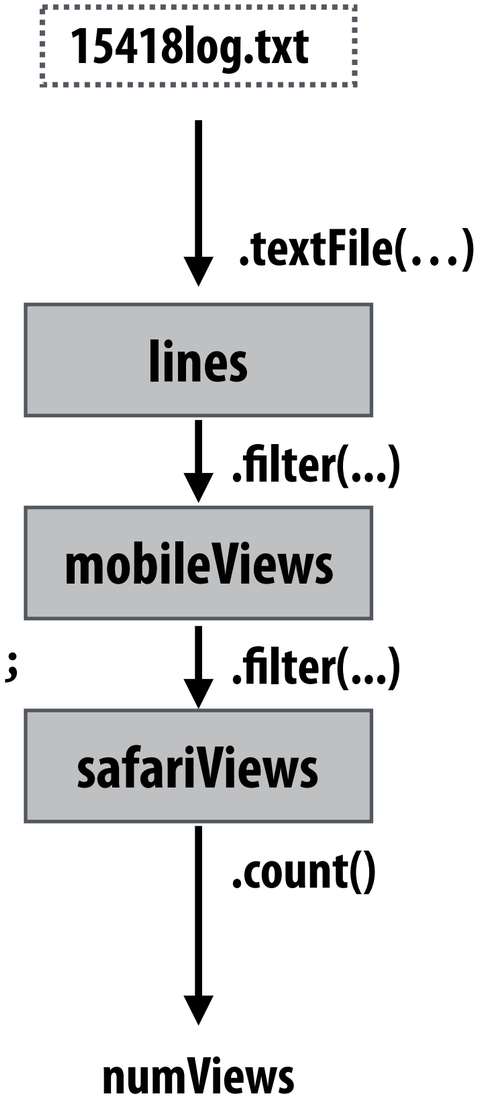
Spark's key programming abstraction:

- 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

RDDs

```
// 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));

// another filter() transformation
var safariViews = mobileViews.filter((x: String) => x.contains("Safari"));

// then count number of elements in RDD via count() action
var numViews = safariViews.count();
```

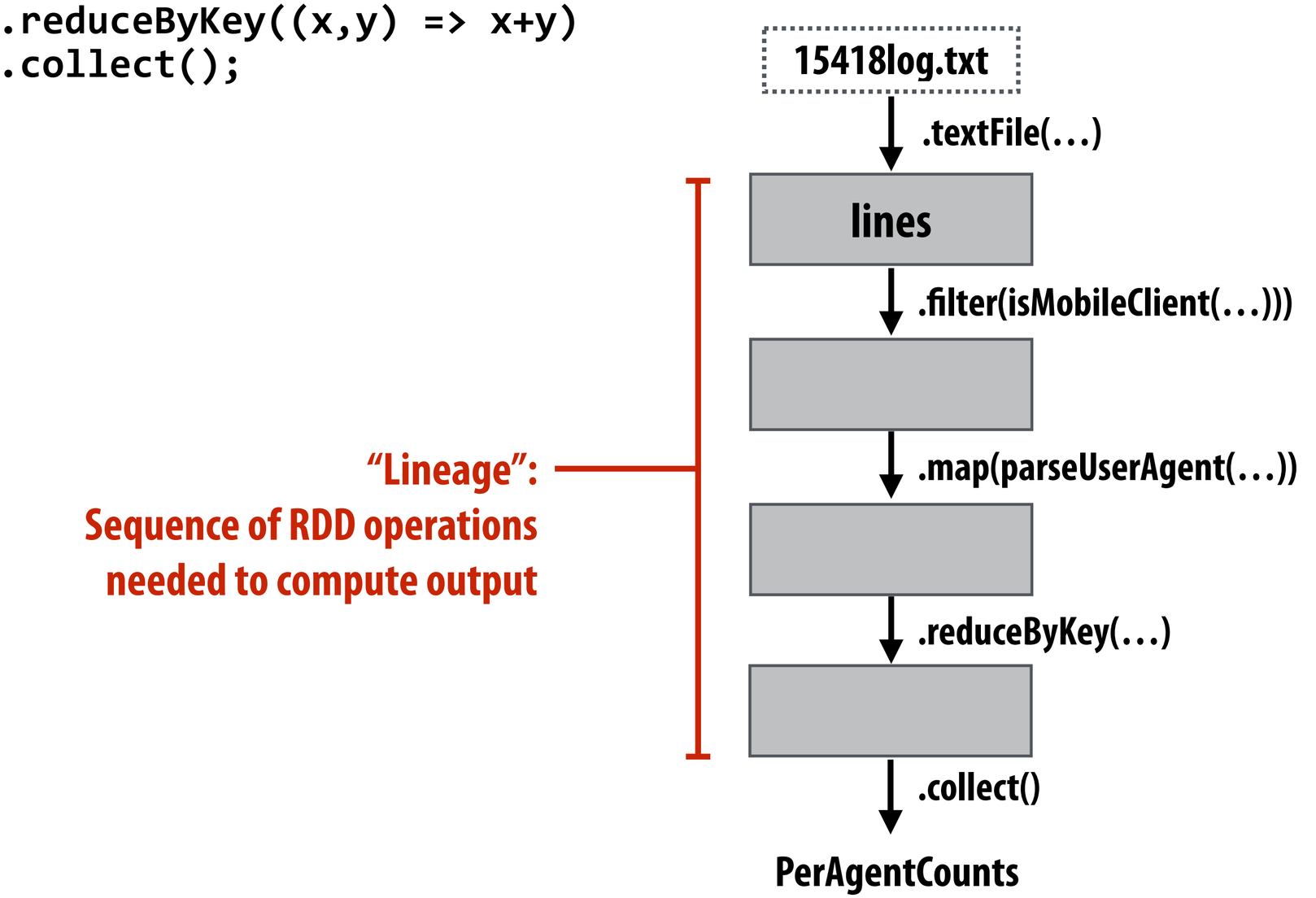
int



Repeating the map-reduce example

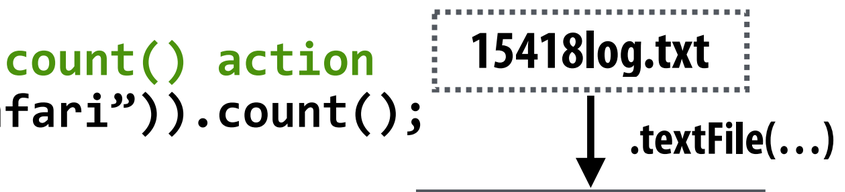
```
// 1. create RDD from file system data
// 2. create RDD with only lines from mobile clients
// 3. create RDD with elements of type (String,Int) from line string
// 4. group elements by key
// 5. call provided reduction function on all keys to count views
var perAgentCounts =  spark.textFile(“hdfs://15418log.txt”)
                           .filter(x => isMobileClient(x))
                           .map(x => (parseUserAgent(x),1));
                           .reduceByKey((x,y) => x+y)
                           .collect();
```

Array[String,int]

