18-847F: Special Topics in Computer Systems

Foundations of Cloud and Machine Learning Infrastructure

Lecture 1: Logistics and Overview

Foundations of Cloud and Machine Learning Infrastructure

Graduate Seminar Class

Few Lectures

Reading research papers

Student presentations

Class Discussions

Final Research Project (No Exams!)

Learning Objectives

- Know the state-of-the-art frameworks in cloud and machine learning and their theoretical foundations
- Read and provide constructive criticism of research papers
- Present to an audience, and answer their questions
- Do creative, collaborate research

Why study Cloud and ML infrastructure?

What are the largest words after ‘Big Data’?

Big Data Gold Rush

Who got rich in the California gold rush?

Big Data Gold Rush

Google Compute Engine

amazon web services

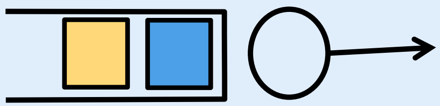
IBM Bluemix™

Who got rich in the California gold rush?

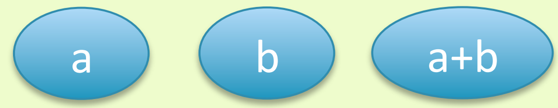
In the Big Data rush, it's the infrastructure companies

Topics Covered

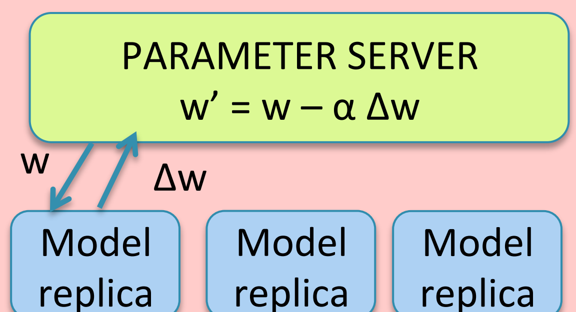
Cloud Computing



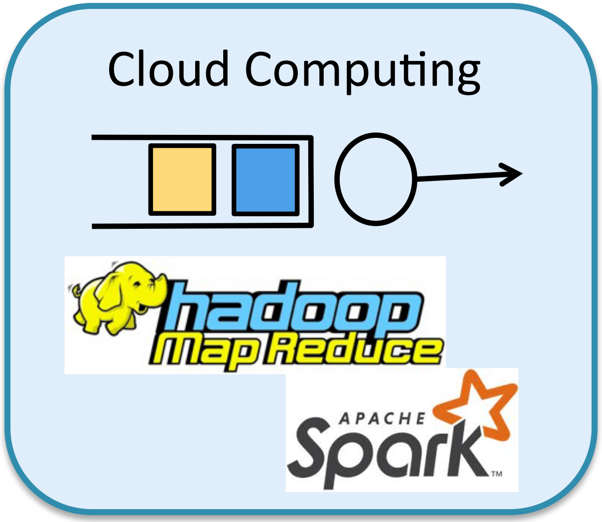
Distributed Storage



Machine Learning



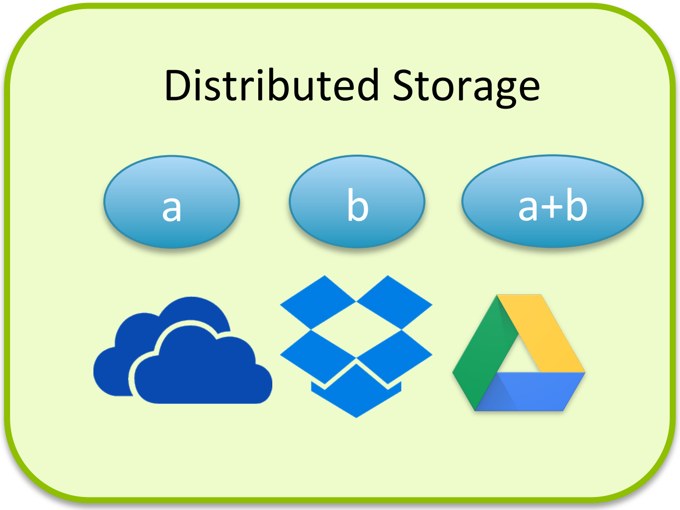
Topics Covered



- Scheduling in Parallel Computing
  - MapReduce, Spark
  - Straggler Replication
- Task Replication in Queueing Systems

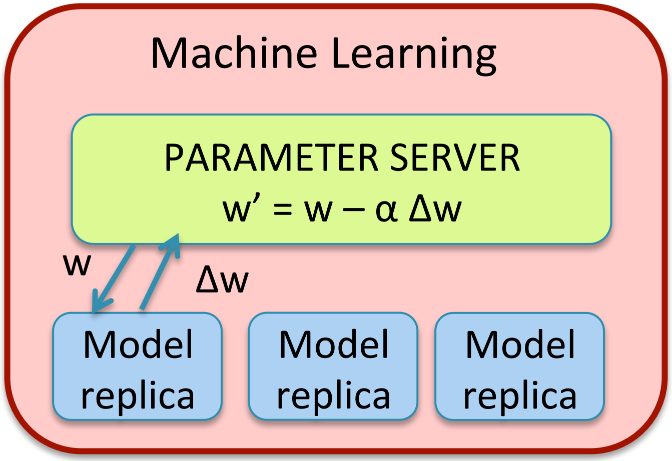
Topics Covered

- Coding for locality/repair
- Systems implementation of codes
- Reducing latency in content download



Topics Covered

- SGD and its convergence
- Distributed Deep Learning
- Hyper-parameter tuning
- GANs, Deep reinforcement learning



Instructor: Gauri Joshi

B.Tech+M.Tech
2005-2010

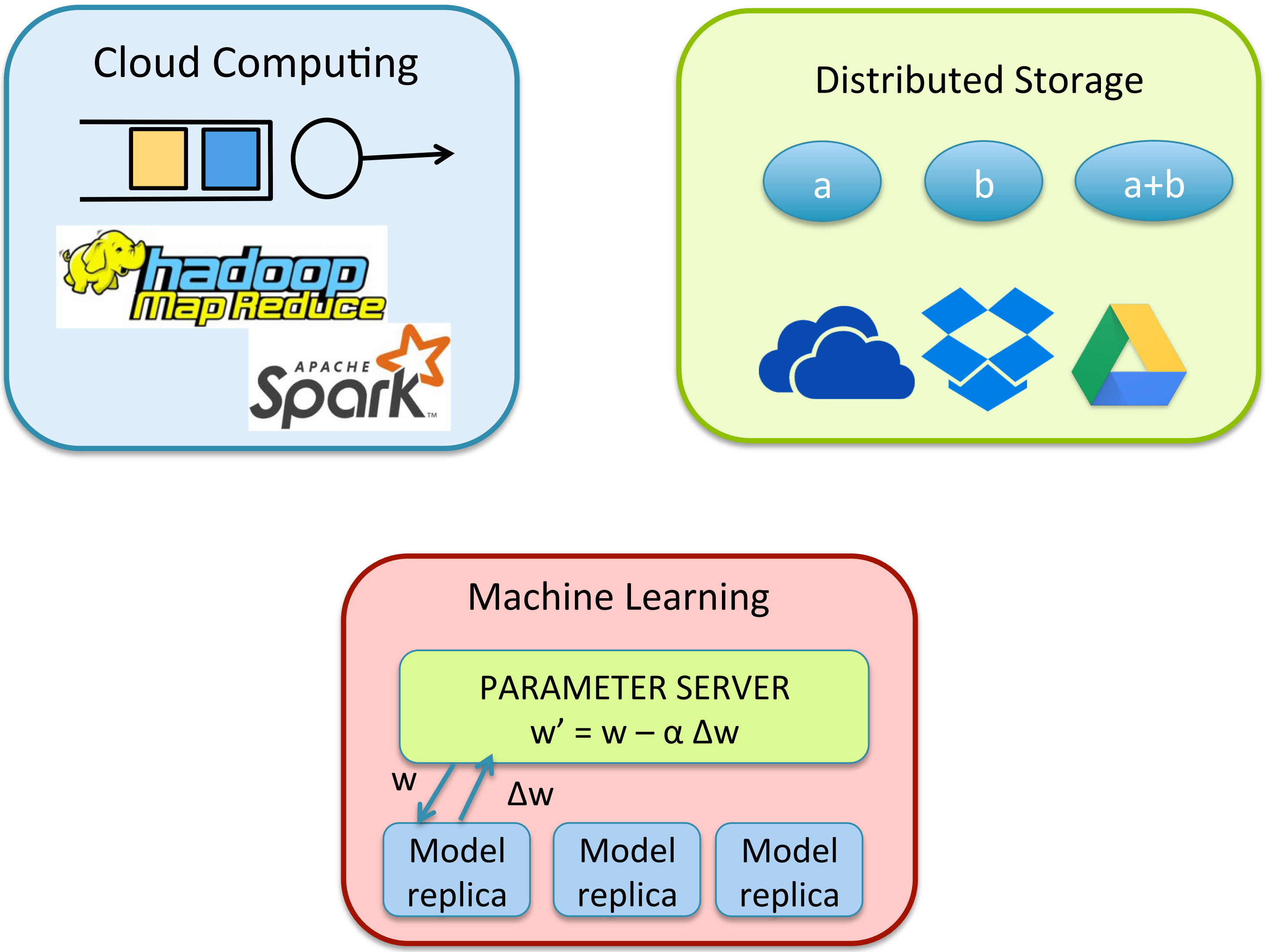
SM + PhD
2010-2016

Research Staff Member
2016-2017

Assistant Professor
Fall 2017 -

Internships

Have worked in all these areas



Student Introductions

- Name?
- Department?
- Undergrad/Masters/PhD?
- Previous related classes (if any)?
- What you are looking to learn from this class?

Waiting list will be cleared soon!

Class Hours and Website(s)

- When: Mon, Wed 4:30-6:00 pm
- Where: Scaife Hall 222

- Class Website (Readings, Schedule): https://www.andrew.cmu.edu/user/gaurij/18-847F-Fall-2018.html
- Canvas Site (Readings, Assignments, Projects): https://canvas.cmu.edu/

- No prerequisites. Basic knowledge of probability and linear algebra is encouraged.

Reading Material

Papers will be posted on the class website or on Canvas

- Book chapters
- Survey papers
- Theory papers (Scheduling, Queuing, Coding, Optimization)
- Systems papers (Cloud, Machine Learning)

Additional reference books listed in the syllabus

Instructor/TA and Office Hours

Instructor: Prof. Gauri Joshi (gaurij [AT]andrew.cmu.edu)

TA: Jianyu Wang (jianyuw1 [AT]andrew.cmu.edu)

Office Location: CIC 4105

Office Hours: By appointment

Graduate Seminar Class

A few lectures

Reading research papers

Student presentations

Class Discussions

Final Research Project

Lectures

- Next week: Deeper Overview of probability and queuing theory
- Guest lectures during the semester by authors of papers relevant to this class

Graduate Seminar Class

A few lectures

Reading research papers

Student presentations

Class Discussions

Final Research Project

Homeworks (~50%)

- Submit paper review (due 10:00 am before class)
  - ~Two reviews per week
- Discussion with classmates is okay, but write reviews in your own words.

Paper Review Format

- Summary of the paper
  - Reflects your understanding of the paper
  - Significance & correctness of results
- Discussion Questions for Class (at least 2)
  - Confusions about the paper, open research directions
- Answers to concept-check questions

Homework Grading Rubric (Total: 10 pts)

- Understanding of the paper (4 pts)
- Discussion Questions (3 pts)
- Concept-check questions (3 pts)

Graduate Seminar Class

A few lectures

Reading research papers

Student presentations

Class Discussions

Final Research Project

Class Presentations (~15%)

- Sign up for presentation at least 1 week in advance
- Each student will present 1-2 times in the semester (depends on # of students registered)
- 20 min presentation, followed by 25 min discussion
  - Motivation and Related work
  - Summary of main results
  - Your views on the paper

Presentation Grading Rubric (Total: 10 pts)

- Motivation (1.5 pts)
- Clarity (1.5 pts)
- Understanding/Correctness (4 pts)
- Peer-review Feedback (3 pts)

Graduate Seminar Class

A few lectures

Reading research papers

Student presentations

Class Discussions

Final Research Project

Class Participation (~15%)

- The class will be divided into groups of 3-4 students each

- Each group will discuss one of the discussion questions among themselves

- Summarize the discussion to the whole class

Participation Grading Rubric (Total: 5 pts)

- Attendance and attention (1.5 pt)
- Speaking up in class (1.5 pt)
- Insightful Questions/Comments (2 pt)

Graduate Seminar Class

A few lectures

Reading research papers

Student presentations

Class Discussions

Final Research Project

Research Project (~20%)

- Groups of 1-3

- Original research on a topic of your choice
  - Topics aligned with your research allowed and encouraged
  - If you can't think of topics, come talk to Jianyu or me

- Possible Project Types:
  - New theoretical analysis
  - Implementation using one of the frameworks discussed
  - In-depth literature survey of a particular topic

Timeline

- 1-page proposal due Oct 3
- Publishable quality report (max 5 pg) in ACM format
  - Initial draft due: Nov 21
  - Final report due: Dec 7
- Last week of class: Presentations (~30 min per group)
- Peer-review other presentations

Project Grading Rubric (Total: 20 pts)

- Originality (1 pts)
- Review of Related Work (1.5 pts)
- Writing and Organization (1.5 pts)
- Technical Results (4 pts)
- Final presentation (10 pts)
- Peer-Review (2 pts)

In Summary..

- Paper Reading
- Submitting Reviews
- Class Presentations
- Final Project

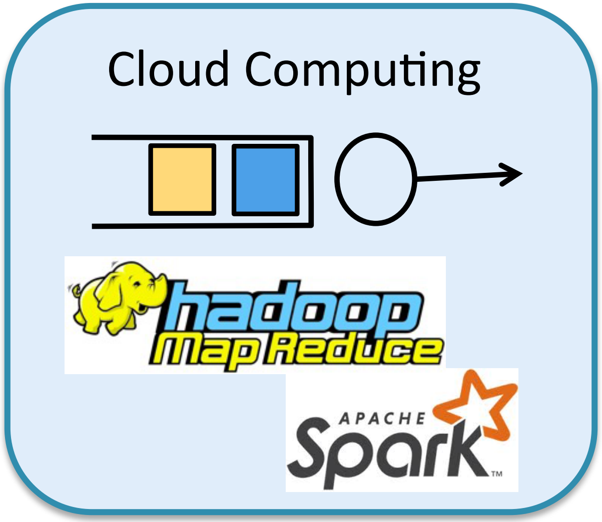
Might seem like a lot of work but..

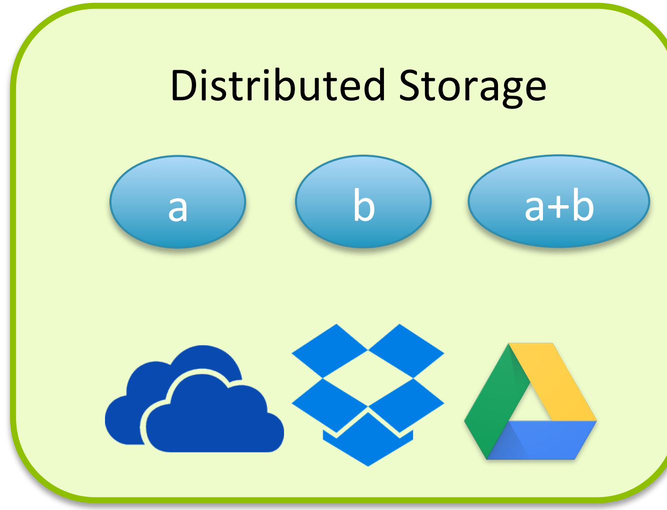
- You will get fast and efficient at reading papers
- The project will be a fun, collaborative exercise
- No exams!

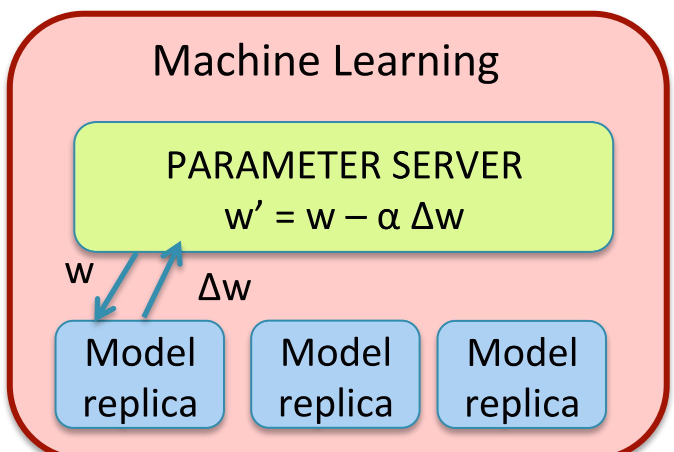
TO DO

- Fill out the sign-up sheet
- Sign-up for presentations
- Start reading the papers
- Form groups for class projects
- Start thinking about projects

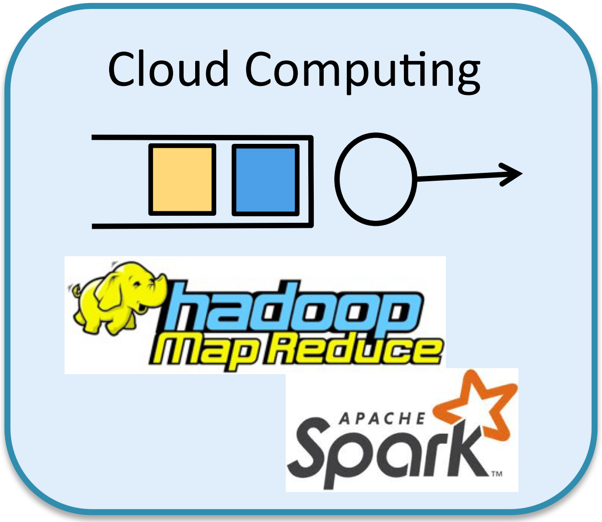
Topics Covered

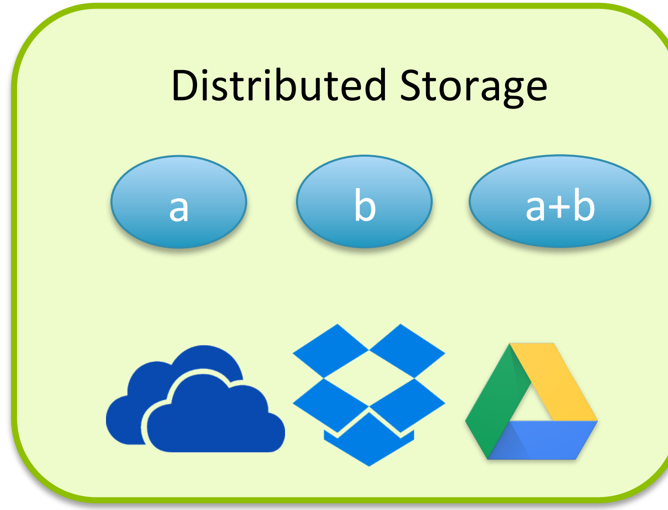




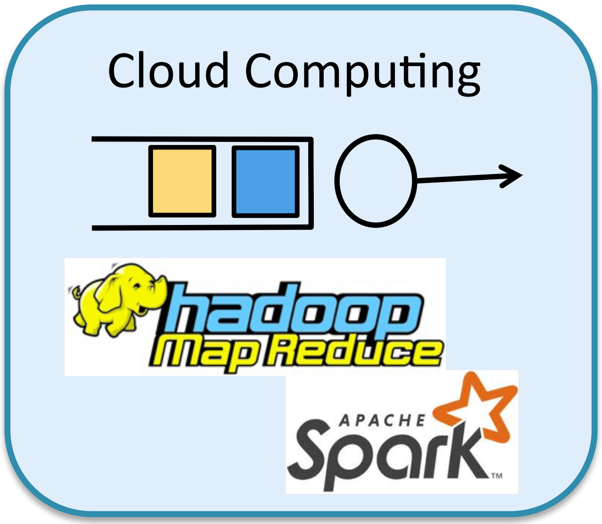


History and Overview





History and Overview



- MapReduce, Spark
- Scheduling in Parallel Computing
- Straggler Replication
- Task Replication in Queueing Systems

What is the cloud?

