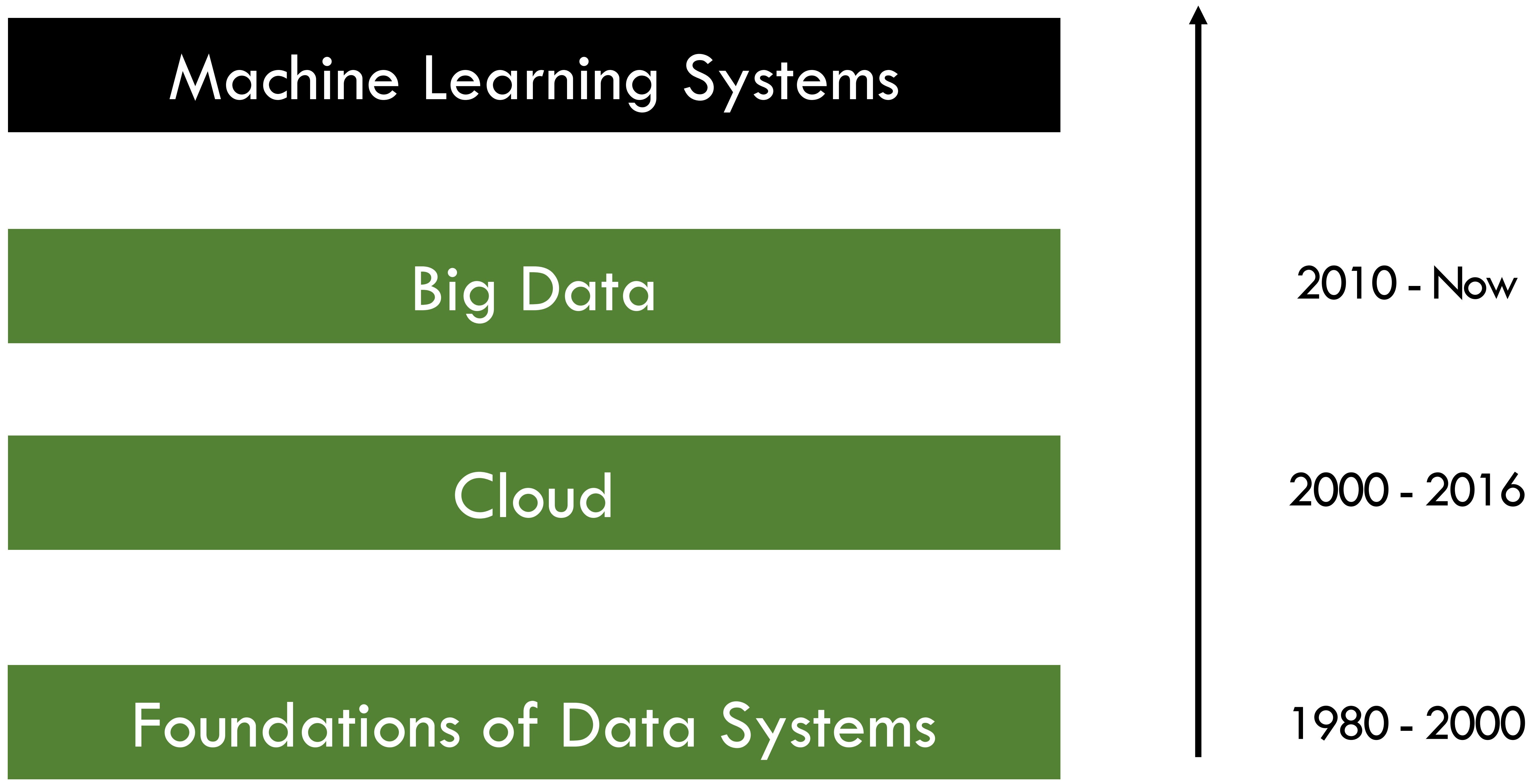
Where We Are


Machine Learning Systems
Big Data
2010 - Now
Cloud
2000 - 2016
Foundations of Data Systems
1980 - 2000

Recap: Batch Processing

- Batch Processing
  - Suitable for latency insensitive tasks
  - Map-reduce prog model: mapper, reducer, (combiner, partitioner)
  - Many Map-reduce jobs to compose dataflows
    - They communicate via disk I/O
  - Pros and Cons
    - Pros: expressive, scalable, and fault tolerant
    - Cons: low performance due to disk I/O

Today’s topic: Stream Processing

- Computation vs. I/O: Arithmetic intensity
  - Loop fusion
- When MapReduce fails
- Spark and RDD
- Why Spark succeeded

Recall: Instruction

Register names

```
add     %rbx,  %rax
```

is

```
rax += rbx
```

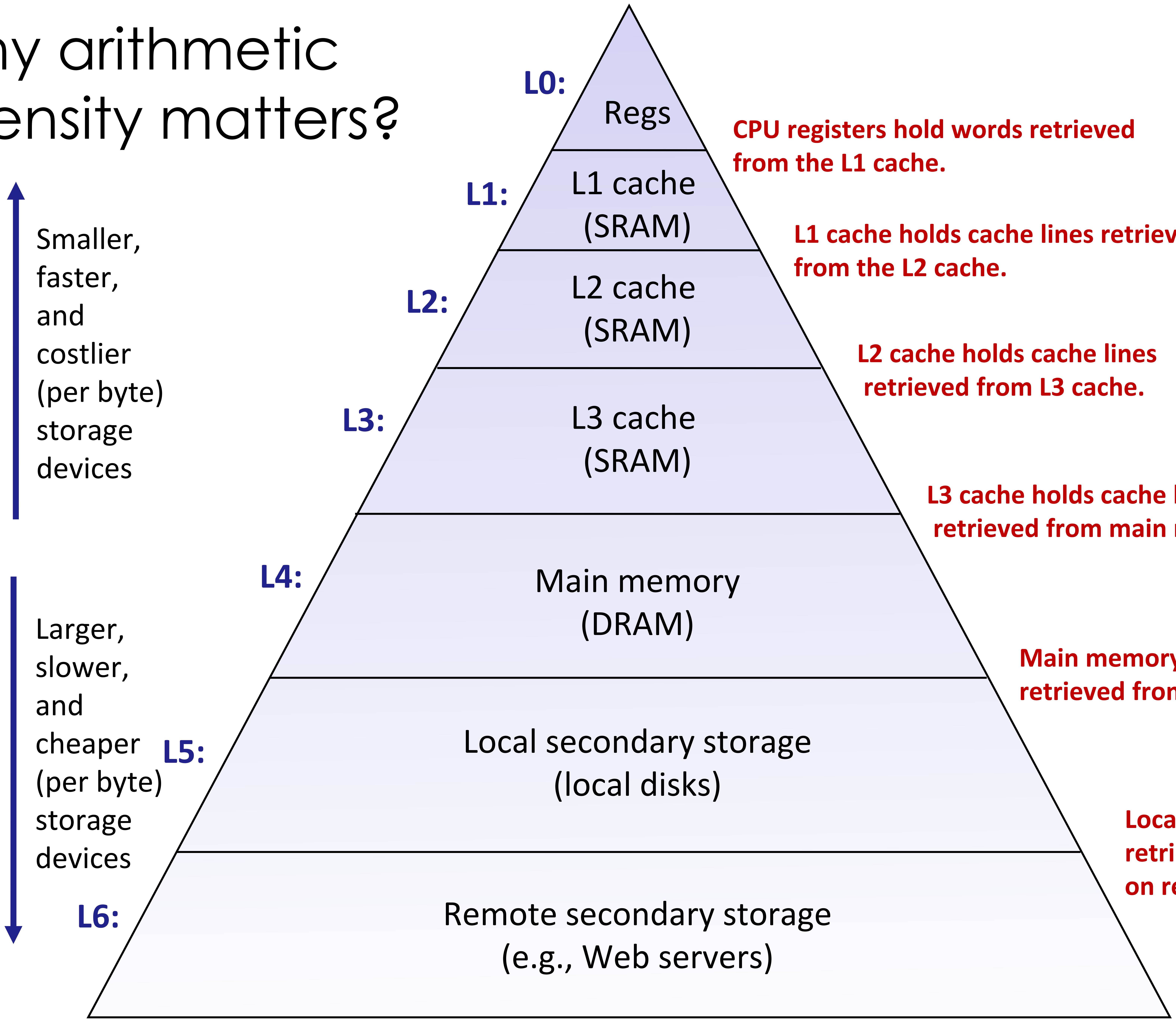
Recall: Basics of Processors

Q: *How does a processor execute machine code?*

- Types of ISA commands to manipulate register contents:
  - Memory access: **load** (copy bytes from a DRAM address to register); **store** (reverse); put constant
  - Arithmetic & logic on data items in registers: **add/multiply/etc**.; bitwise ops; compare, etc.; handled by ALU
  - Control flow (branch, call, etc.); handled by CU
- Caches: Small local memory to buffer instructions/data

If interested in more details: https://www.youtube.com/watch?v=cNN_tTXABUA

Why arithmetic intensity matters?



Smaller, faster, and costlier (per byte) storage devices
Larger, slower, and cheaper (per byte) storage devices
L0:
Regs
L1:
L1 cache (SRAM)
L2:
L2 cache (SRAM)
L3:
L3 cache (SRAM)
L4:
Main memory (DRAM)
L5:
Local secondary storage (local disks)
L6:
Remote secondary storage (e.g., Web servers)
CPU registers hold words retrieved from the L1 cache.
L1 cache holds cache lines retrieved from the L2 cache.
L2 cache holds cache lines retrieved from L3 cache.
L3 cache holds cache lines retrieved from main memory.
Main memory holds disk blocks retrieved from local disks.
Local disks hold files retrieved from disks on remote servers.

How to measure the impact of I/O

- I/O is the primary enemy of computer engineers/scientists: it will always slow down computation in every levels of the memory hierarchy
  - Processor reads/writes cache or memory
  - Map-reduce save and load results from distributed storage
- Q: how we measure such slowdown?
  - Arithmetic intensity

Arithmetic Intensity

$$AI = \frac{\#Compute\ Op}{\#\mathrm{I/O}\ op}$$

Arithmetic intensity

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

Two loads, one store per math op

1. Read A[i]
2. Read B[i]
3. Add A[i]+B[i]
4. Store C[i]

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

Which program performs better? Program 2

```
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);




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);
```

Overall arithmetic intensity = 1/3

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

computation fusion!

