18-847F: Special Topics in Computer Systems

Foundations of Cloud and Machine Learning Infrastructure

Lecture 2: Overview and Key Concepts

Foundations of Cloud and Machine Learning Infrastructure

Graduate Seminar Class

(Almost) no lectures

Reading research papers

Student presentations

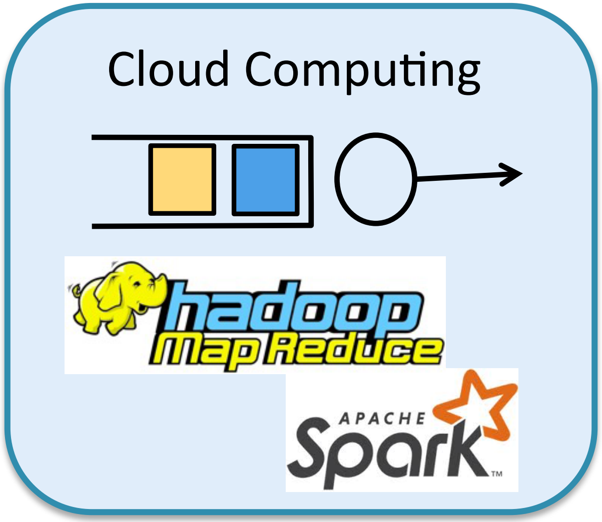
Class Discussions

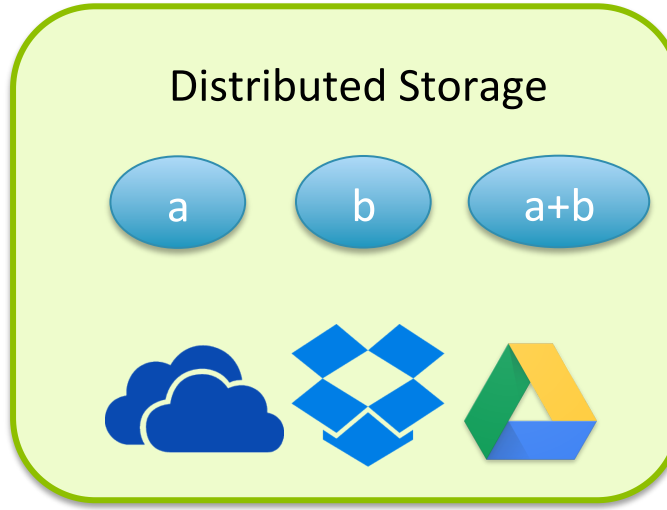
Final Research Project (No Exams!)

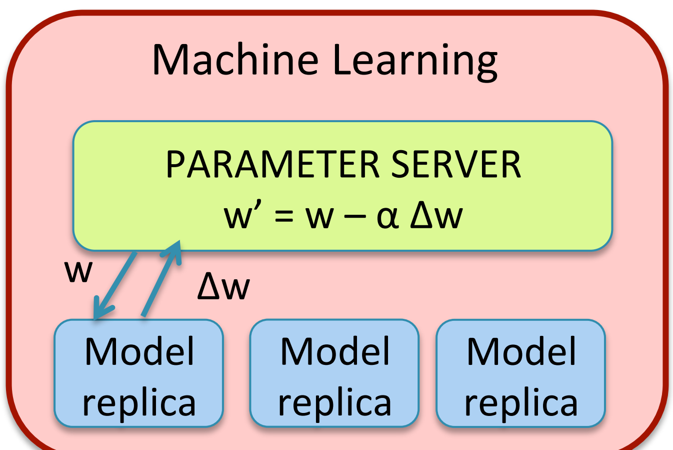
TO DO

- Sign-up for presentation
- Form groups for class projects
- Start thinking about projects

Topics Covered

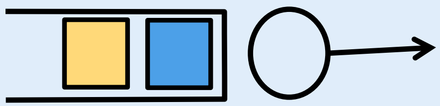




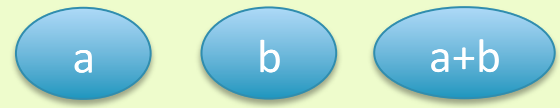


History and Overview

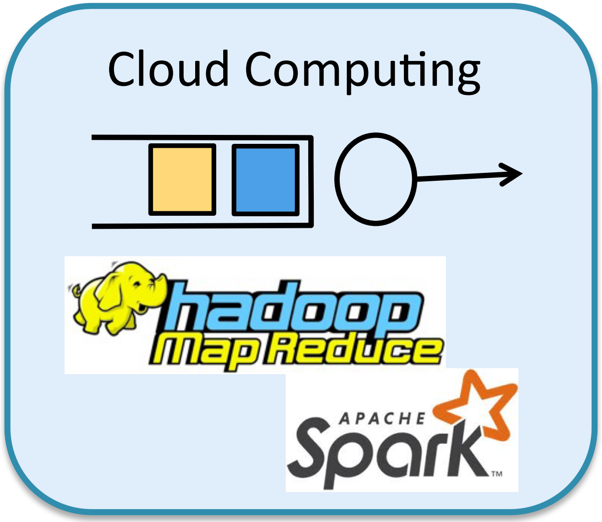
Cloud Computing



Distributed Storage



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)

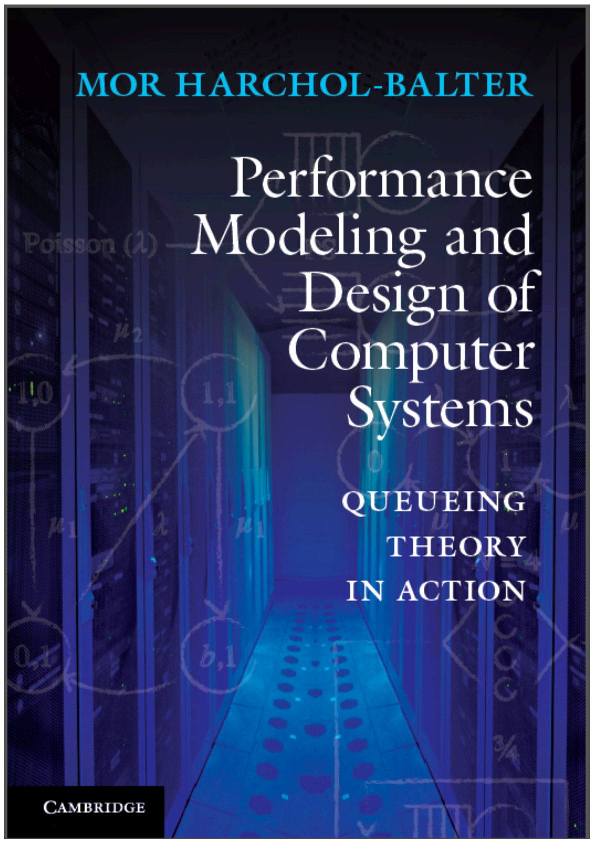
1980's-1990's: Internet and PCs

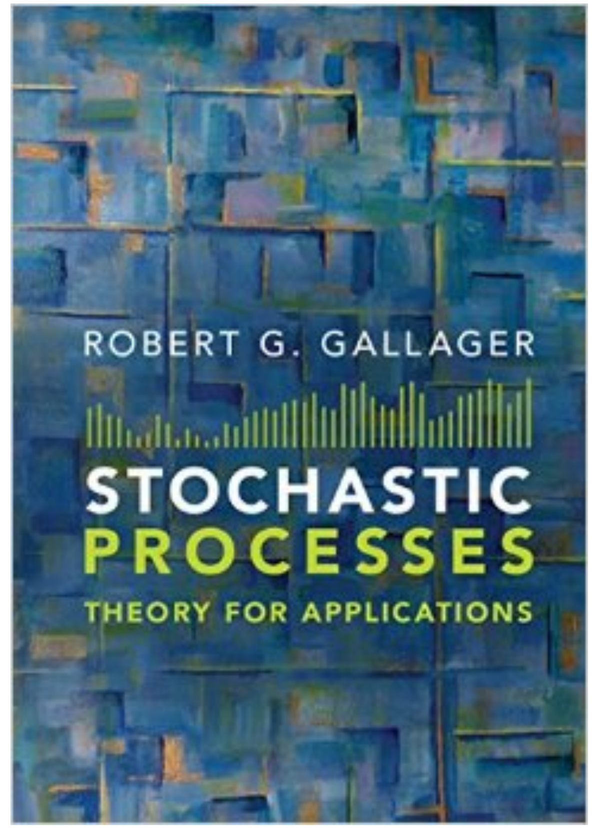
- PCs become affordable
- Internet connectivity went on improving

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

A Short Tutorial on Queueing Theory

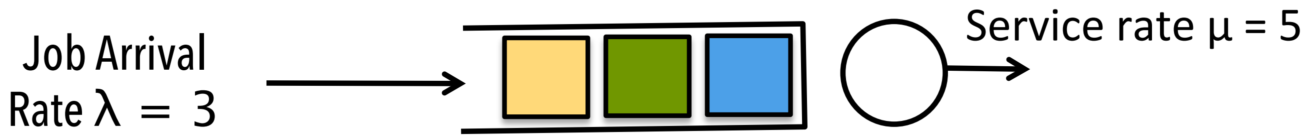
Reference Textbooks





Design Question 1

What if the arrival rate doubles?

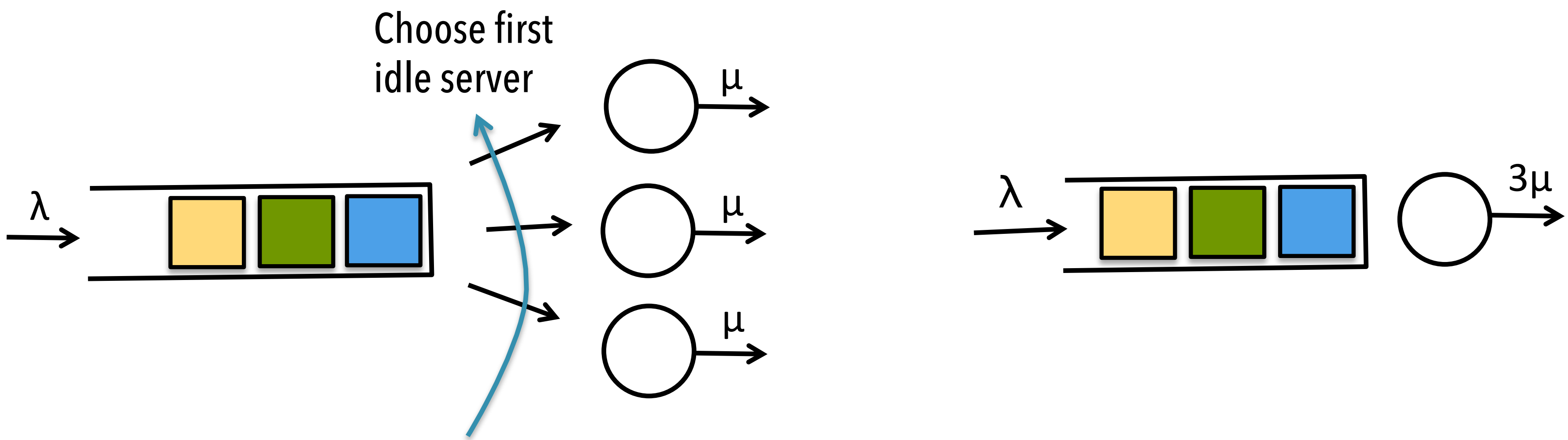


Mean Response Time T = Waiting time in Queue + Service Time

Q: If λ doubles, do you need a server of 2x rate to achieve the same E[T]?

Design Question 2

Many slow, or one fast server?

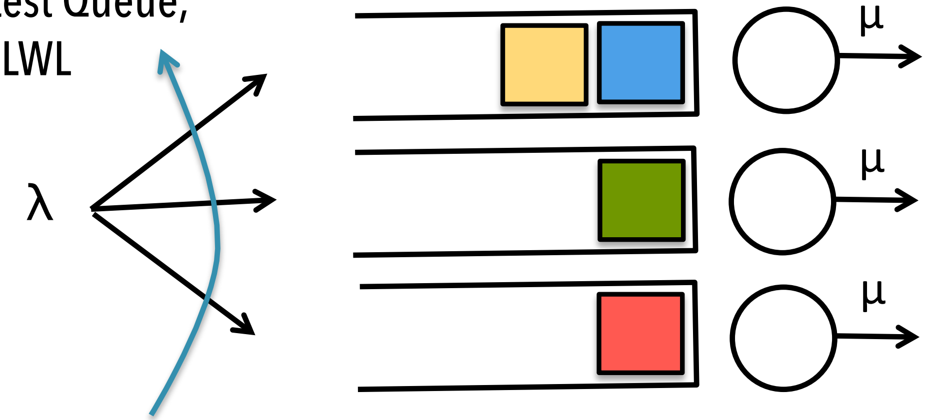


Q: Which of the two systems gives lower E[T]?

Design Question 3

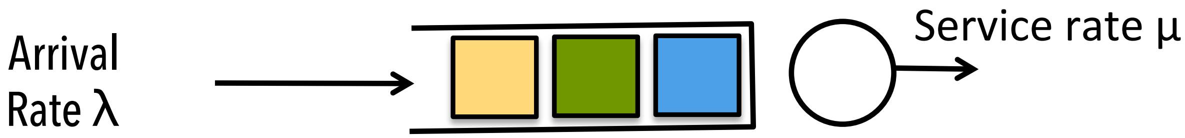
How to assign jobs to servers

Random, Round-robin,
Shortest Queue,
SITA, LWL



Q: Which policy works the best?

Queueing Terminology



Mean Service Time	$E[S] = 1/\mu$
Mean Waiting Time	$E[W]$
Mean Response Time	$E[T] = E[W] + E[S]$
Mean # Customers in Queue	$E[N]$
Server Utilization	$\rho = \lambda/\mu$

Little's Law

Theorem: For any ergodic open system we have

$$E[N] = \lambda E[T]$$

Very general and hence powerful law

- Any # of servers, scheduling policy, queue size limit

Some Variants

$$E[N_w] = \lambda E[W]$$

$$\rho = \lambda E[S]$$

Little's Law: Quiz

A professor takes 2 new students in even-numbered years, and 1 new student in odd-numbered years.