A collection of servers that can function as a single computing node, and can be accessed from multiple devices

1960's: The Mainframe Era

- Large, expensive machines
- Only one per university/institution

IBM 704 (1964)

1970's: Virtualization

- IBM released a VM OS that allowed multiple users to share the mainframe computer

IBM 704 (1964)

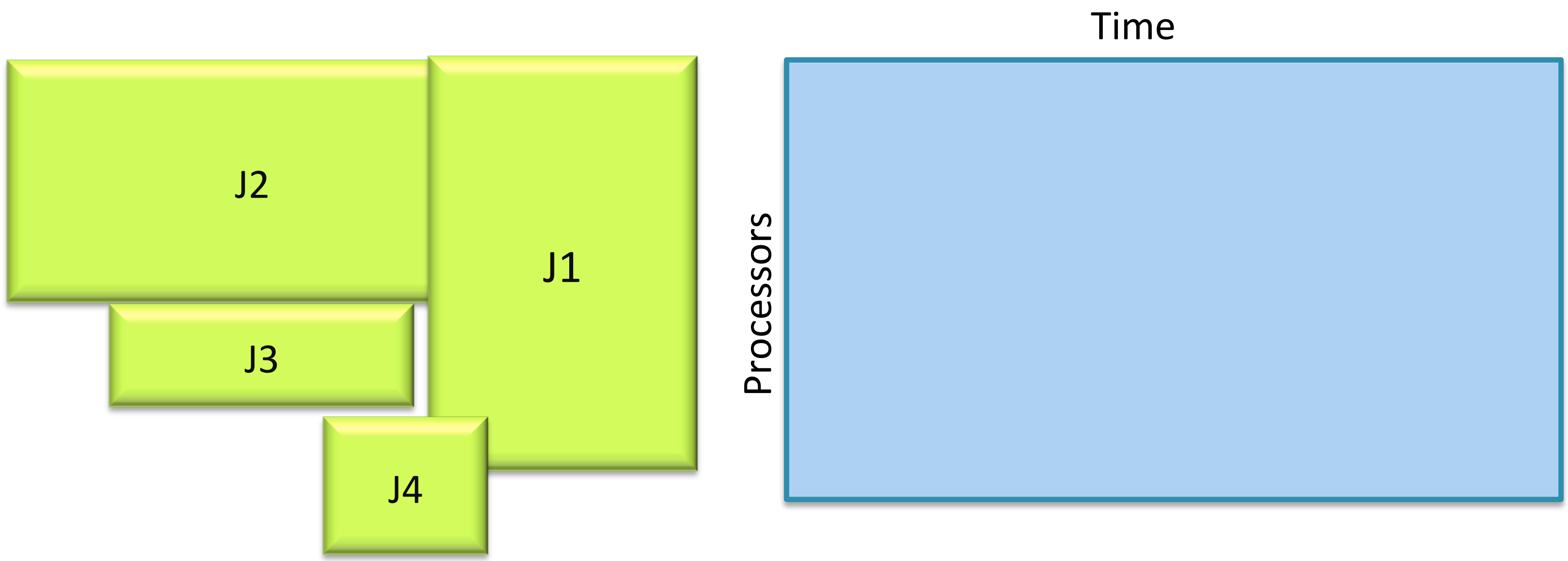
1980’s-1990’s: Internet and PCs

- PCs become affordable
- Internet connectivity went on improving

- Virtual Private Networks (VPNs)
- Grid Computing: Connect cheap PCs via the Internet
- On the theory side, queuing theory, traditionally used in operations management rebounded

1990's: Scheduling in Parallel Computing

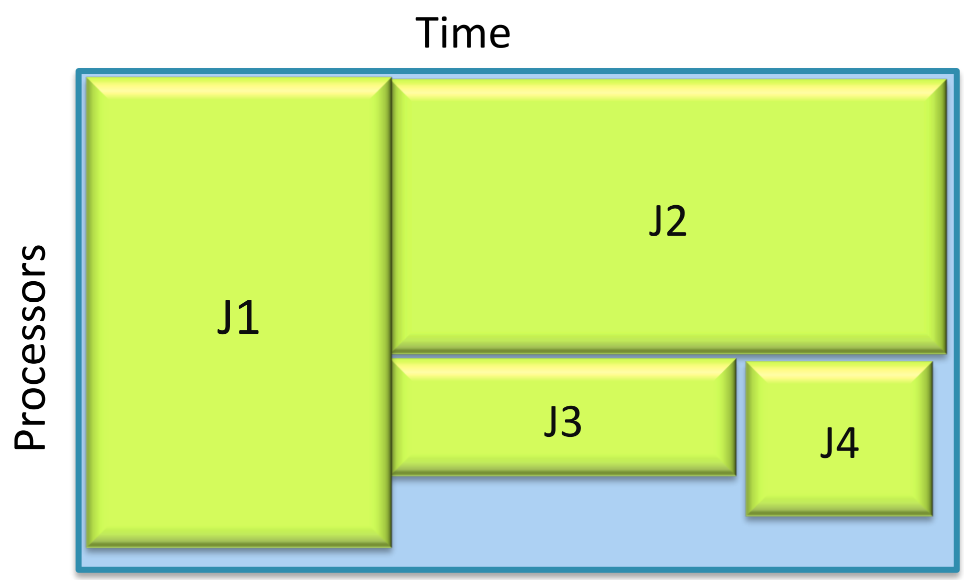
- **Bin-Packing**



For references see survey
[Weinberg 2008]

1990's: Scheduling in Parallel Computing

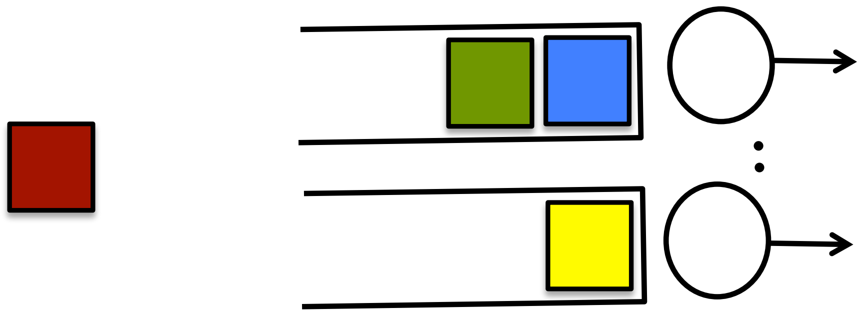
- **Bin-Packing**
  - Need job size estimates



For references see survey [Weinberg 2008]

1990's: Scheduling in Parallel Computing

- **Bin-Packing**
  - Need job size estimates
- **Processor Sharing**, i.e. switching b/w threads for different jobs
  - Need processor speed estimates
- **Load-balancing**: Work stealing, Power-of-choice
  - Need queue length estimates



1990's: Internet and PCs

- PCs become affordable
- Internet connectivity went on improving

- Virtual Private Networks (VPNs)
- Grid Computing: Connect cheap PCs via the Internet

- Many Internet Companies bought their own servers and managed them privately
- But then the Dotcom bubble burst..

2000's: The Cloud Computing Era

- The idea of a flexible, low-cost, scalable, shared computing environment developed

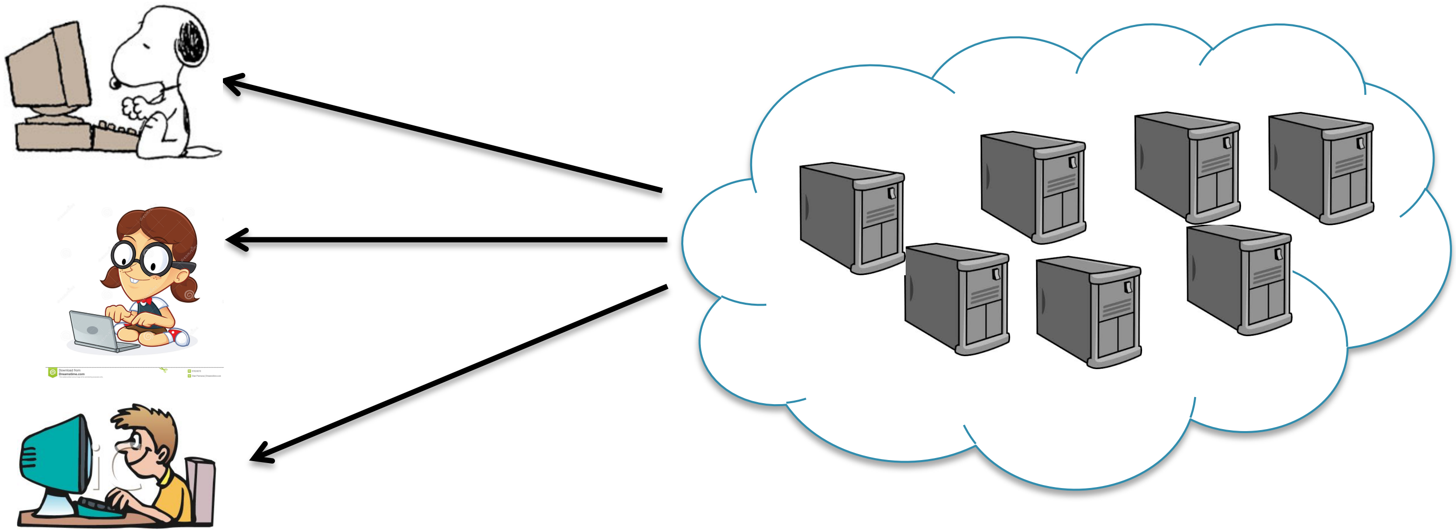
- Computing become a utility, like electricity

2000's: The Cloud Computing Era

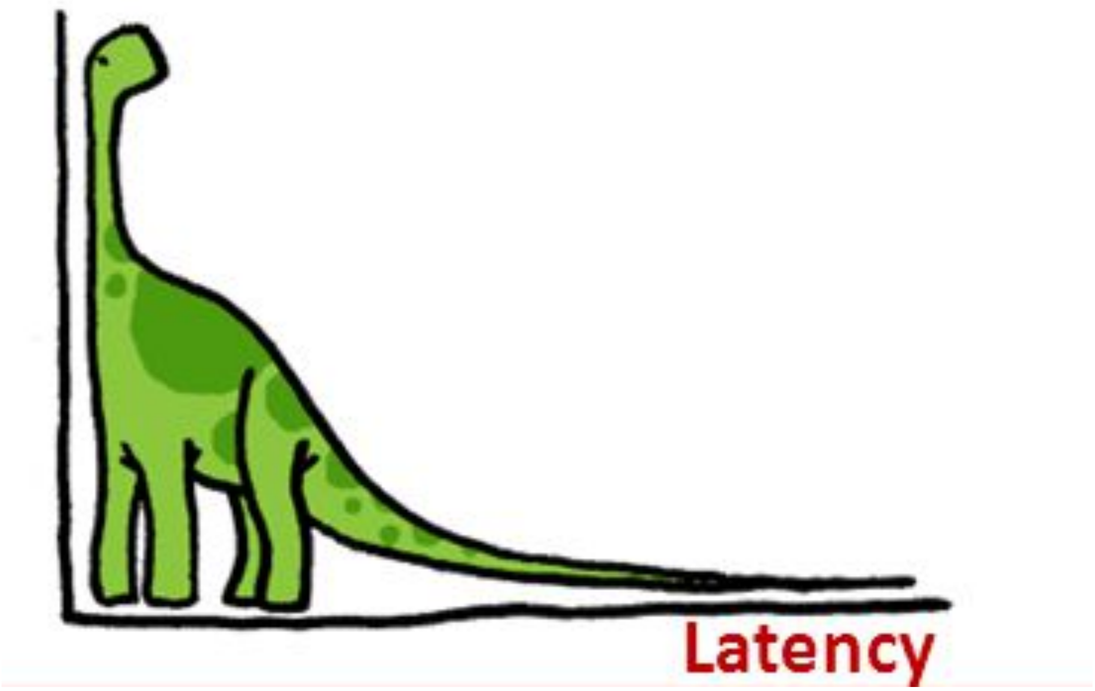
KEY ISSUE: Job sizes, server speeds & queue lengths are unpredictable

REASON: Large-scale resource sharing → Variability in service

- Virtualization, server outages etc.
- Norm and not an exception [Dean-Barroso 2013]

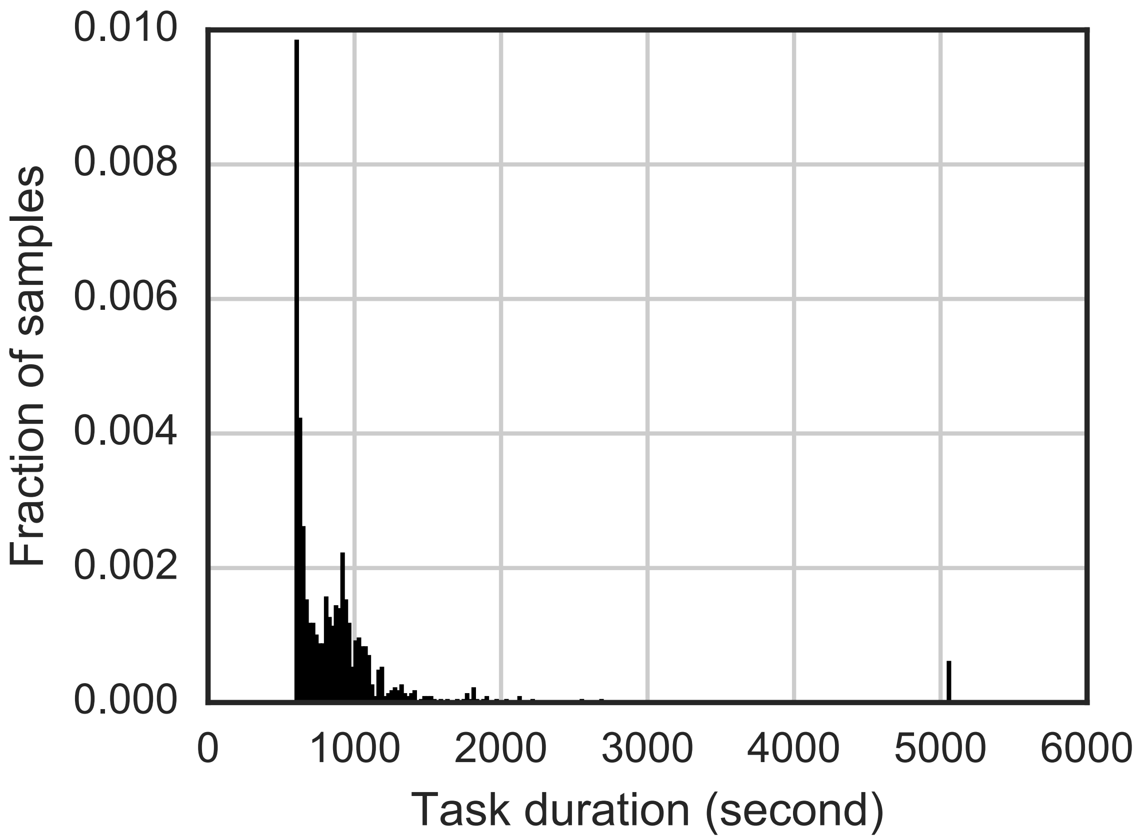


The Tale of Tails



Tail at Scale: 99%ile latency can be much higher than average

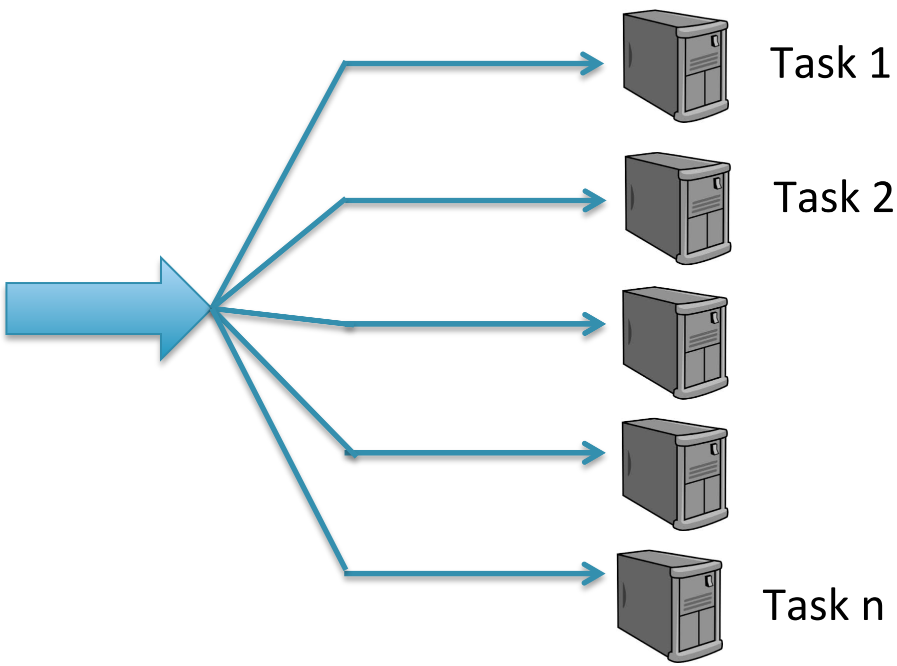
The Tale of Tails



Tail at Scale: 99%ile latency much higher than average

Problem: Stragglers in Parallel Computing

- A job with hundreds of parallel tasks
- Machine response time can vary due to virtualization, congestion etc.
- The slowest tasks are the bottleneck in job completion



[Dean "Tail at Scale" 2013]

Latency	50%ile	99%ile
1 task finishes	1ms	10ms
All tasks finish	40ms	140 ms

Exercise: Tale of Tails

A server finishes a task in 1 sec with probability 0.9, and 10 sec with probability 0.1

- What is the expected task execution time?

- If 100 tasks are run in parallel of 100 servers, what is the expected time to complete all of them.

Exercise: Tale of Tails

A server finishes a task in 1 sec with probability 0.9, and 10 sec with probability 0.1

- What is the expected task execution time?

  1*0.9 + 10*0.1 = 1.9

- If 100 tasks are run in parallel of 100 servers, what is the expected time to complete all of them.

Exercise: Tale of Tails

A server finishes a task in 1 sec with probability 0.9, and 10 sec with probability 0.1

- What is the expected task execution time?

  $1*0.9 + 10*0.1 = 1.9$

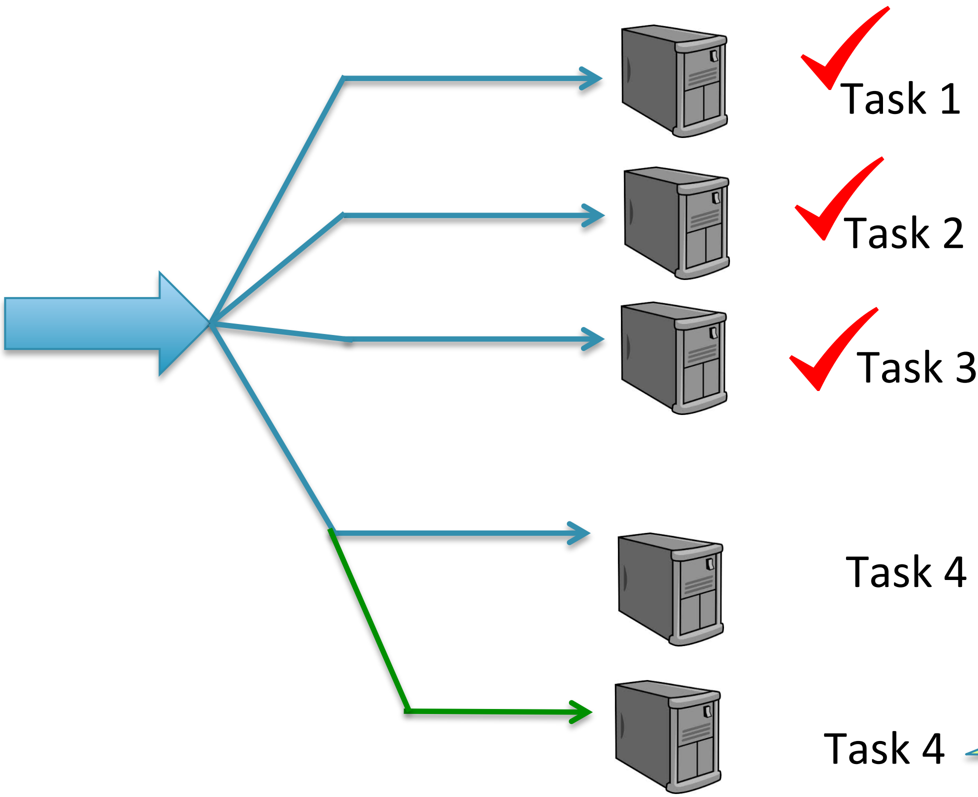
- If 100 tasks are run in parallel of 100 servers, what is the expected time to complete all of them.

  $1*0.9^{100} + 10*(1- 0.9^{100}) \sim 10$

Straggler Replication

PROBLEM: Slowest tasks become a bottleneck

SOLUTION: Replicate the stragglers and wait for one copy



PARAMETERS

p: Frac. of tasks replicated

r: # additional replicas

c: kill/keep original task

Eg. MapReduce, Apache Spark launch 1 replica, keep original copy

Straggler Replication Analysis

[Wang-GJ-Wornell SIGMETRICS 2014, 15]

PARAMETERS

p: Frac. of tasks replicated

r: # additional replicas

c: kill/keep original task

METRICS

E[T] = Time to finish all tasks

E[C] = Total server runtime per task

$$\mathbb{E}[T] = \boxed{\mathbb{E}[X_{(1-p)n:n}]} + \boxed{\mathbb{E}[Y_{pn:pn}]}$$