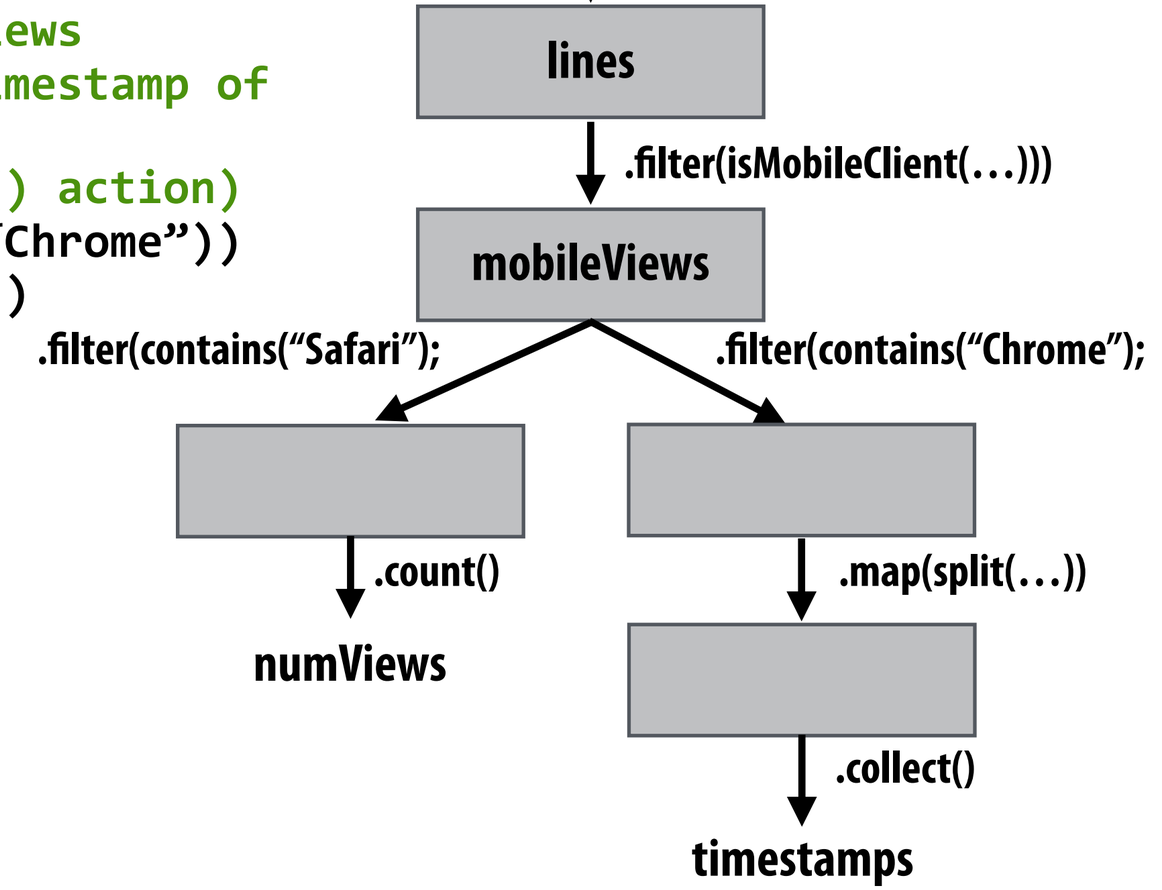


Another Spark program

```
// 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 timestamp of
//    page view
// 3. convert RDD to a scalar sequence (collect() action)
var timestamps = mobileViews.filter(_.contains(“Chrome”))
                            .map(_.split(“ ”)(0))
                            .collect();
```



RDD transformations and actions

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

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

Actions: (provide data back to the "host" application)

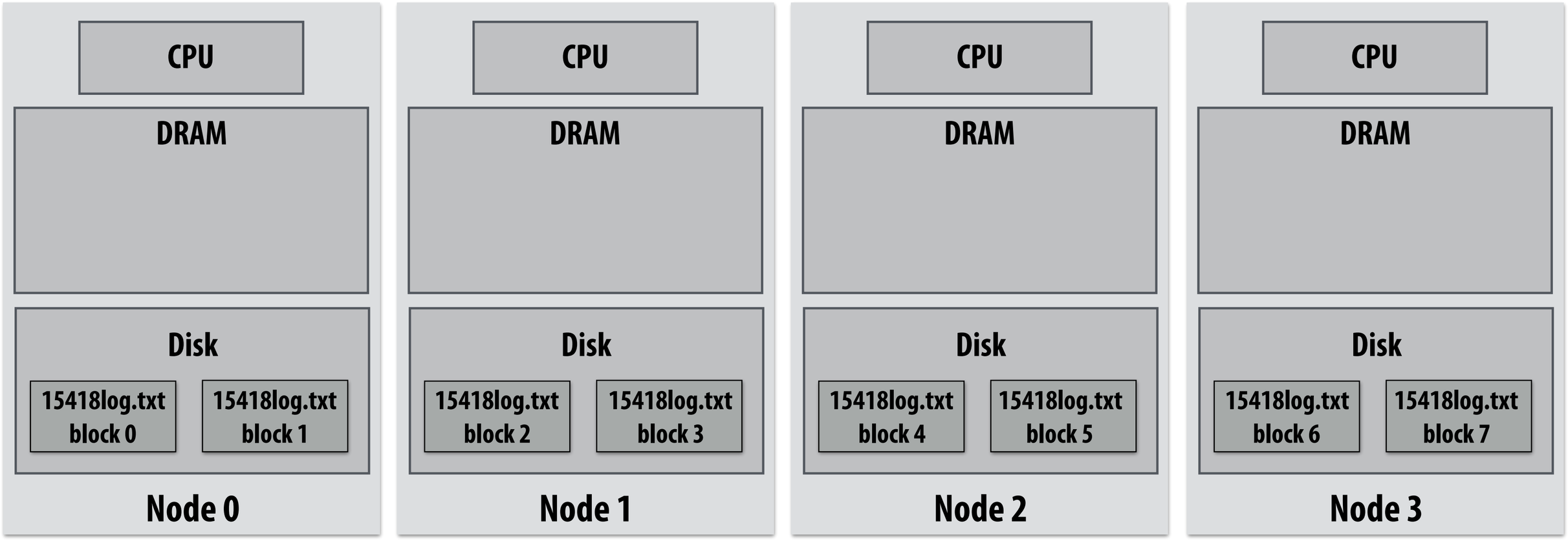
$$
\begin{aligned}
count() &: \mathrm{RDD[T]} \Rightarrow \mathrm{Long} \\
collect() &: \mathrm{RDD[T]} \Rightarrow \mathrm{Seq[T]} \\
reduce(f : (\mathrm{T}, \mathrm{T}) \Rightarrow \mathrm{T}) &: \mathrm{RDD[T]} \Rightarrow \mathrm{T} \\
lookup(k : \mathrm{K}) &: \mathrm{RDD[(K, V)]} \Rightarrow \mathrm{Seq[V]}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

How do we implement RDDs?