Core Problem of Map-reduce

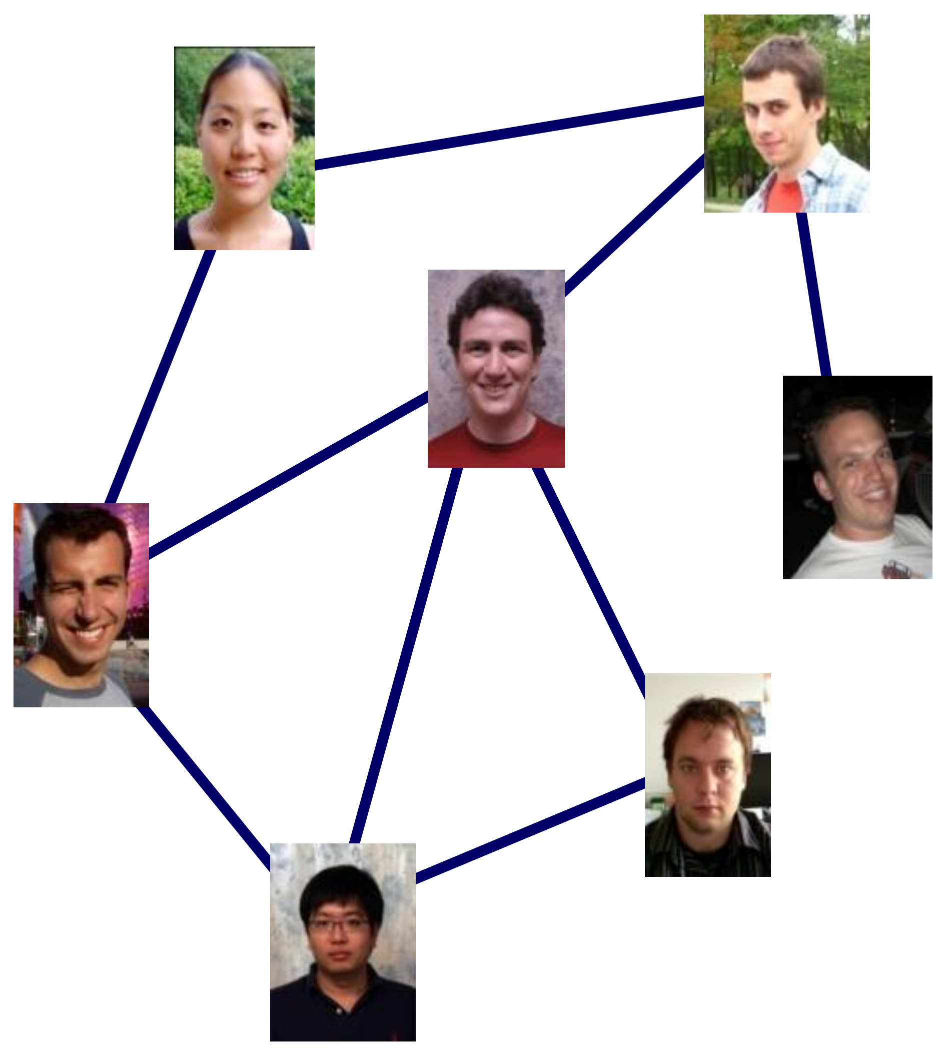
Low arithmetic intensity
due to Disk I/O

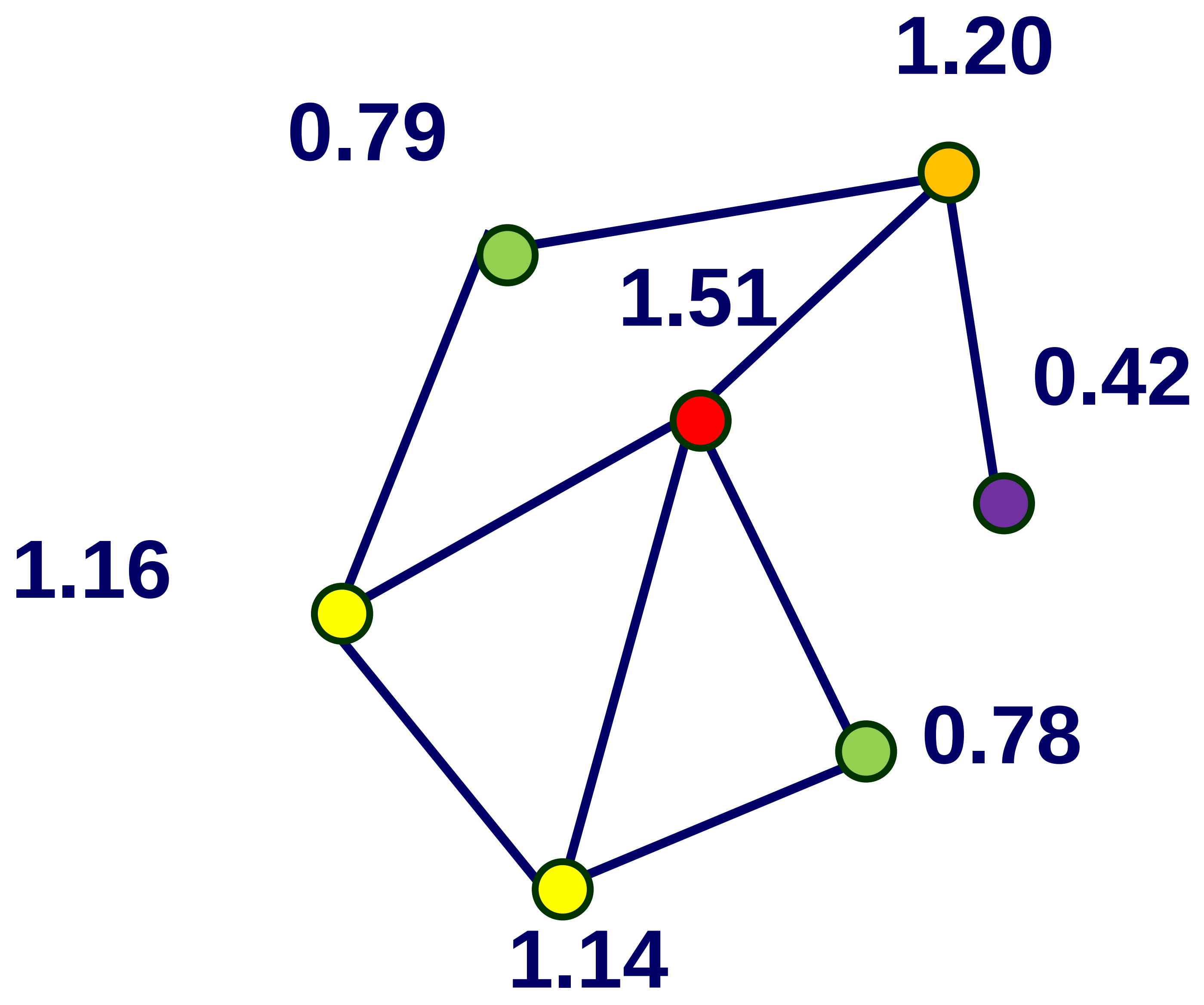
PageRank

PageRank Computation

- Larry Page & Sergey Brinn, 1998

Rank “Importance” of Web Pages





PageRank Computation

Initially

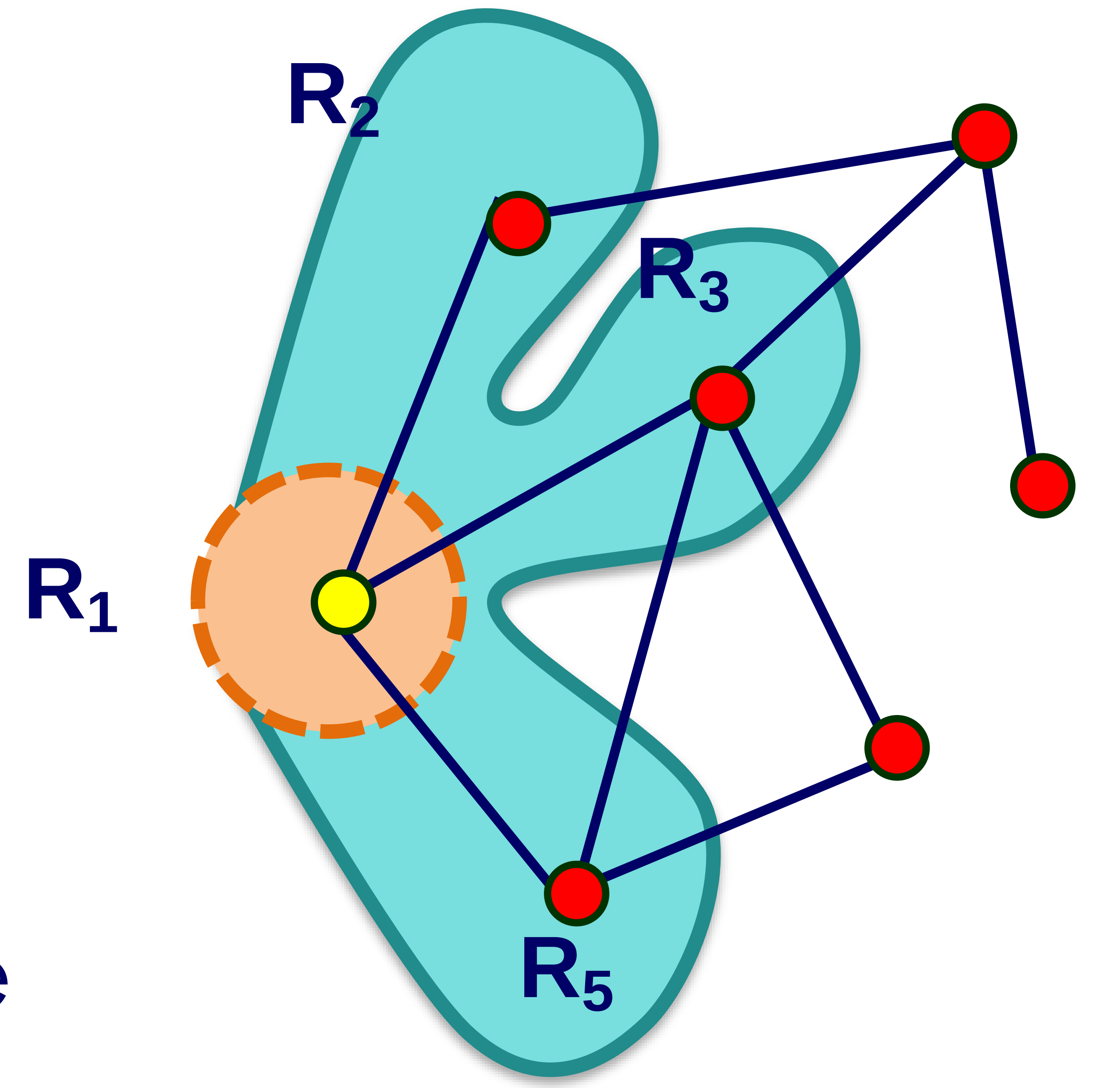
- Assign weight 1.0 to each page

Iteratively

- Select arbitrary node and update its value

Convergence

- Results unique, regardless of selection ordering



$R_1 \leftarrow \quad 0.1 + 0.9 * (\frac{1}{2} R_2 + \frac{1}{4} R_3 + \frac{1}{3} R_5)$

PageRank with Map/Reduce

$R_1 \leftarrow 0.1 + 0.9 * (\frac{1}{2} R_2 + \frac{1}{4} R_3 + \frac{1}{3} R_5)$

Each Iteration: Update all nodes

- Map: Generate values to pass along each edge
  - Key value 1: $(1, \frac{1}{2} R_2)$ $(1, \frac{1}{4} R_3)$ $(1, \frac{1}{3} R_5)$
  - Similar for all other keys
- Reduce: Combine edge values to get new rank
  - $R_1 \leftarrow 0.1 + 0.9 * (\frac{1}{2} R_2 + \frac{1}{4} R_3 + \frac{1}{3} R_5)$
  - Similar for all other nodes

Iterative algorithms must load from disk each iteration

```
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]);
}
```

Low
Arithmetic Intensity!