Y is the residual service time after adding replicas

Central Value Theorem

n -> ∞

$$F_X^{-1}(1-p)$$

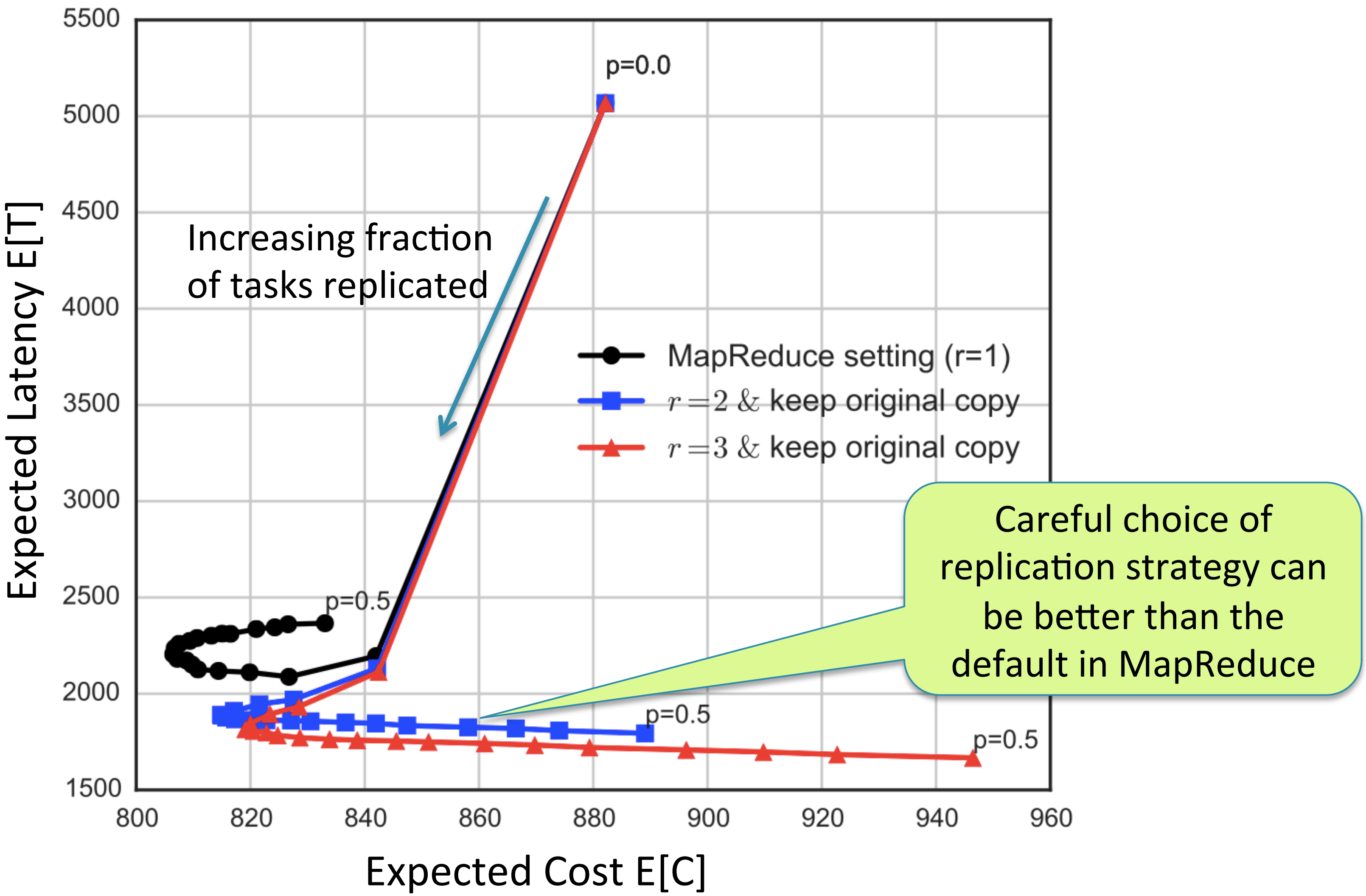
Extreme Value Theorem

n -> ∞

Different behavior for Exponential, Light or Heavy tailed Y

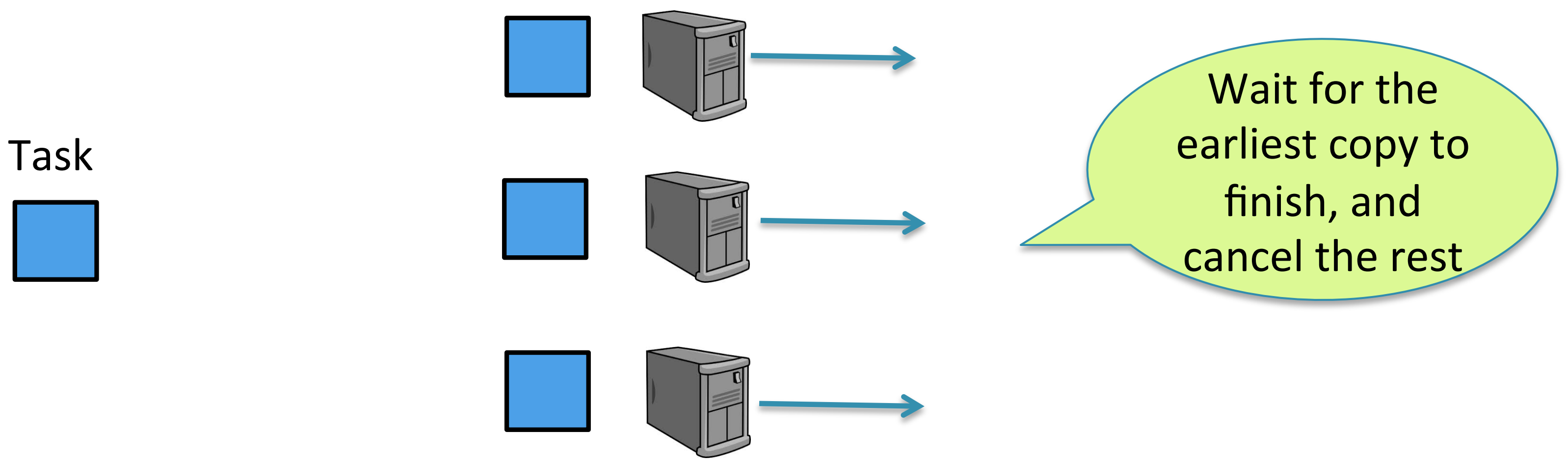
Simulations using Google Cluster Data

Latency-Cost Trade-off



Task Replication in Queueing Systems

IDEA: Assign task to multiple servers and wait for earliest copy

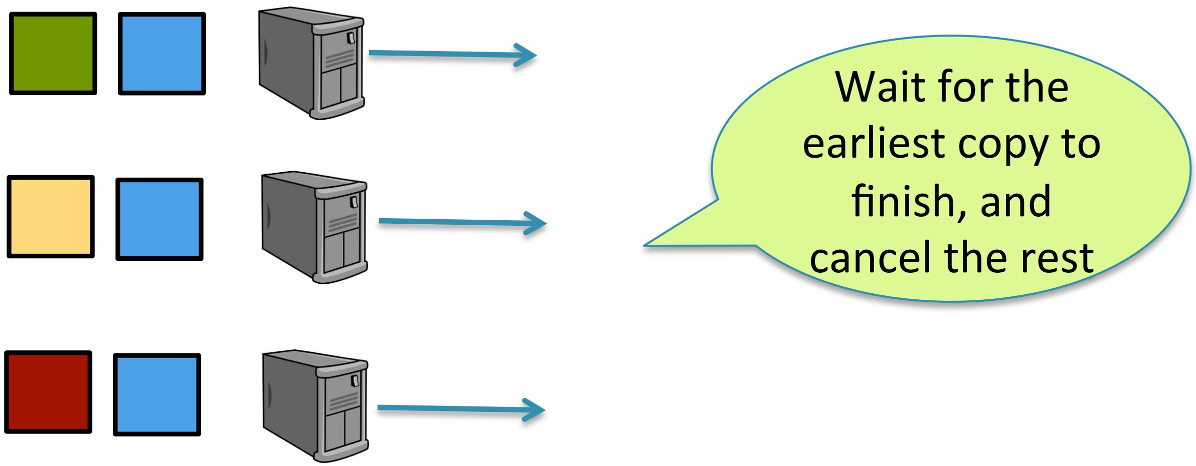


COST

- Additional computing time at servers

Task Replication in Queueing Systems

IDEA: Assign task to multiple servers and wait for earliest copy

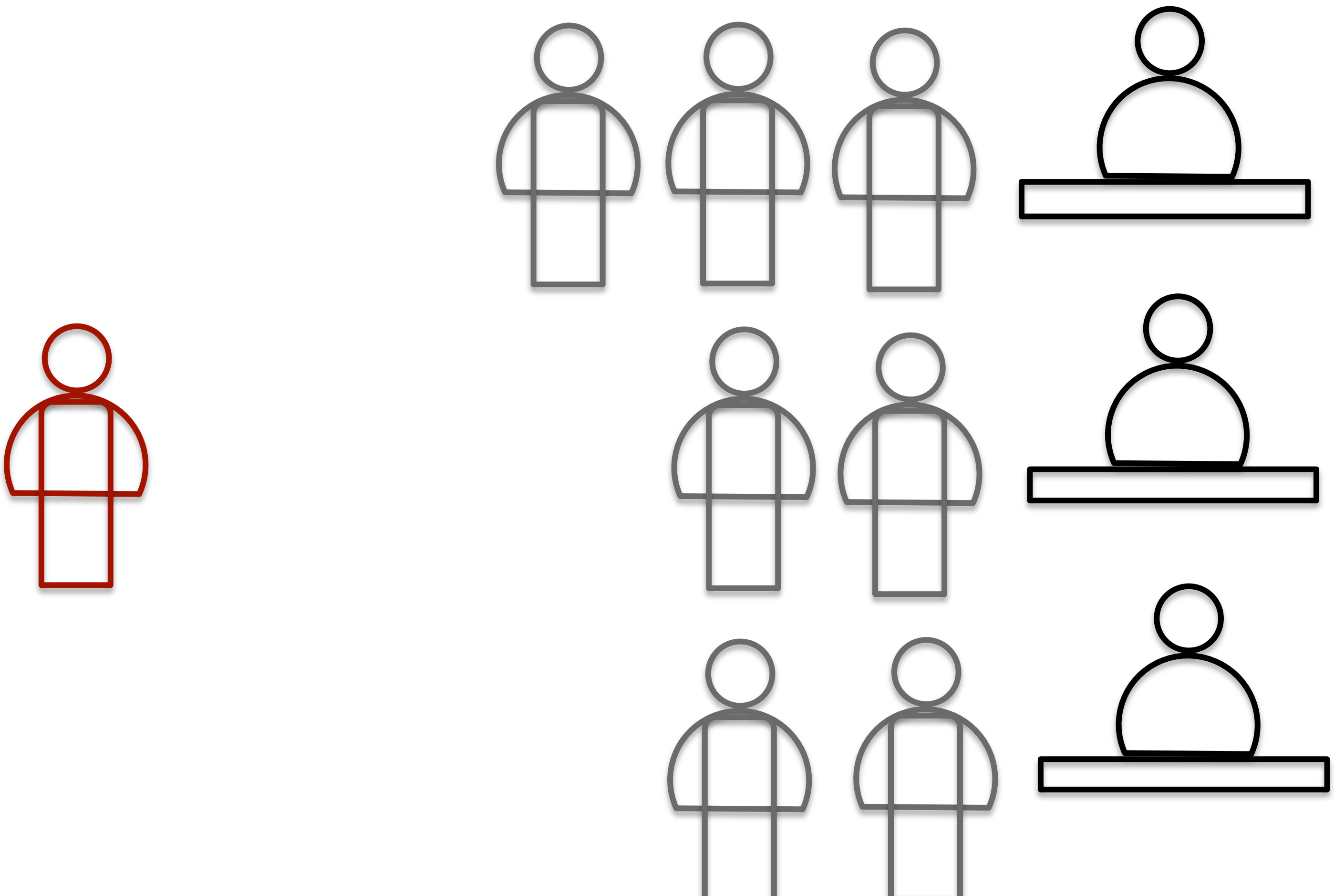


COST

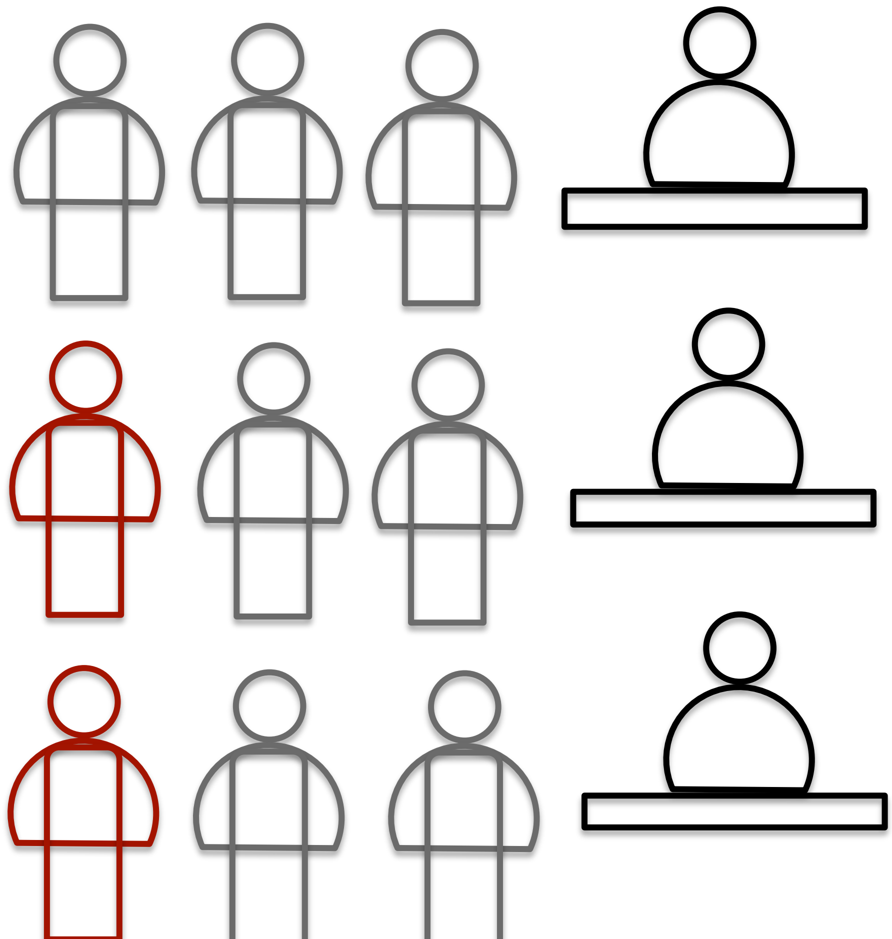
- Additional computing time at servers
- Increased queuing delay for other tasks

Analogy: Supermarket Queues

Supermarket Queues



Supermarket Queues

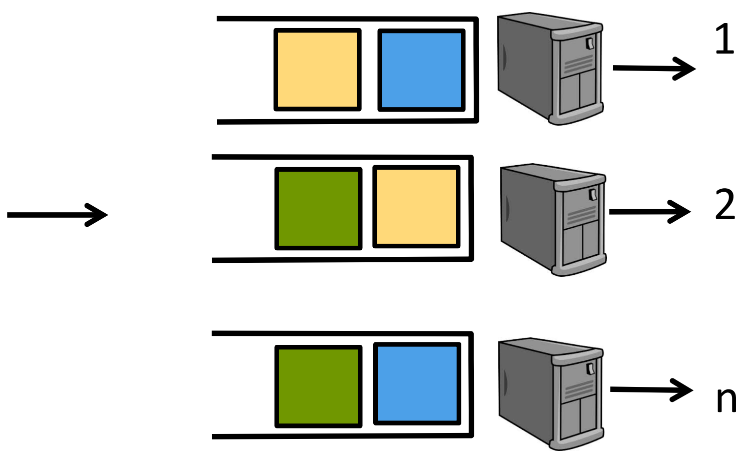


Get a friend to join the other queue!

What if everyone in the supermarket uses this strategy?

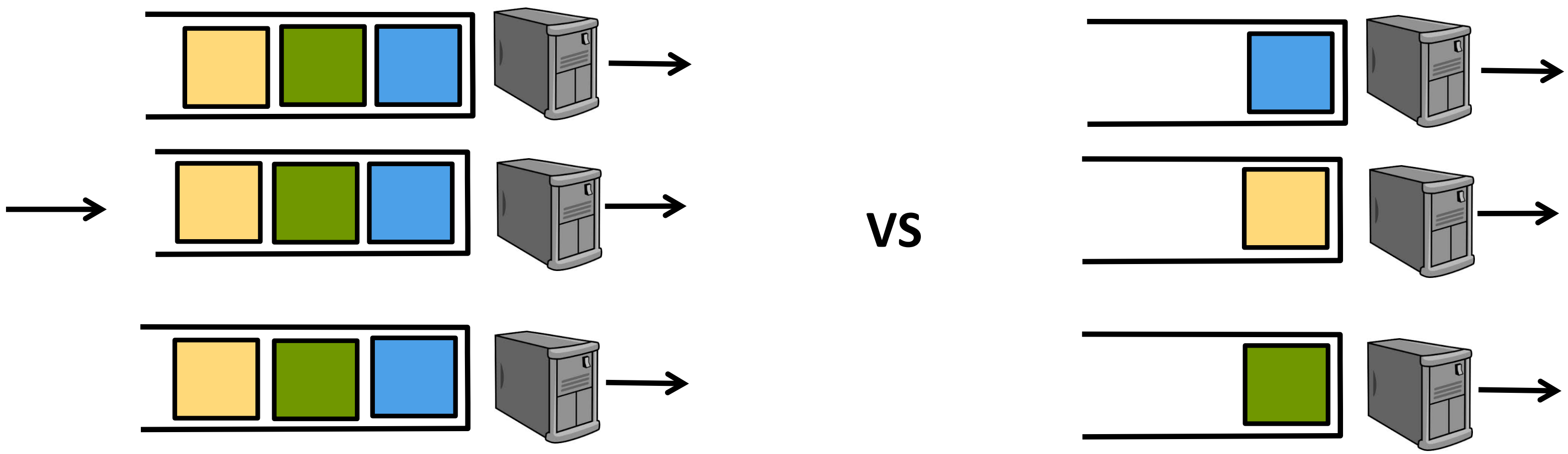
Design Questions

- How many replicas to launch?
- Which queues to join?
- When to issue and cancel the replicas?



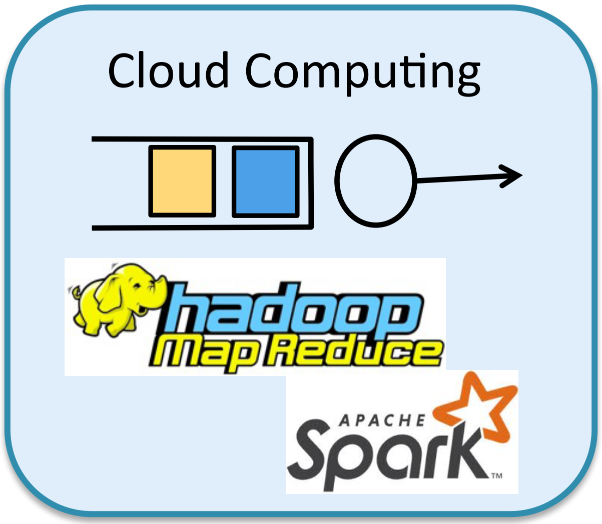
Surprising Insight

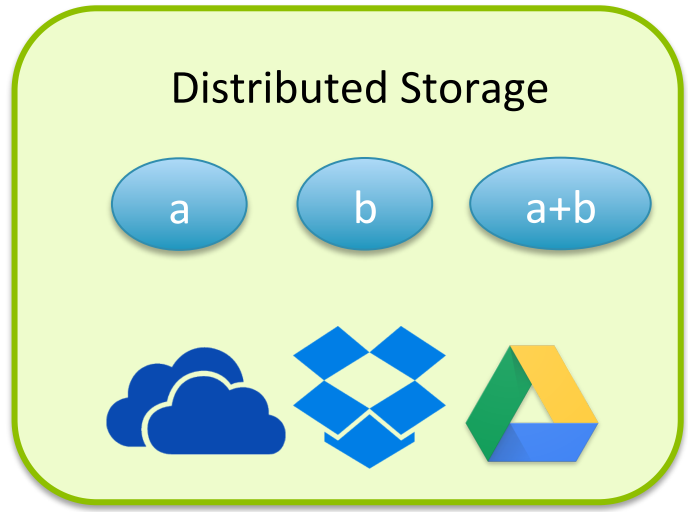
In certain regimes, replication could make the whole system faster, and cheaper!



Effective service rate > Sum of individual servers

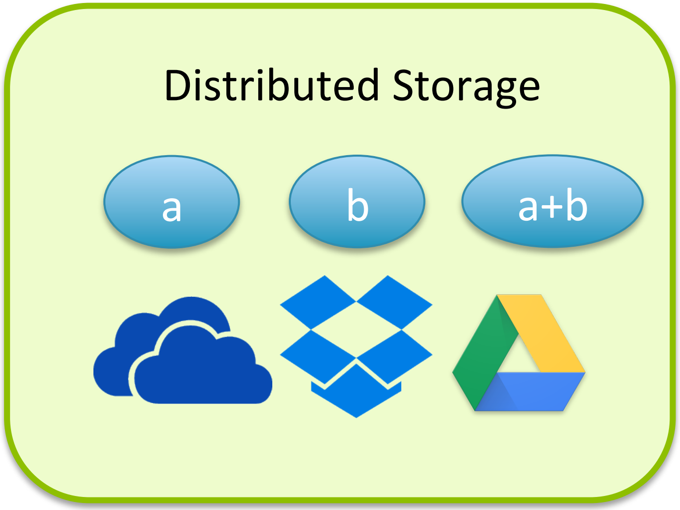
History and Overview





History and Overview

- RAID systems
- Coding for locality/repair
- Systems implementation of codes
- Reducing latency in content download



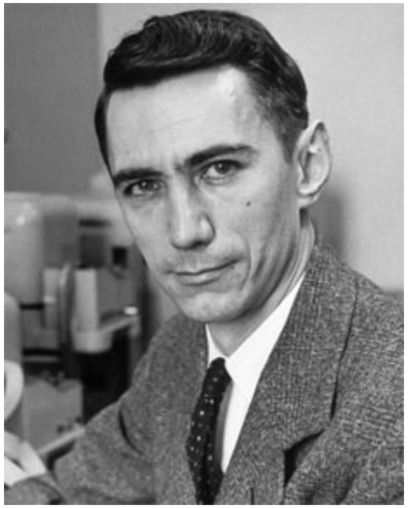
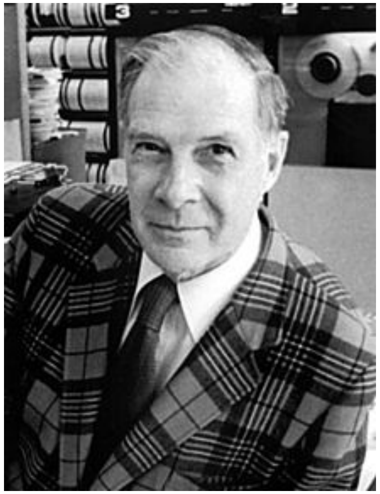
RAID: Redundant Array of Independent Disks (1987)

- Levels RAID 0, RAID 1, … : design for different goals such as reliability, availability, capacity etc.