If avg. graduation time = 6 yrs, how many students will the professor have on average?

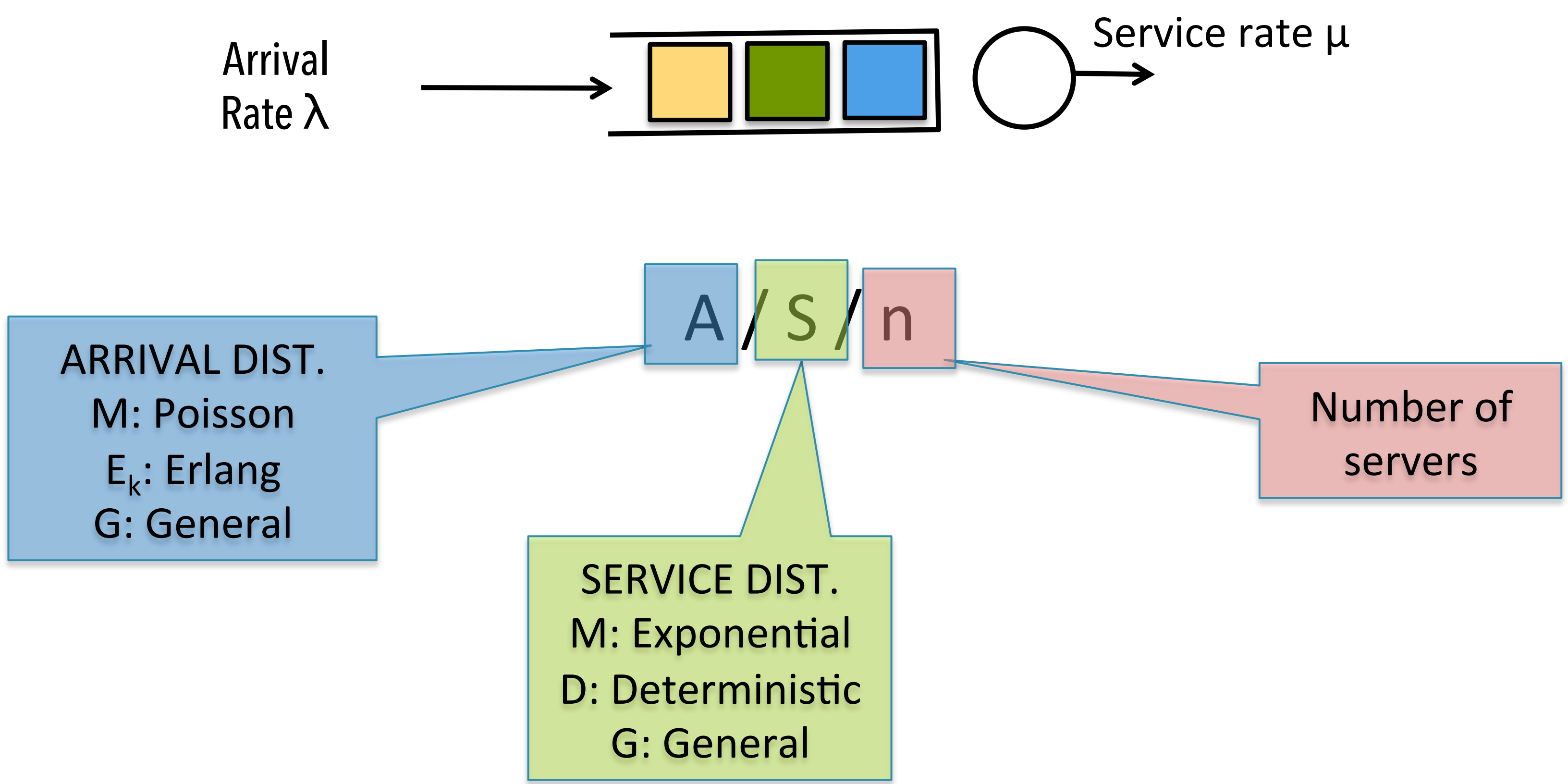
Little's Law: Answer

A professor takes 2 new students in even-numbered years, and 1 new student in odd-numbered years.

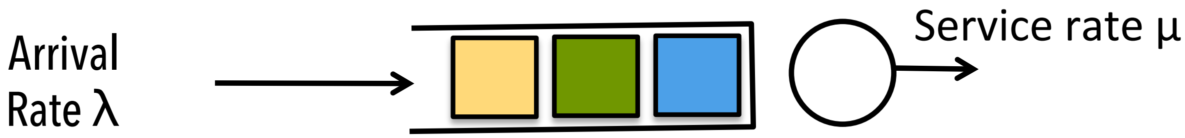
If avg. graduation time = 6 yrs, how many students will the professor have on average?

$$E[N] = \lambda E[T]$$
$$= 1.5 * 6$$
$$= 9$$

Kendall's Notation



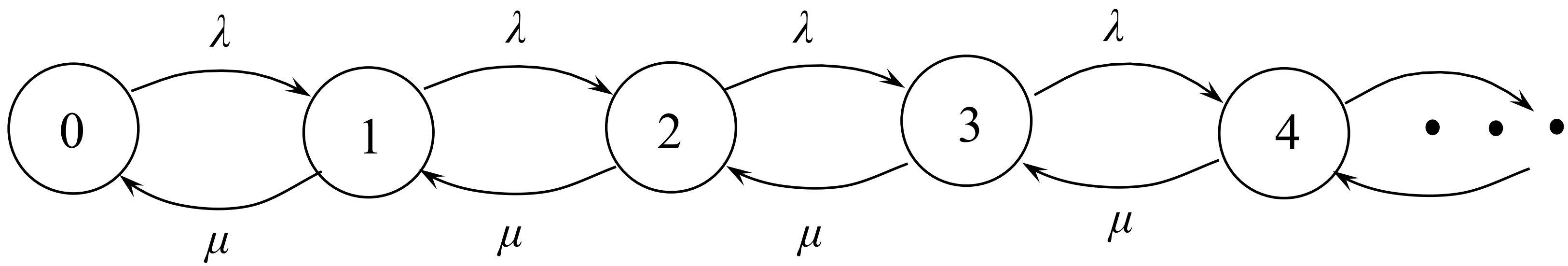
M/M/1 Queue



WANT TO FIND

1. Mean Response Time E[T]
2. Mean Waiting Time E[W]

M/M/1: Markov Model

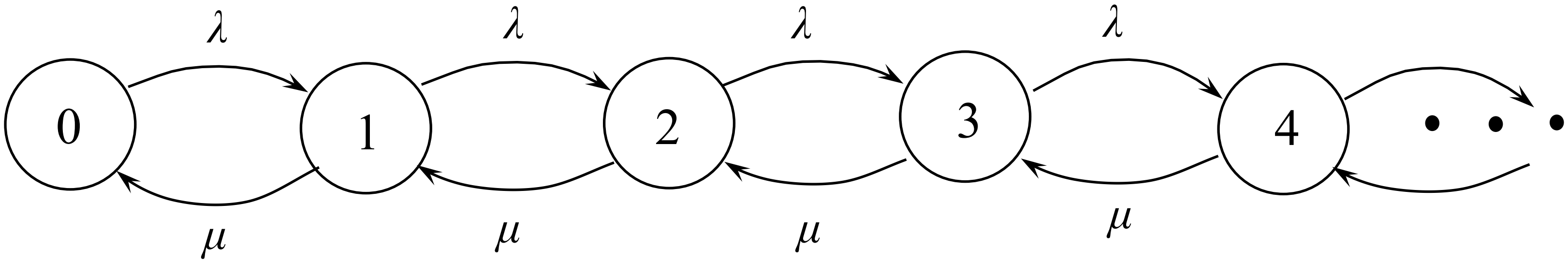


$$\pi_i = \rho^i(1-\rho)$$
$$\pi_0 = (1-\rho)$$

where $\rho = \dfrac{\lambda}{\mu}$

$$\mathbb{E}[N] = \sum_{i=0}^{\infty} i\pi_i = \rho(1-\rho)\sum_{i=1}^{\infty} i\rho^{i-1} = \frac{\rho}{1-\rho}$$

M/M/1: Mean Response Time



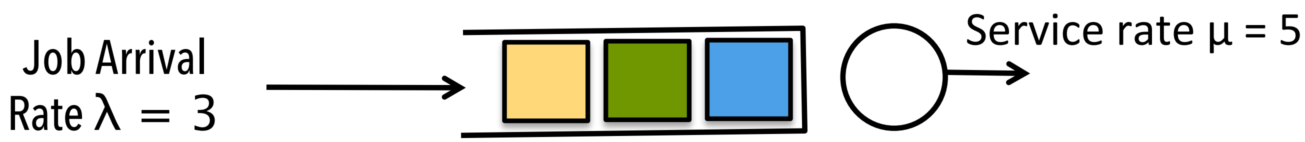
$$\mathbb{E}[N] = \sum_{i=0}^{\infty} i\pi_i = \rho(1-\rho)\sum_{i=1}^{\infty} i\rho^{i-1} = \frac{\rho}{1-\rho}$$

$$\mathbb{E}[T] = \frac{\mathbb{E}[N]}{\lambda} = \frac{\rho}{\lambda(1-\rho)} = \boxed{\frac{1}{\mu-\lambda}}$$

$$\mathbb{E}[W] = \frac{1}{\mu-\lambda} - \frac{1}{\mu} = \boxed{\frac{\rho}{\mu-\lambda}}$$

Quiz: Design Question 1

What if the arrival rate doubles?

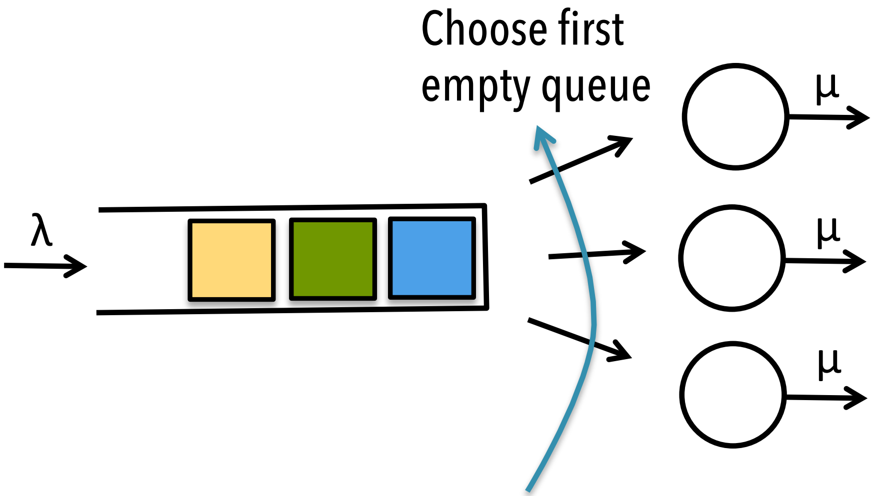


Mean Response Time T = Waiting time in Queue + Service Time

Q: If λ doubles, do you need a server of 2x rate to achieve the same E[T]?

A: Service rate 6+2 = 8 is sufficient

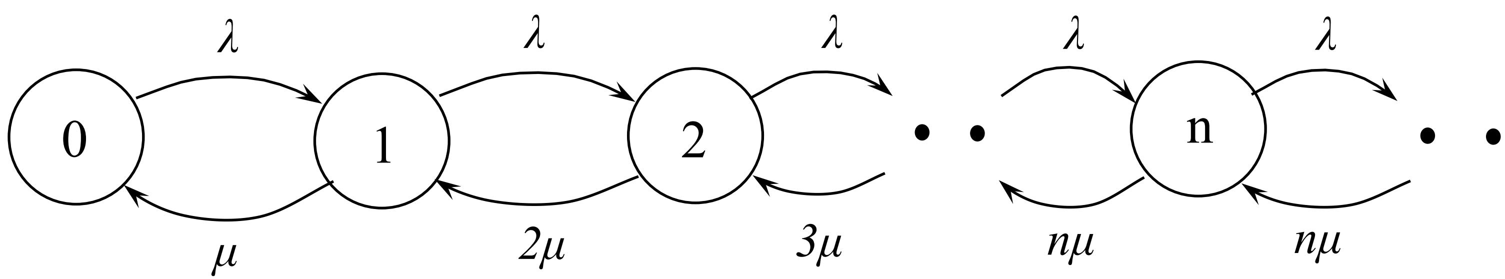
M/M/n Queue



WANT TO FIND

1. Mean Response Time E[T]
2. Mean Waiting Time E[W]

M/M/n Queue



$$P_Q = \sum_{i=n}^{\infty} \pi_i$$

$$\rho = \frac{\lambda}{n\mu}$$

$$= \pi_0 \frac{n^n}{n!} \sum_{i=n}^{\infty} \rho^i$$ where $\pi_0 = \left[\sum_{i=0}^{n-1} \frac{(n\rho)^i}{i!} + \frac{(n\rho)^n}{n!(1-\rho)} \right]^{-1}$