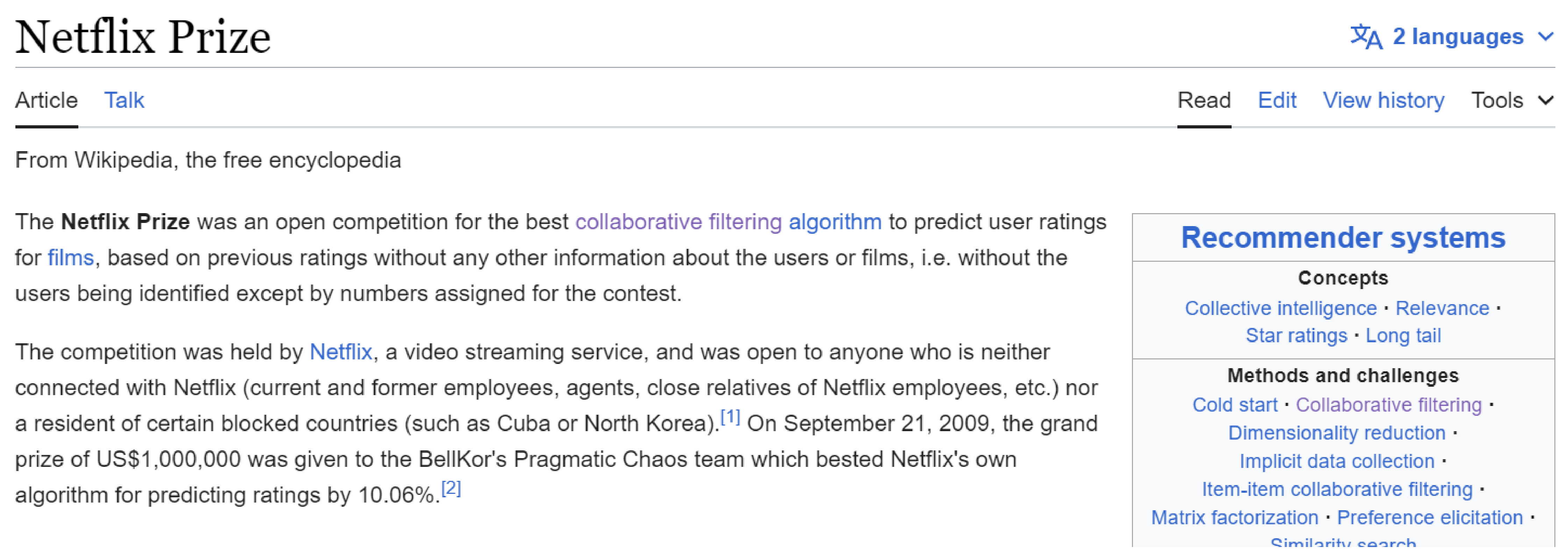
APACHE Spark™

in-memory, fault-tolerant distributed computing

http://spark.apache.org/

Goals

- This guy felt UC Grad student salary too low so he decided to make some money (roughly 1M) via the Netflix challenge.



Netflix Prize

2 languages

Article Talk Read Edit View history Tools

From Wikipedia, the free encyclopedia

The **Netflix Prize** was an open competition for the best collaborative filtering algorithm to predict user ratings for films, based on previous ratings without any other information about the users or films, i.e. without the users being identified except by numbers assigned for the contest.

The competition was held by Netflix, a video streaming service, and was open to anyone who is neither connected with Netflix (current and former employees, agents, close relatives of Netflix employees, etc.) nor a resident of certain blocked countries (such as Cuba or North Korea).[1] On September 21, 2009, the grand prize of US$1,000,000 was given to the BellKor's Pragmatic Chaos team which bested Netflix's own algorithm for predicting ratings by 10.06%.[2]

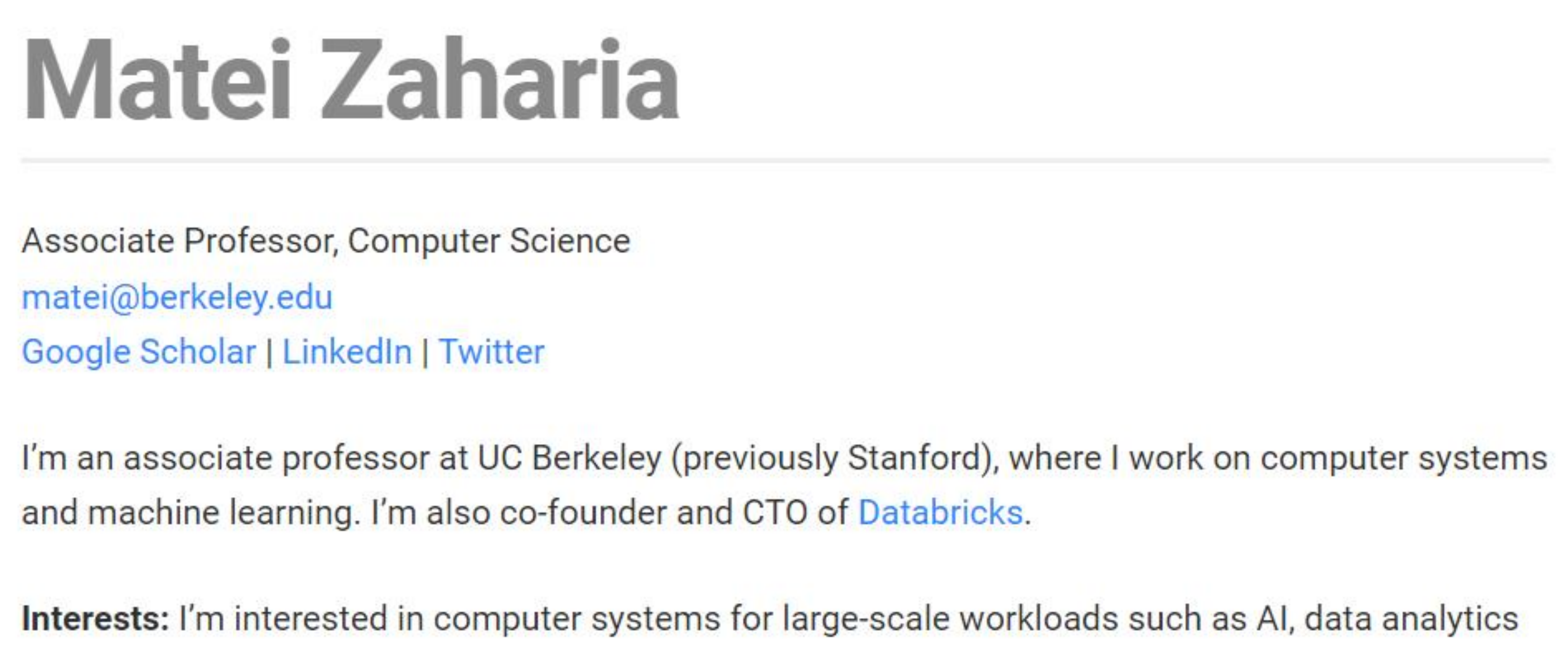
Recommender systems

Concepts

Collective intelligence · Relevance · Star ratings · Long tail

Methods and challenges

Cold start · Collaborative filtering · Dimensionality reduction · Implicit data collection · Item-item collaborative filtering · Matrix factorization · Preference elicitation · Similarity search



Matei Zaharia

Associate Professor, Computer Science
matei@berkeley.edu
Google Scholar | LinkedIn | Twitter

I'm an associate professor at UC Berkeley (previously Stanford), where I work on computer systems and machine learning. I'm also co-founder and CTO of Databricks.

Interests: I'm interested in computer systems for large-scale workloads such as AI, data analytics