- One of the inventors, Garth Gibson was here at CMU

Coding Theory

- For reliable communication in presence of noise
- Bell Labs was one of the leaders in 1950’s
- Key figures: Claude Shannon and Richard Hamming

Simplest Codes

- Repetition Code
  - 0 → 0 0 0 : Rate: 1/3
  - If receive 0 ? ? we can recover from 2 erasures

- (3,2) code: Data bits: a, b Parity bit: (a XOR b)
  - Example: 0 1 1, 1 1 0: Rate 2/3
  - If we receive 0 ? 1 or ? 1 0 we can correct the failed bit
  - 2 bit symbols: (0 1) ? (1 1)

(n,k) Reed-Solomon Codes: 1960

- Data: $d_1, d_2, d_3, \ldots d_k$
- Polynomial: $d_1 + d_2\, x + d_3\, x^2 + \ldots\, d_k\, x^{k-1}$
- Parity bits: Evaluate at n-k points:

$x=1$:	$d_1 + d_2 + d_3 + d_4$
$x=2$:	$d_1 + 2\, d_2 + 4\, d_3 + 8\, d_4$
$x=3$:	….
$x=4$:	….
$x=n$:	…

- Can solve for the coefficients from any k coded symbols

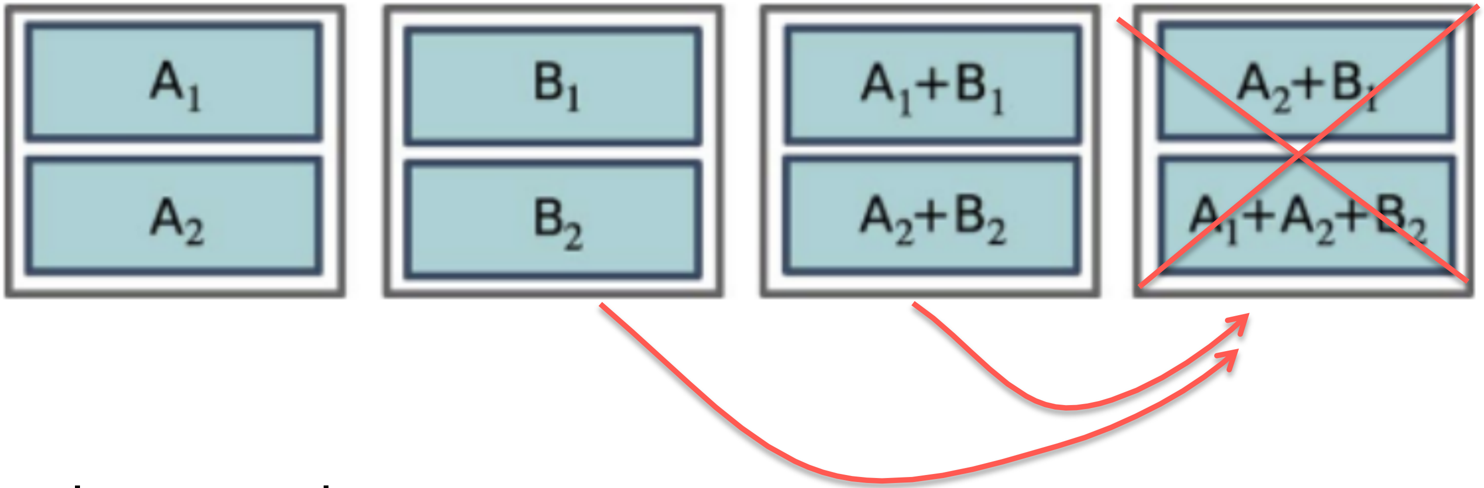
Example: (4,2) Reed-Solomon Code

- Data: d_1, d_2 → Polynomial: $d_1 + d_2\,x + d_3\,x^{2} + \ldots d_k\,x^{k-1}$

- Can solve for the coefficients from any k coded symbols
- Microsoft uses (7, 4) code
- Facebook uses (14,10) code

Locality and Repair Issues

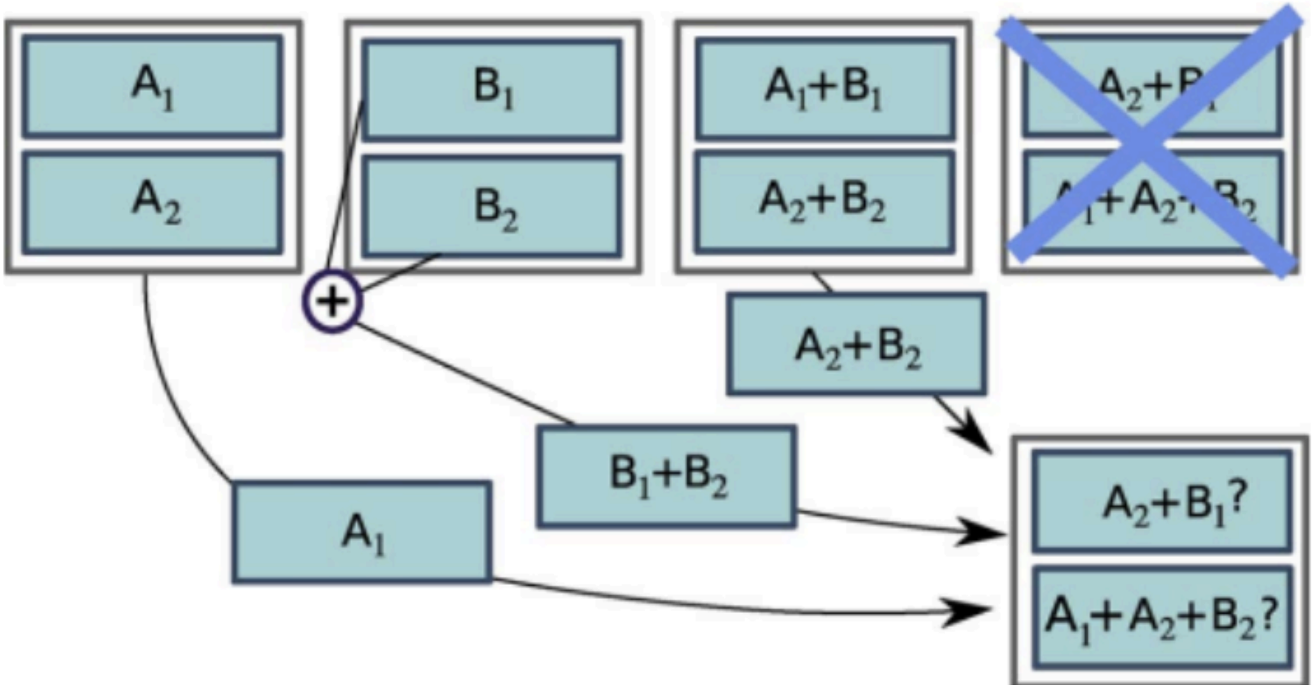
- Repairing failed nodes is hard with Reed-Solomon Codes..



- If we lose 1 node:
  - Need to contact k other nodes
  - Need to download k times the lost data

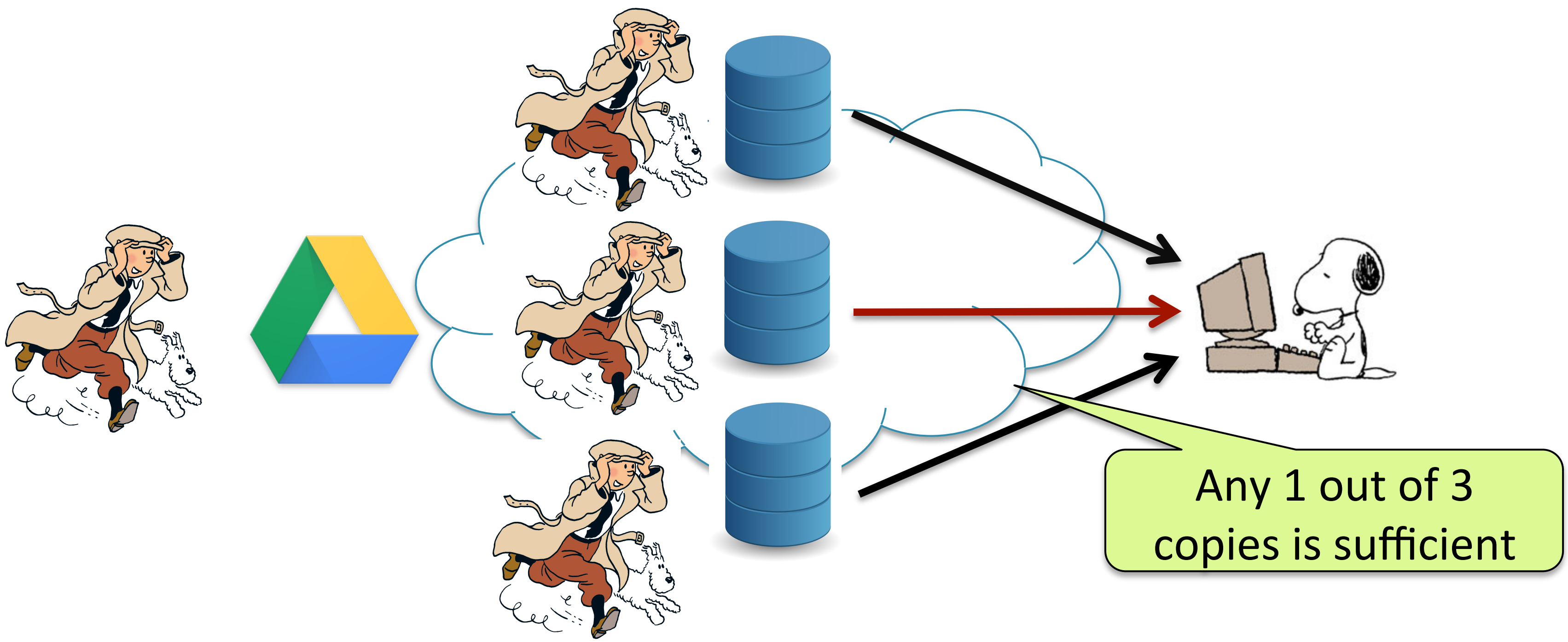
Solution: Locally Repairable Codes

- Codes designed to minimize:
  - Repair Bandwidth
  - Number of nodes contacted



Replicated Storage

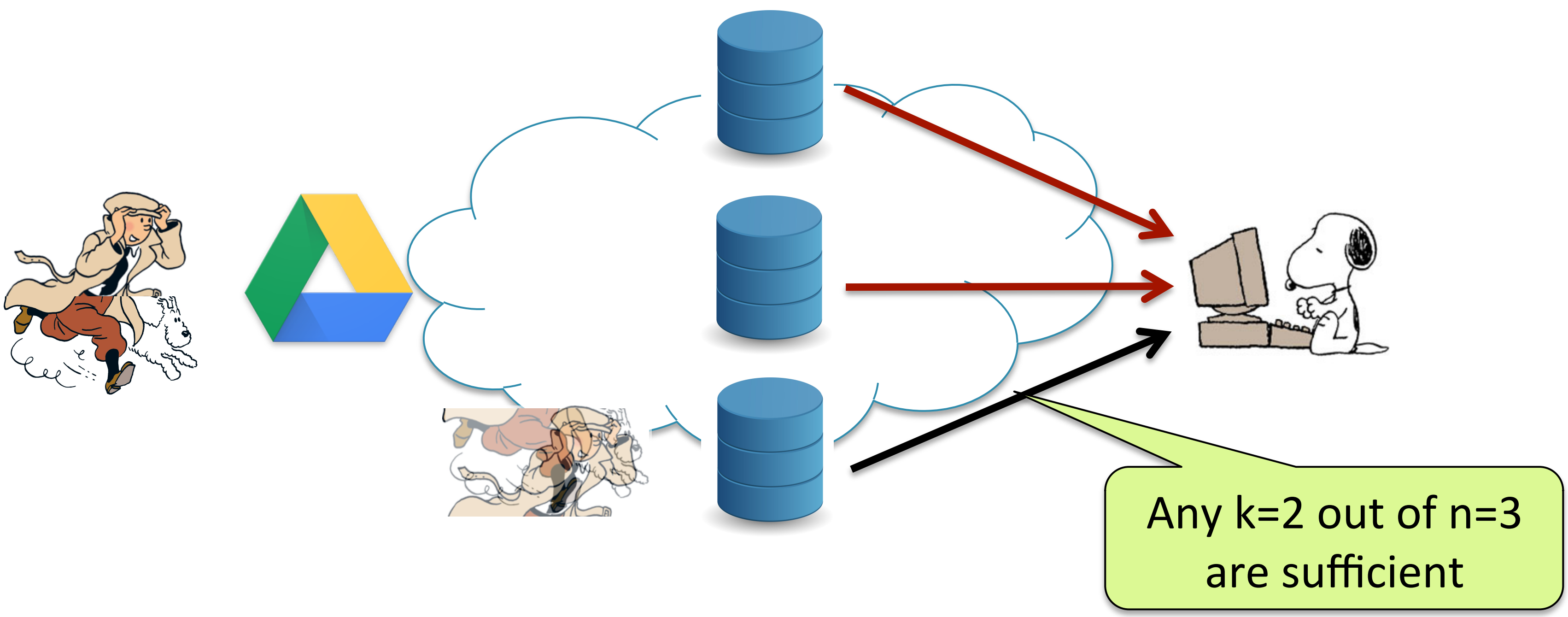
- Content is replicated on the cloud for reliability



- Can support more users simultaneously
- Replicated used for “hot” data, i.e. more frequent accessed

Erasure Coded Storage

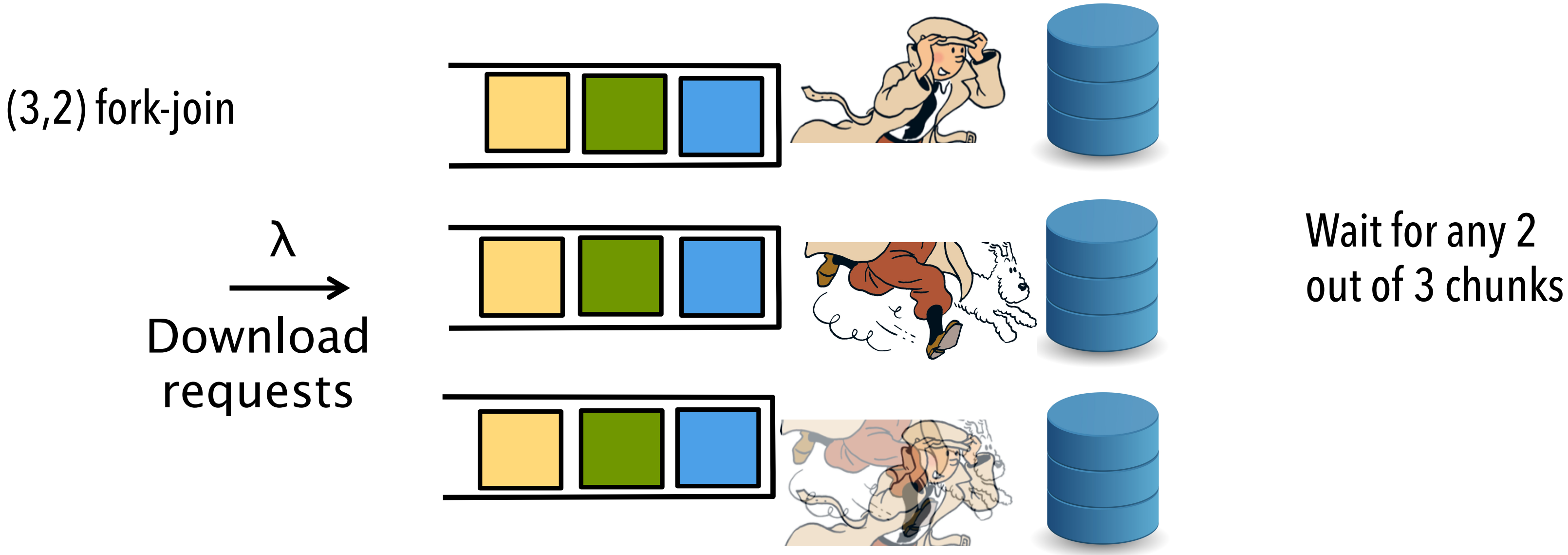
- With an (n,k) MDS code, any k out of n chunks are sufficient
  - Facebook, Google, Microsoft use (14,10) or (7,4) codes
  - Currently used for cold data, increasing for hot data



Q: How many users can we serve, and how fast?

The (n,k) fork-join model

- Request all n chunks, wait for any k to be downloaded
- Each chunk takes service time $X \sim F_X$

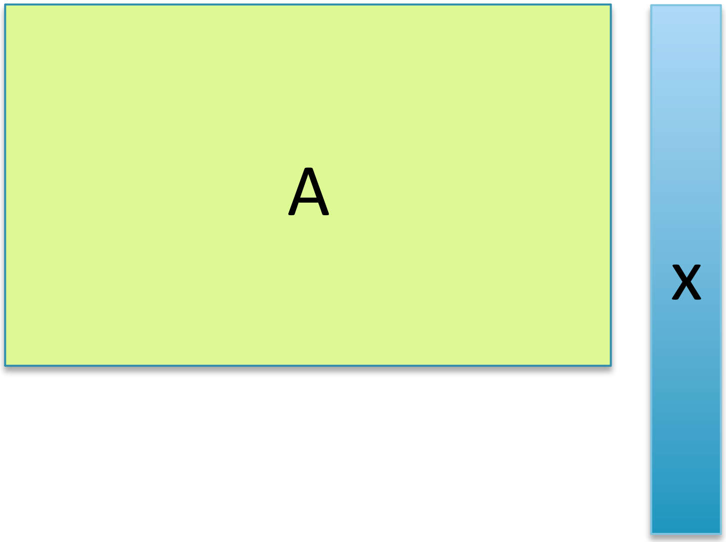


k = 1: Replicated Case

k = n: Fork-join system actively studied in 90's

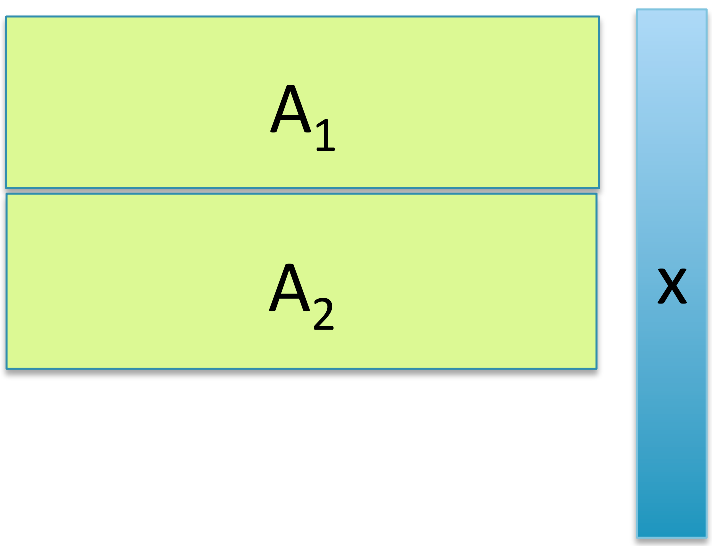
Coded Computing and ML

- So far: Coding for storage
- Codes can also speed up computing and machine learning!
- Example: Matrix-Vector Multiplication



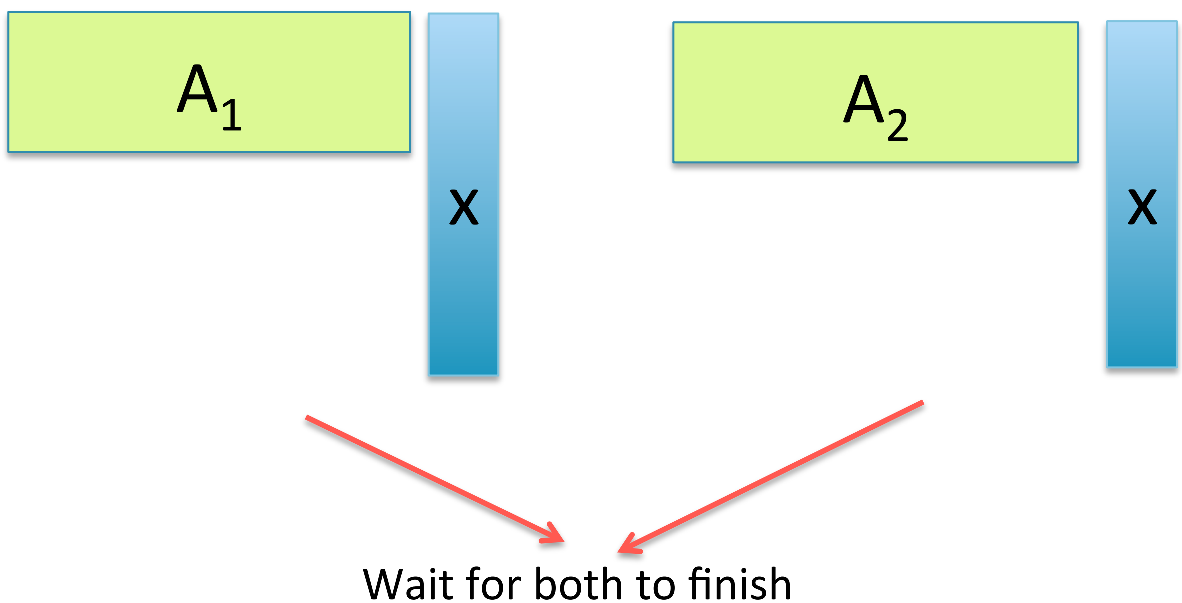
Coded Computing and ML

- So far: coding for storage
- Codes can also speed up computing and machine learning!
- Example: Matrix-Vector Multiplication



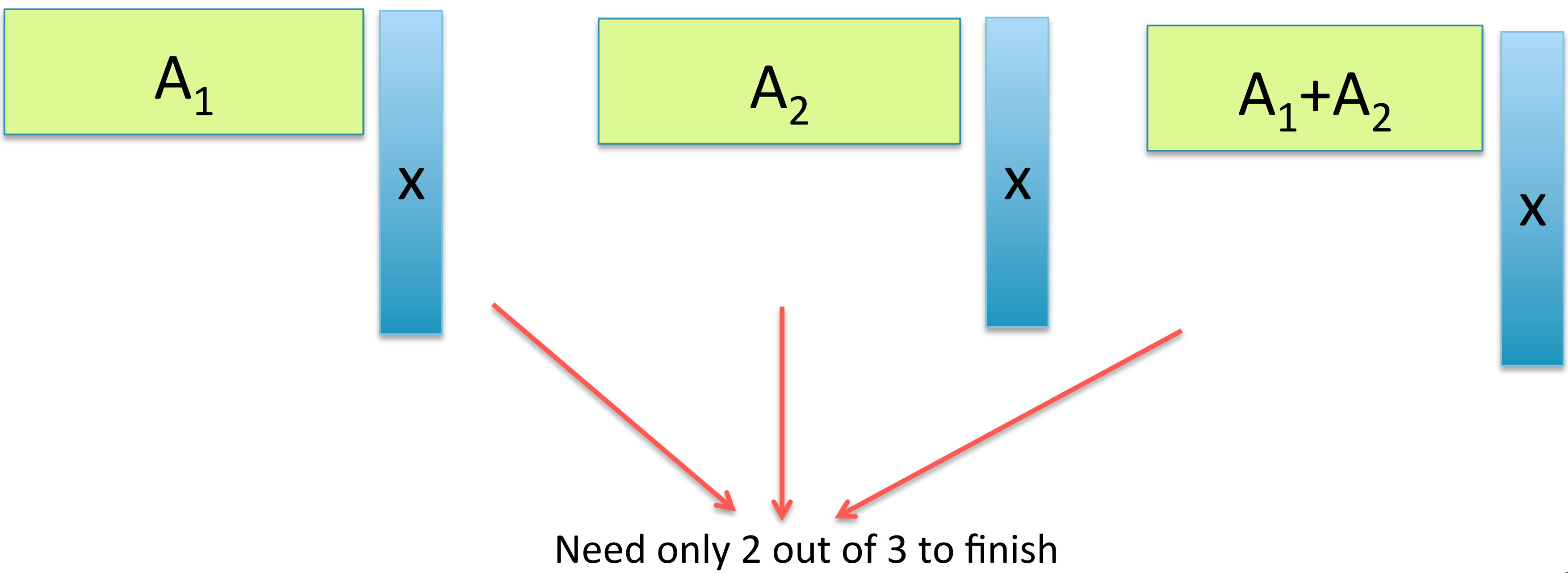
Coded Computing and ML

- So far: coding for storage
- Codes can also speed up computing and machine learning!
- Example: Matrix-Vector Multiplication