$$= \frac{n^n \pi_0}{n!(1-\rho)}$$ Erlang-C Formula

Used in call centers to determine number of agents required

M/M/n Queue

$$\mathbb{E}[N_w] = \sum_{i=n}^{\infty} \pi_i (i-n)$$

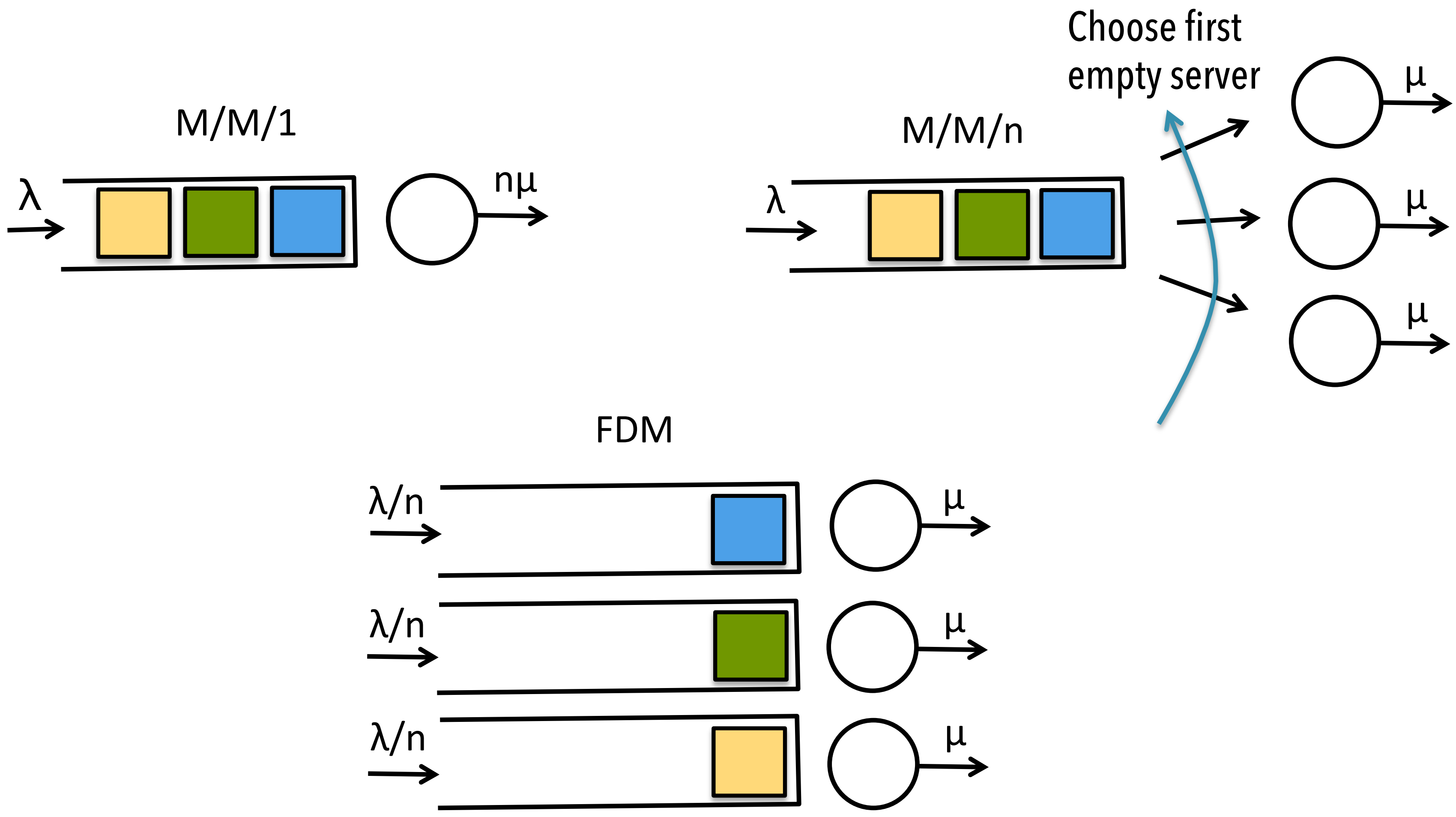
$$= \pi_0 \sum_{i=n}^{\infty} \frac{\rho^i n^n}{n!}(i-n)$$

$$= P_Q \frac{\rho}{1-\rho}$$

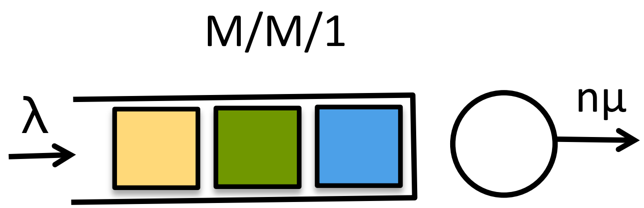
$$\mathbb{E}[W] = \frac{\mathbb{E}[N_w]}{\lambda} = P_Q \frac{\rho}{\lambda(1-\rho)}$$

$$\mathbb{E}[T] = P_Q \frac{\rho}{\lambda(1-\rho)} + \frac{1}{\mu}$$

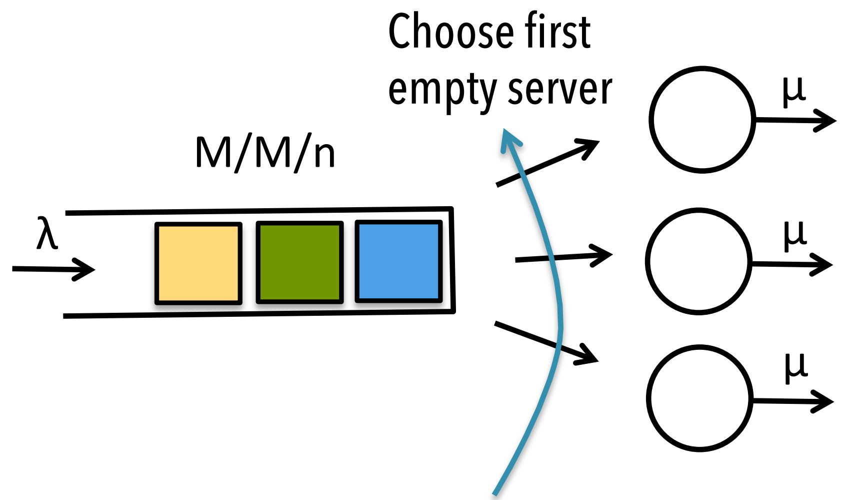
Quiz: Comparison of 3 systems

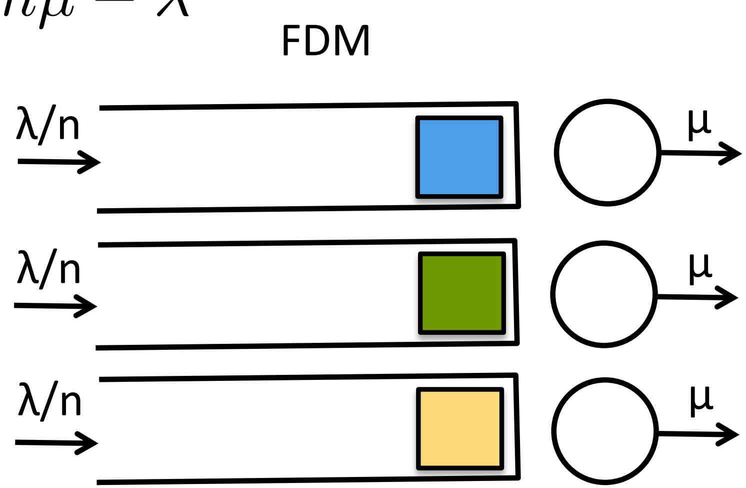


Quiz: Comparison of 3 systems



$$\mathbb{E}[T]^{M/M/1} = \frac{1}{n\mu - \lambda}$$

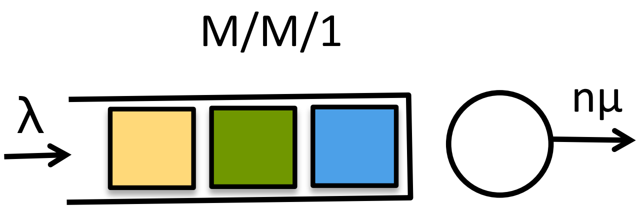


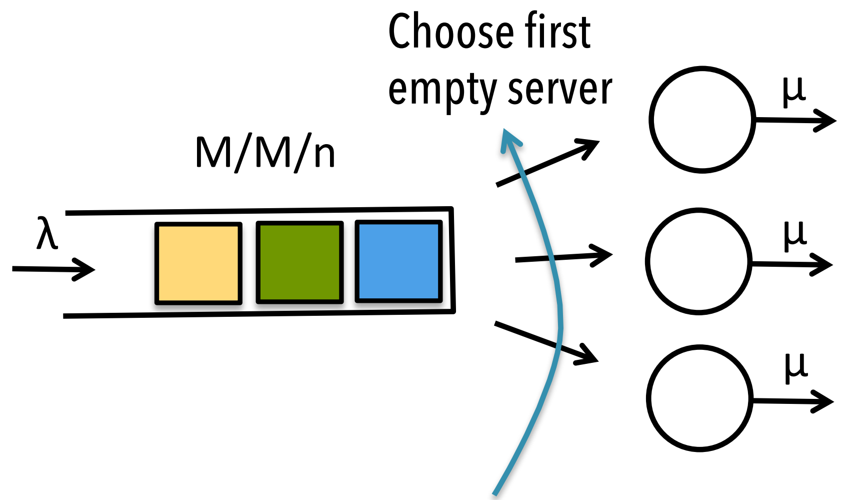


FDM is n times slower than M/M/1

$$\mathbb{E}[T]^{FDM} = \frac{n}{n\mu - \lambda}$$

Quiz: Comparison of 3 systems





$$\mathbb{E}[T]^{M/M/1} = \frac{\rho}{\lambda(1-\rho)}$$

$$\mathbb{E}[T]^{M/M/n} = P_Q \frac{\rho}{\lambda(1-\rho)} + \frac{1}{\mu}$$

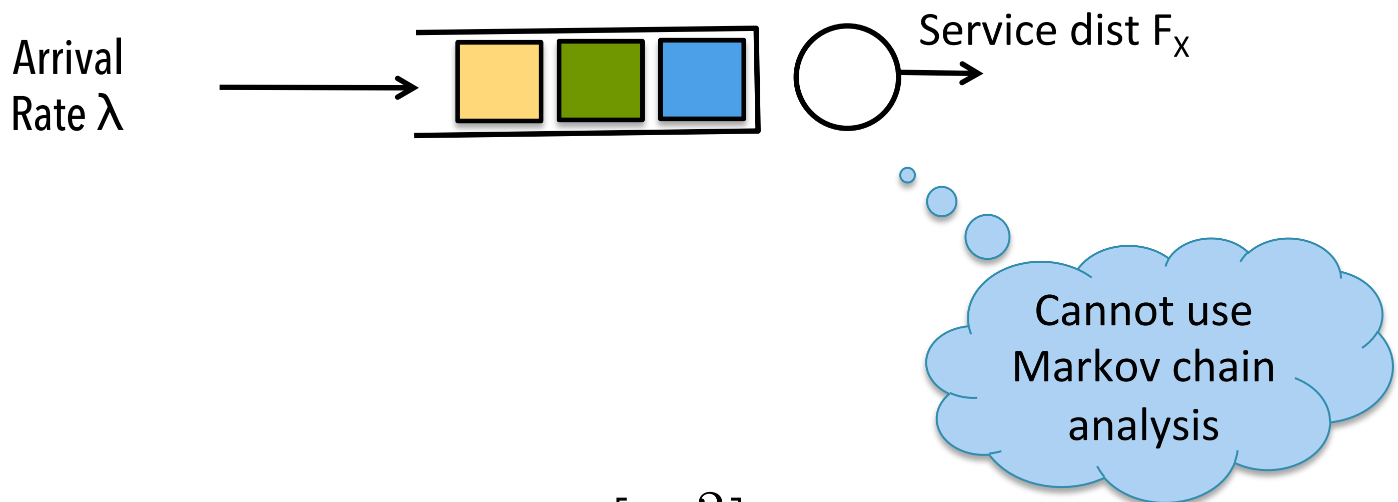
M/M/n is n times slower when ρ→0

$$\frac{\mathbb{E}[T]^{M/M/n}}{\mathbb{E}[T]^{M/M/1}} = P_Q + n(1-\rho)$$

M/M/n and M/M/1 are almost equal when ρ→ 1

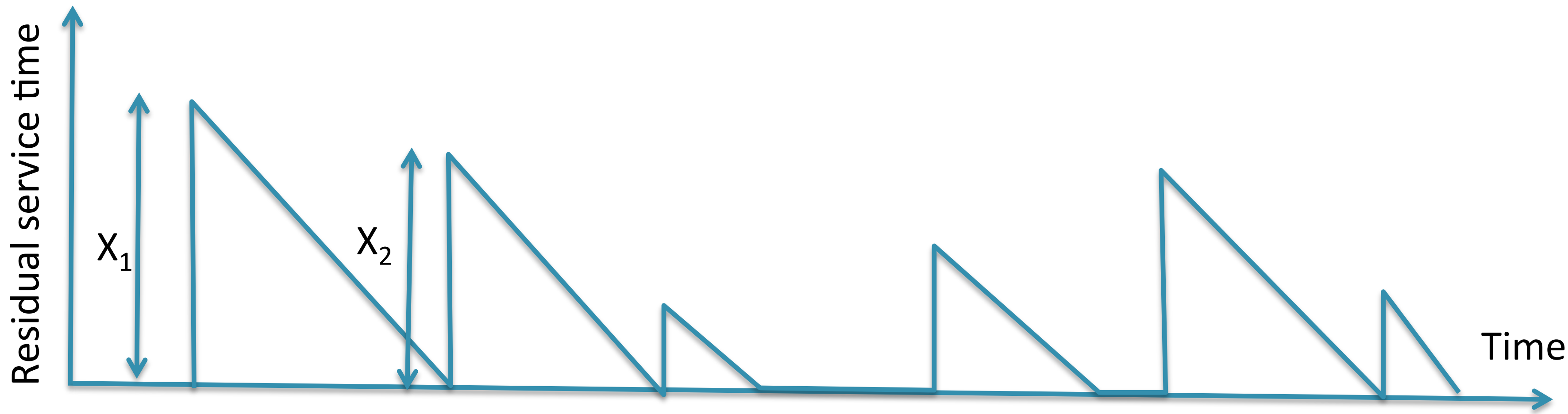
M/G/1 Queue

Pollaczek-Khinchine Formula



$$\mathbb{E}[T] = \mathbb{E}[X] + \frac{\mathbb{E}[X^2]}{2(1 - \lambda \mathbb{E}[X])}$$