In particular, how should they be stored?

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

Question: should we think of RDD’s like arrays?

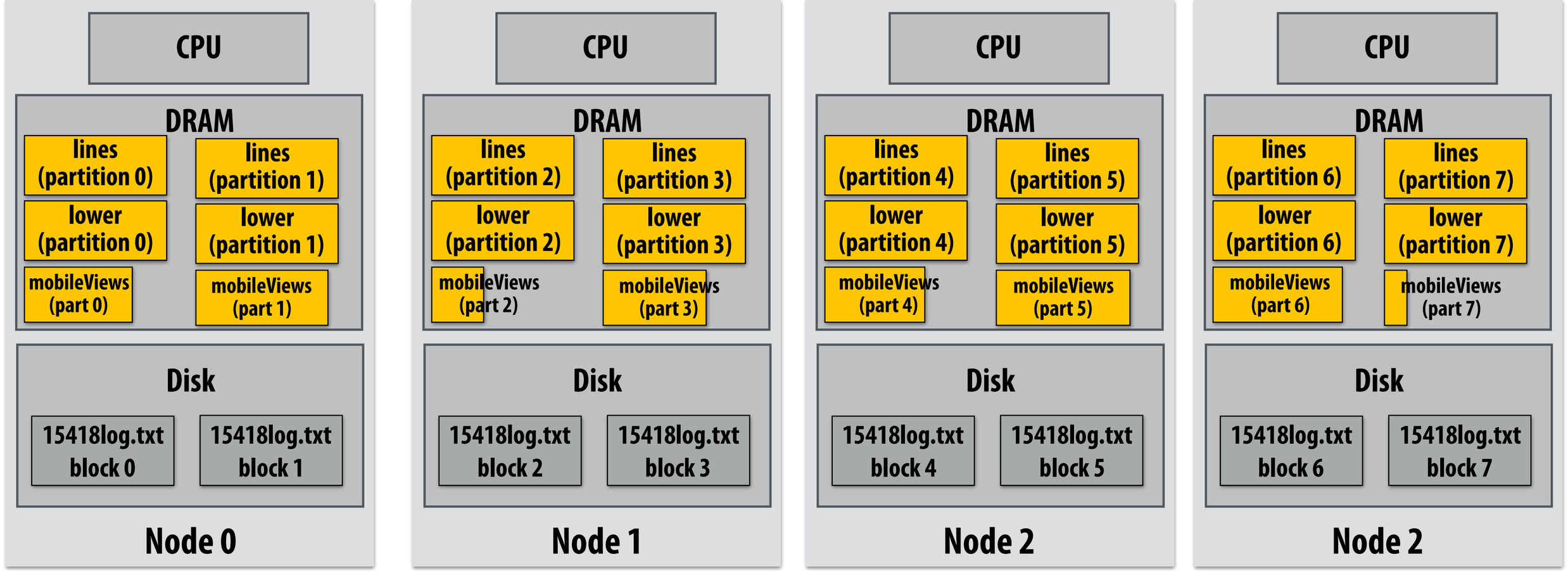


How do we implement RDDs?

In particular, how should they be stored?

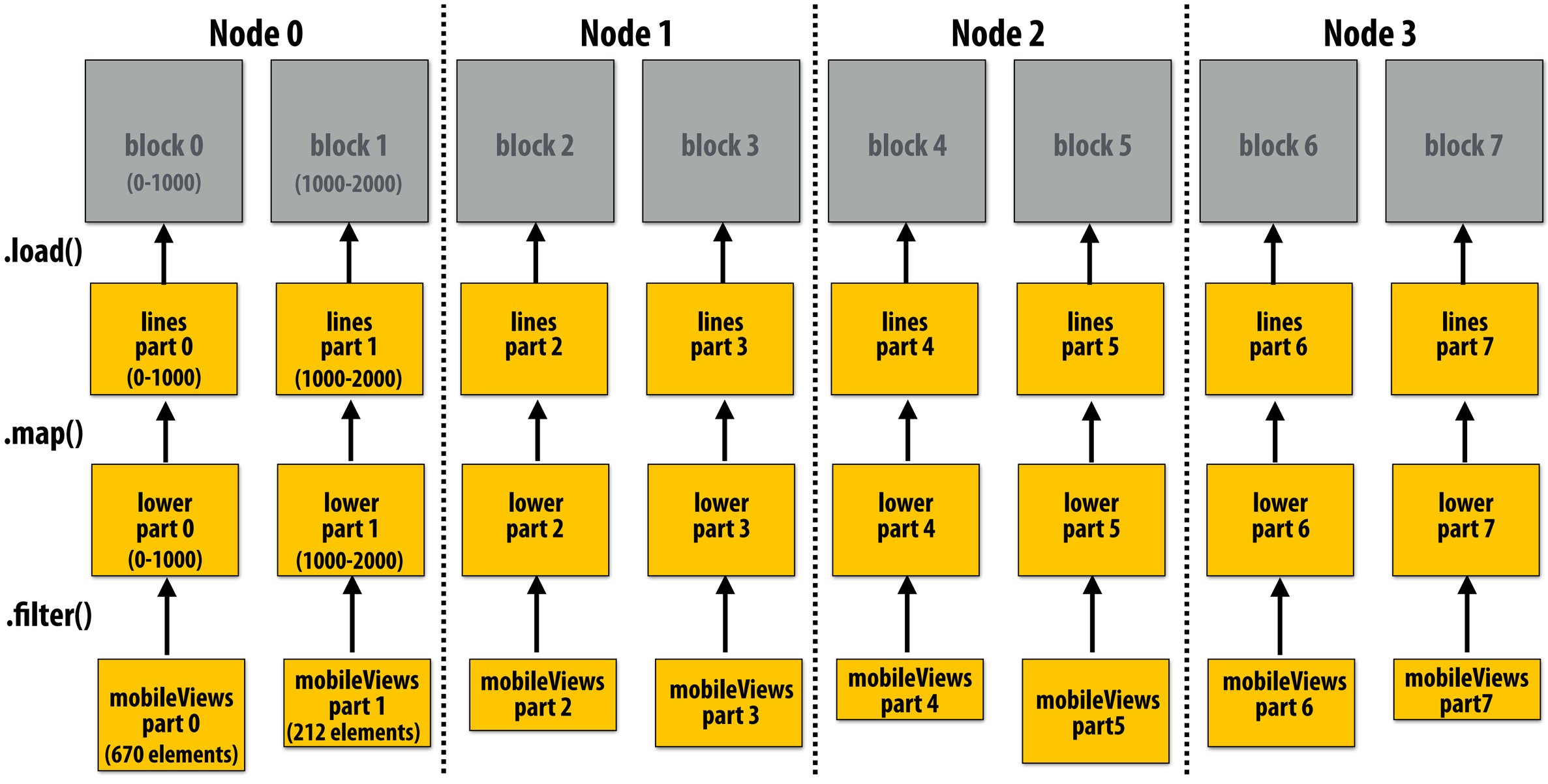
```
var lines = spark.textFile(“hdfs://15418log.txt”);
var lower = lines.map(_.toLower());
var mobileViews = lower.filter(x => isMobileClient(x));
var howMany = mobileViews.count();
```

In-memory representation would be huge! (larger than original file on disk)



RDD partitioning and dependencies

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



Black lines show dependencies between RDD partitions.