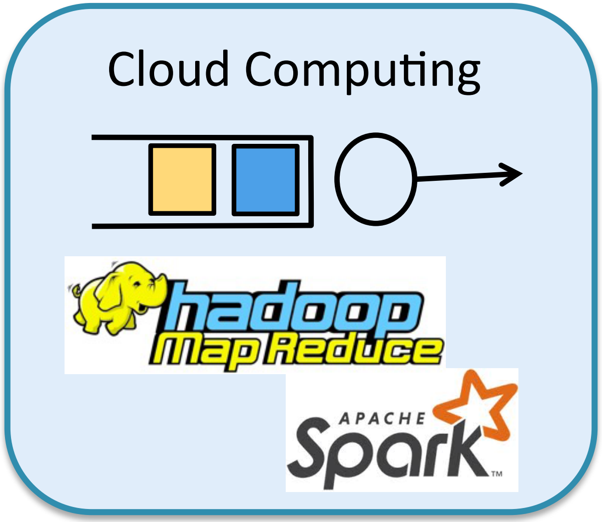


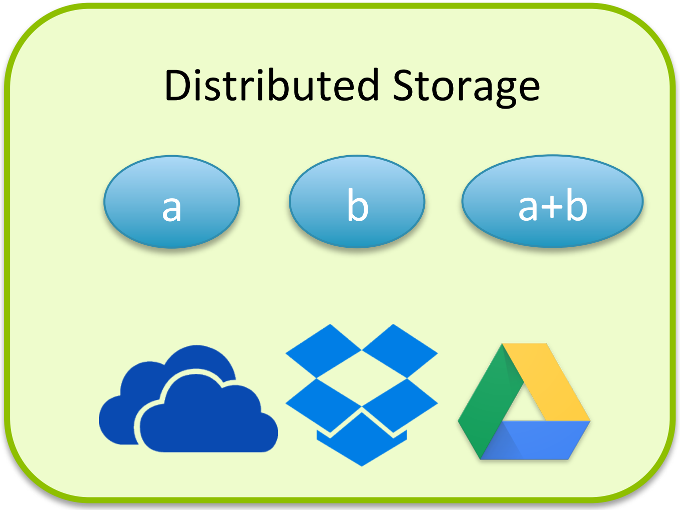
Coded Computing and ML

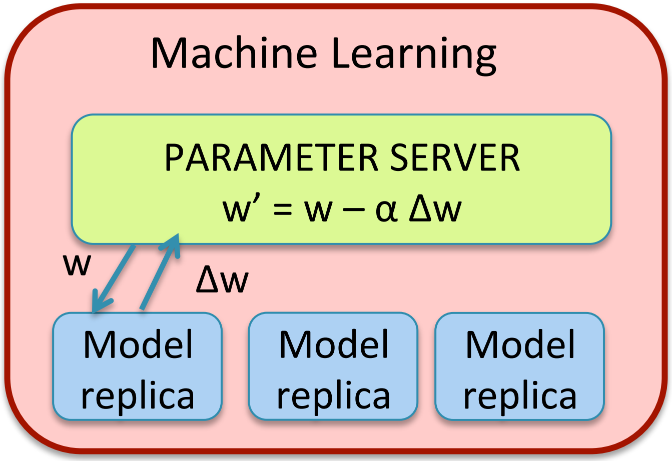
- So far: coding for storage
- Codes can also speed up computing and machine learning!
- Example: Matrix-Vector Multiplication



Second half: Machine Learning

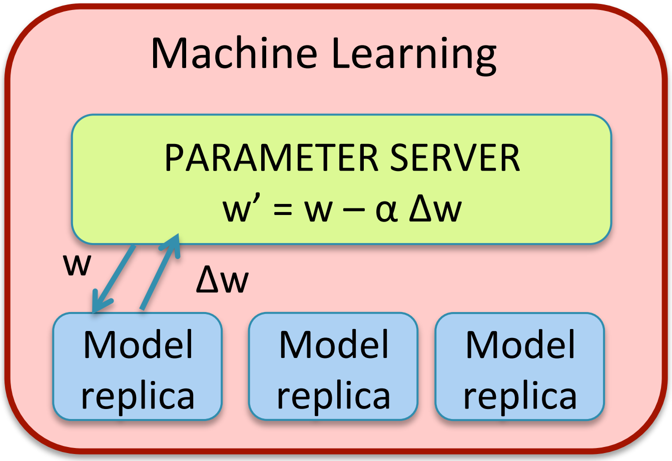




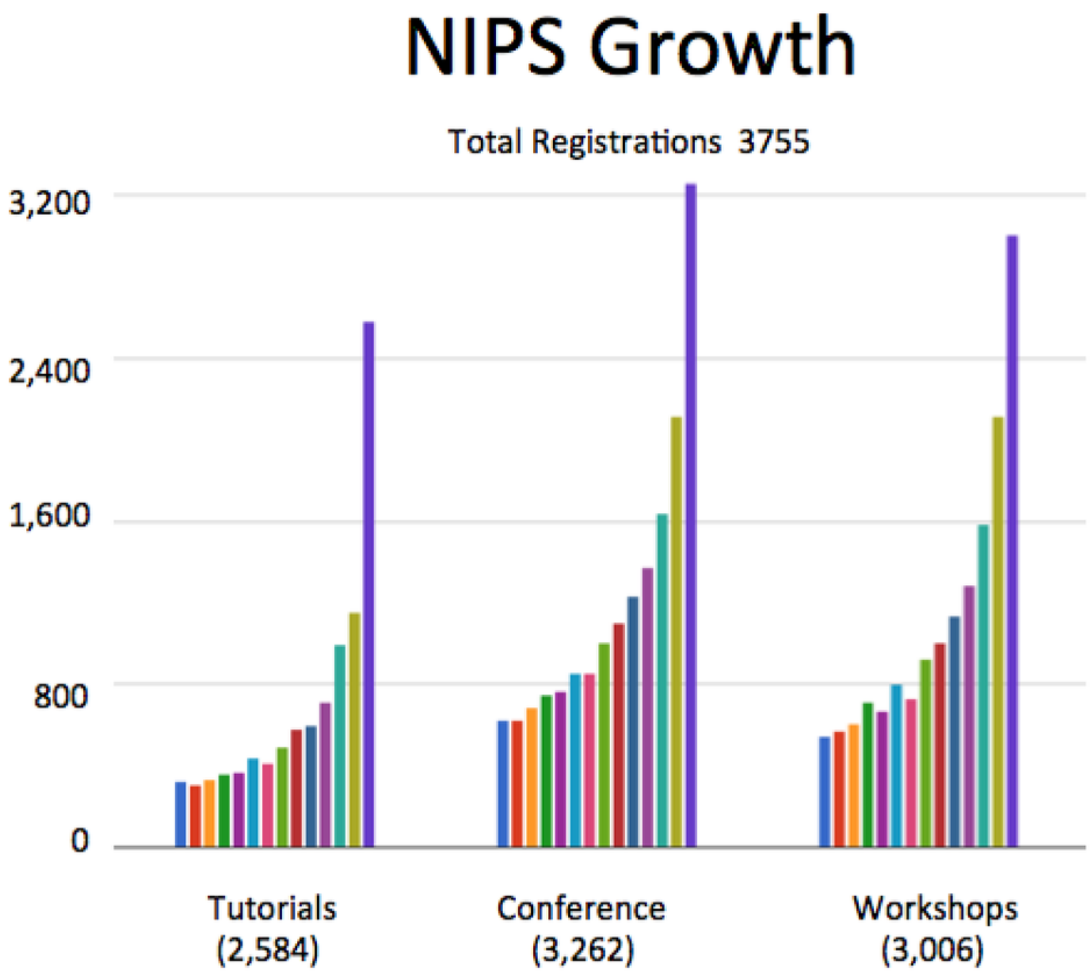


Second half: Machine Learning

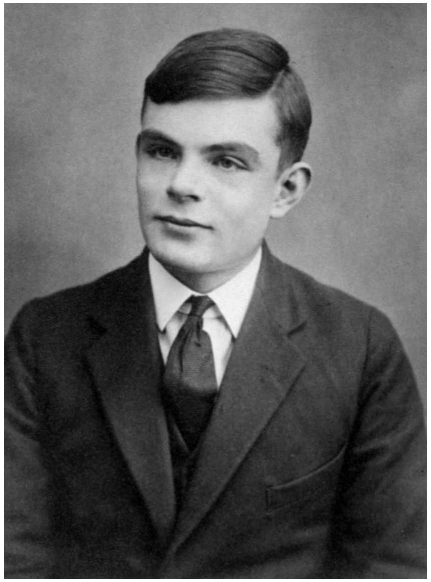
- SGD Methods, Convergence
- DistBelief, Alexnet
- Synchronous, Asynchronous SGD
- GANs, Reinforcement Learning



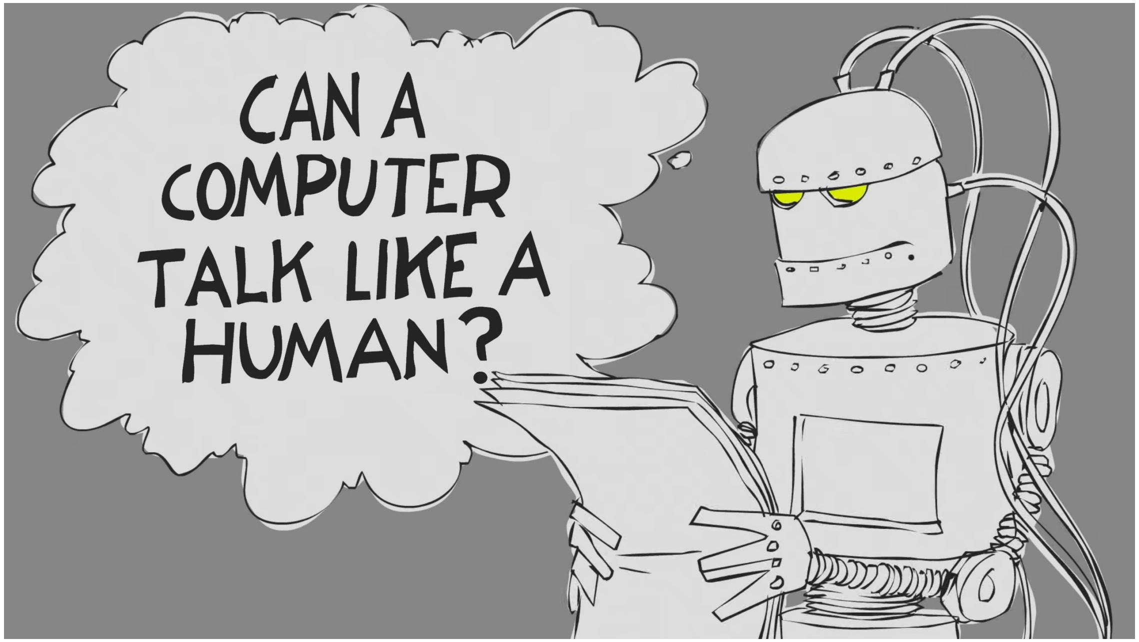
The unprecedented ML boom



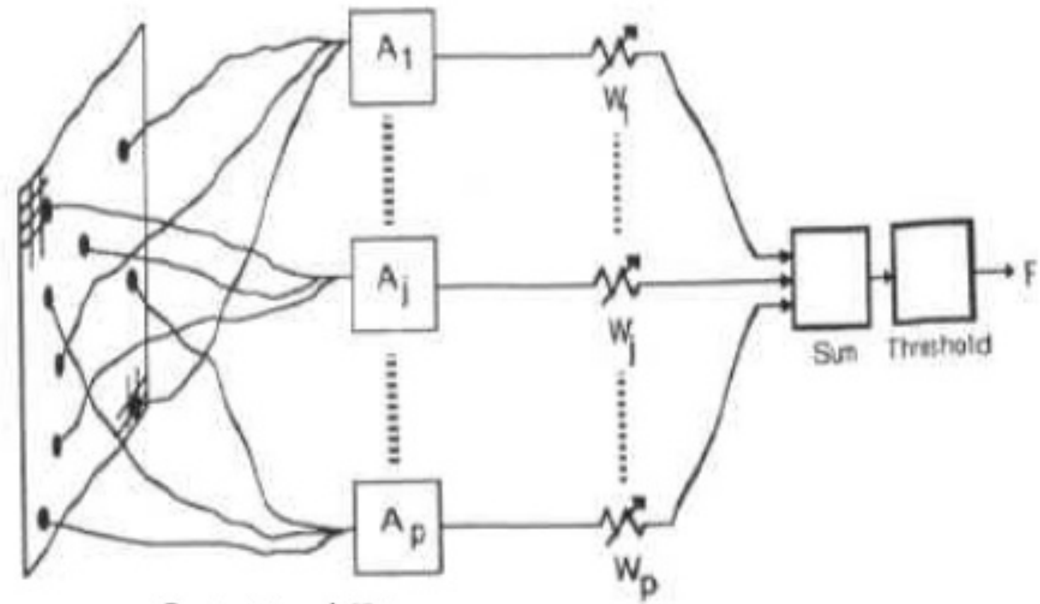
The Origins: 1950



Alan Turing



Neural Networks: Perceptron 1957

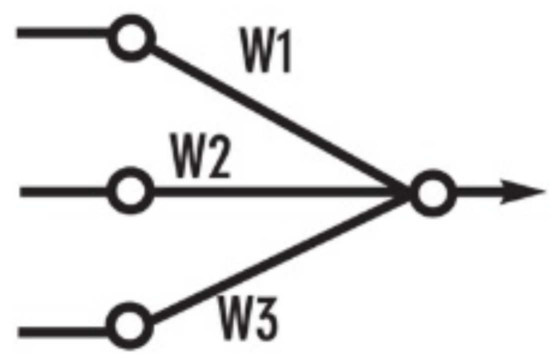


Original Perceptron

(From Perceptrons by M. L Minsky and S. Papert, 1969, Cambridge, MA: MIT Press. Copyright 1969 by MIT Press.

Frank Rosenblatt
(1928-1971)

Simplified model:



Back-propagation Algorithm (1986)

Geoff Hinton (U. Toronto, Google)

nature.com | about npg | news@nature.com | naturejobs | natureevents | help | site index

nature

my account | e-alerts | subscribe | register

SEARCH JOURNAL Go

Sunday 22 October 2017

Journal Home
Current Issue
AOP
Archive

THIS ARTICLE

Download PDF
References

Export citation
Export references

Send to a friend

More articles like this

Table of Contents
< Previous | Next >

letters to nature

Nature **323**, 533 - 536 (09 October 1986); doi:10.1038/323533a0

Learning representations by back-propagating errors

DAVID E. RUMELHART[*], GEOFFREY E. HINTON[†] & RONALD J. WILLIAMS[*]

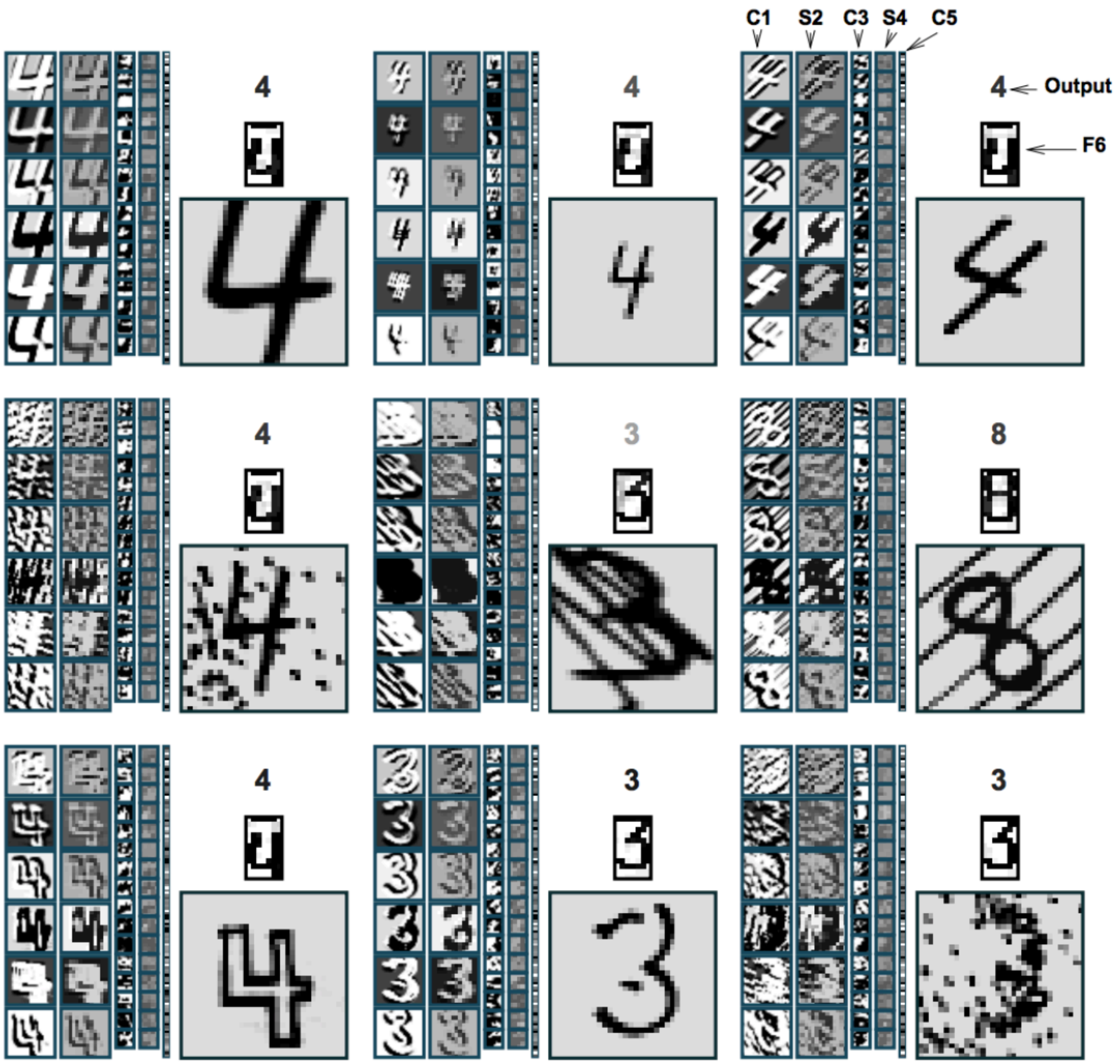
[*]Institute for Cognitive Science, C-015, University of California, San Diego, La Jolla, California 92093, USA
[†]Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Philadelphia 15213, USA
[†]To whom correspondence should be addressed.

We describe a new learning procedure, back-propagation, for networks of neurone-like units. The procedure repe Alan_Turing_Aged_16.jpg **ights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the net and the desired output vector. As a result of the weight adjustments, internal 'hidden' units which are not part of the input or output come to represent important features of the task domain, and the regularities in the task are captured by the interactions of these units. The ability to create useful new features distinguishes back-propagation from earlier, simpler methods such as the perceptron-convergence procedure[1].**

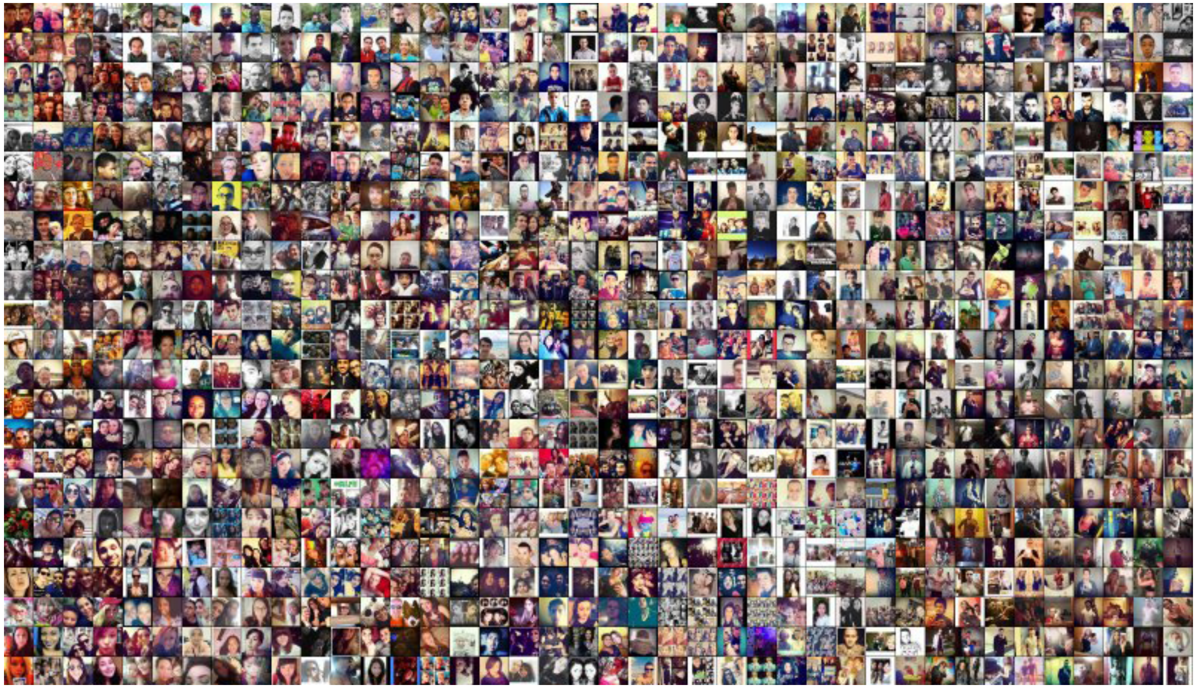
References

1. Rosenblatt, F. *Principles of Neurodynamics* (Spartan, Washington, DC, 1961).

MNIST (LeCun et al 1998)

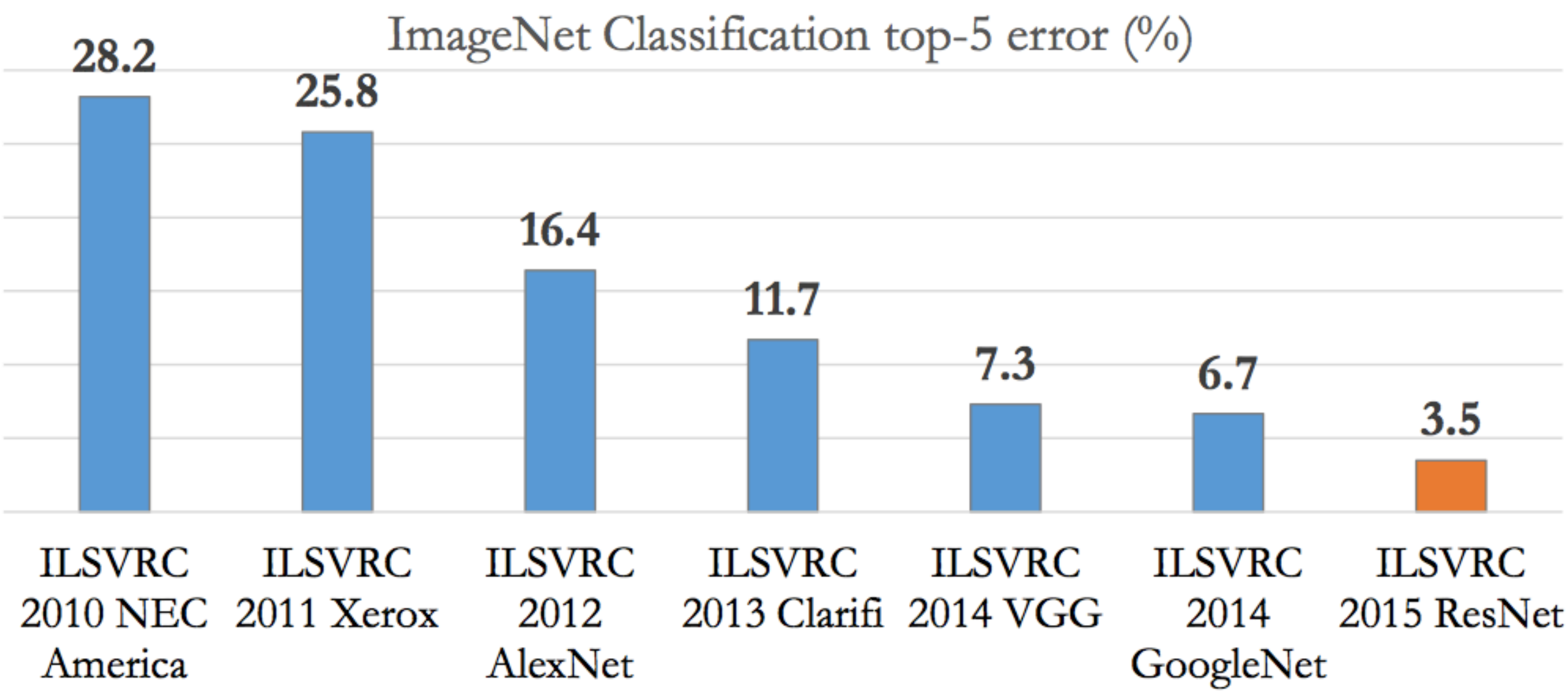


ImageNet and ILSVRC (2012)



Fei-Fei Li, Stanford

ImageNet and ILSVRC



Why the sudden success?