Proof of PK formula

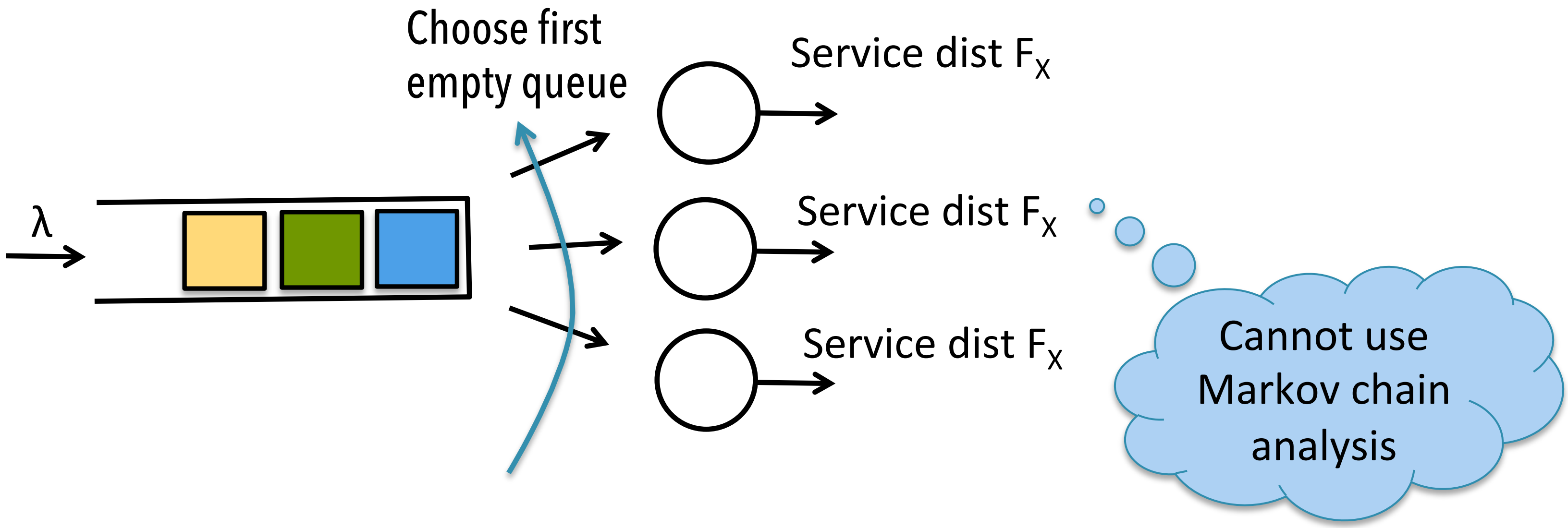


$$\mathbb{E}[T_w] = \mathbb{E}[N_w] \cdot \mathbb{E}[X] + E[R]$$

$$= \lambda \mathbb{E}[T_w] \cdot \mathbb{E}[X] + \frac{\mathbb{E}[X^2]}{2}$$

$$= \frac{\mathbb{E}[X^2]}{2(1 - \lambda \mathbb{E}[X])}$$

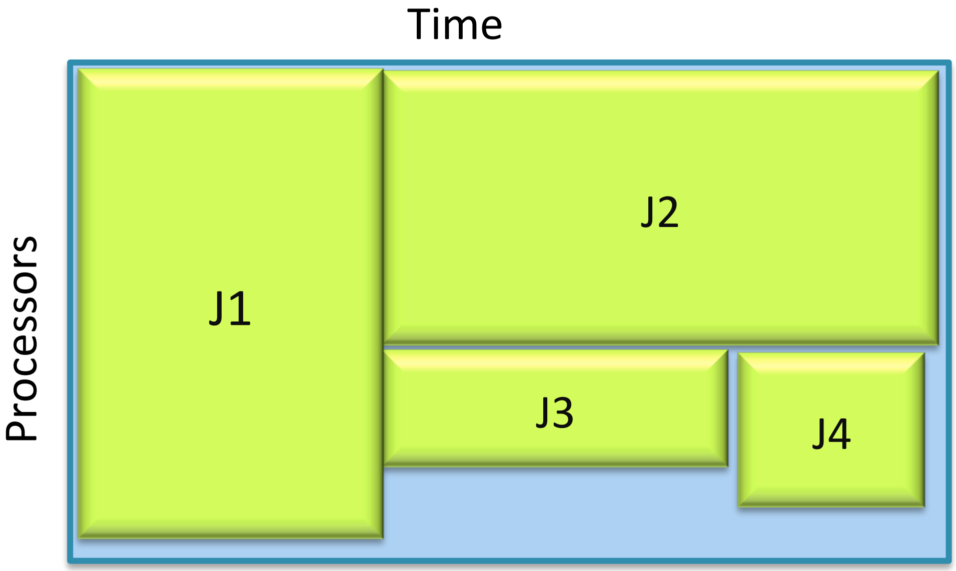
M/G/n Queue



$$\mathbb{E}[T] \approx \mathbb{E}[X] + \frac{\mathbb{E}[X^2]}{2\mathbb{E}[X]} \cdot \mathbb{E}[W^{M/M/n}]$$

1990's: Scheduling in Parallel Computing

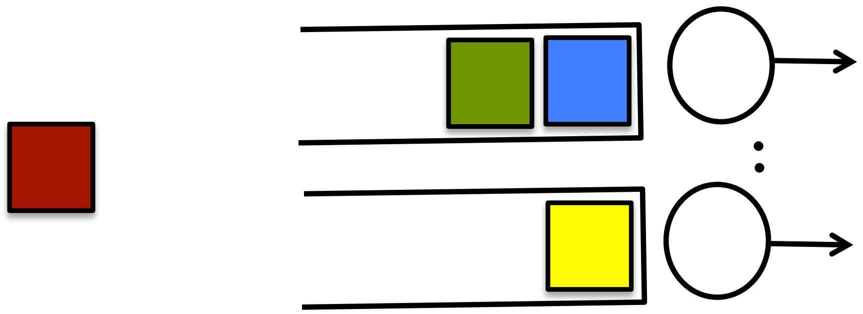
- **Bin-Packing**
  - Need job size estimates



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

amazon web services

Google Cloud Platform

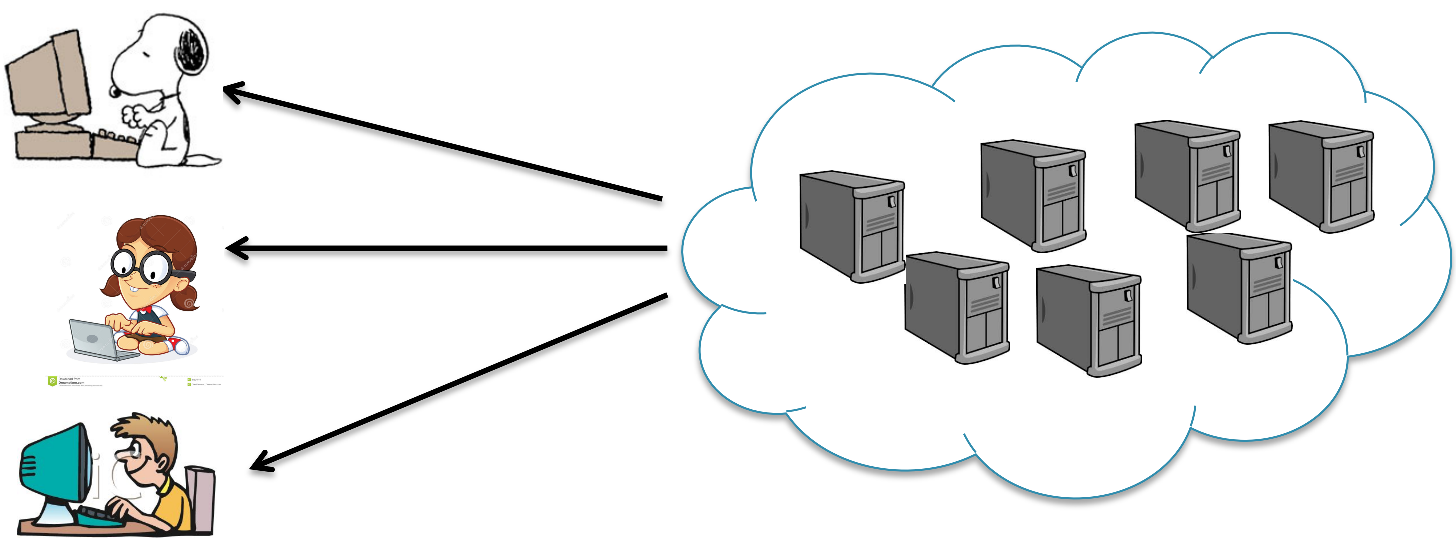
- Computing become a utility, like electricity

2000's: The Cloud Computing Era

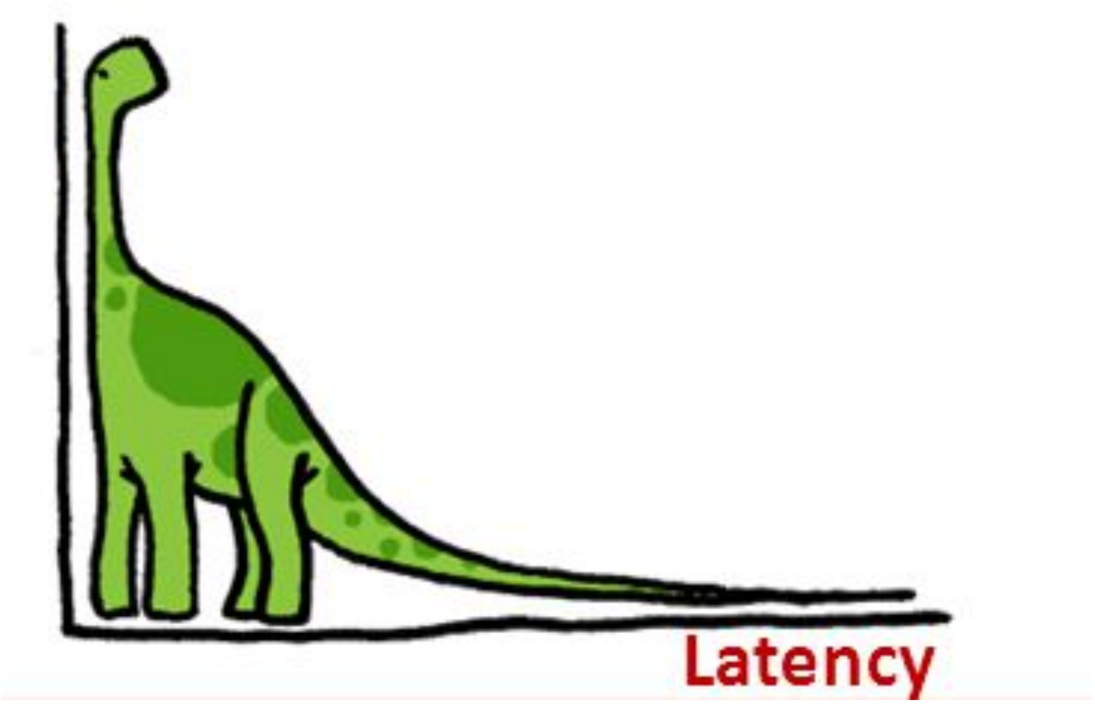
KEY ISSUE: Job sizes, server speeds & queue lengths are unpredictable

REASON: Large-scale resource sharing → Variability in service

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

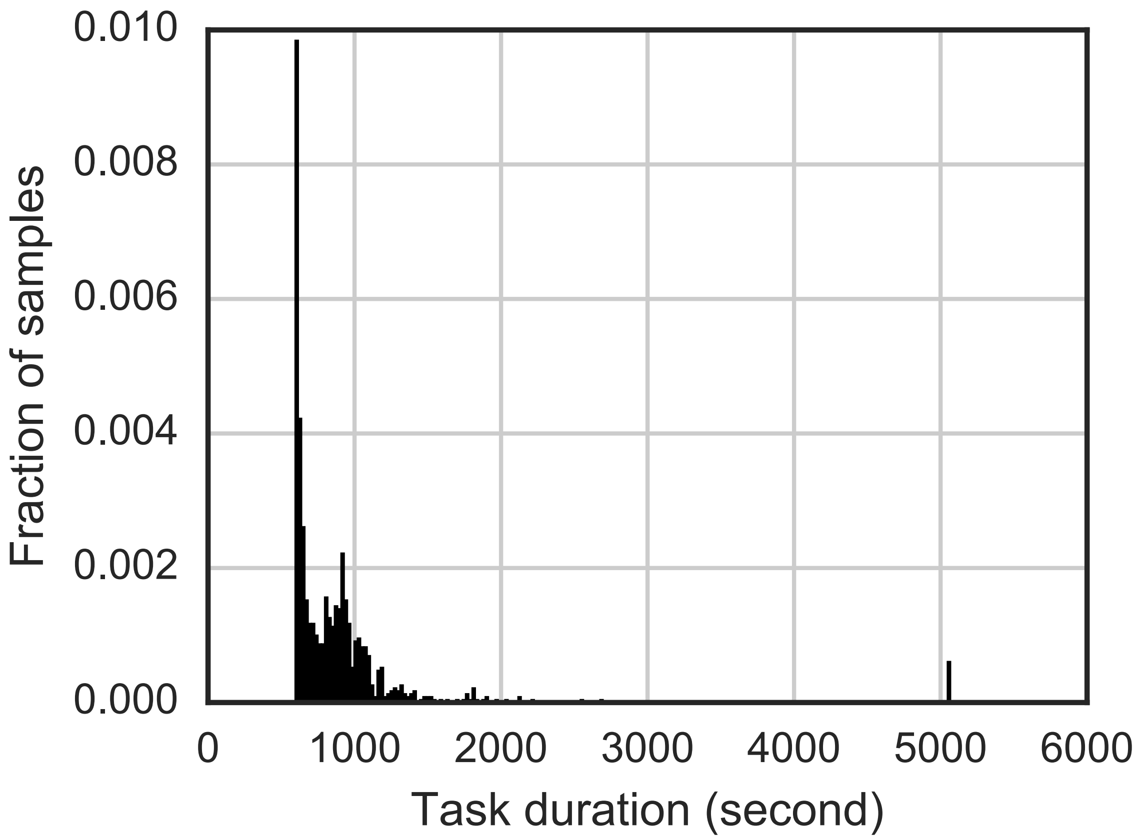


The Tale of Tails



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

Tale of Tails: Quiz

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.