Goals

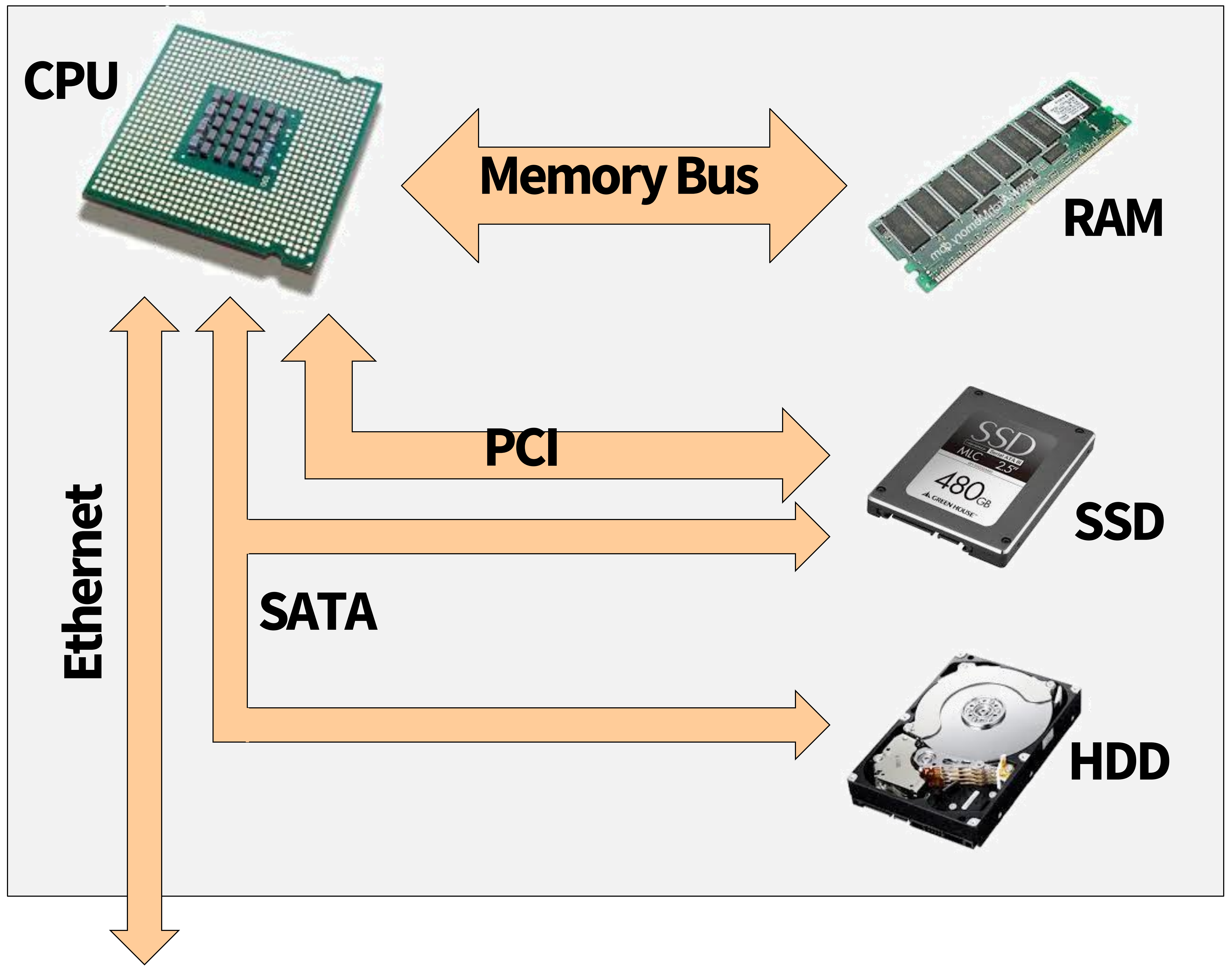
- Programming model for cluster-scale computations where there is **significant reuse** of intermediate datasets
  - Iterative machine learning and graph algorithms
  - Interactive data mining: load large dataset into aggregate memory of cluster and then perform multiple ad-hoc queries

- 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.

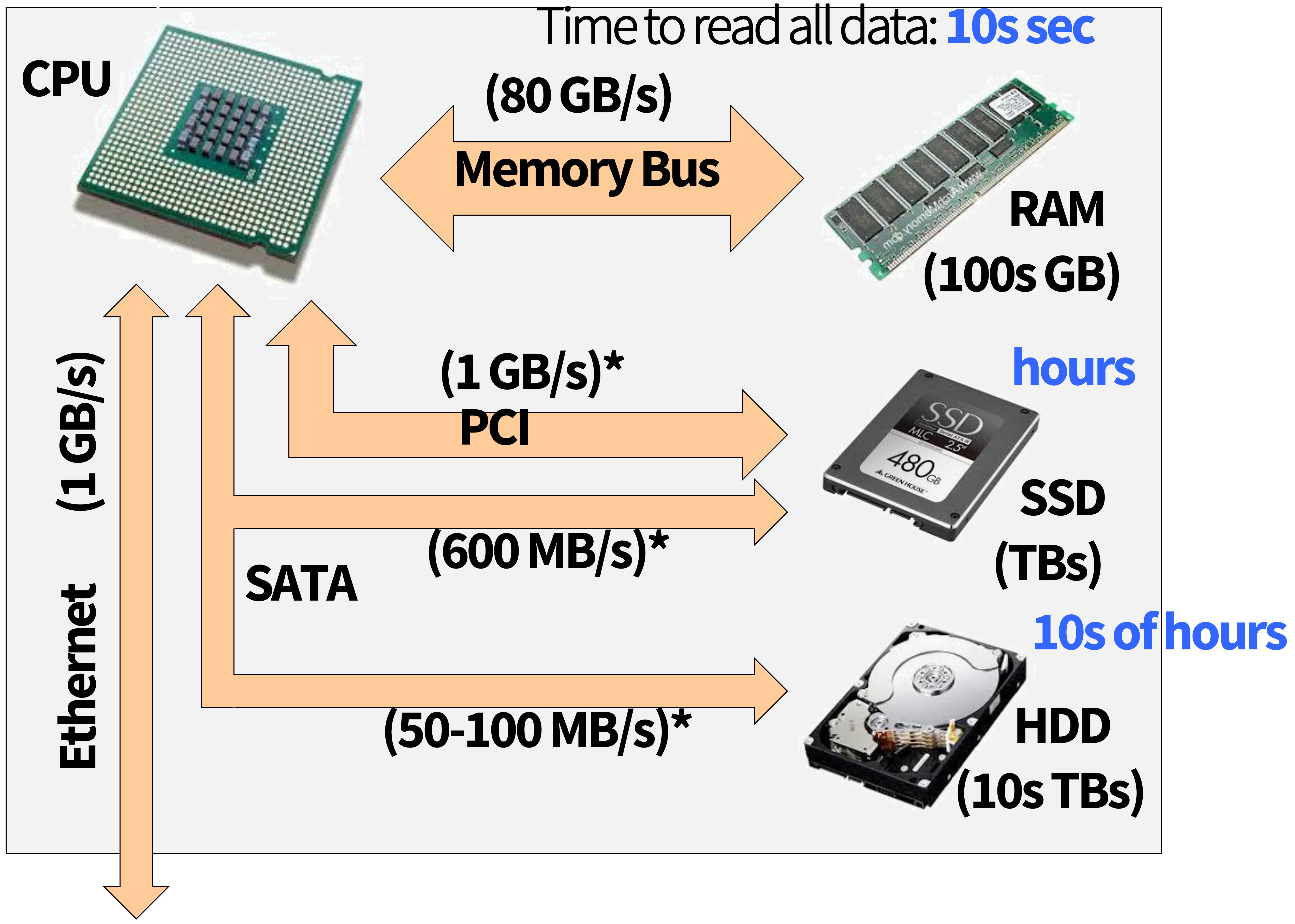
Three Necessary Conditions

- Memory: large (cheap) enough
- Network: fast (cheap) enough
- fault tolerance: at least as good as map-reduce