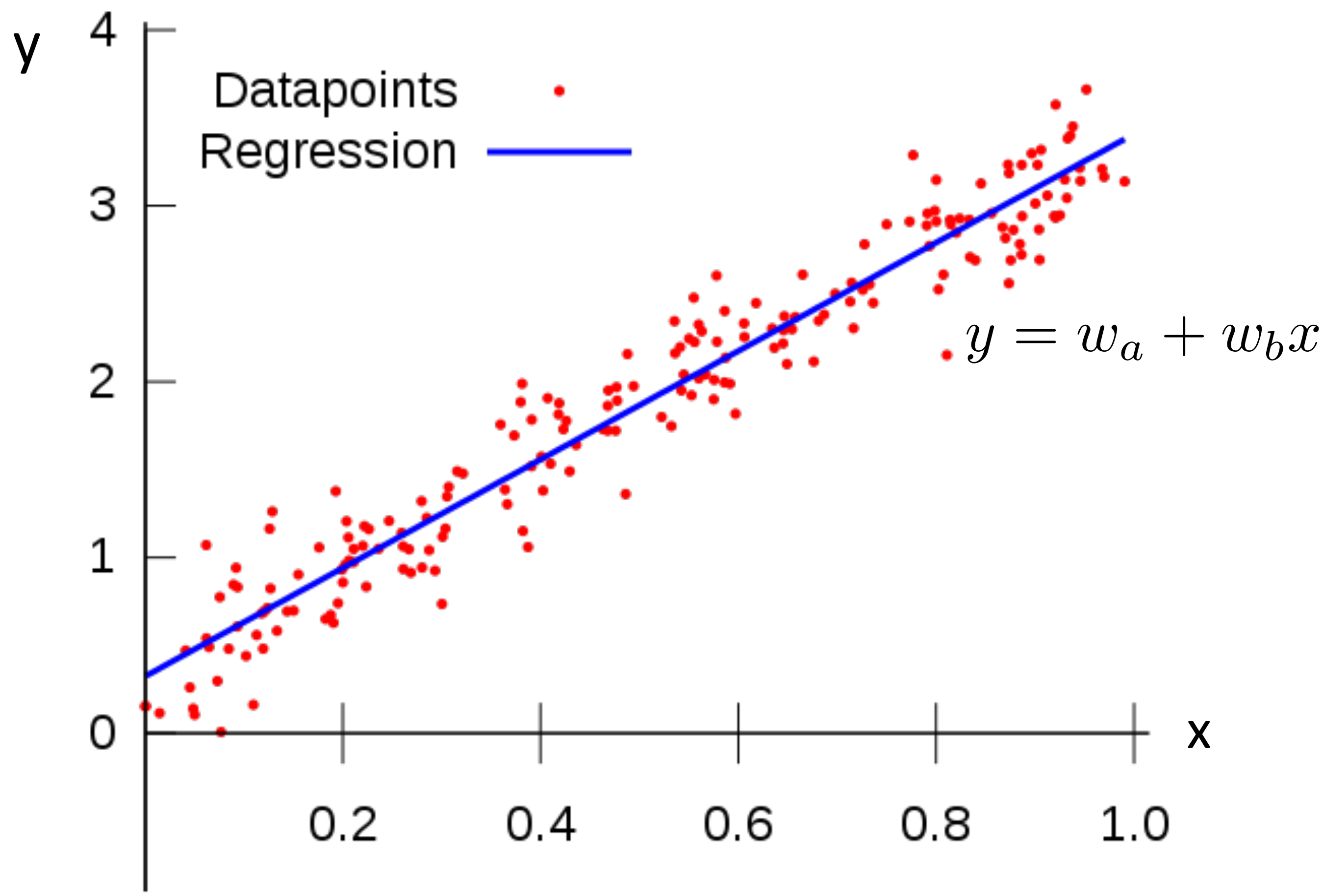
- Availability of massive datasets like Imagenet

- Computing power to train deep neural networks
  - Parallelization
  - GPUs

- Algorithmic advances:
  - Momentum, Adagrad, Adam etc.

Core of ML: Stochastic Gradient Descent (SGD)

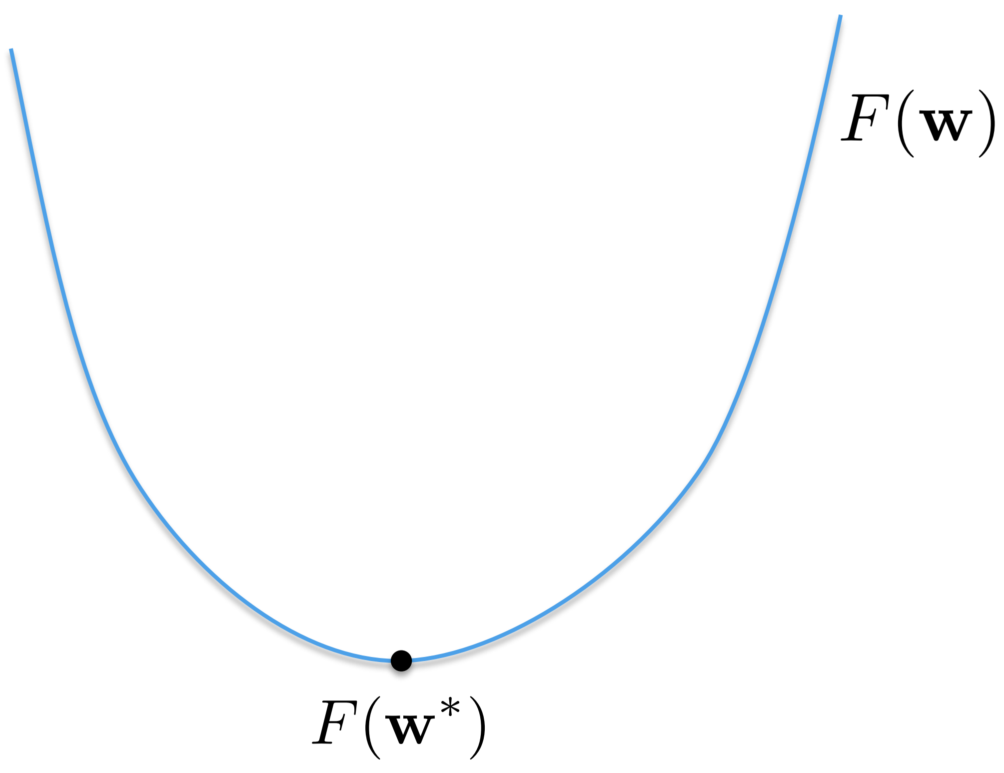
Simplest ML example: Regression



Given a big dataset of ($\mathbf{x_1}$, y_1), ($\mathbf{x_2}$, y_2), ($\mathbf{x_3}$, y_3), ($\mathbf{x_4}$, y_4), ….($\mathbf{x_N}$, y_N)
Find the optimal weights **w**

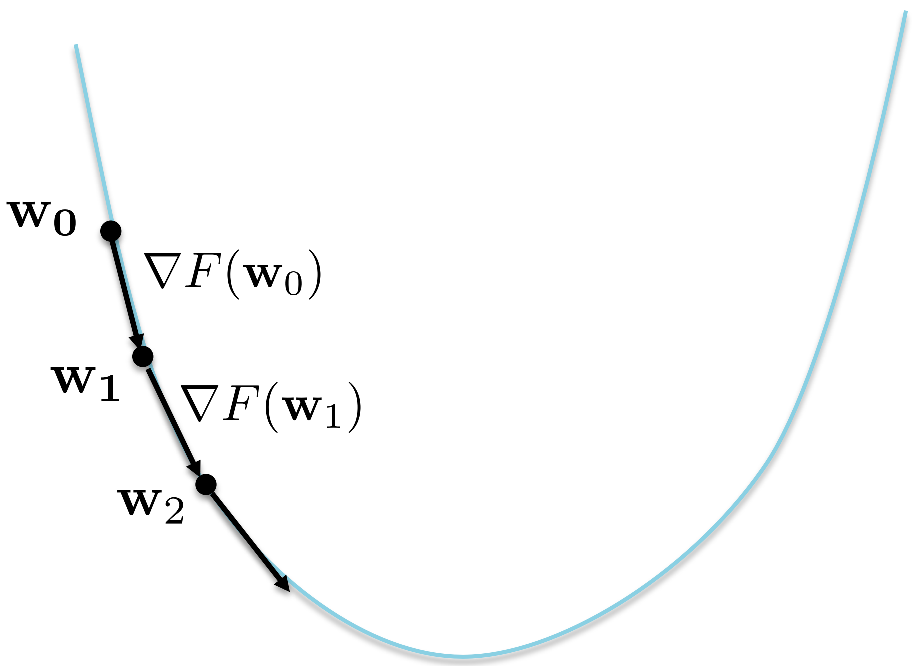
Core of ML: Gradient Descent (GD)

$$\min_{\mathbf{w}} F(\mathbf{w}) = \min_{\mathbf{w}} \frac{1}{N} \sum_{i=1}^{N} \nabla (y_i - \mathbf{w}^T \mathbf{x})^2$$

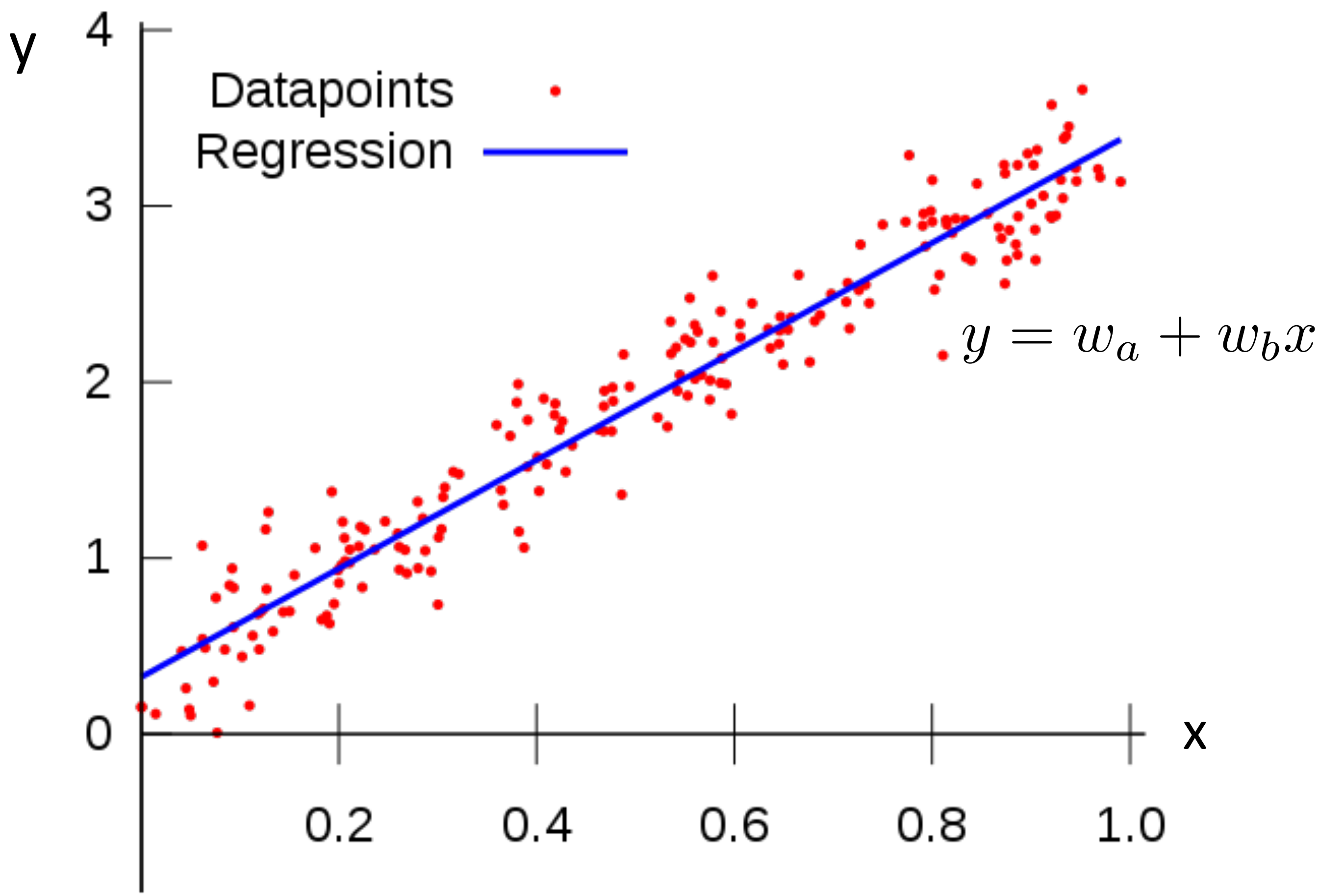


Core of ML: Gradient Descent (GD)

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla F(\mathbf{w_t})$$



Exercise: Find the update rule for w_a and w_b



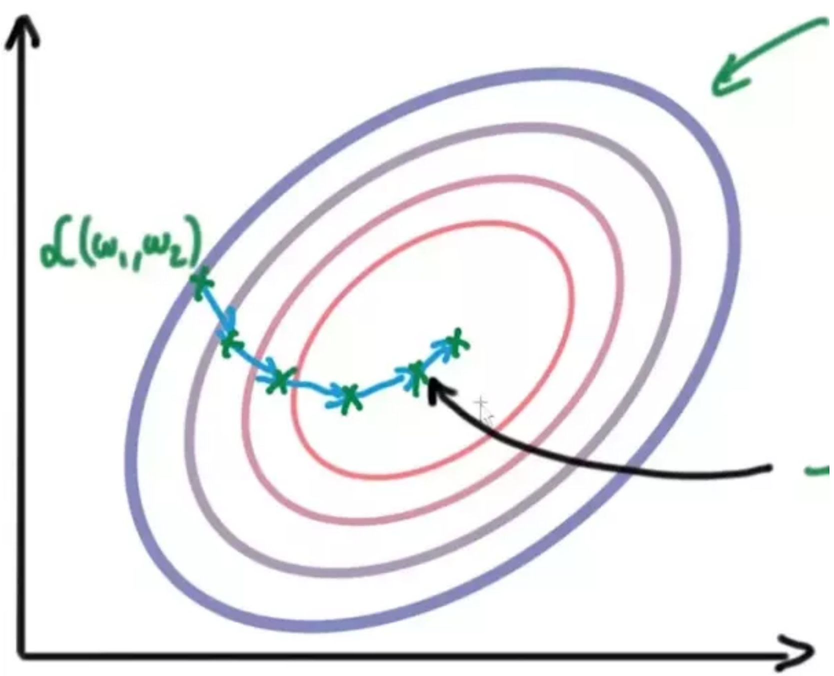
Given a big dataset of ($\mathbf{x_1}$, y_1), ($\mathbf{x_2}$, y_2), ($\mathbf{x_3}$, y_3), ($\mathbf{x_4}$, y_4), ….($\mathbf{x_N}$, y_N)

Find the optimal weights $\mathbf{w}$ = (w_a, w_b)

Gradient Descent (GD)

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \boxed{\frac{1}{N}\sum_{i=1}^{N} \nabla (y_i - \mathbf{w}^T \mathbf{x}_i)^2}$$

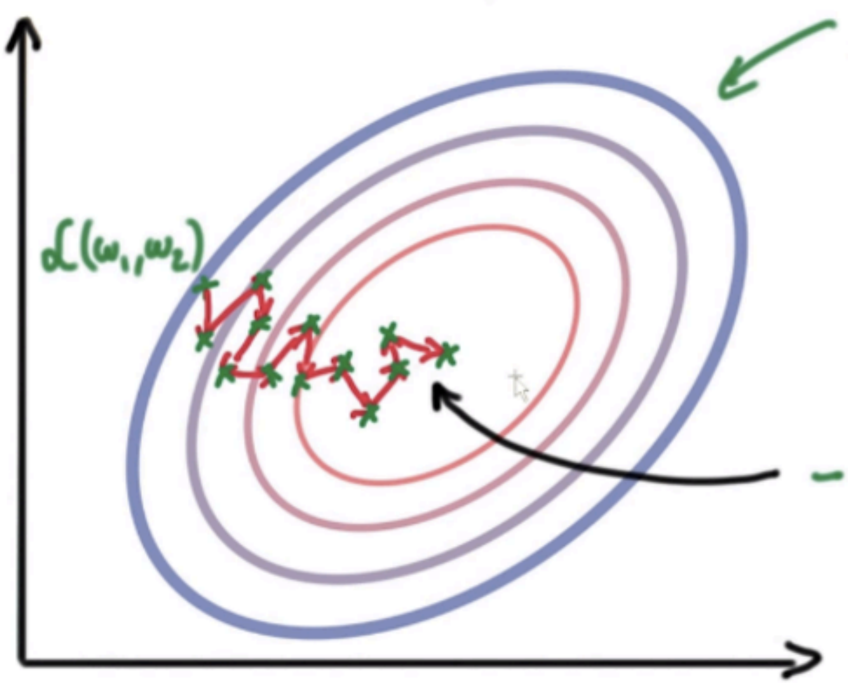
Too expensive for large datasets



Stochastic Gradient Descent (SGD)

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \boxed{\nabla (y_i - \mathbf{w}^T \mathbf{x}_i)^2}$$

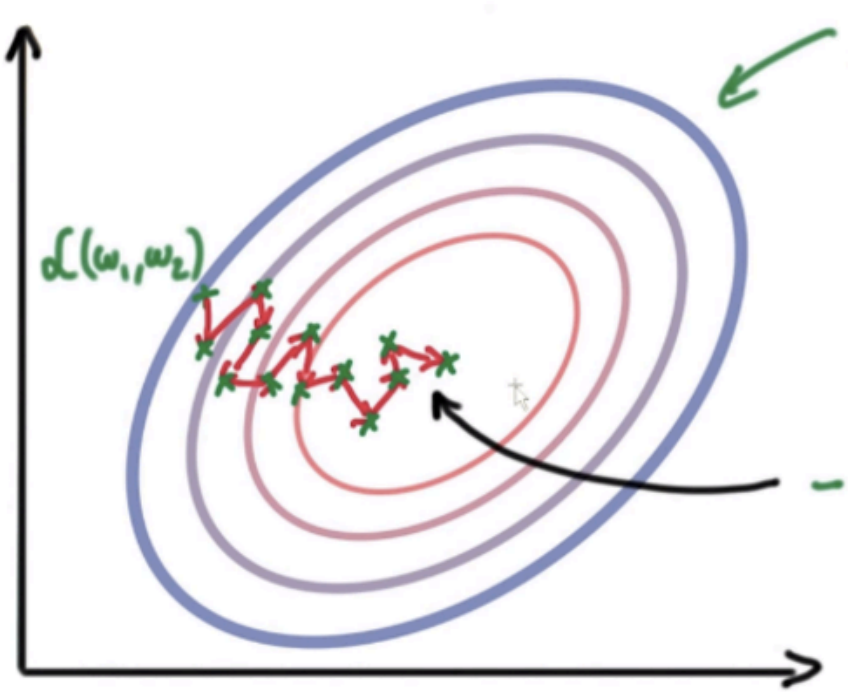
Easy, but possibly too noisy



Mini-batch SGD

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \boxed{\frac{1}{m} \sum_{i=1}^{m} \nabla (y_i - \mathbf{w}^T \mathbf{x}_i)^2}$$

Less noisy, but also computationally tractable



Exercise: How does variance scale with m?

If $Var(\nabla F(\mathbf{w}, \xi_i)) = \sigma^2$

What is the variance of the gradient update in mini-batch SGD?

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \sum_{i=1}^{m} \frac{1}{m} \nabla F(\mathbf{w}_t, \xi_i)$$

Convergence of SGD

$$\mathbb{E}[F(\mathbf{w}_k) - F_*] \leq \underbrace{\boxed{\frac{\eta L M}{2c}}}_{\text{Error Floor}} + \underbrace{\boxed{(1 - \eta c)}}_{\text{Decay Rate}}^{k-1}\left(F(\mathbf{w}_0) - F_* - \frac{\eta L M}{2c}\right)$$

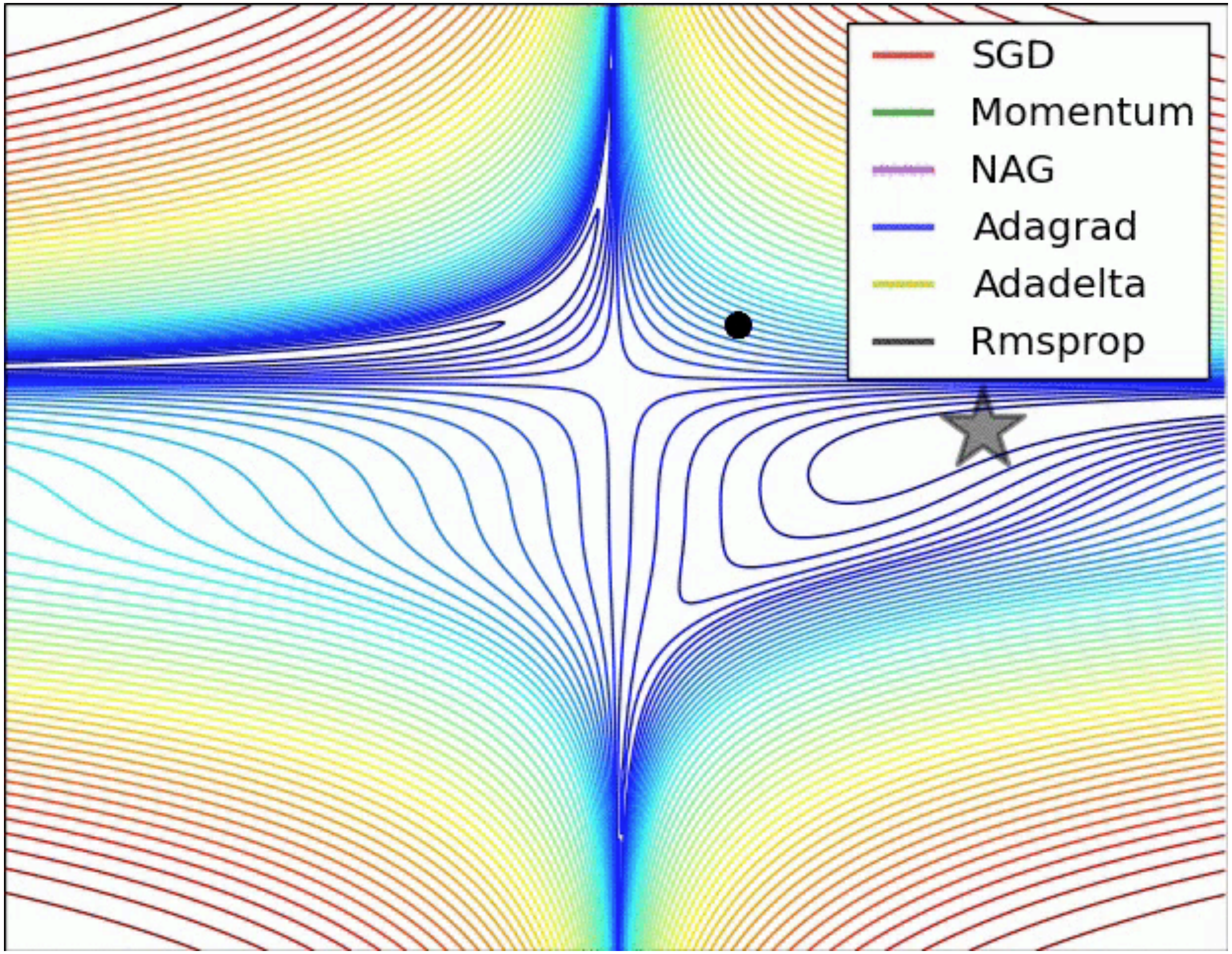
How does decay rate and error floor change with

- η (Learning Rate) ?
- M (Second moment of gradient) ?