Tale of Tails: Quiz

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.

Tale of Tails: Quiz

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?

  $1*0.9 + 10*0.1 = 1.9$

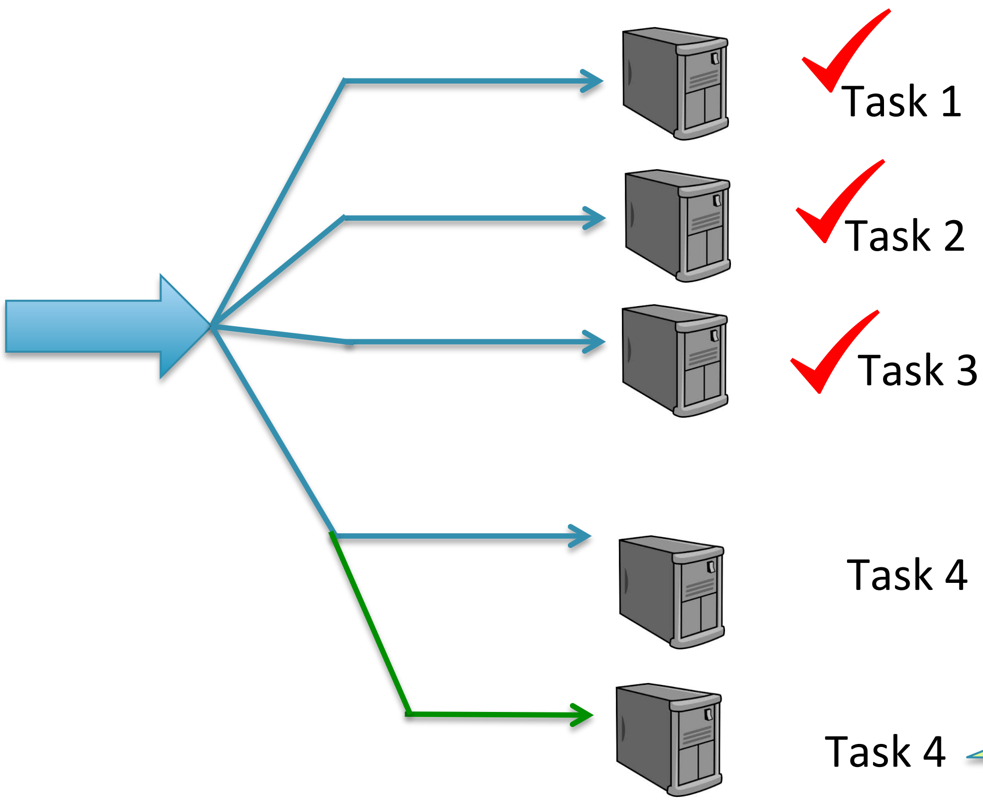
- If 100 tasks are run in parallel of 100 servers, what is the expected time to complete all of them.

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

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

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

Y is the residual service time after adding replicas

Central Value Theorem, n -> ∞:

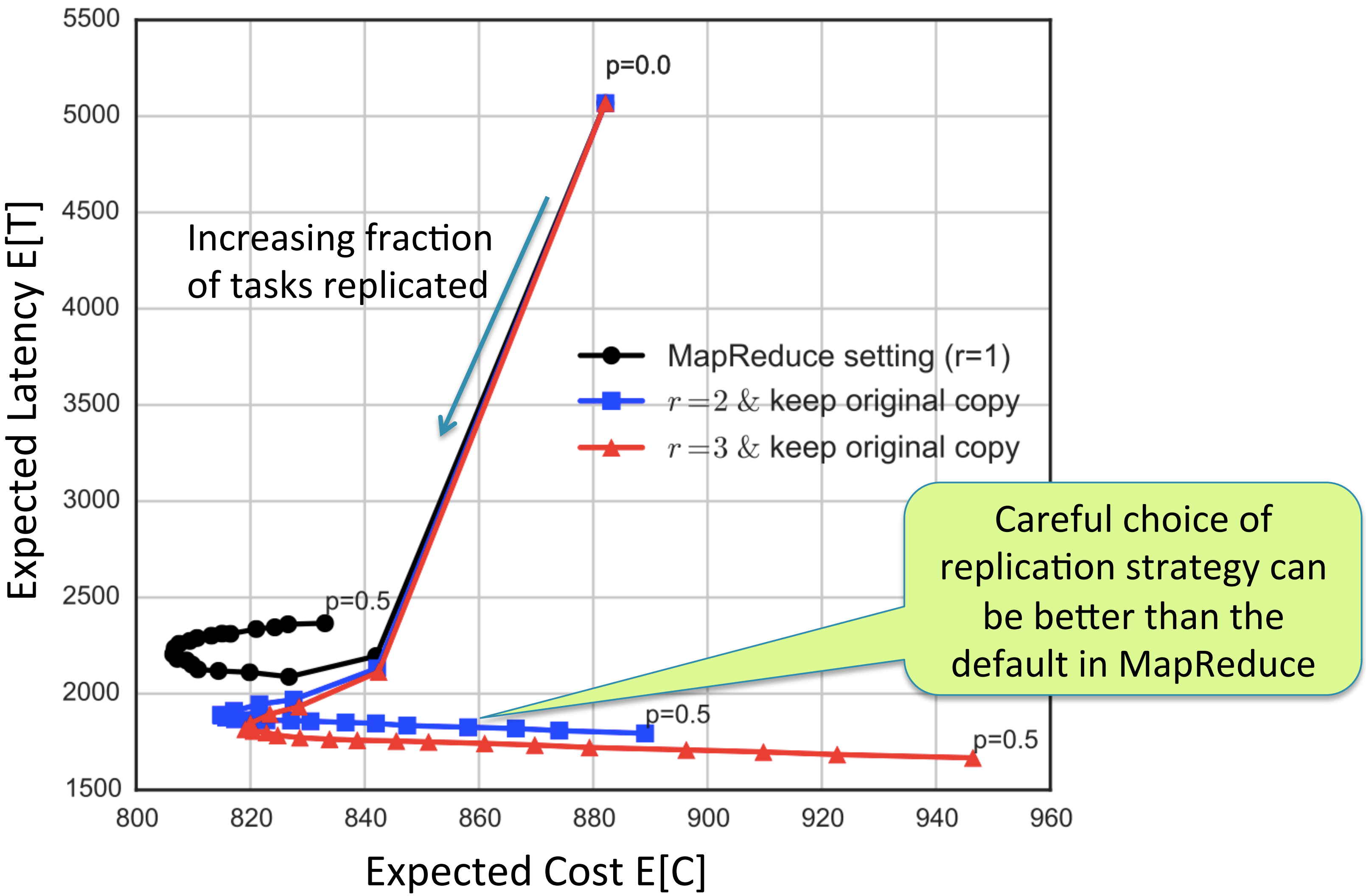
$$F_X^{-1}(1-p)$$

Extreme Value Theorem, n -> ∞:

Different behavior for Exponential, Light or Heavy tailed Y

Simulations using Google Cluster Data

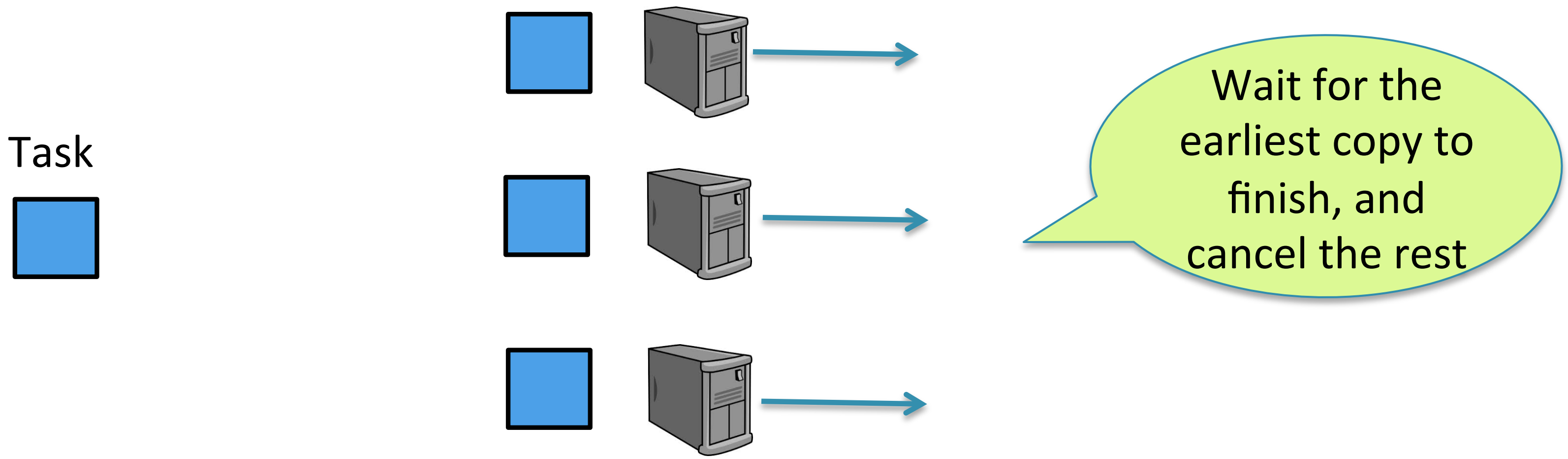
Latency-Cost Trade-off



Task Replication in Queueing Systems

Task Replication in Cloud Computing

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

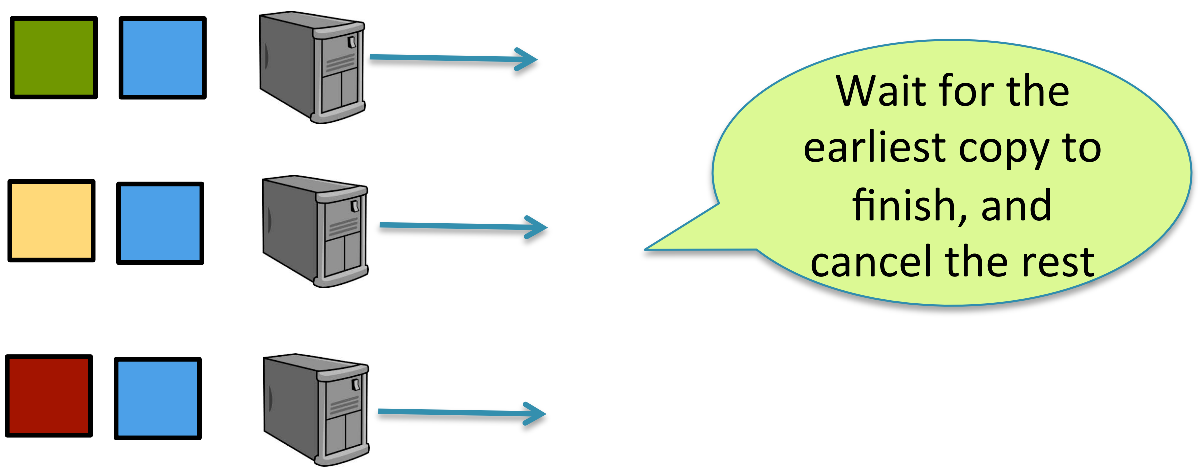


COST

- Additional computing time at servers

Task Replication in Cloud Computing

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

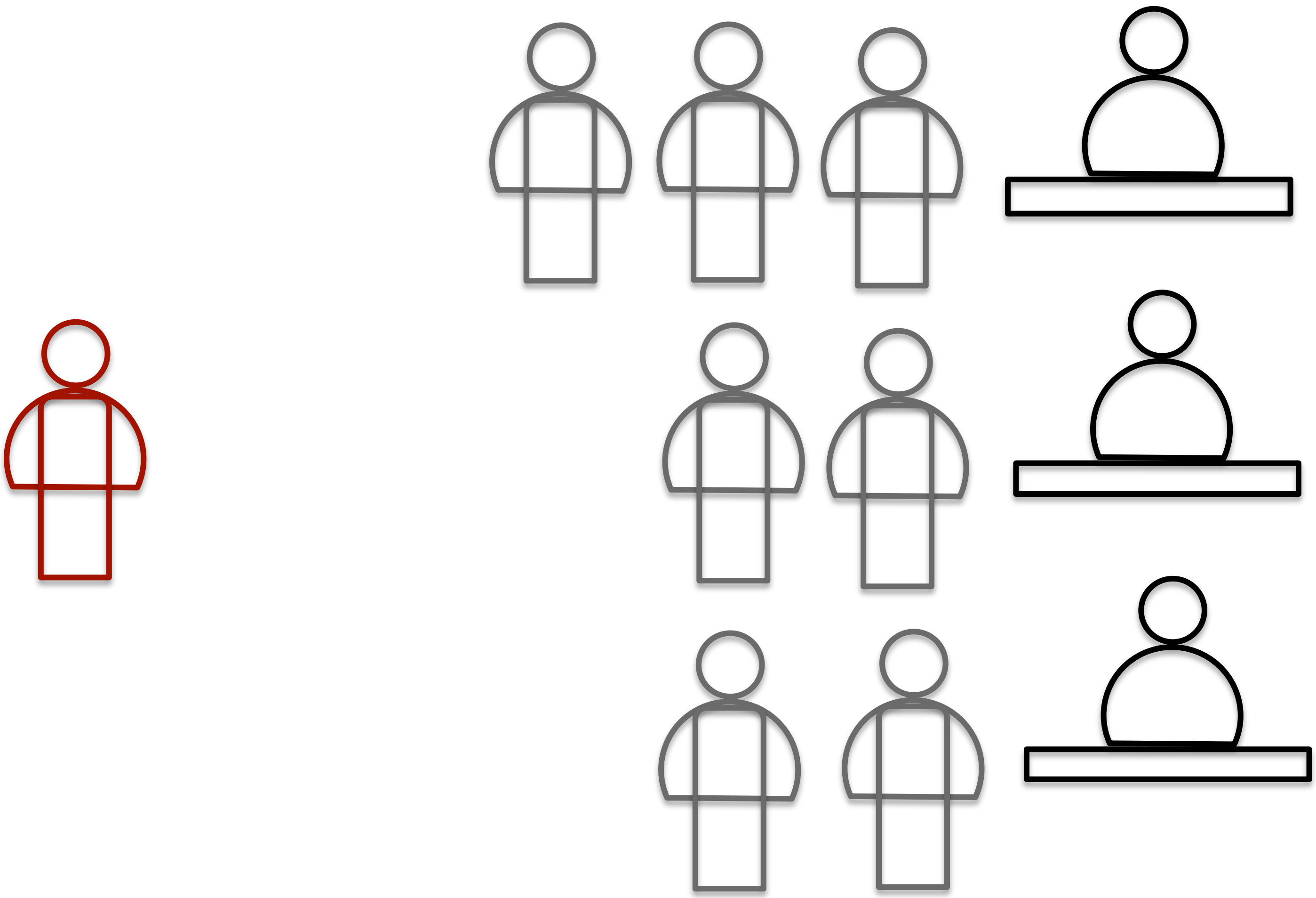


COST

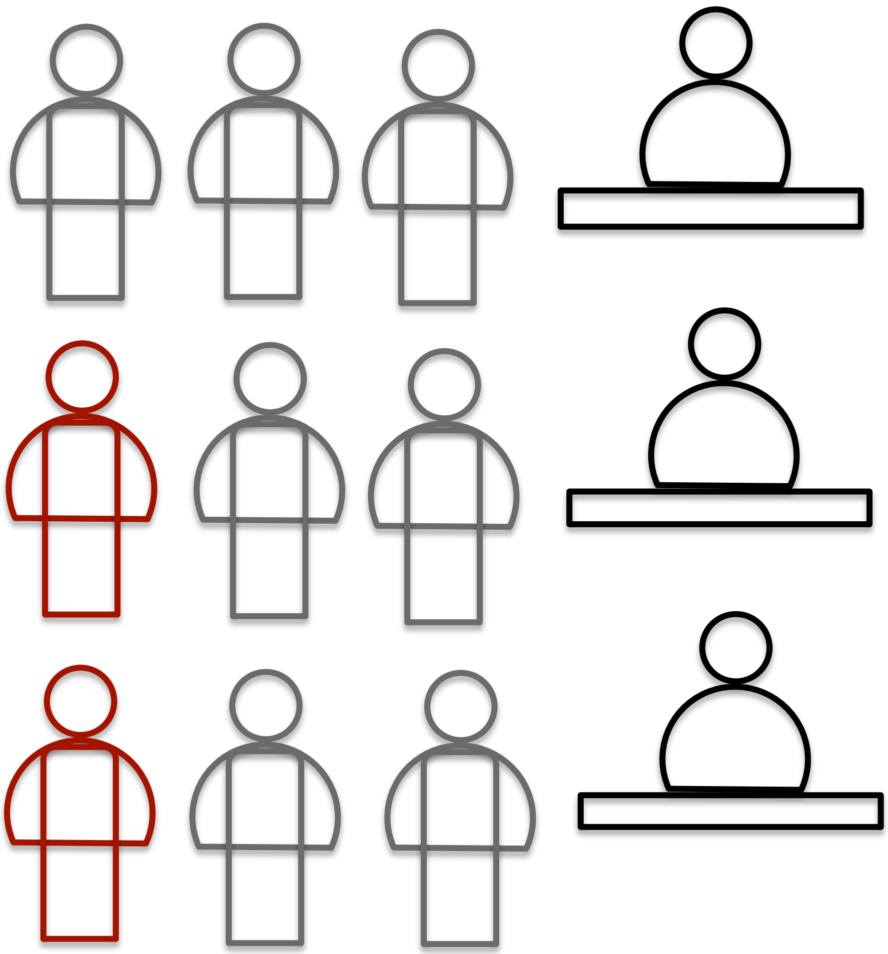
- Additional computing time at servers
- Increased queuing delay for other tasks

Analogy: Supermarket Queues

Supermarket Queues



Supermarket Queues

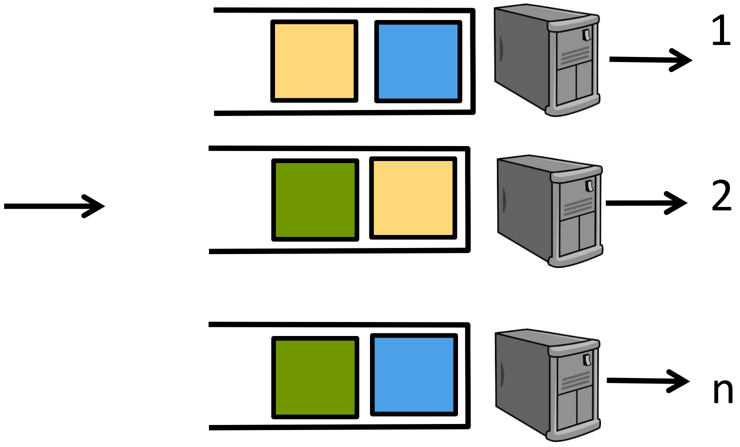


Get a friend to join the other queue!

What if everyone in the supermarket uses this strategy?

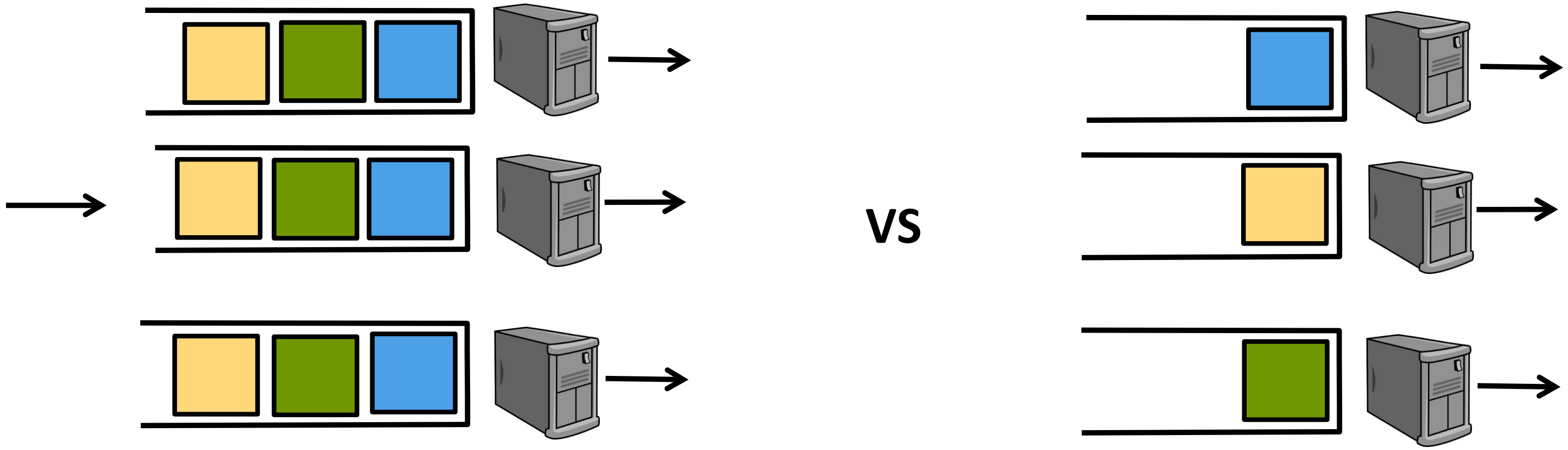
Design Questions

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



Surprising Insight

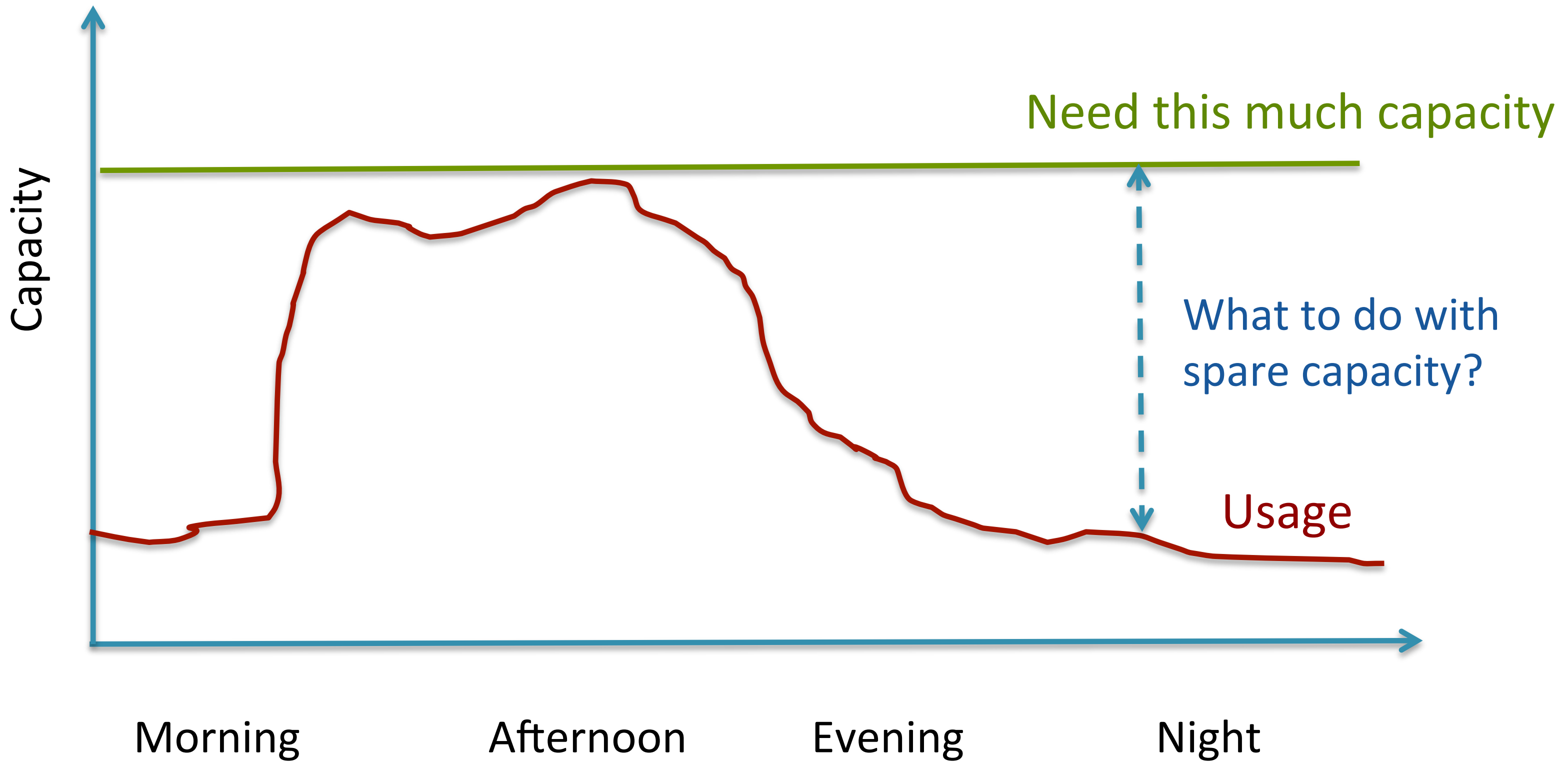
In certain regimes, replication could make the whole system faster, and cheaper!