Implementing sequence of RDD ops efficiently

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

Recall “loop fusion” from start of lecture

The following code stores only a line of the log file in memory, and only reads input data from disk once (“streaming” solution)

```
int count = 0;
while (inputFile.eof()) {
   string line = inputFile.readLine();
   string lower = line.toLower;
   if (isMobileClient(lower))
     count++;
}
```

A simple interface for RDDs

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

```
// create RDD by mapping fun onto input (parent) RDD
RDD::map(RDD parent, func) {
   return new RDDFromMap(parent, func);
}

// create RDD from text file on disk
RDD::textFile(string filename) {
   return new RDDFromTextFile(open(filename));
}

// count action (forces evaluation of RDD)
RDD::count() {
   int count = 0;
   while (hasMoreElements()) {
      var el = next();
      count++;
   }
}
```

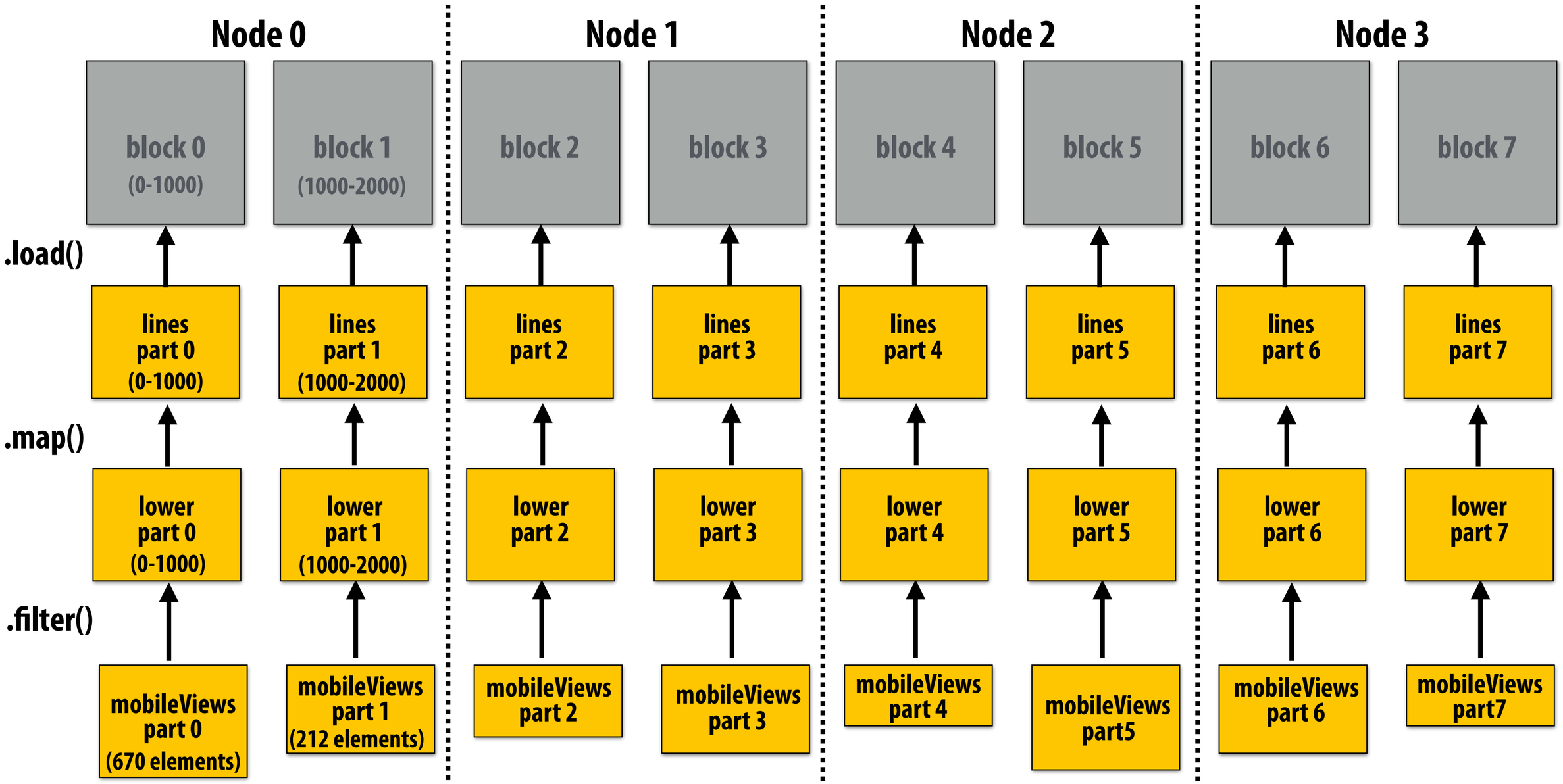
```
RDD::hasMoreElements() {
   parent.hasMoreElements();
}

// overloaded since no parent exists
RDDFromTextFile::hasMoreElements() {
   return !inputFile.eof();
}

RDDFromTextFile::next() {
   return inputFile.readLine();
}

RDDFromMap::next() {
   var el = parent.next();
   return el.toLower();
}

RDDFromFilter::next() {
  while (parent.hasMoreElements()) {
    var el = parent.next();
    if (isMobileClient(el))
      return el;
}
```

Narrow dependencies

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

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

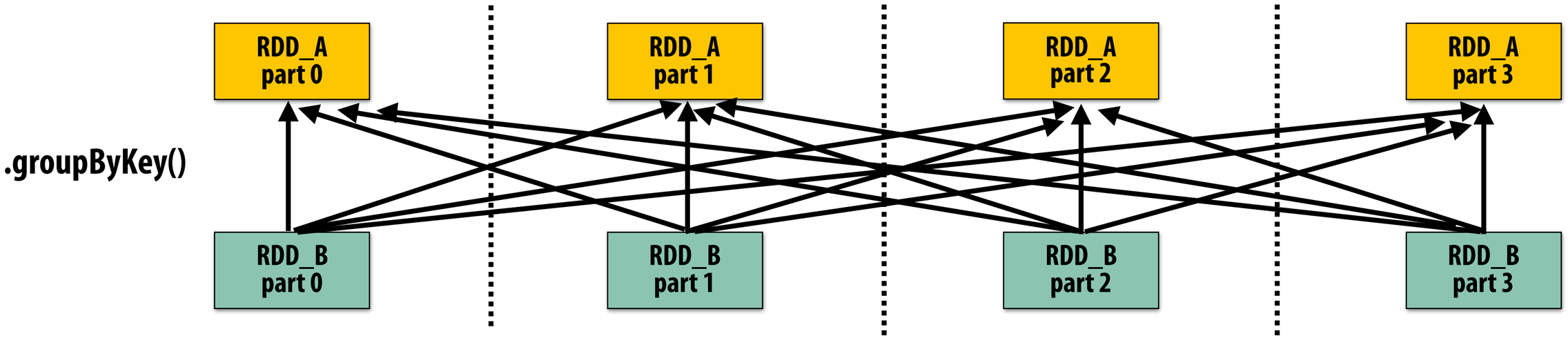
- **Allows for fusing of operations (here: can apply map and then filter all at once on input element)**
- **In this example: no communication between nodes of cluster (communication of one int at end to perform count() reduction)**



Wide dependencies

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

"Make a new RDD where each element is a sequence containing all values from the parent RDD with the same key."