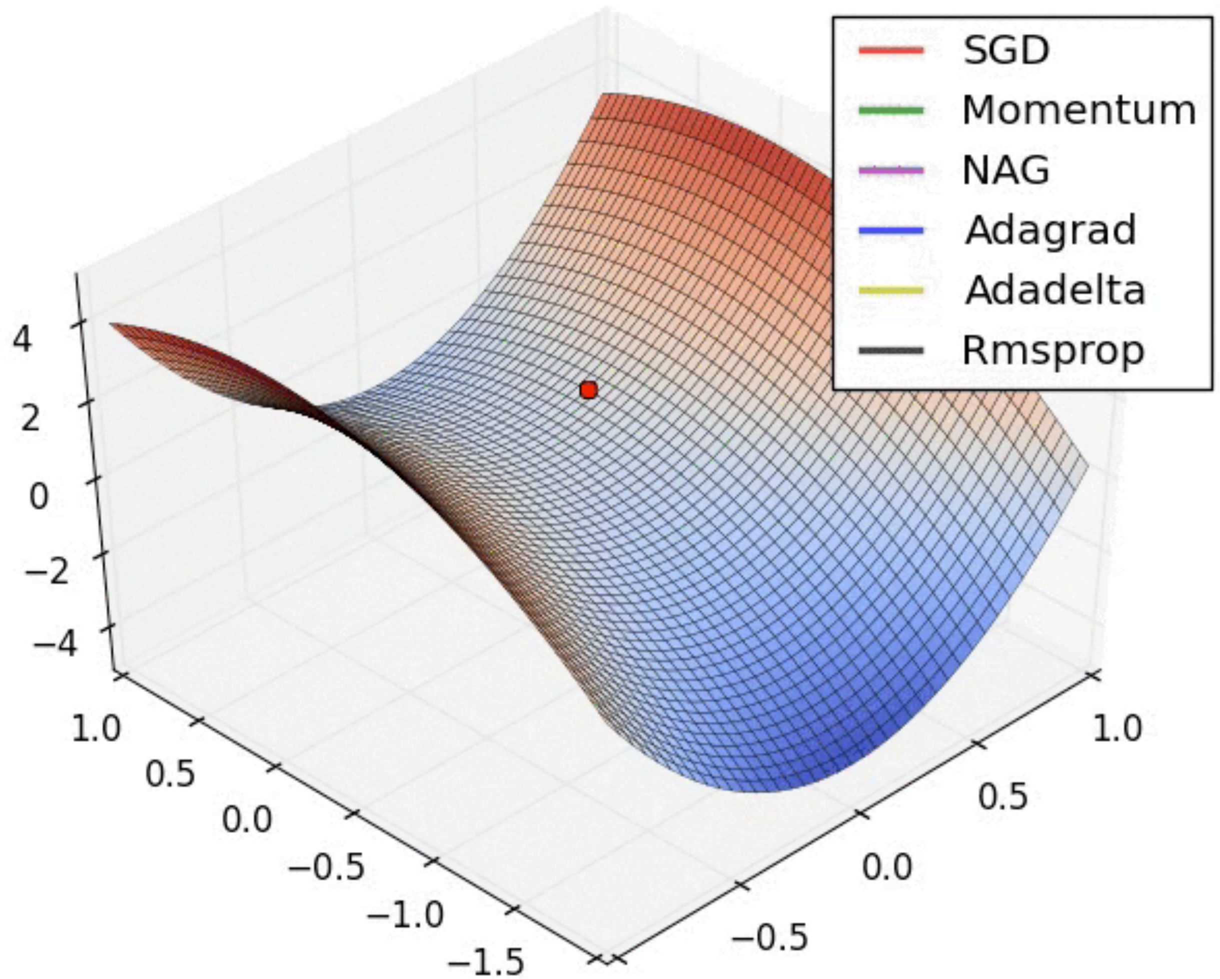
Many other variants of SGD

- Momentum SGD
- Nesterov Momentum
- AdaGrad
- Adam
- AdaDelta
- RMS prop

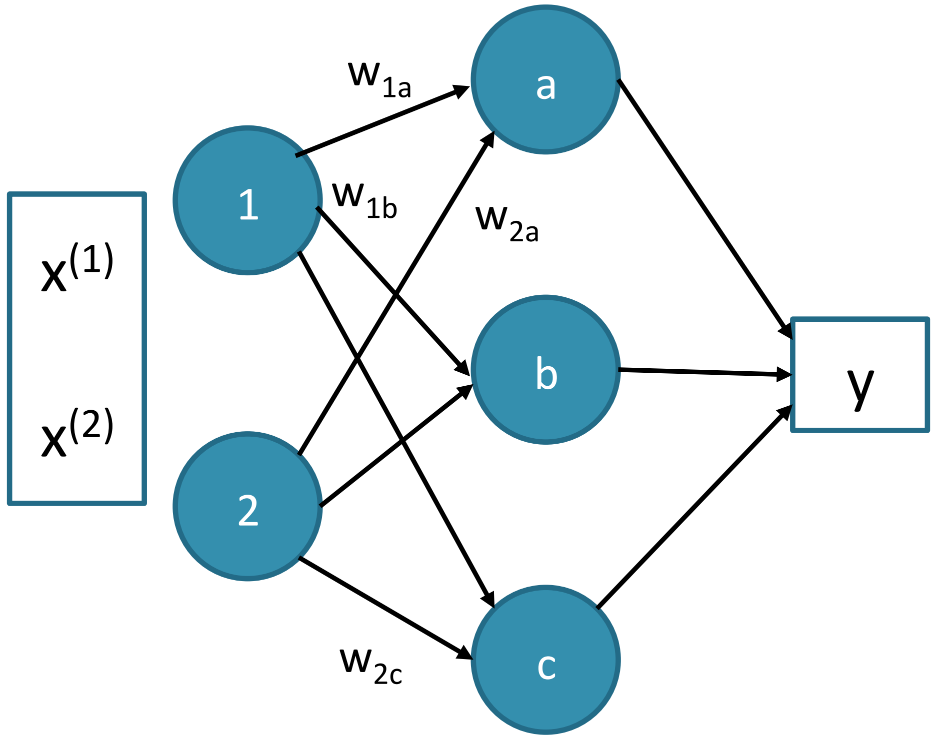
Many other variants of SGD



Many other variants of SGD

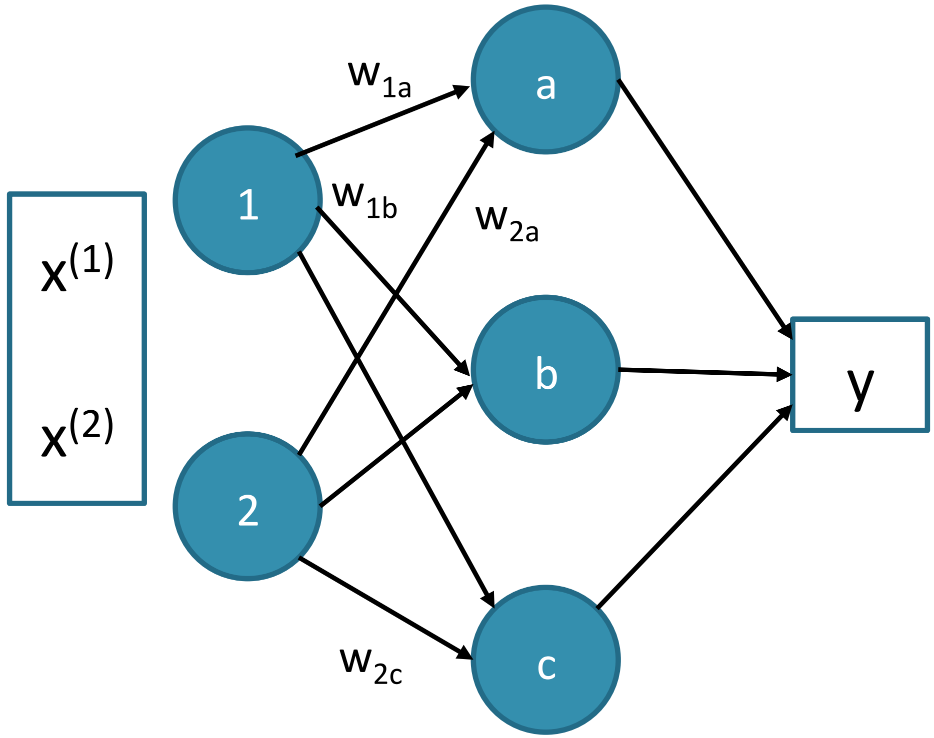


SGD and Backpropagation



Given a big dataset of $(\mathbf{x_1}, y_1)$, $(\mathbf{x_2}, y_2)$, $(\mathbf{x_3}, y_3)$, $(\mathbf{x_4}, y_4)$, ….$(\mathbf{x_N}, y_N)$
Find the optimal weights $\mathbf{w}$

SGD and Backpropagation

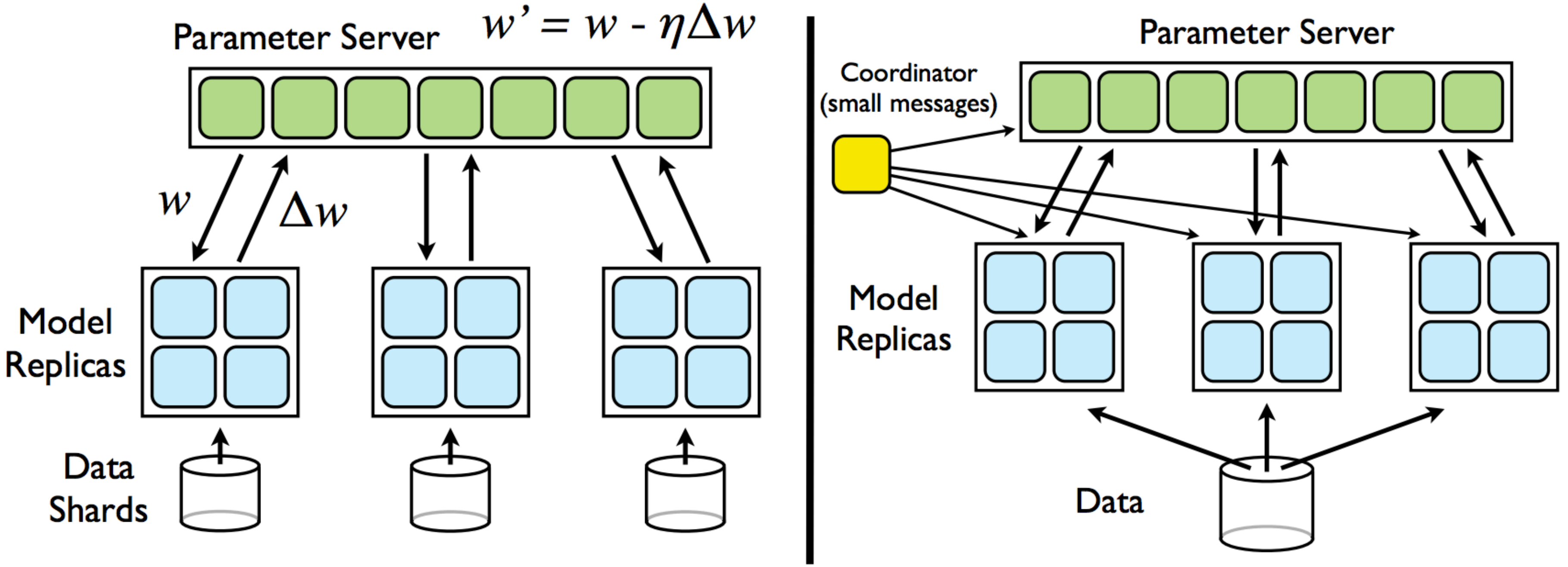


Input to a $=$ inp_a $=$ $w_{1a}\, x_1 + w_{2a}\, x_2$

Output of a $=$ $out_a = g\,(inp_a)$

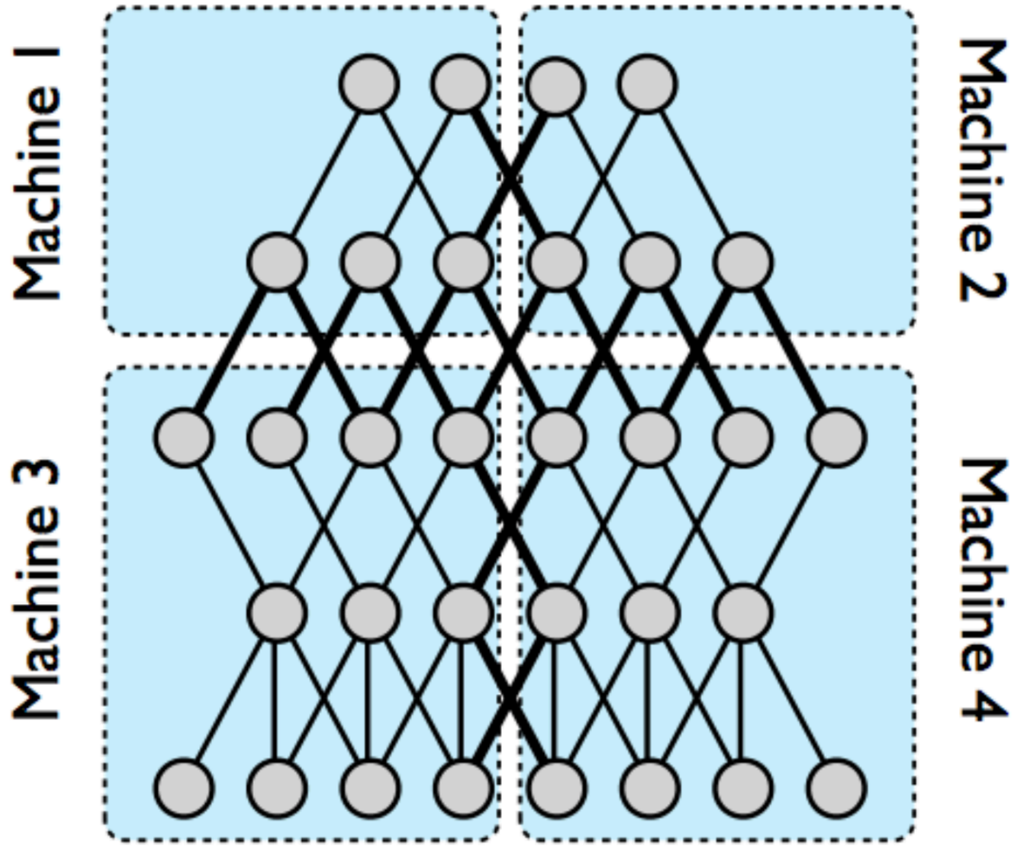
Distributed Deep Learning

Data Parallelism



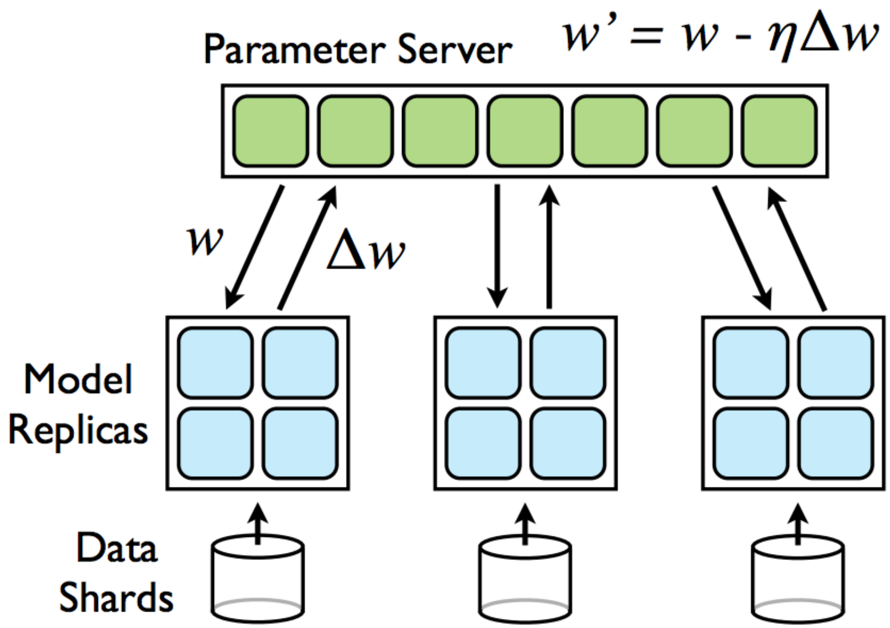
Distributed Deep Learning

Model Parallelism



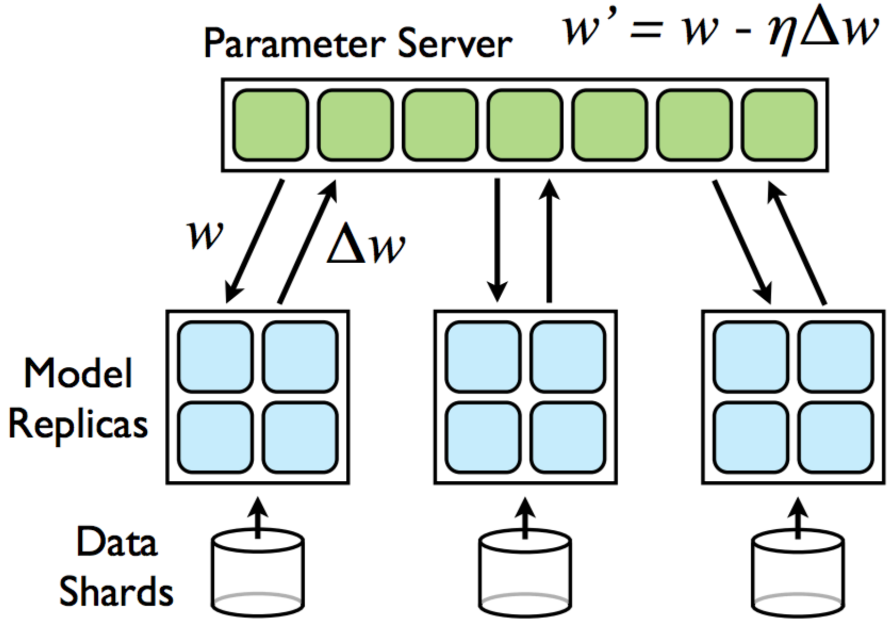
Synchronous SGD

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \sum_{k=1}^{K} \frac{1}{K} \nabla F(\mathbf{w}_t, \xi_k)$$



Q: What is the convergence rate and error floor?