Typical Server Node

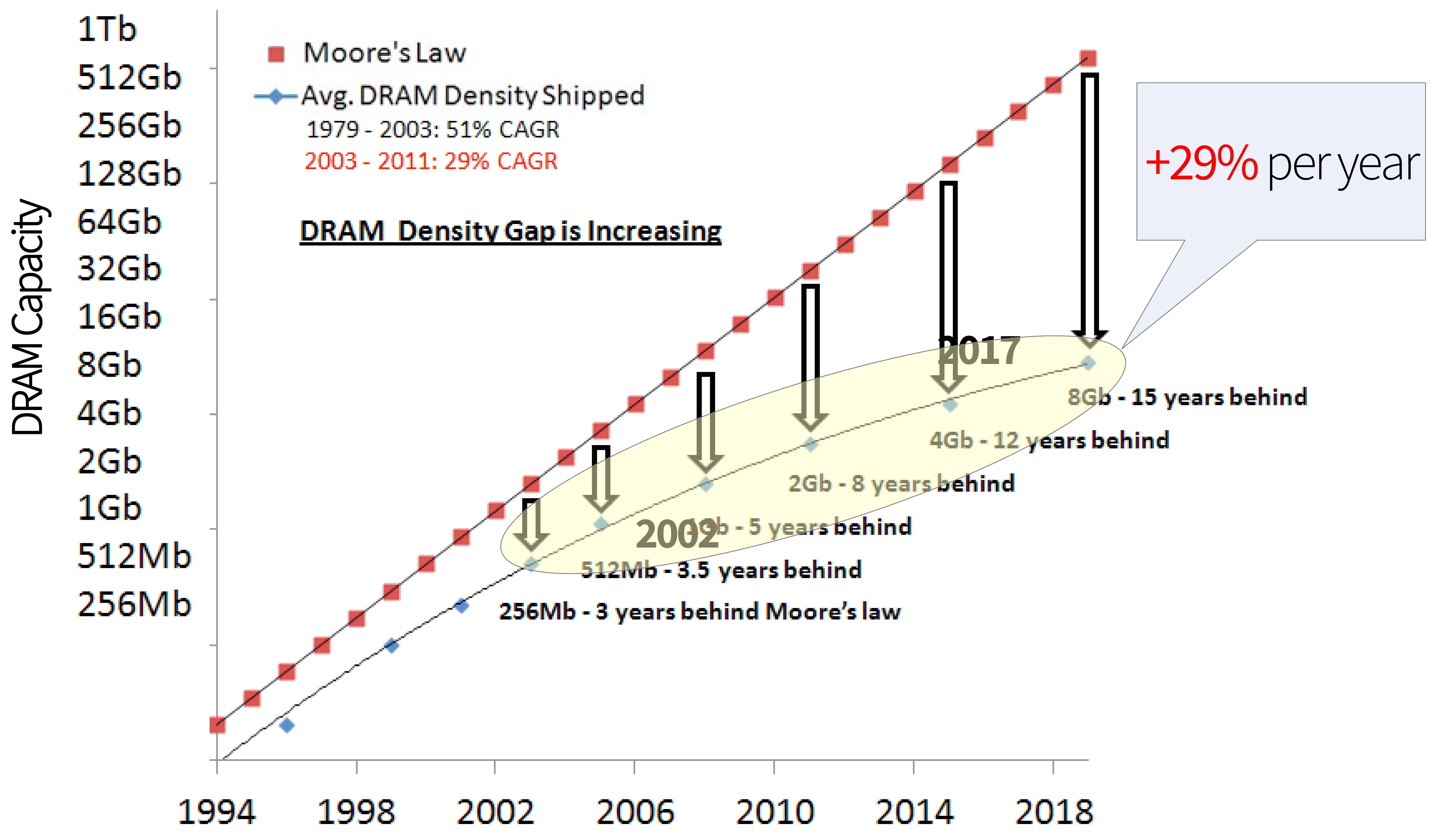


Typical Server Node



* multiple channels

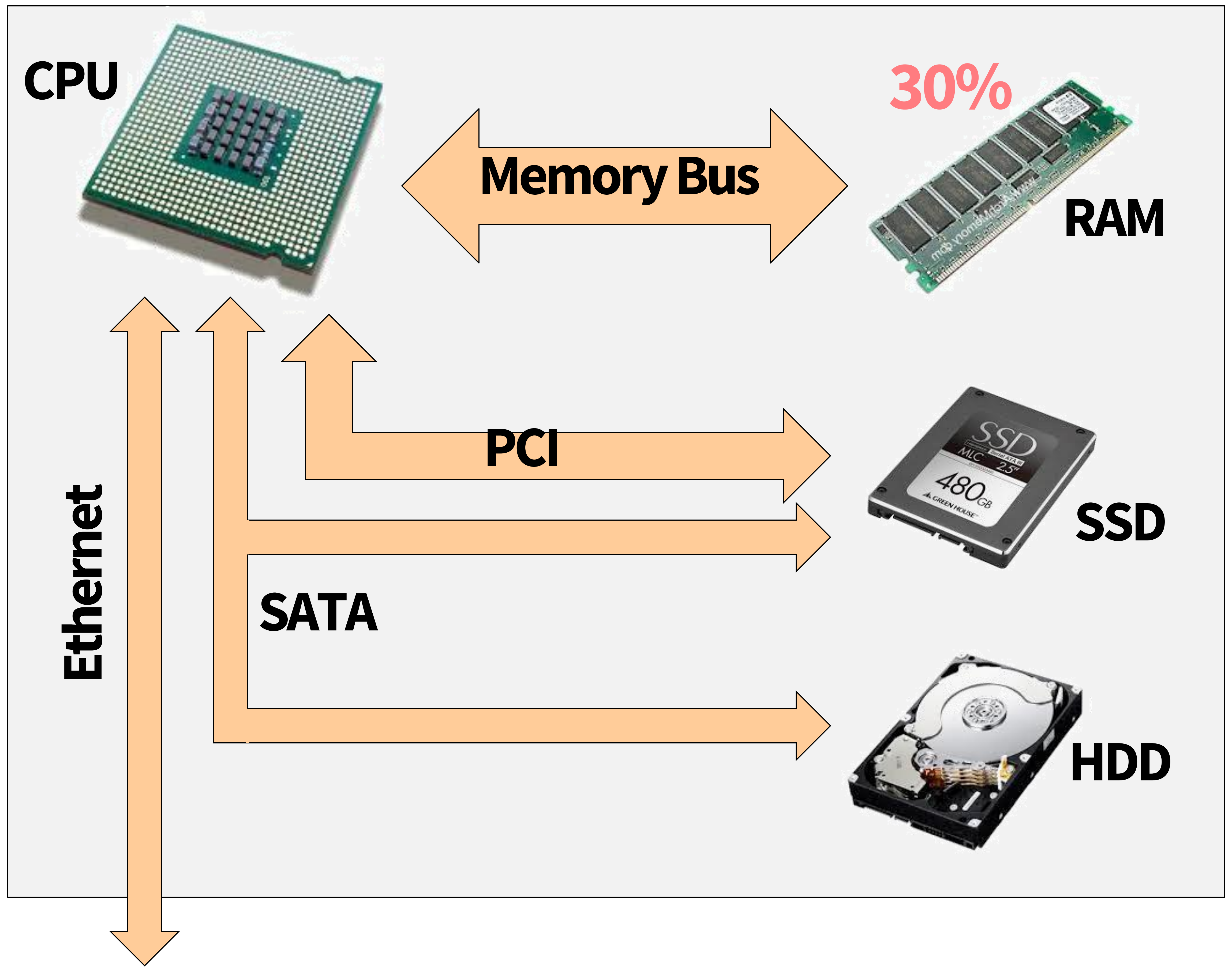
Memory Capacity



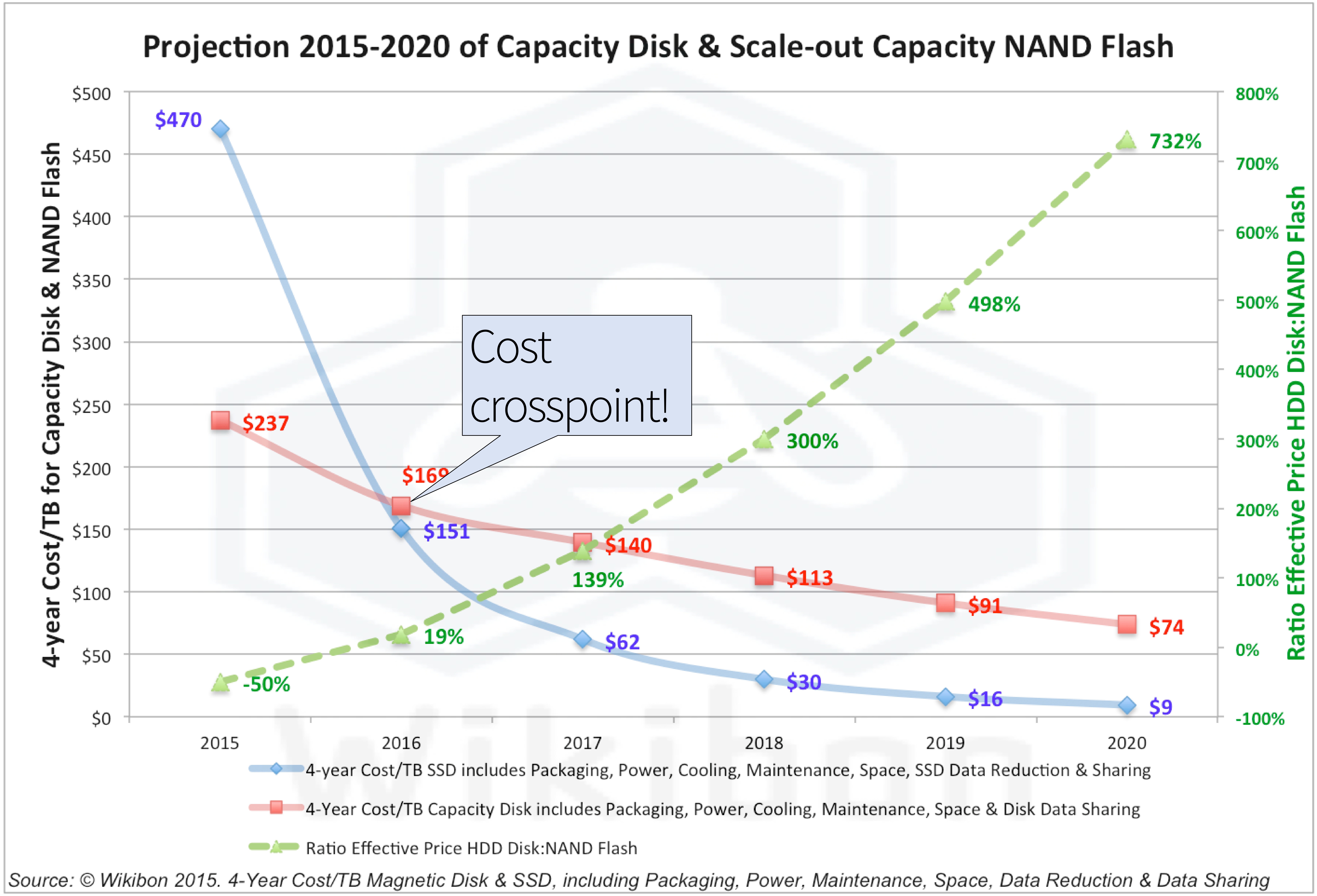
Memory Price/Byte Evolution

- 1990-2000: -54% per year
- 2000-2010: -51% per year
- 2010-2015: -32% per year
- (http://www.jcmit.com/memoryprice.htm)

Typical Server Node



d


Projection 2015-2020 of Capacity Disk & Scale-out Capacity NAND Flash
4-year Cost/TB for Capacity Disk & NAND Flash
Ratio Effective Price HDD Disk:NAND Flash
$500
$450
$400
$350
$300
$250
$200
$150
$100
$50
$0
800%
700%
600%
500%
400%
300%
200%
100%
0%
-100%
2015
2016
2017
2018
2019
2020
$470
$151
$62
$30
$16
$9
$237
$169
$140
$113
$91
$74
-50%
19%
139%
300%
498%
732%
Cost crosspoint!
4-year Cost/TB SSD includes Packaging, Power, Cooling, Maintenance, Space, SSD Data Reduction & Sharing
4-Year Cost/TB Capacity Disk includes Packaging, Power, Cooling, Maintenance, Space & Disk Data Sharing
Ratio Effective Price HDD Disk:NAND Flash
Source: © Wikibon 2015. 4-Year Cost/TB Magnetic Disk & SSD, including Packaging, Power, Maintenance, Space, Data Reduction & Data Sharing