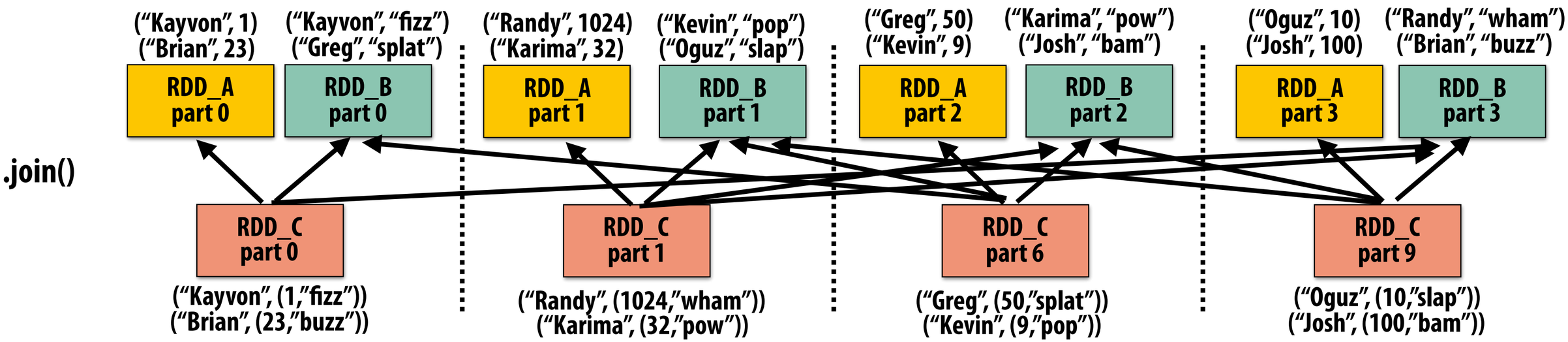
- Wide dependencies = each partition of parent RDD referenced by multiple child RDD partitions
- Challenges:
  - Must compute all of RDD_A before computing RDD_B
    - Example: groupByKey() may induce all-to-all communication as shown above
  - May trigger significant recompilation of ancestor lineage upon node failure (will address resilience in a few slides)

Cost of operations depends on partitioning

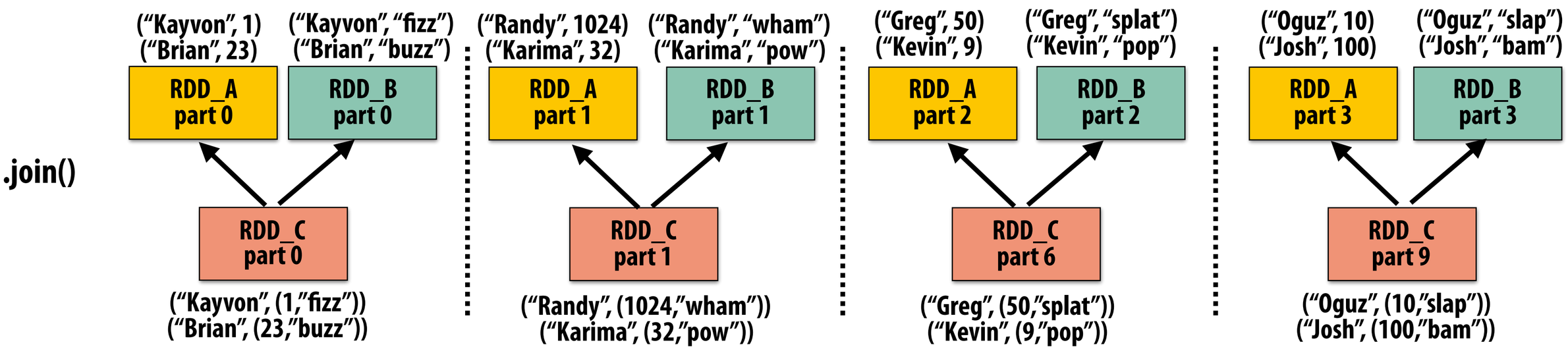
join: RDD[(K,V)], RDD[(K,W)] → RDD[(K,(V,W))]

Assume data in RDD_A and RDD_B are partitioned by key: hash username to partition id