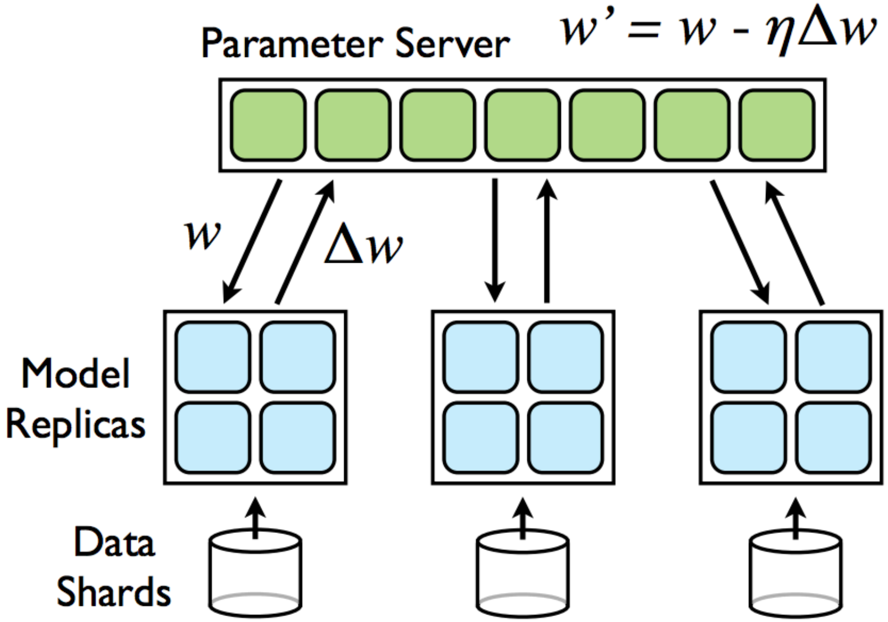
$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \sum_{k=1}^{K} \frac{1}{K} \nabla F(\mathbf{w}_t, \xi_k)$$

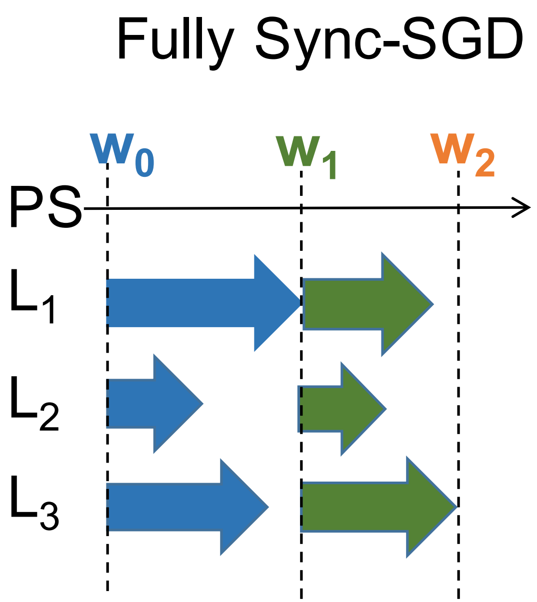


Q: What is the time to complete each iteration?

$$\mathbb{E}[T] = \mathbb{E}[\max(X_1, X_2, \ldots X_K)]$$

Slowest Learner is the bottleneck

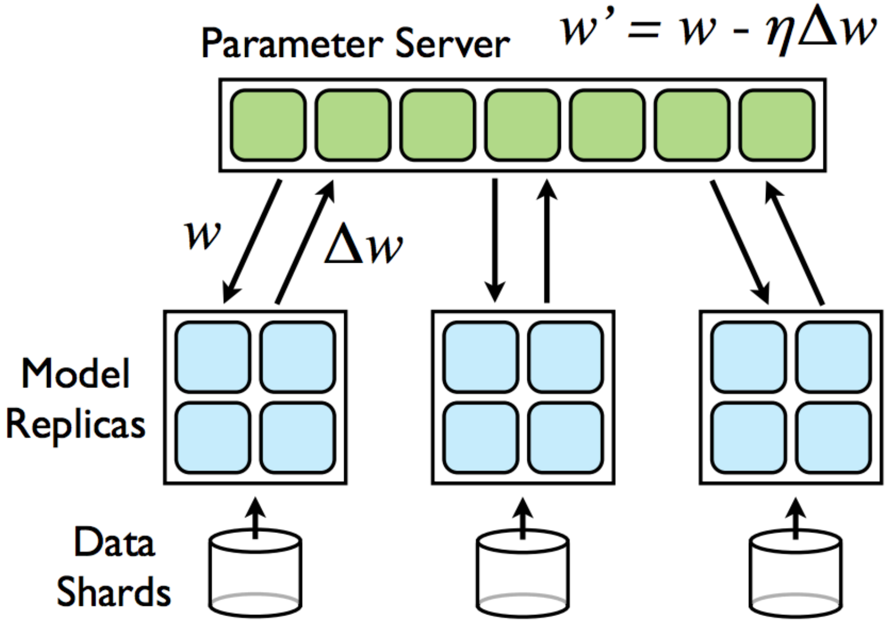




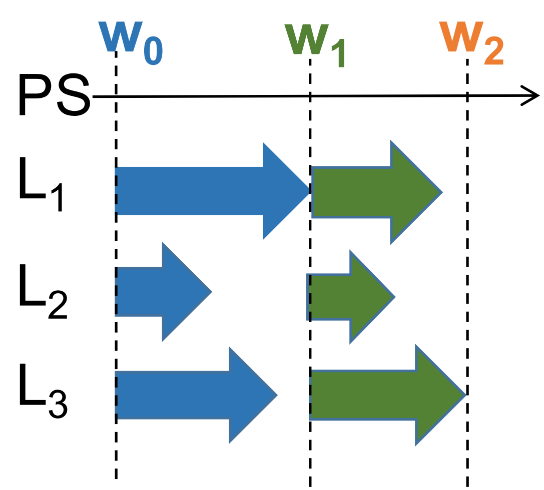
Q: How can we reduce it?

$$\mathbb{E}[T] = \mathbb{E}[\max(X_1, X_2, \ldots X_K)]$$

Slowest Learner is the bottleneck



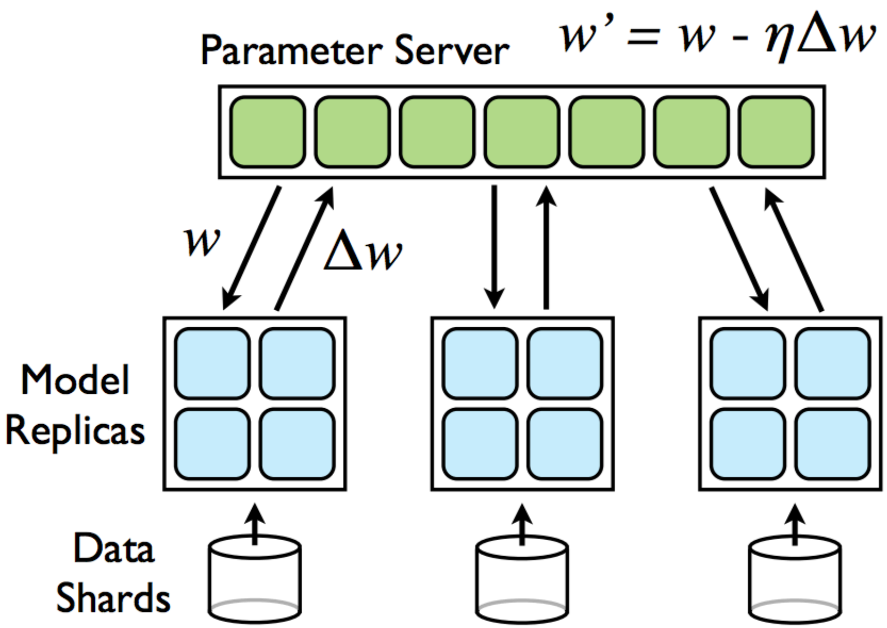
Fully Sync-SGD

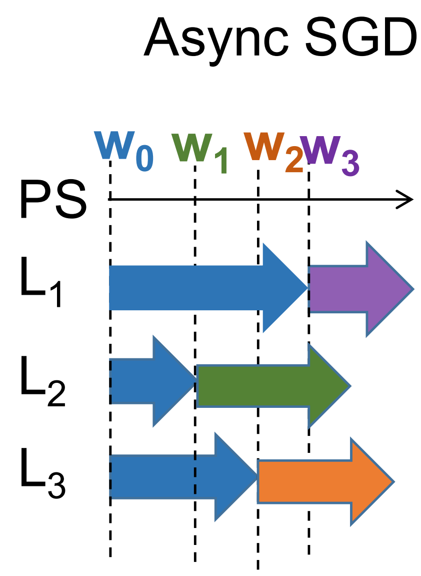


Asynchronous SGD: Don’t wait for all

Asynchronous SGD cuts the latency tail.

But, what effect does it have on the error?





Variants of Distributed SGD

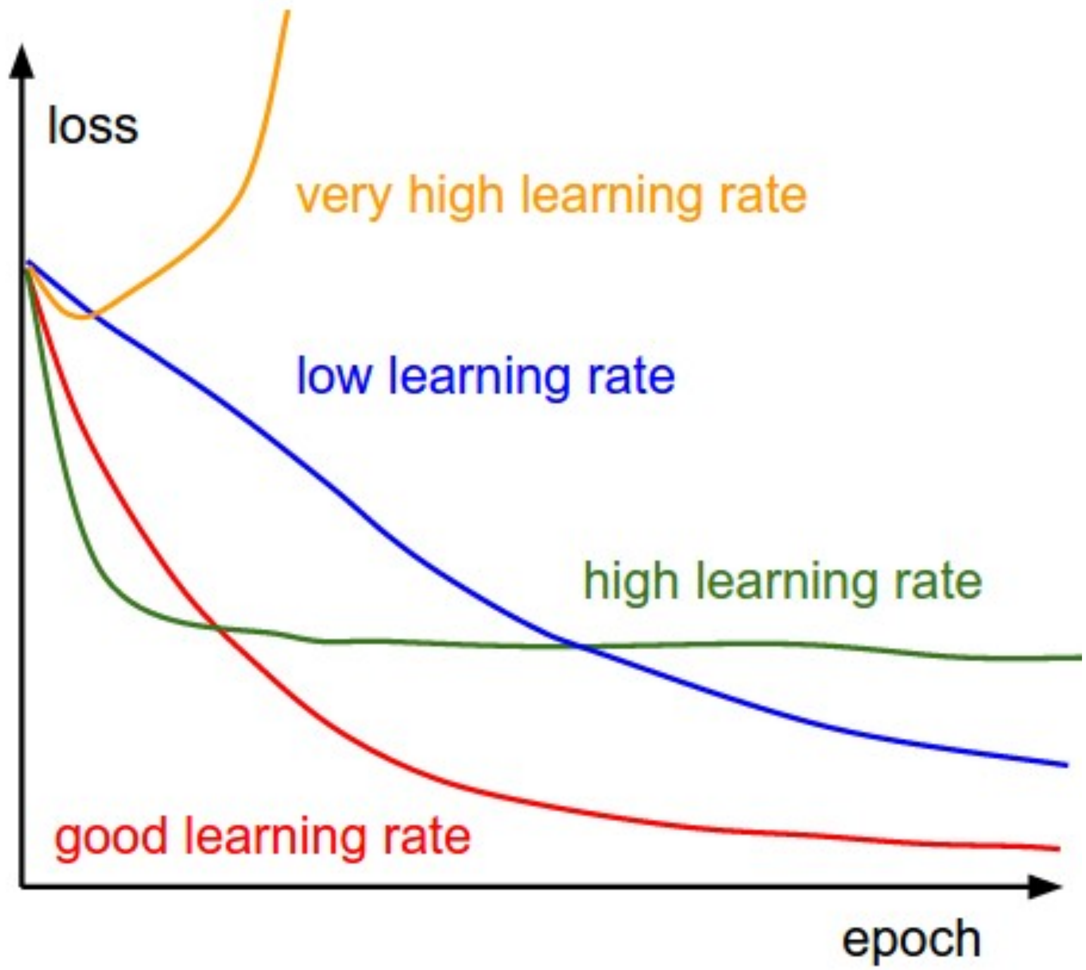
- Synchronous SGD
- Asynchronous SGD
- HogWild
- Elastic-Averaging SGD

Hyper-Parameter Tuning

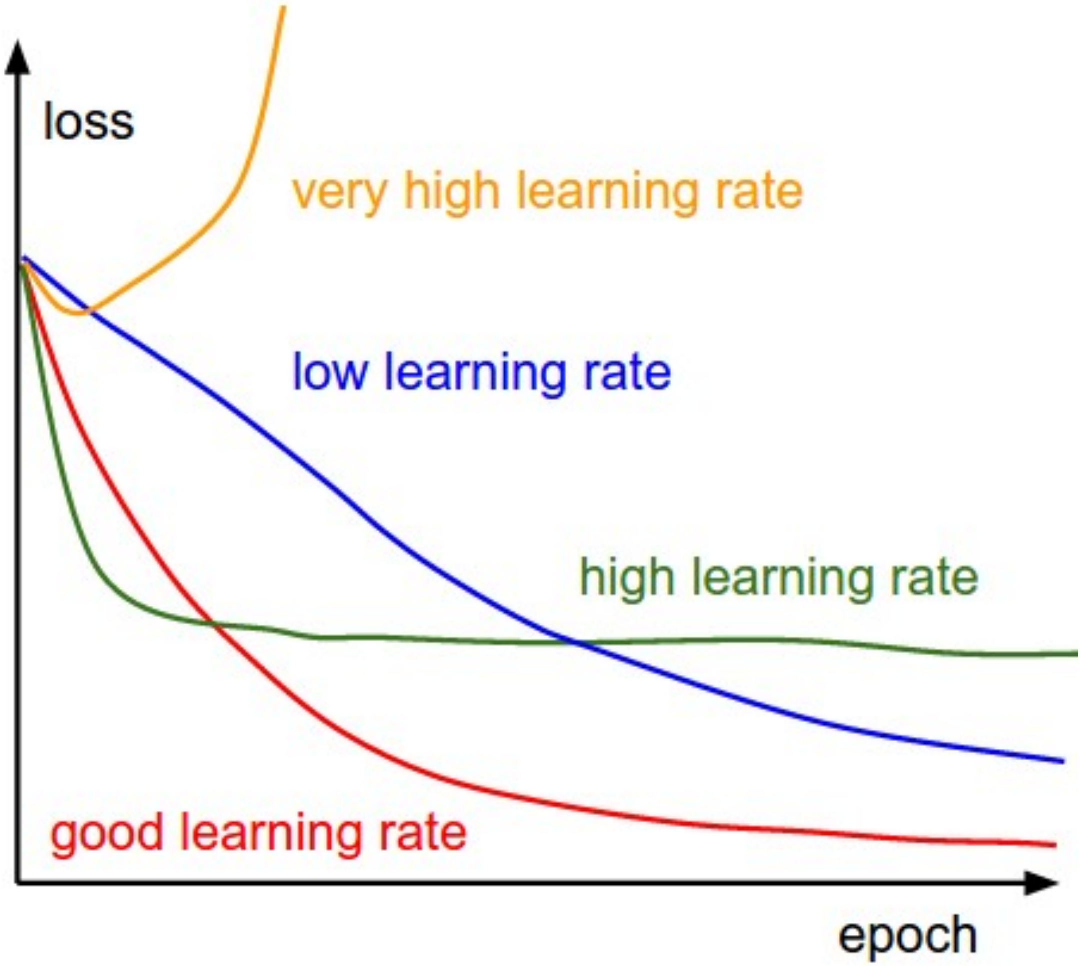
Need to choose the right

- Learning rate
- Mini-batch size
- Momentum
- Number of layers
- Number of neurons per layer

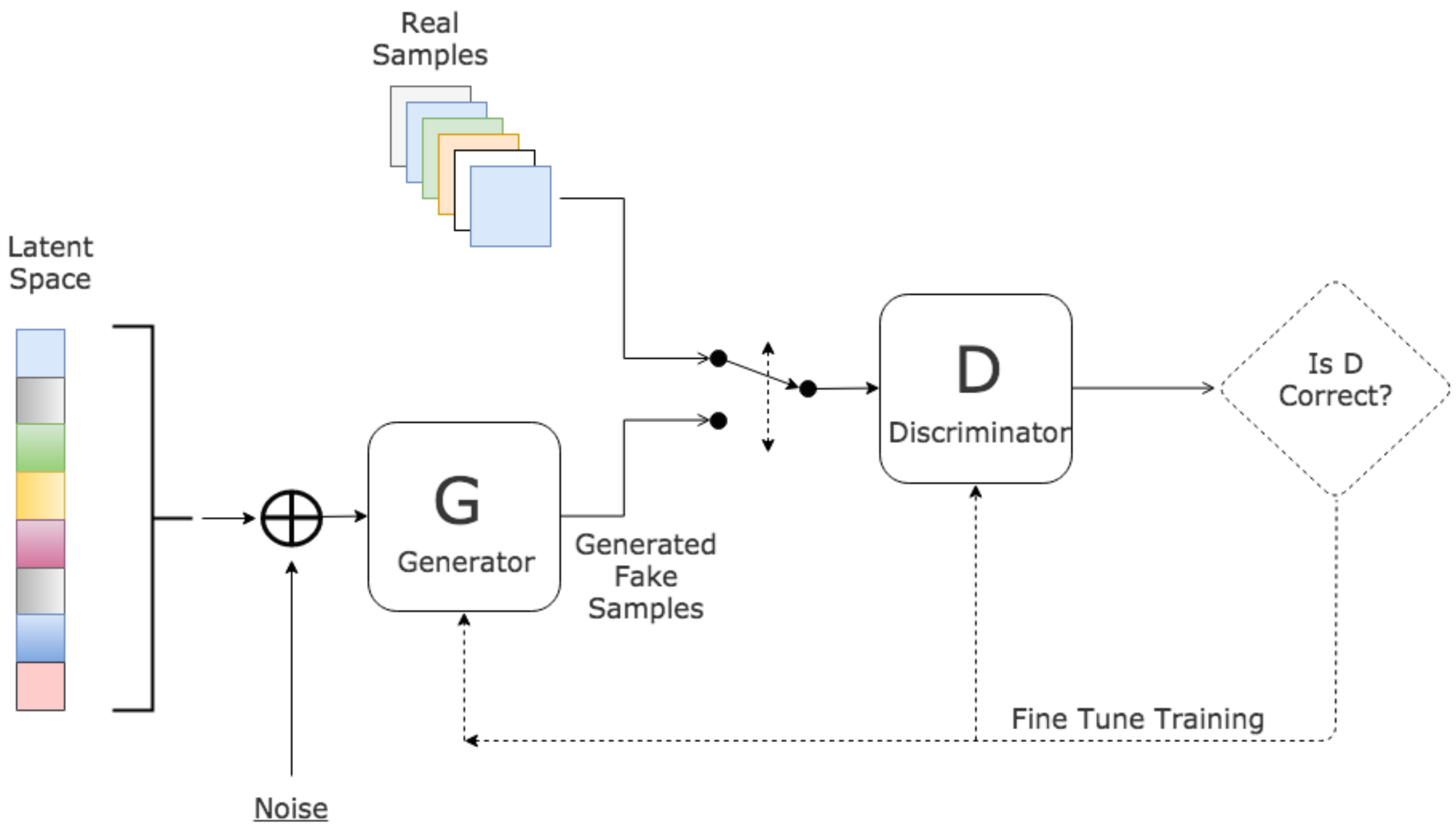
Hyper-Parameter Tuning



Multi-armed Bandits and Bayesian Optimization

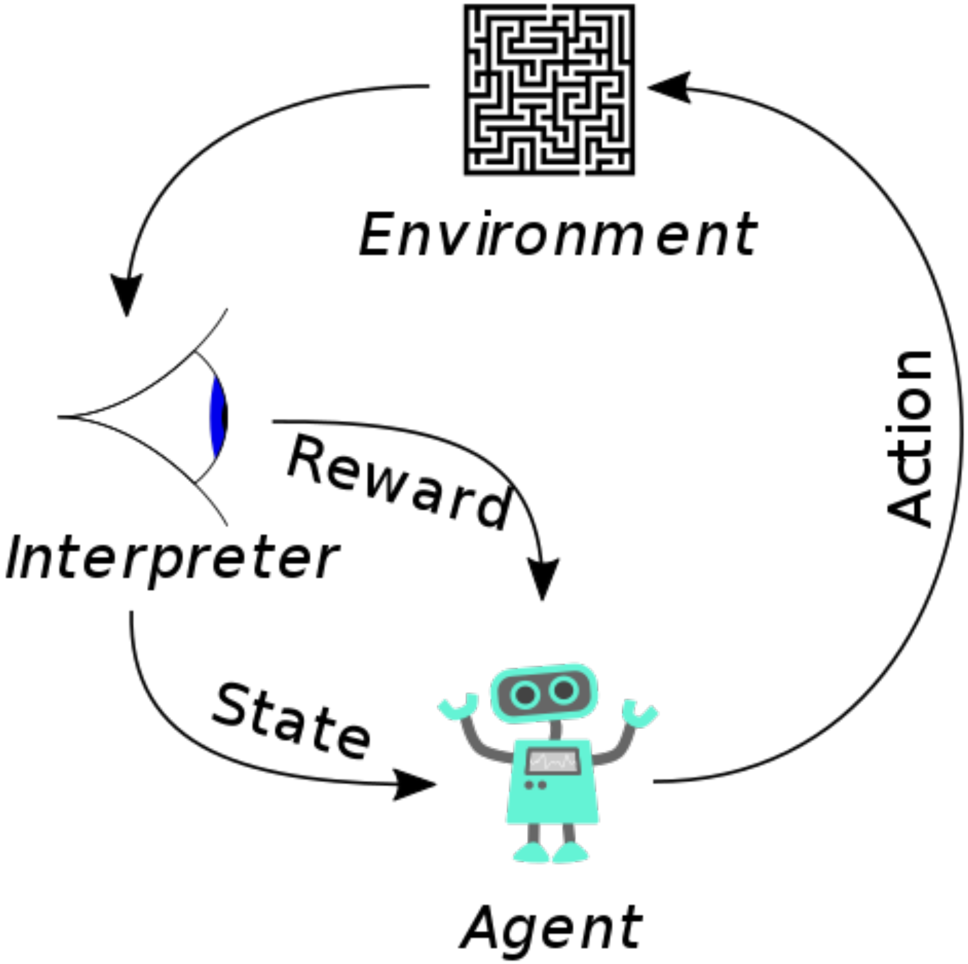


Generative Adversarial Networks

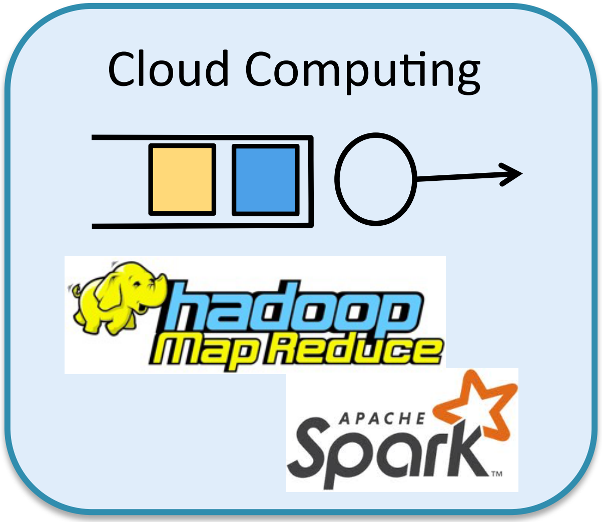


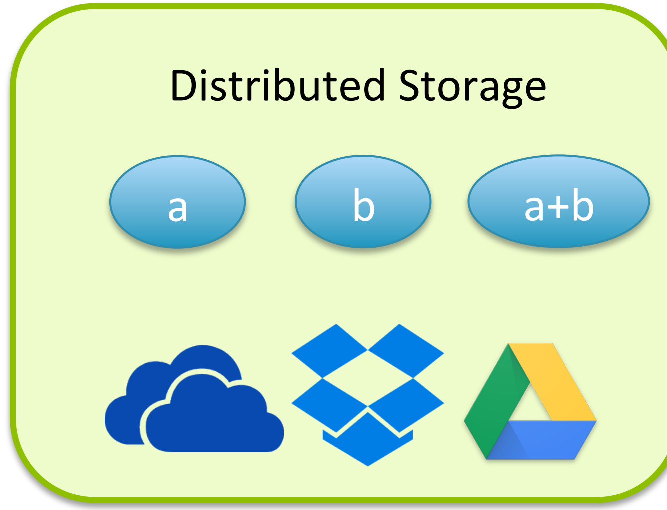
Reinforcement Learning

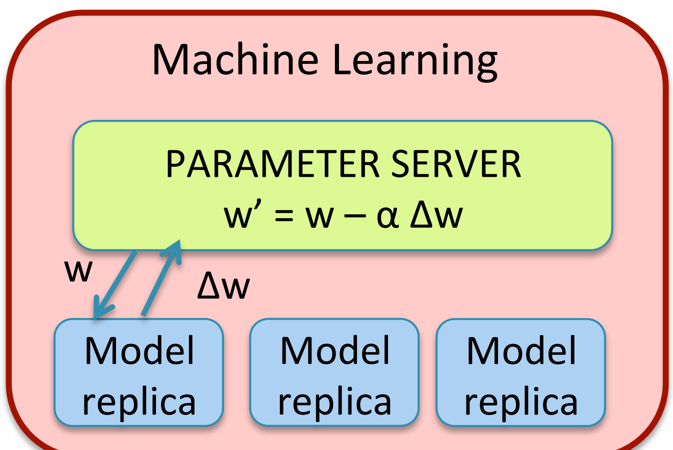
Reinforcement Learning



Topics Covered







TO DO

- Fill out the sign-up sheet
- Sign-up for presentations
- Start reading the papers
- Form groups for class projects
- Start thinking about projects