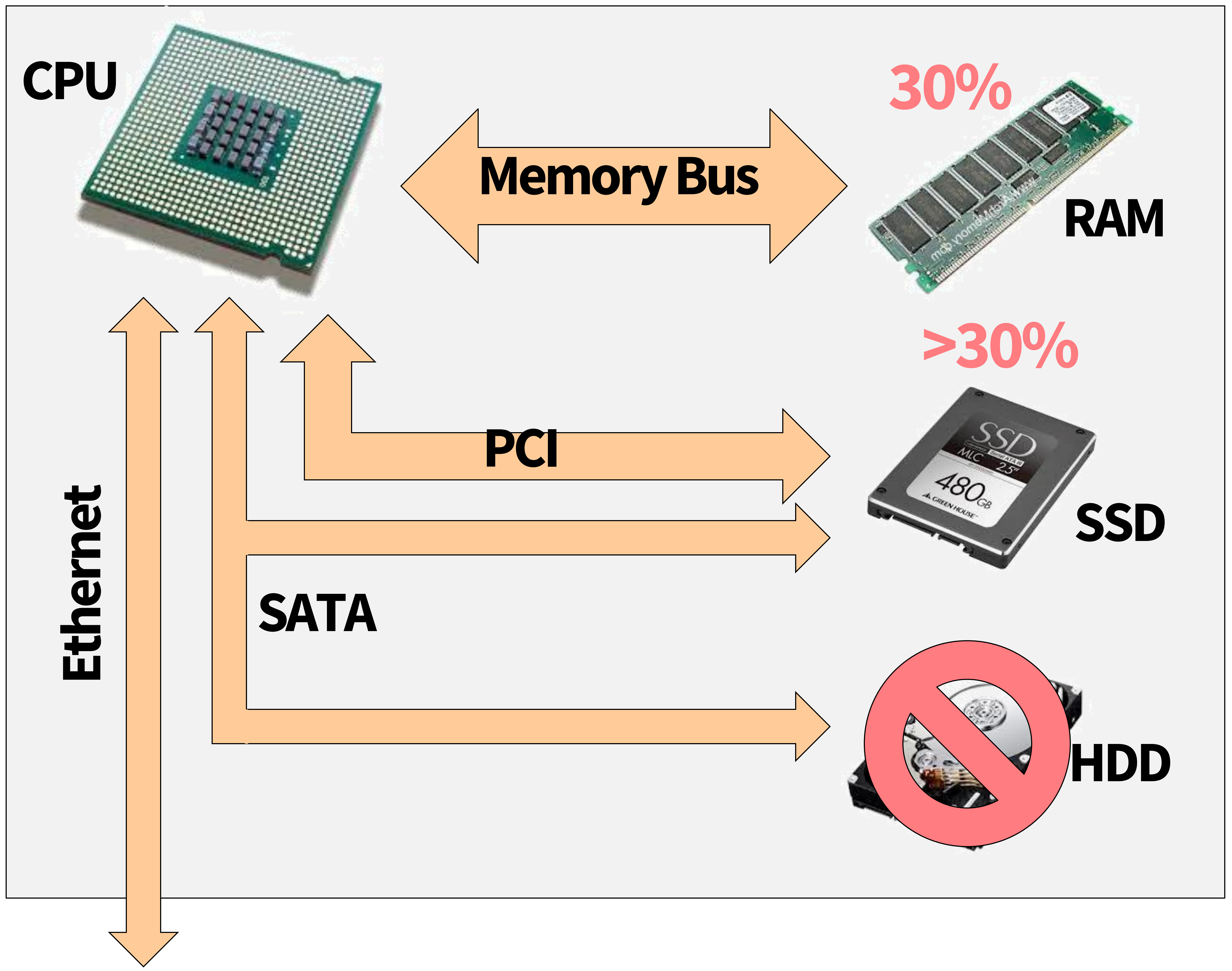
SSDs vs. HDDs

- SSDs has become cheaper than (or as cheap as to) HDDs

- Transition from HDDs to SSDs has accelerate
  - Already most instances in AWS have SSDs
  - Digital Ocean instances are SSD only

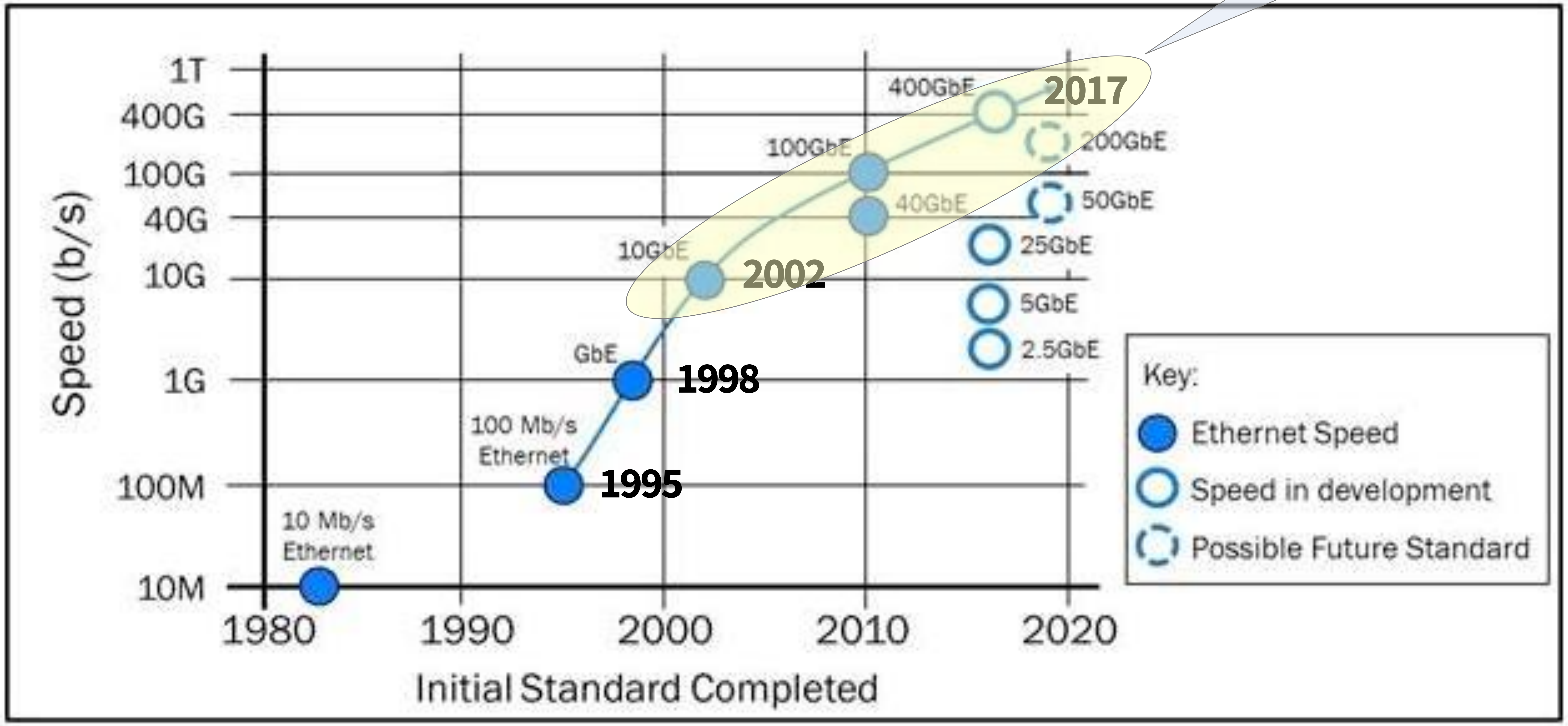
- Going forward we can assume SSD only clusters