Effective service rate > Sum of individual servers

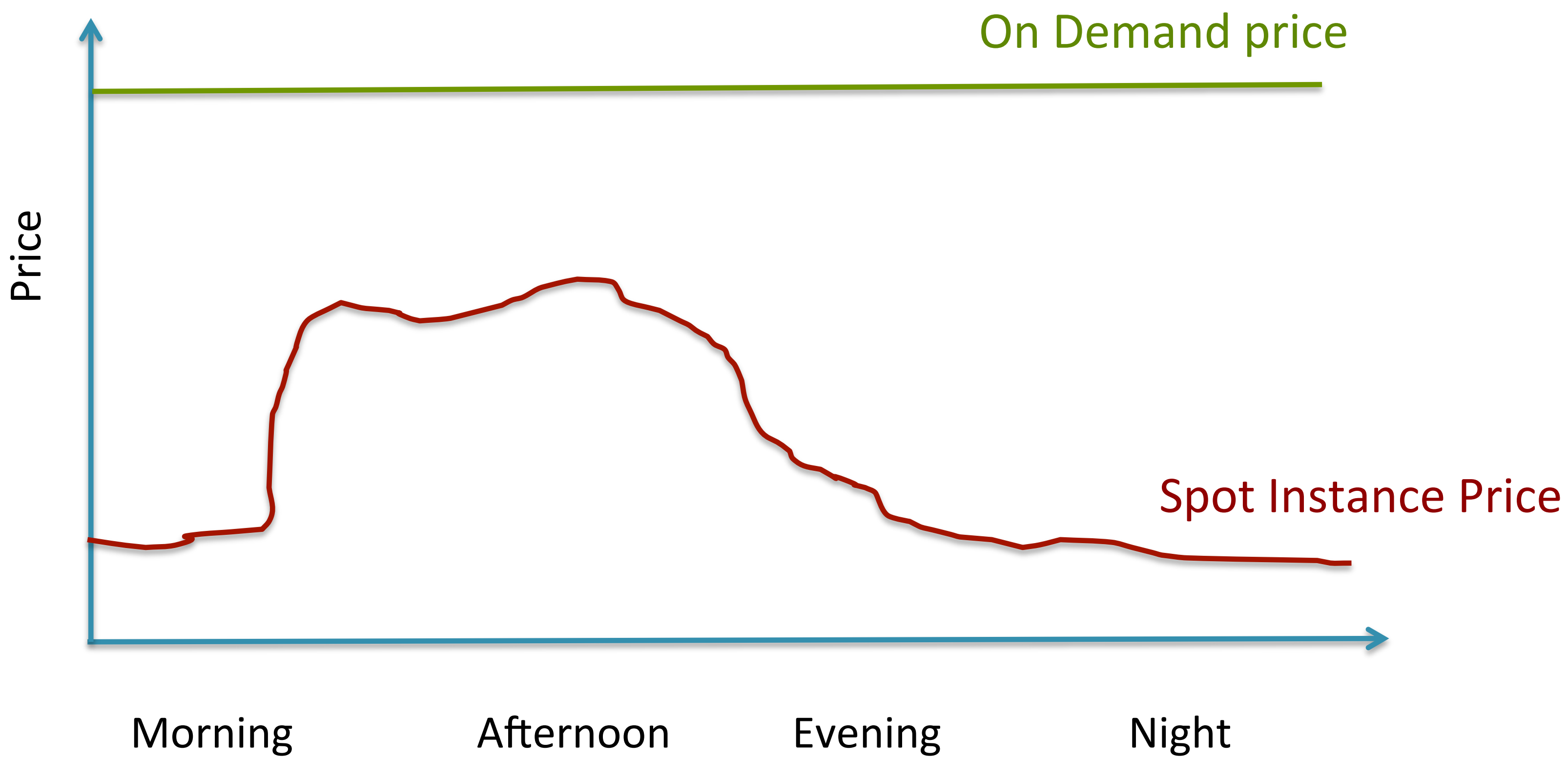
Cloud Spot Markets

- Spare capacity in cloud computing



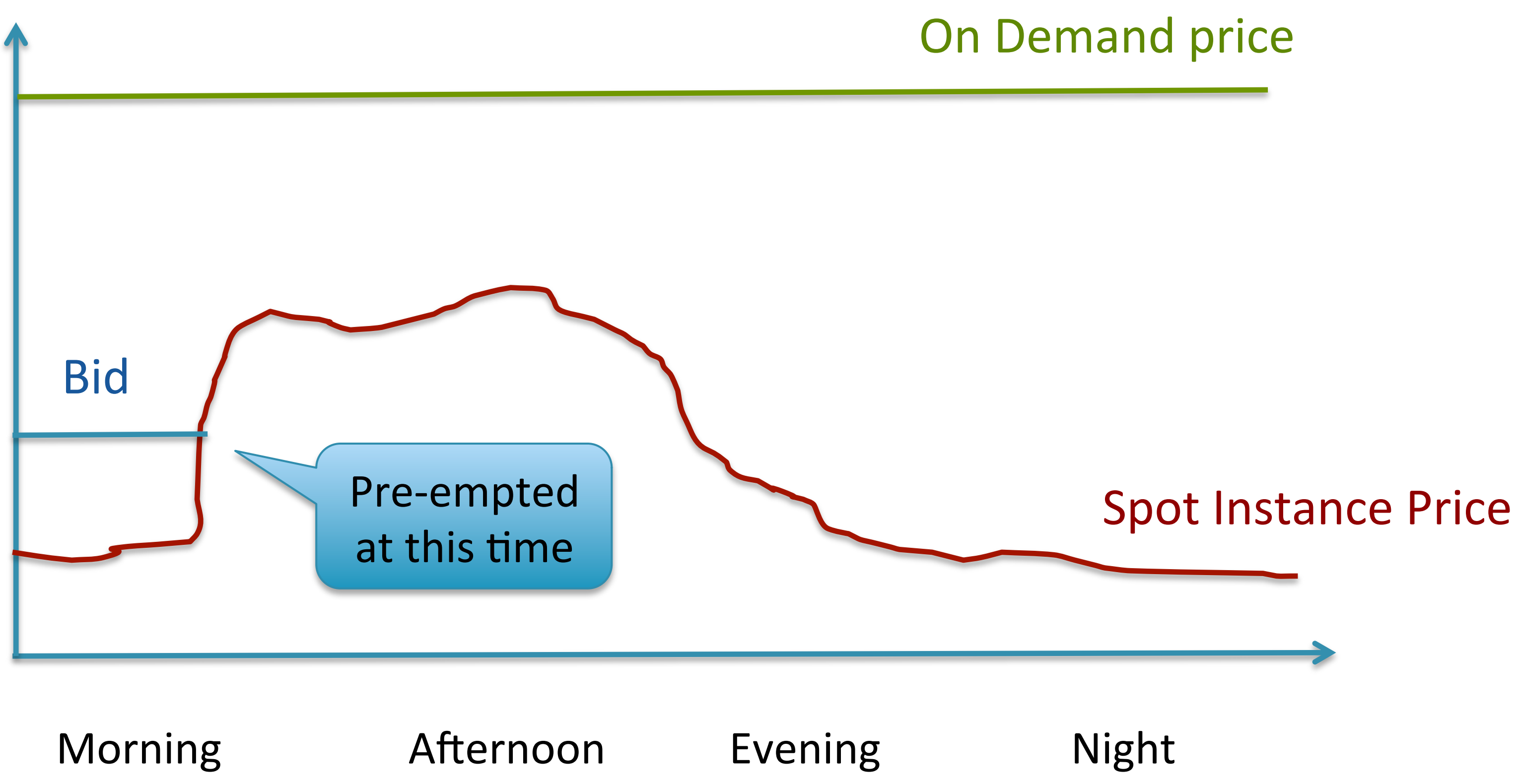
Cloud Spot Markets

- Sell it on the spot market for a lower price!



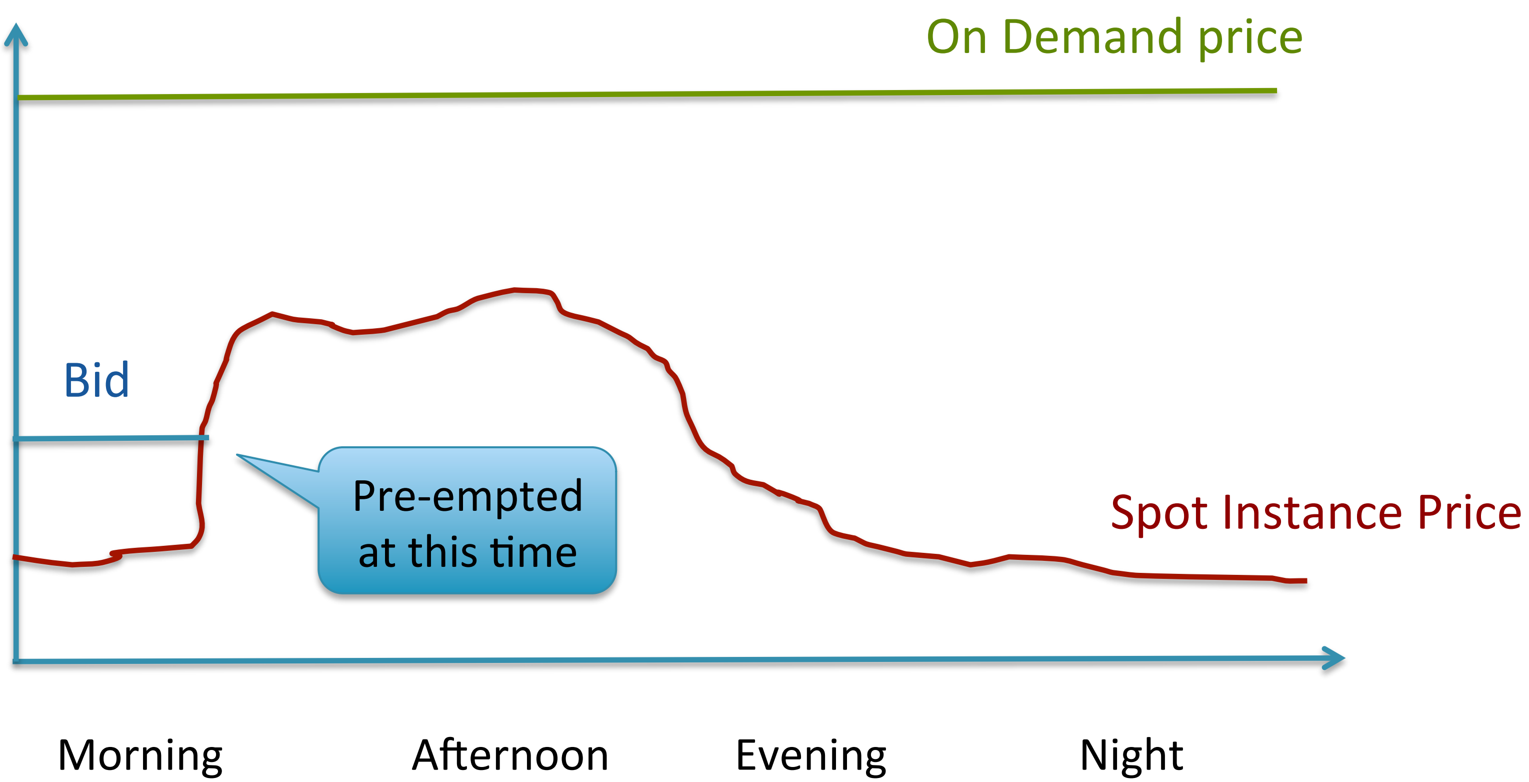
Bidding for Spot Instances

- Sell it on the spot market for a lower price



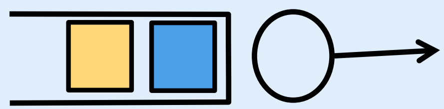
Sept 27 Guest Lecture: Prof. Carlee Joe-Wong