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



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



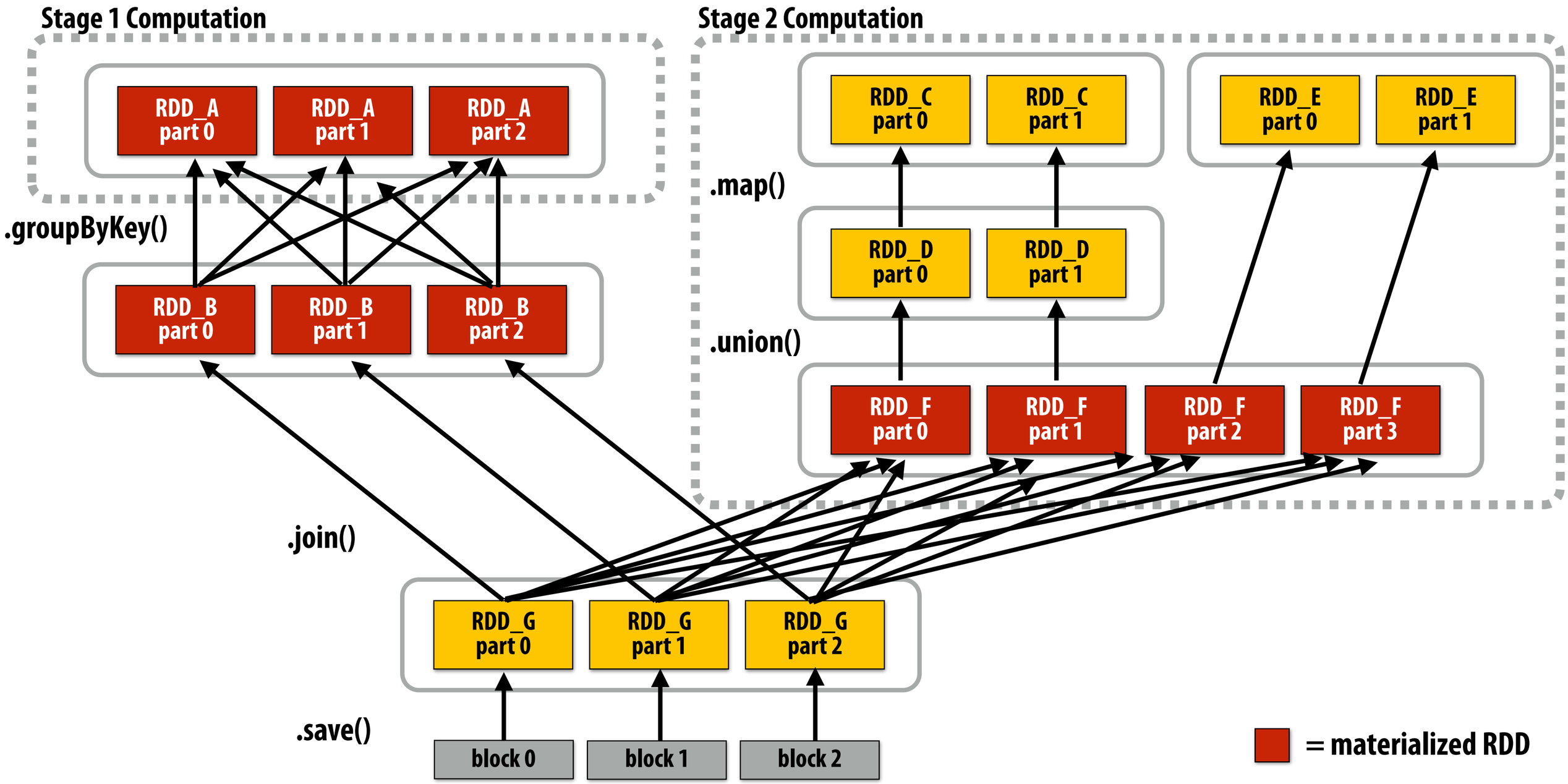
PartitionBy() transformation

- 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);

// inform Spark of partition
// .persist() also instructs Spark to try to keep dataset in memory
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

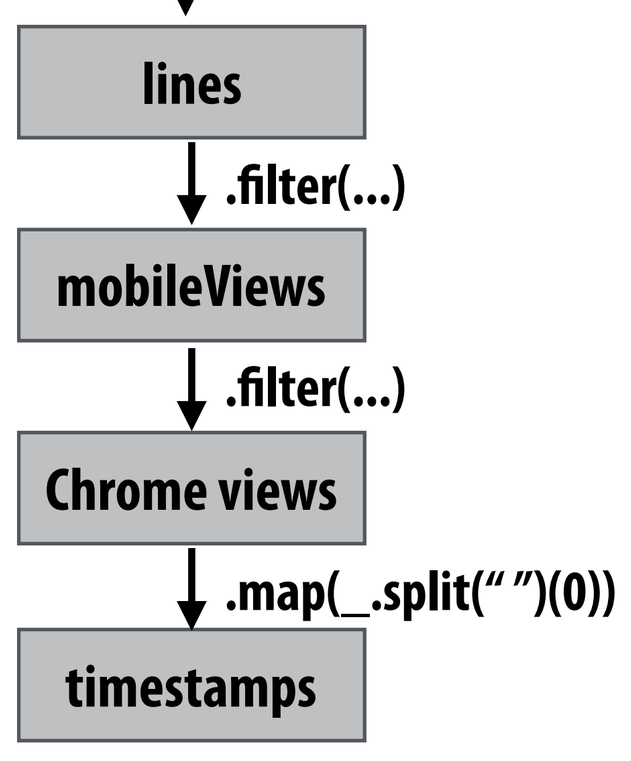
Stage 2 Computation: evaluate map in fused manner, only actually materialize RDD F