Typical Server Node

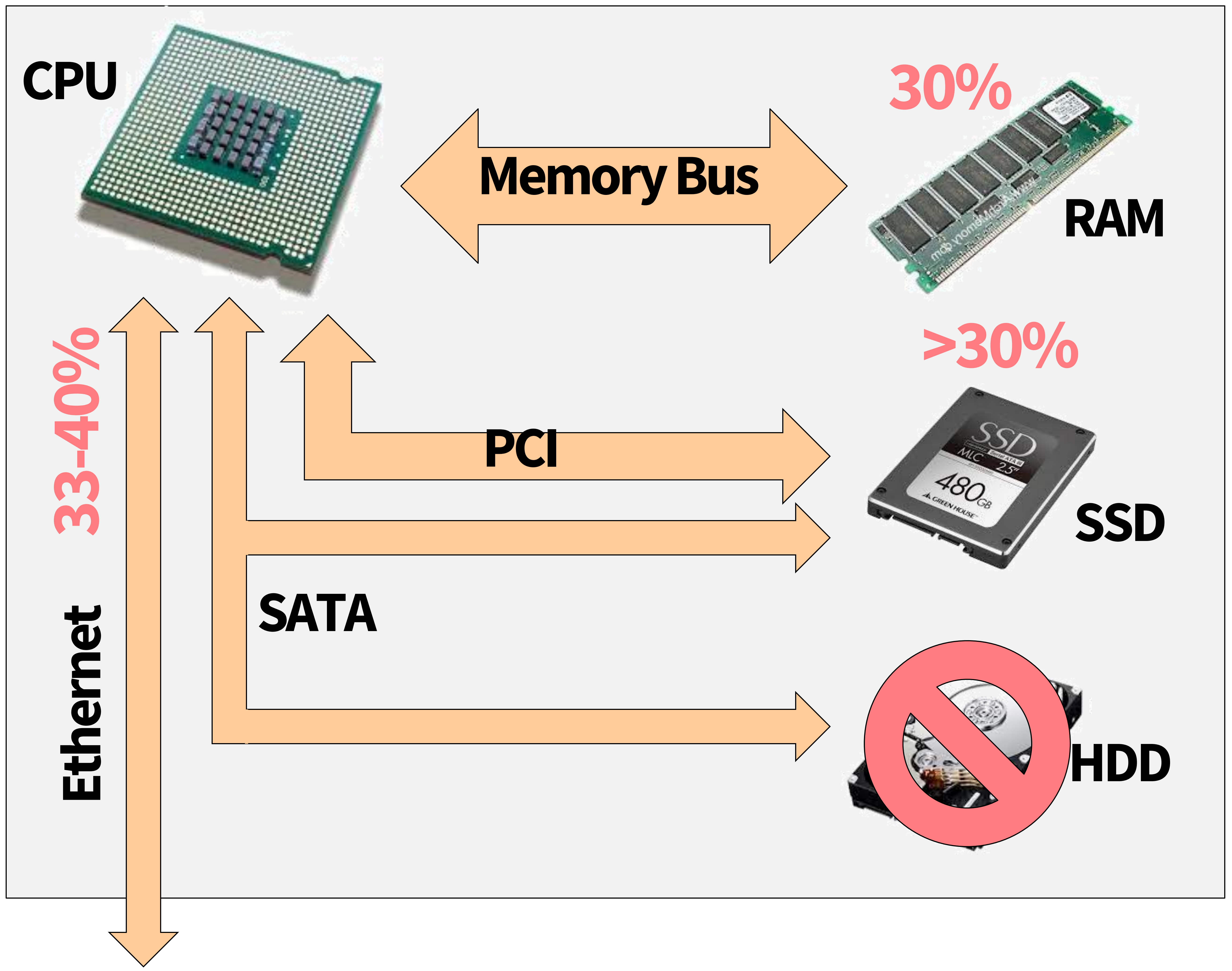


Ethernet Bandwidth

33-40% per year



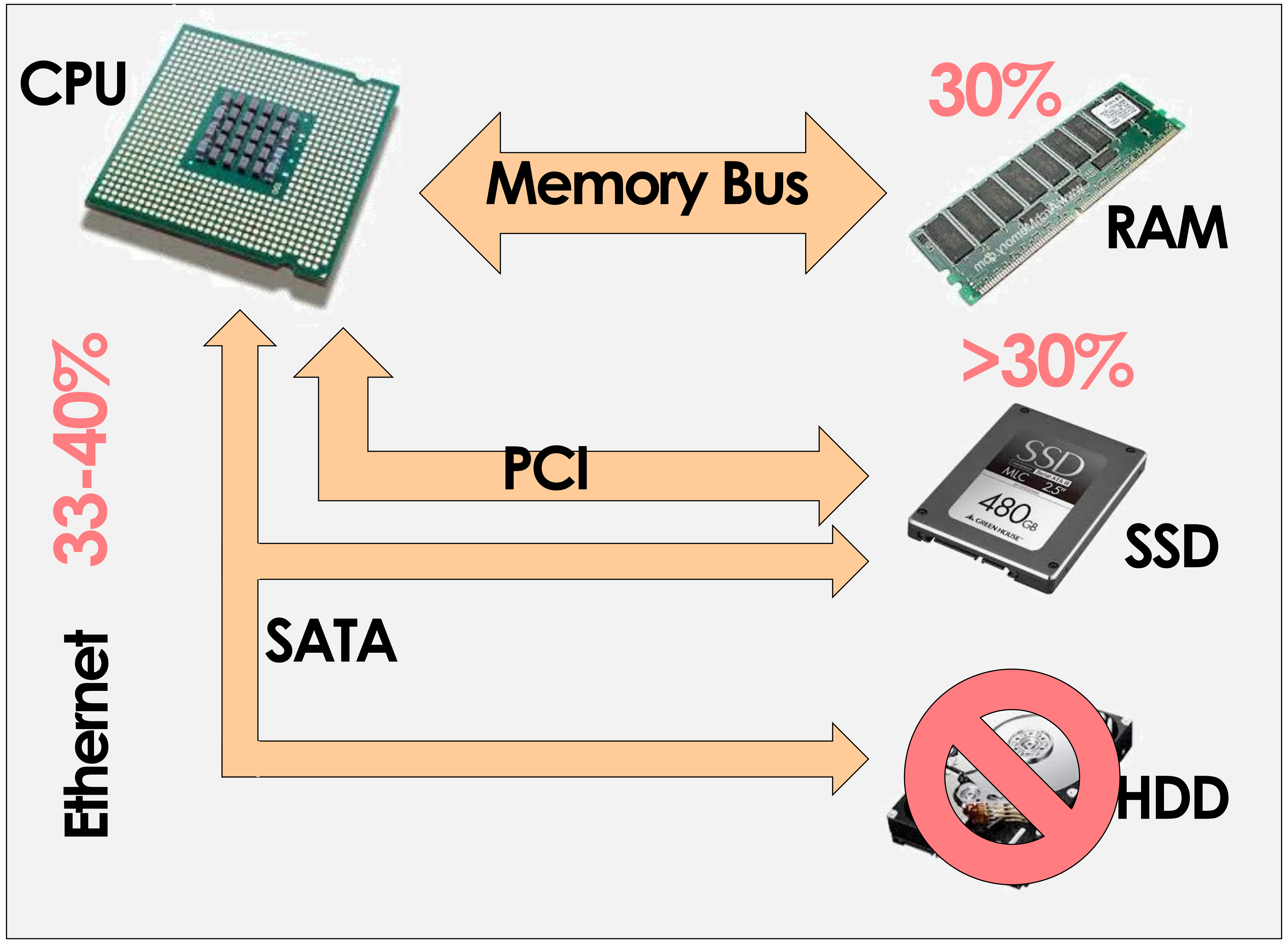
Typical Server Node



What Does This Mean?

- Memory hierarchy has shift one layer up

- HDD is virtually dead
- We have unlimited space of SSD
- Today's RAM space = yesterday's SSD space
- Today's SSD space = yesterday's HDD space
- Ethernet may become faster than PCI/SATA bandwidth



Three Necessary Conditions

- Memory: large (cheap) enough ✓
- Network: fast (cheap) enough ✓
- Fault tolerance: at least as good as map-reduce

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

Resilient distributed dataset (RDD)

Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing

Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, Ion Stoica
University of California, Berkeley

Abstract

We present Resilient Distributed Datasets (RDDs), a distributed memory abstraction that lets programmers perform in-memory computations on large clusters in a fault-tolerant manner. RDDs are motivated by two types of applications that current computing frameworks handle inefficiently: iterative algorithms and interactive data mining tools. In both cases, keeping data in memory can improve performance by an order of magnitude. To achieve fault tolerance efficiently, RDDs provide a restricted form of shared memory, based on coarse-grained transformations rather than fine-grained updates to shared state. However, we show that RDDs are expressive enough to capture a wide class of computations, including recent specialized programming models for iterative jobs, such as Pregel, and new applications that these models do not capture. We have implemented RDDs in a system called Spark, which we evaluate through a variety of user applications and benchmarks.

1 Introduction

Cluster computing frameworks like MapReduce [10] and Dryad [19] have been widely adopted for large-scale data analytics. These systems let users write parallel computations using a set of high-level operators, without having to worry about work distribution and fault tolerance.

Although current frameworks provide numerous abstractions for accessing a cluster's computational resources, they lack abstractions for leveraging distributed memory. This makes them inefficient for an important class of emerging applications: those that *reuse* intermediate results across multiple computations. Data reuse is common in many *iterative* machine learning and graph algorithms, including PageRank, K-means clustering, and logistic regression. Another compelling use case is *interactive* data mining, where a user runs multiple ad-hoc queries on the same subset of the data. Unfortunately, in most current frameworks, the only way to reuse data between computations (*e.g.,* between two MapReduce jobs) is to write it to an external stable storage system, which can dominate application execution times.

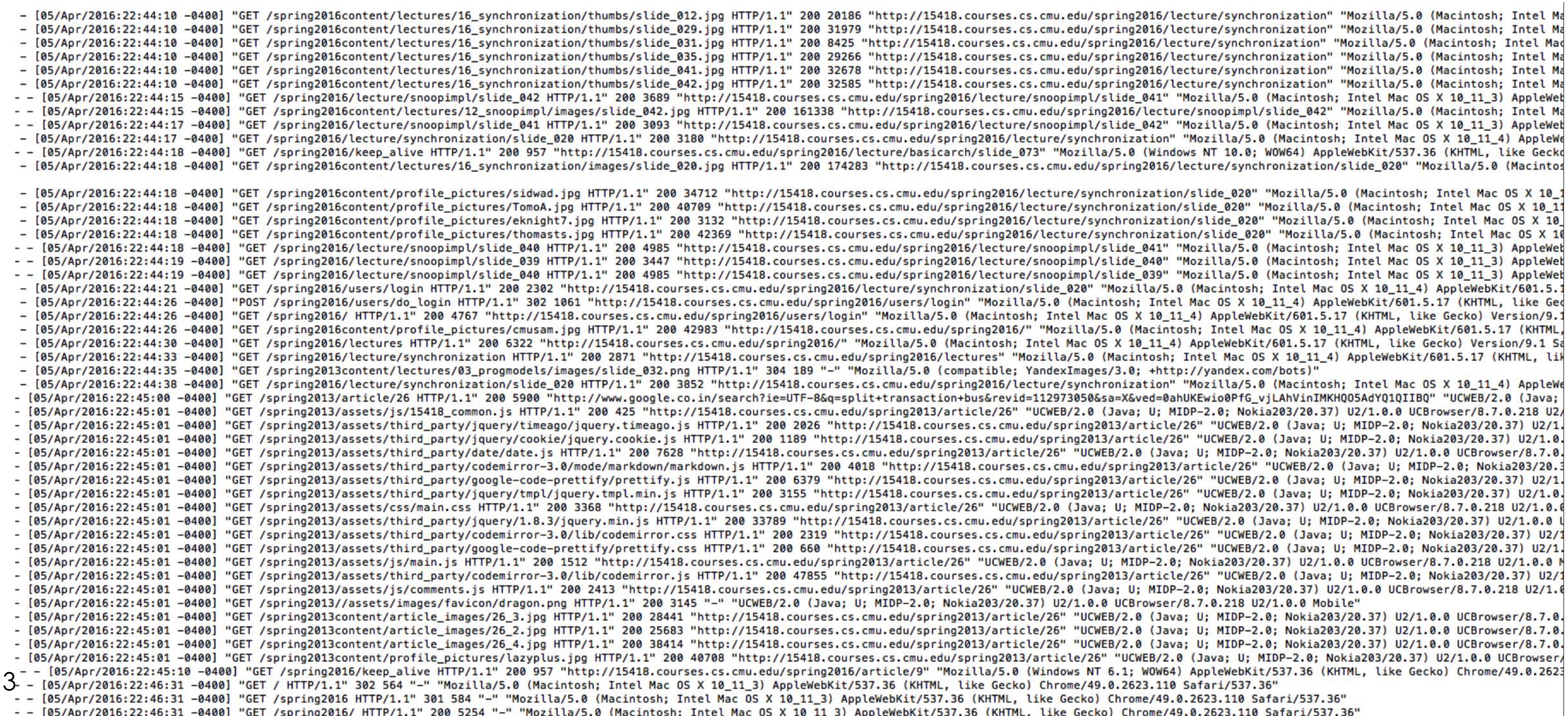
Recognizing this problem, researchers have developed specialized frameworks for some applications that require data reuse. For example, Pregel [22] is a system for iterative graph computations that keeps intermediate data in memory, while HaLoop [7] offers an iterative MapReduce interface. However, these frameworks only support specific computation patterns (*e.g.,* looping a series of MapReduce steps), and perform data sharing implicitly for these patterns. They do not provide abstractions for more general reuse, *e.g.,* to let a user load several datasets into memory and run ad-hoc queries across them.

In this paper, we propose a new abstraction called *resilient distributed datasets (RDDs)* that enables efficient data reuse in a broad range of applications. RDDs are fault-tolerant, parallel data structures that let users explicitly persist intermediate results in memory, control their partitioning to optimize data placement, and manipulate them using a rich set of operators.

The main challenge in designing RDDs is defining a programming interface that can provide fault tolerance *efficiently*. Existing abstractions for in-memory storage on clusters, such as distributed shared memory [24], key-value stores [25], databases, and Piccolo [27], offer an interface based on fine-grained updates to mutable state (*e.g.,* cells in a table). With this interface, the only ways to provide fault tolerance are to replicate the data across machines or to log updates across machines. Both approaches are expensive for data-intensive workloads, as they require copying large amounts of data over the cluster network, whose bandwidth is far lower than that of RAM, and they incur substantial storage overhead.