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

Implementing resilience via lineage

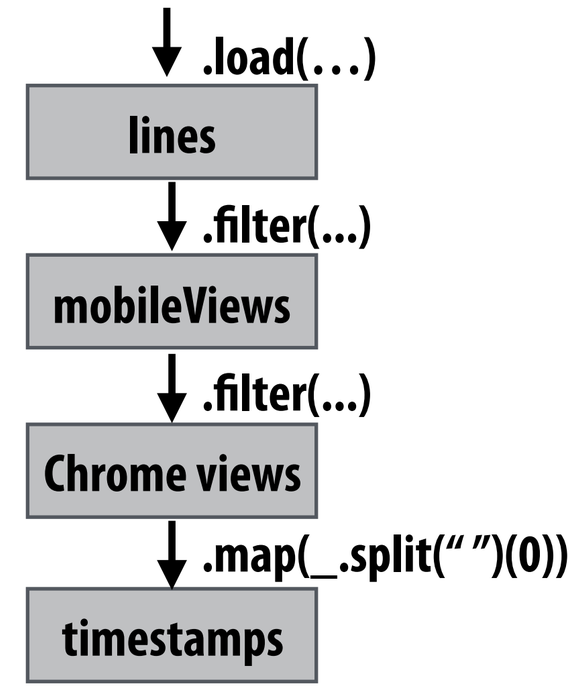
- RDD transformations are bulk, deterministic, and functional
  - Implication: runtime can always reconstruct contents of RDD from its lineage (the sequence of transformations used to create it)
  - Lineage is a log of transformations
  - Efficient: since log records bulk data-parallel operations, overhead 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 timestamp of
//    page view (first element)
// 3. convert RDD To a scalar sequence (collect() action)
var timestamps = mobileView.filter(_.contains(“Chrome”))
                           .map(_.split(“ ”)(0));
```

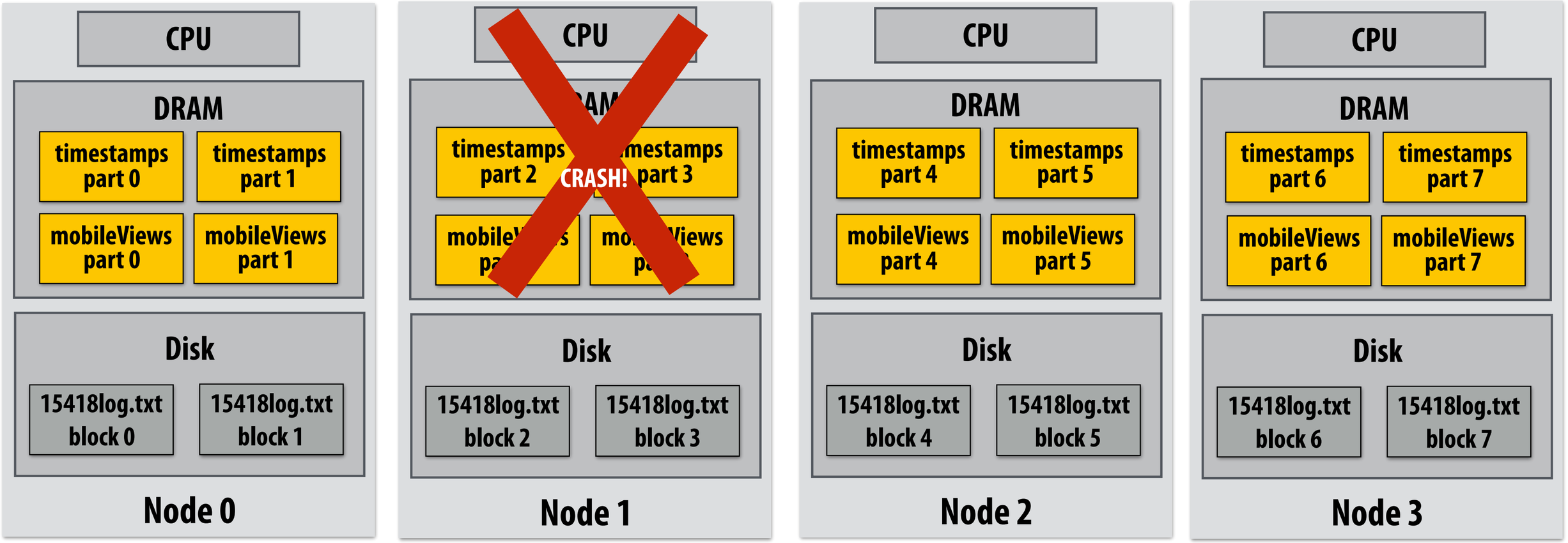


Upon node failure: recompute lost RDD partitions from lineage

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



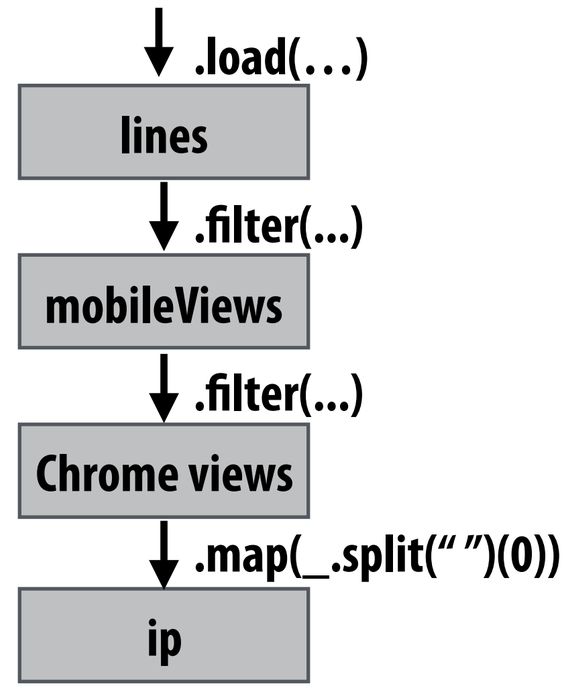
Must reload required subset of data from disk and recompute entire sequence of operations given by lineage to regenerate partitions 2 and 3 of RDD timestamps.



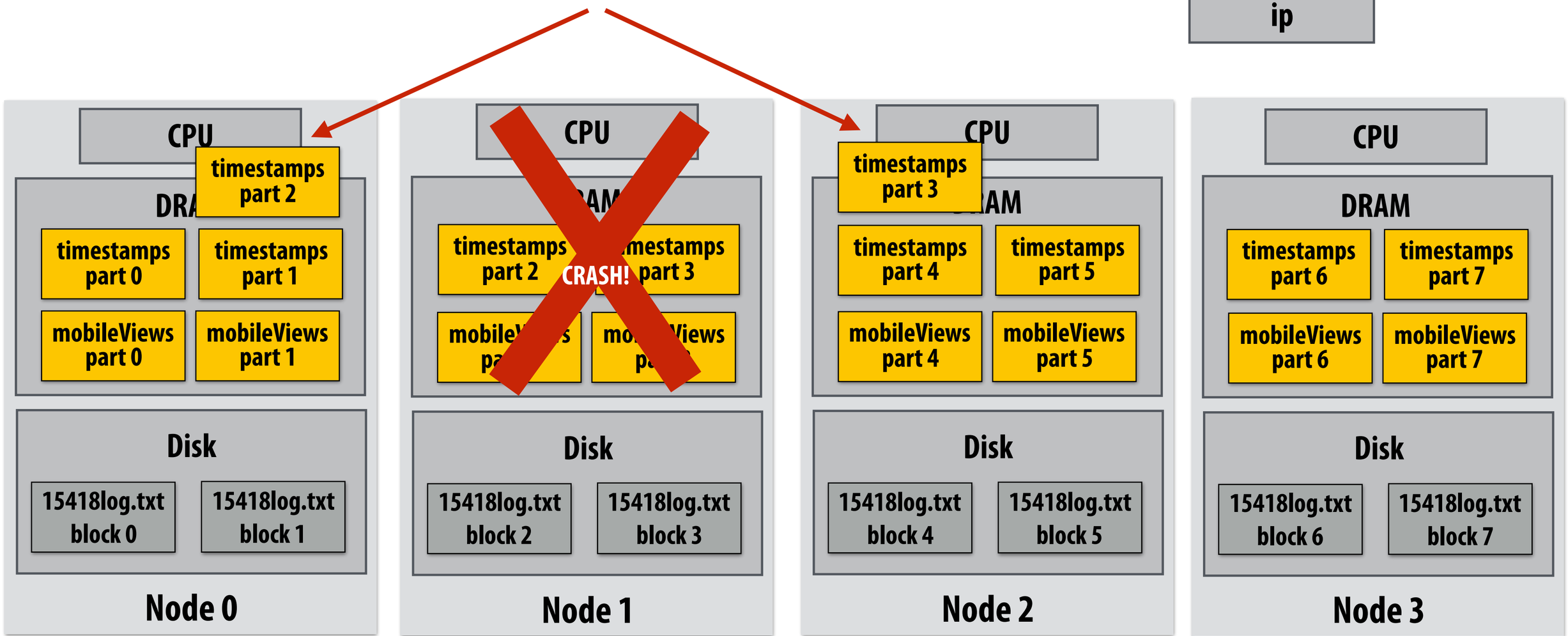
Note: (not shown): file system data is replicated so assume blocks 2 and 3 remain accessible to all nodes

Upon node failure: recompute lost RDD partitions

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

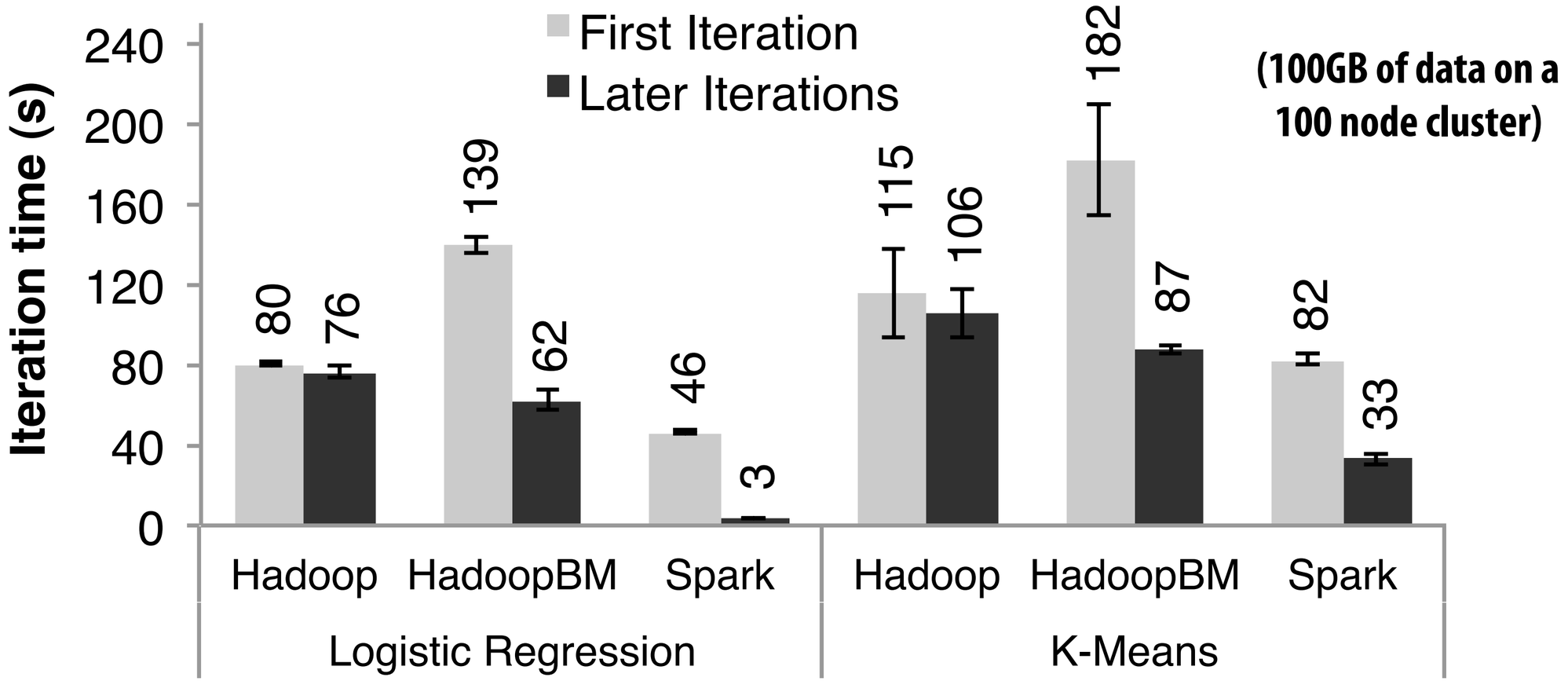


Must reload required subset of data from disk and recompute entire sequence of operations given by lineage to regenerate partitions 2 and 3 of RDD timestamps.



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

Spark 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**

Caution: "scale out" is not the entire story

- Distributed systems designed for cloud execution address many difficult challenges, and have been instrumental in the explosion of "big-data" computing and large-scale analytics
  - Scale-out parallelism to many machines
  - Resiliency in the face of failures
  - Complexity of managing clusters of machines

- But scale out is not the whole story:

20 Iterations of Page Rank

scalable system	cores	twitter	uk-2007-05
GraphChi [10]	2	3160s	6972s
Stratosphere [6]	16	2250s	-
X-Stream [17]	16	1488s	-
Spark [8]	128	857s	1759s
Giraph [8]	128	596s	1235s
GraphLab [8]	128	249s	833s
GraphX [8]	128	419s	462s
Single thread (SSD)	1	300s	651s
Single thread (RAM)	1	275s	-

name	twitter_rv [11]	uk-2007-05 [4]
nodes	41,652,230	105,896,555
edges	1,468,365,182	3,738,733,648
size	5.76GB	14.72GB

Further optimization of the baseline brought time down to 110s

Vertex order (SSD)	1	300s	651s
Vertex order (RAM)	1	275s	-
Hilbert order (SSD)	1	242s	256s
Hilbert order (RAM)	1	110s	-

["Scalability! At what COST?" McSherry et al. HotOS 2015]

Caution: "scale out" is not the entire story

Label Propagation

[McSherry et al. HotOS 2015]

scalable system	cores	twitter	uk-2007-05
Stratosphere [6]	16	950s	-
X-Stream [17]	16	1159s	-
Spark [8]	128	1784s	$\geq$ 8000s
Giraph [8]	128	200s	$\geq$ 8000s
GraphLab [8]	128	242s	714s
GraphX [8]	128	251s	800s
Single thread (SSD)	1	153s	417s

from McSherry 2015:

"The published work on big data systems has fetishized scalability as the most important feature of a distributed data processing platform. While nearly all such publications detail their system's impressive scalability, few directly evaluate their absolute performance against reasonable benchmarks. To what degree are these systems truly improving performance, as opposed to parallelizing overheads that they themselves introduce?"

COST: "Configuration that Outperforms a Single Thread"

Perhaps surprisingly, many published systems have unbounded COST—i.e., no configuration outperforms the best single-threaded implementation—for all of the problems to which they have been applied.

BID Data Suite (1 GPU accelerated node)

[Canny and Zhao, KDD 13]

Page Rank

System	Graph VxE	Time(s)	Gflops	Procs
Hadoop	?x1.1B	198	0.015	50x8
Spark	40Mx1.5B	97.4	0.03	50x2
Twister	50Mx1.4B	36	0.09	60x4
PowerGraph	40Mx1.4B	3.6	0.8	64x8
BIDMat	60Mx1.4B	6	0.5	1x8
BIDMat+disk	60Mx1.4B	24	0.16	1x8

Latency Dirichlet Allocation (LDA)

System	Docs/hr	Gflops	Procs
Smola[15]	1.6M	0.5	100x8
PowerGraph	1.1M	0.3	64x16
BIDMach	3.6M	30	1x8x1

Performance improvements to Spark

- With increasing DRAM sizes and faster persistent storage (SSD), there is interest in improving the CPU utilization of Spark applications
  - Goal: reduce "COST"

- Efforts looking at efficient code generation to Spark ecosystem (e.g., generate SIMD kernels, target accelerators like GPUs, etc.) to close the gap on single node performance
  - See Spark's Project Tungsten

- High-performance computing ideas are influencing design of future performance-oriented distributed systems
  - Conversely: the scientific computing community has a lot to learn from the distributed computing community about elasticity and utility computing

Spark summary

- Introduces opaque sequence abstraction (RDD) to encapsulate intermediates of cluster computations (previously… frameworks like Hadoop/MapReduce 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) => ...
}
```

If you enjoyed today's topic…

- I recommend looking at Legion
  - legion.stanford.edu
- Designed from Supercomputing perspective
  - Spark was designed from a distributed computing perspective
- Key idea: programming via collections called "logical regions"
  - Systems provides 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 programming
  - Overheads are amortized over very large data-parallel operations