In contrast to these systems, RDDs provide an interface based on *coarse-grained* transformations (*e.g.,* map, filter and join) that apply the same operation to many data items. This allows them to efficiently provide fault tolerance by logging the transformations used to build a dataset (its *lineage*) rather than the actual data.[1] If a partition of an RDD is lost, the RDD has enough information about how it was derived from other RDDs to recompute

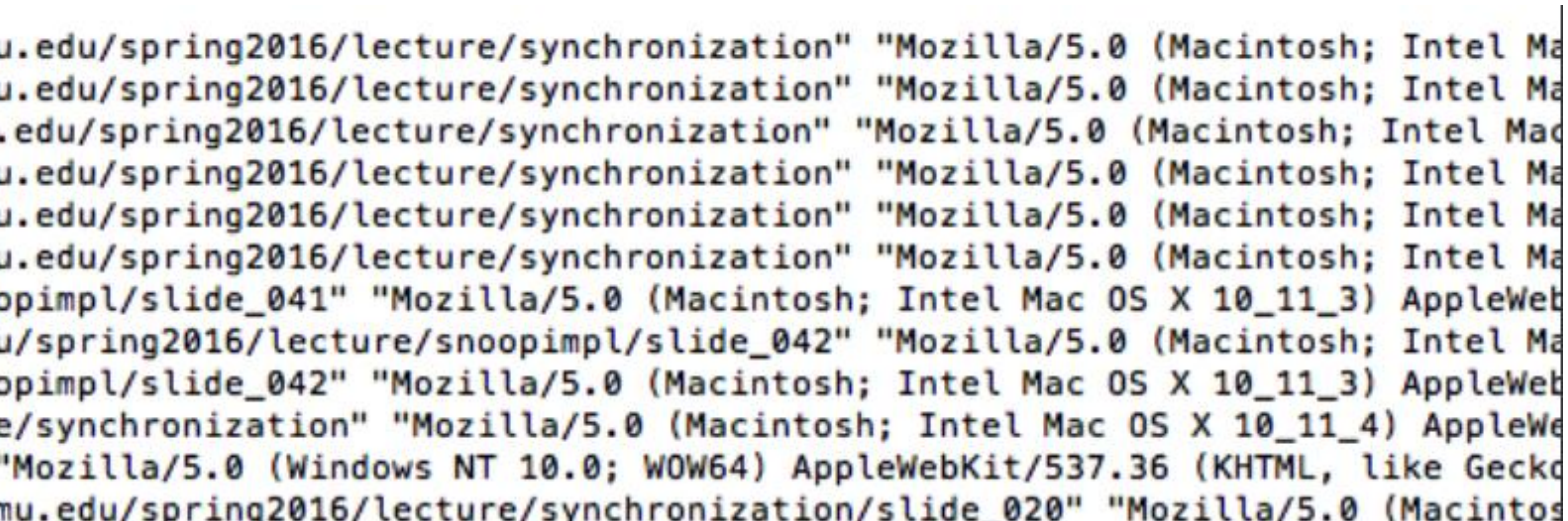
A log of page views on a web site



```
 - [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_1
 - [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; Nokia203/20.3
- [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.
- [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"
```

Example query:

“What type of mobile phone are all the visitors using?”



```
u.edu/spring2016/lecture/synchronization" "Mozilla/5.0 (Macintosh; Intel Ma
u.edu/spring2016/lecture/synchronization" "Mozilla/5.0 (Macintosh; Intel Ma
.edu/spring2016/lecture/synchronization" "Mozilla/5.0 (Macintosh; Intel Mac
u.edu/spring2016/lecture/synchronization" "Mozilla/5.0 (Macintosh; Intel Ma
u.edu/spring2016/lecture/synchronization" "Mozilla/5.0 (Macintosh; Intel Ma
u.edu/spring2016/lecture/synchronization" "Mozilla/5.0 (Macintosh; Intel Ma
opimpl/slide_041" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_3) AppleWeb
u/spring2016/lecture/snoopimpl/slide_042" "Mozilla/5.0 (Macintosh; Intel Ma
opimpl/slide_042" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_3) AppleWeb
e/synchronization" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_4) AppleWe
"Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Geck
mu.edu/spring2016/lecture/synchronization/slide_020" "Mozilla/5.0 (Macintos
```

Using MapReduce

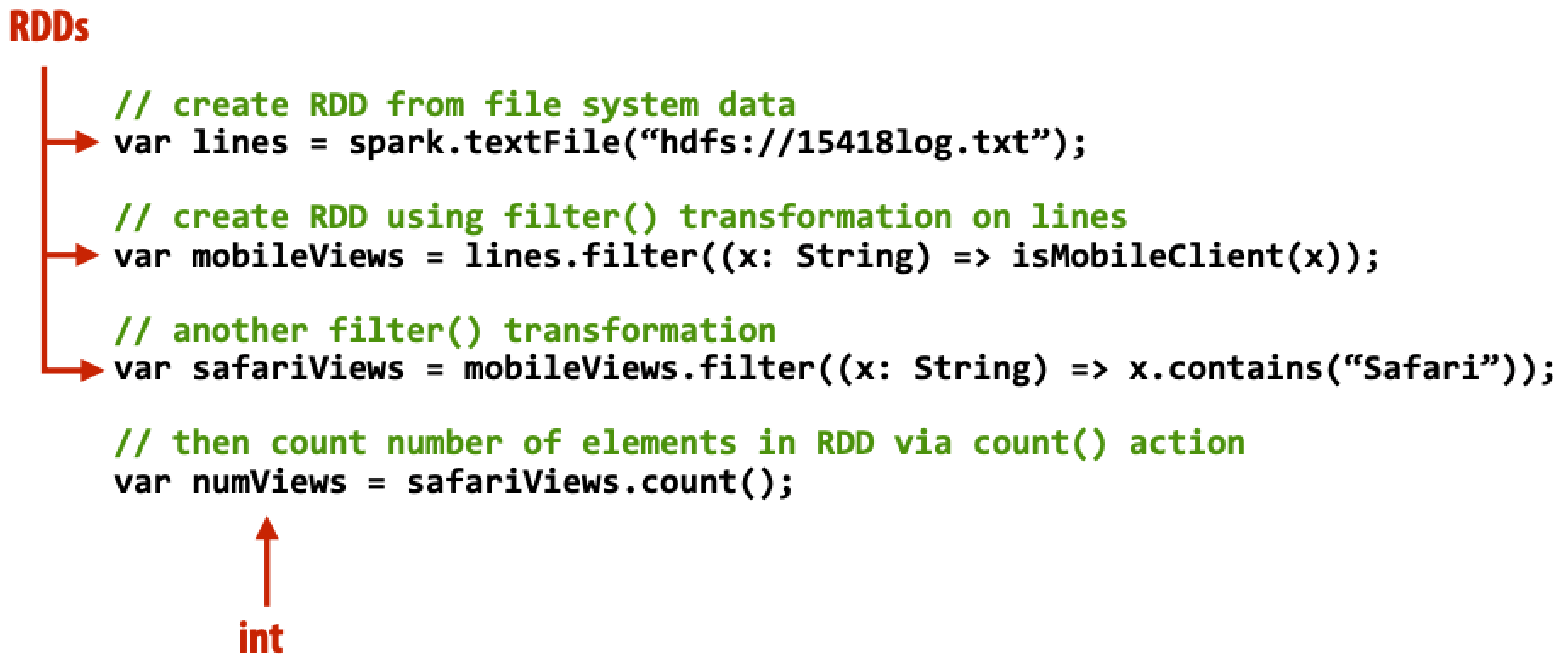
```
// called once per line in input file by runtime
// input: string (line of input file)
// output: adds (user_agent, 1) entry to list
void mapper(string line, multimap<string,string>& results) {
   string user_agent = parse_requester_user_agent(line);
   if (is_mobile_client(user_agent))
     results.add(user_agent,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

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

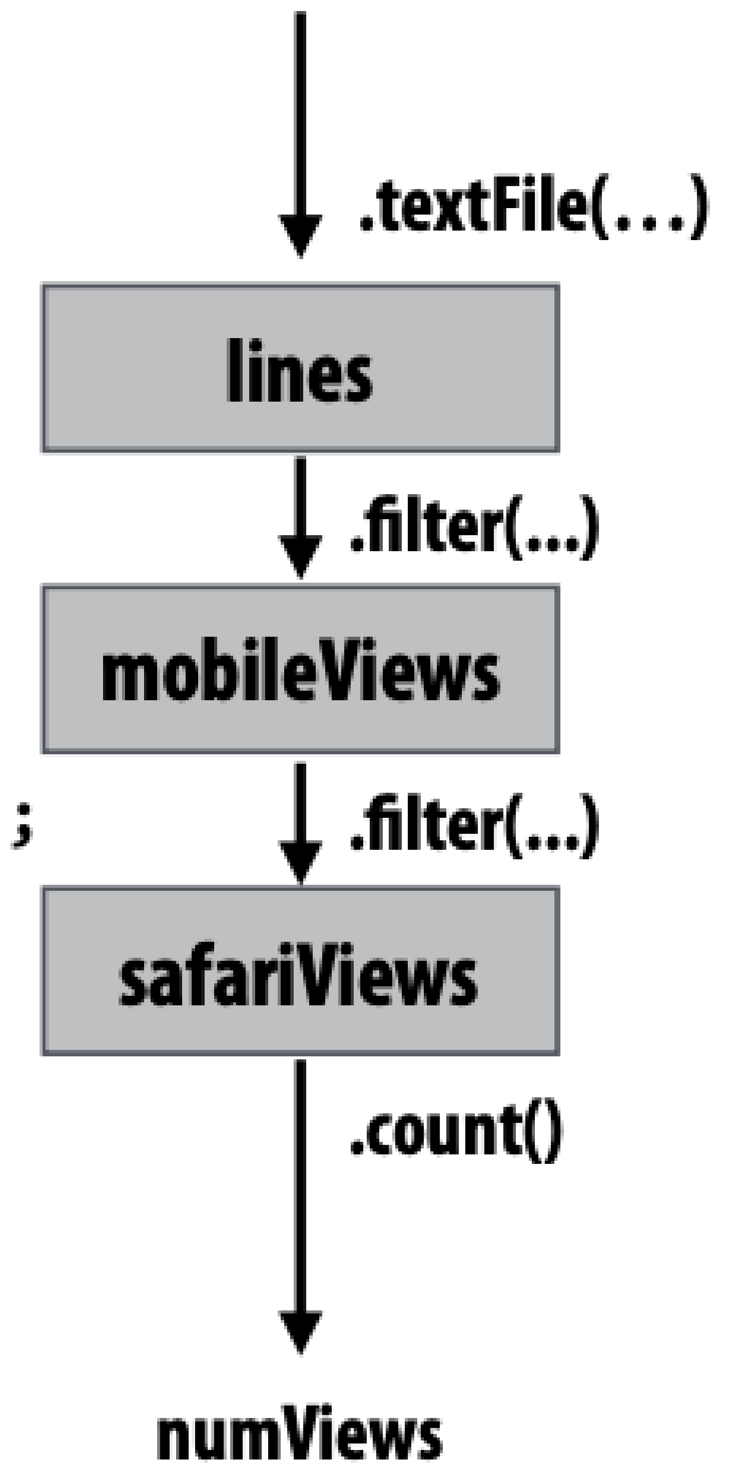
The code left computes the count of page views by each type of mobile phone.

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



```
// 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();
```


.textFile(...)
lines
.filter(...)
mobileViews
.filter(...)
safariViews
.count()
numViews