- Bidding and pricing strategies for spot markets

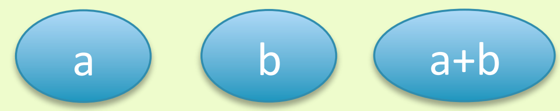


History and Overview

Cloud Computing

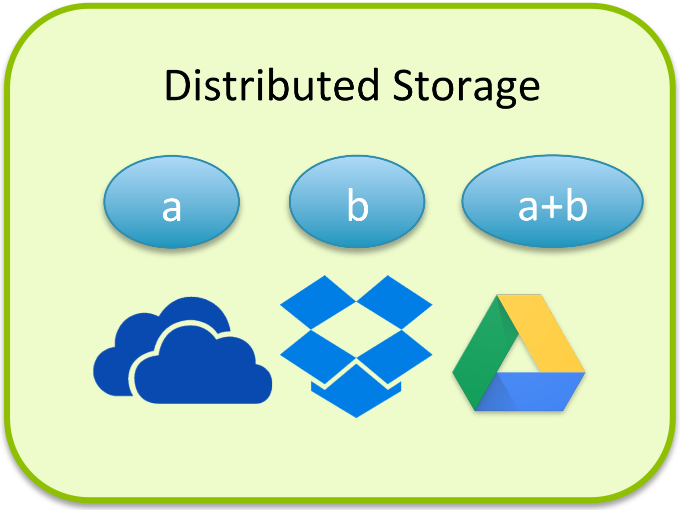


Distributed Storage



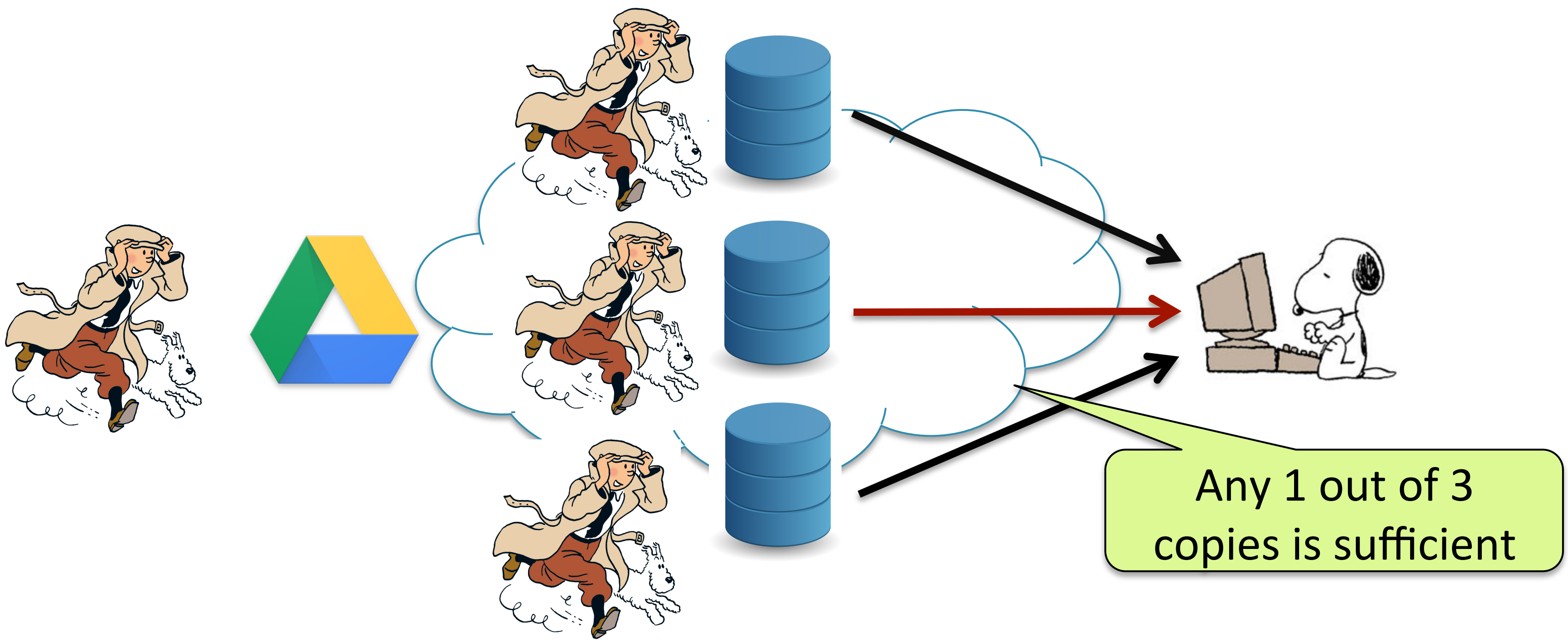
History and Overview

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



Replicated Storage

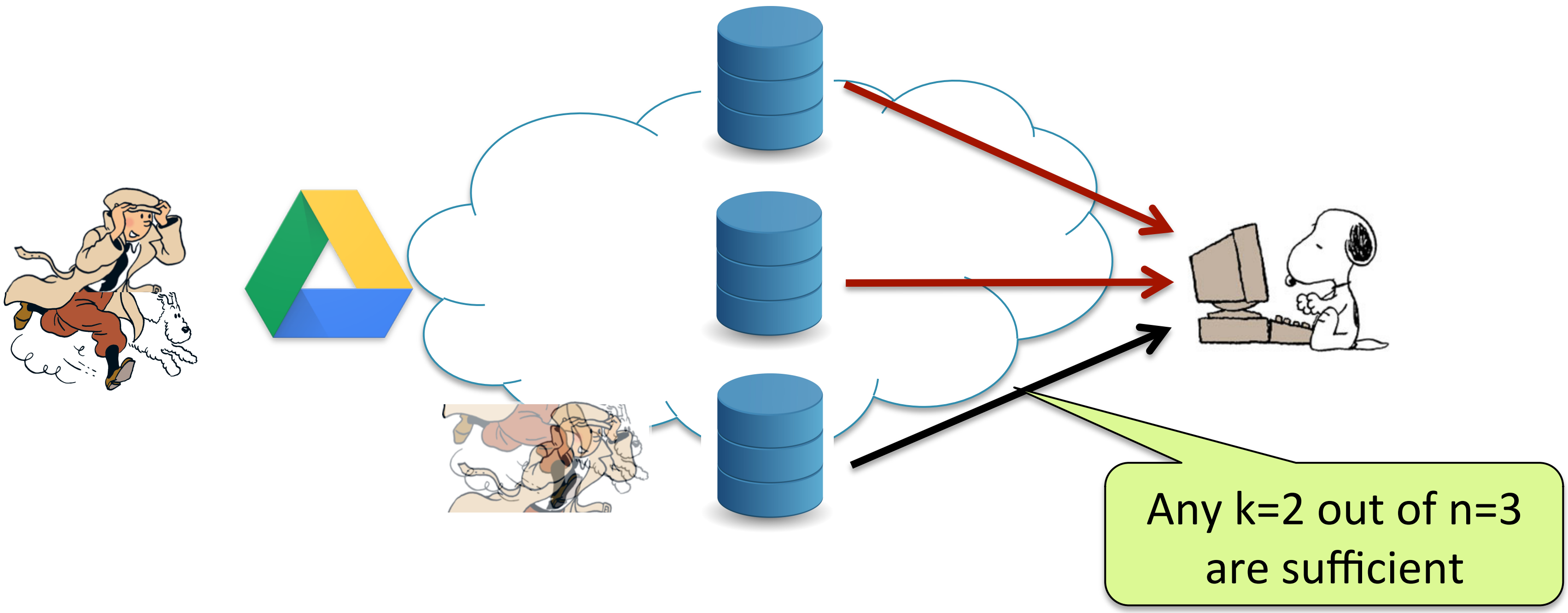
- Content is replicated on the cloud for reliability



- 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



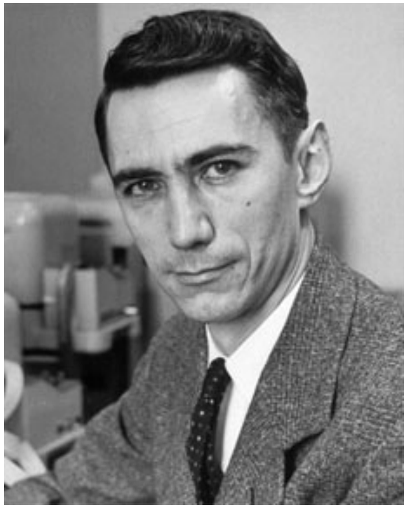
RAID: Redundant Array of Independent Disks (1987)

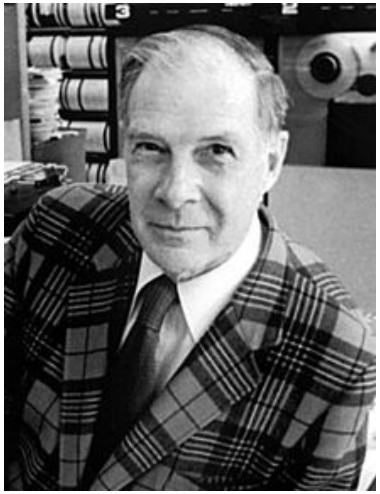
- Levels RAID 0, RAID 1, … : design for different goals such as reliability, availability, capacity etc.

- One of the inventors, Garth Gibson is 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



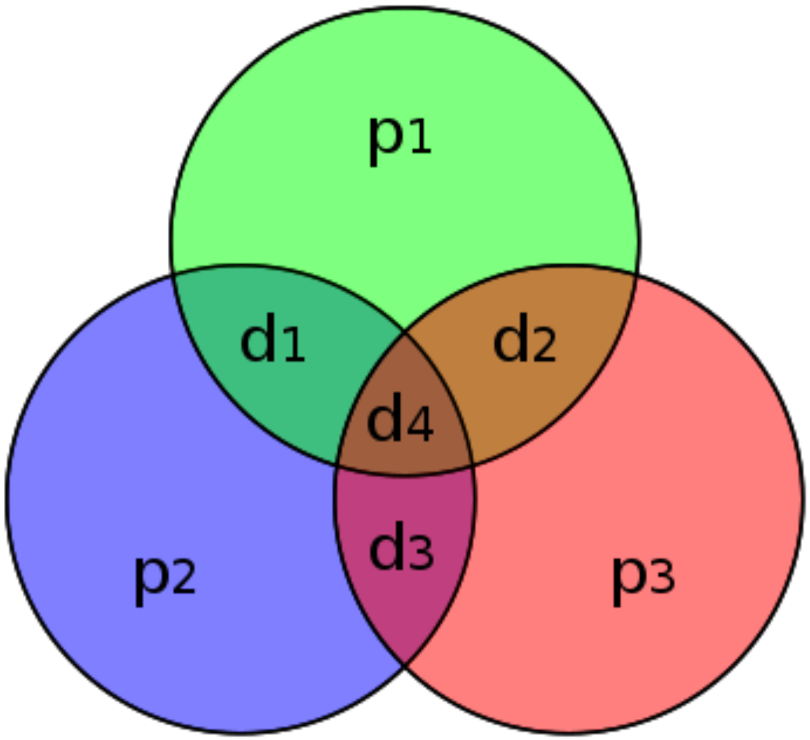


Simplest Codes

- 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)

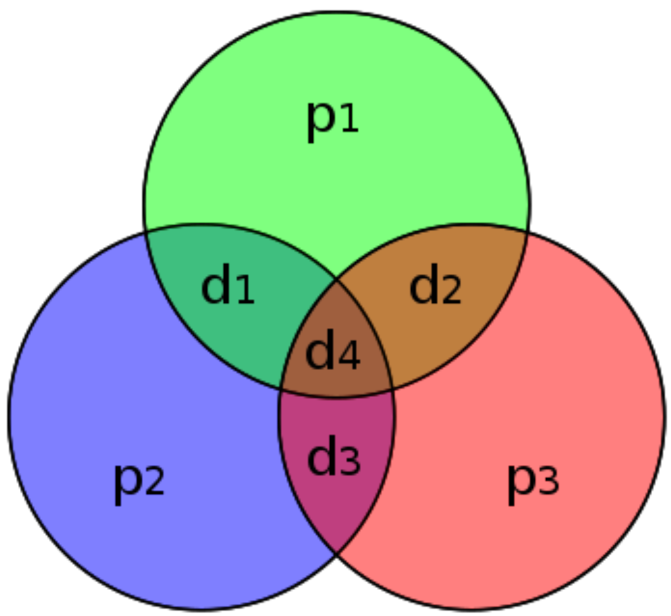
Hamming Codes

- (7,4) Hamming Code: 4 data bits, 3 parity bits
- Parity $p_1 = d_1 \oplus d_2 \oplus d_4$



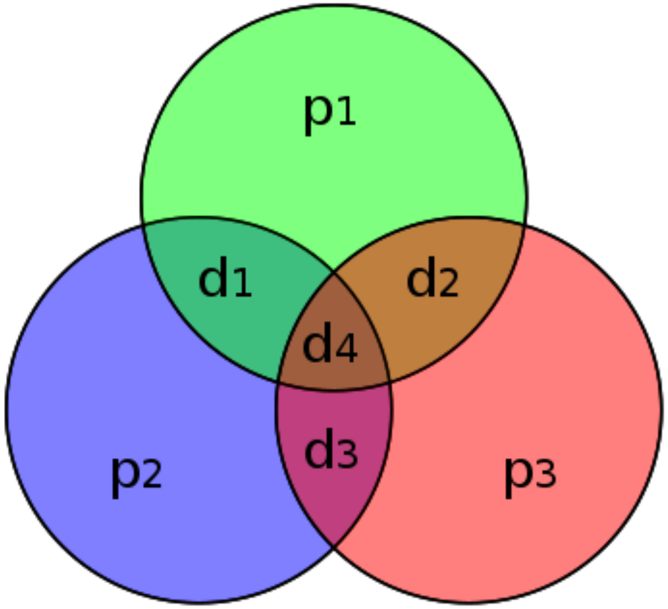
Hamming Codes: Quiz

- What is the rate of the code?
- Correct the 2 erasures
  - $(d_1, d_2, d_3, d_4, p_1, p_2, p_3) = (0, ?, 1, ?, 1, 0, 0)$



Hamming Codes: Answer

- What is the rate of the code? R = 4/7
- Correct the 2 erasures
  - (d1, d2, d3, d4, p1, p2, p3) = (0, 0, 1, 1, 1, 0, 0)



(n,k) Reed-Solomon Codes: 1960

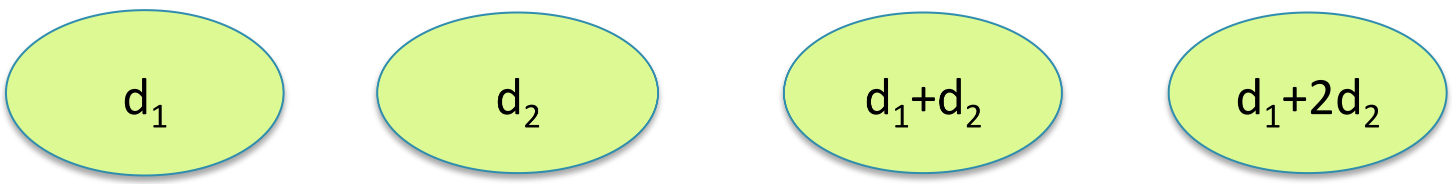
- Data: $d_1, d_2, d_3, \ldots d_k$
- Polynomial: $d_1 + d_2\, x + d_3\, x^2 + \ldots d_k\, x^{k-1}$
- Parity bits: Evaluate at n-k points:

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

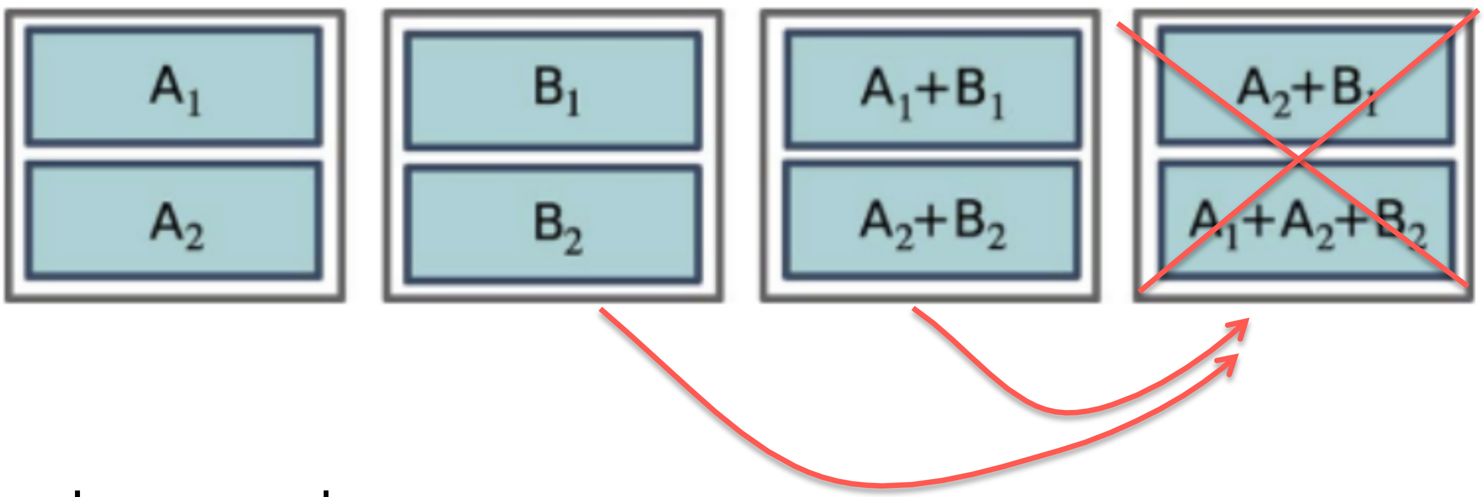
- Data: d_1, d_2 → Polynomial: $d_1 + d_2 x + d_3 x^{2} + \dots d_k x^{k-1}$



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

Locality and Repair Issues

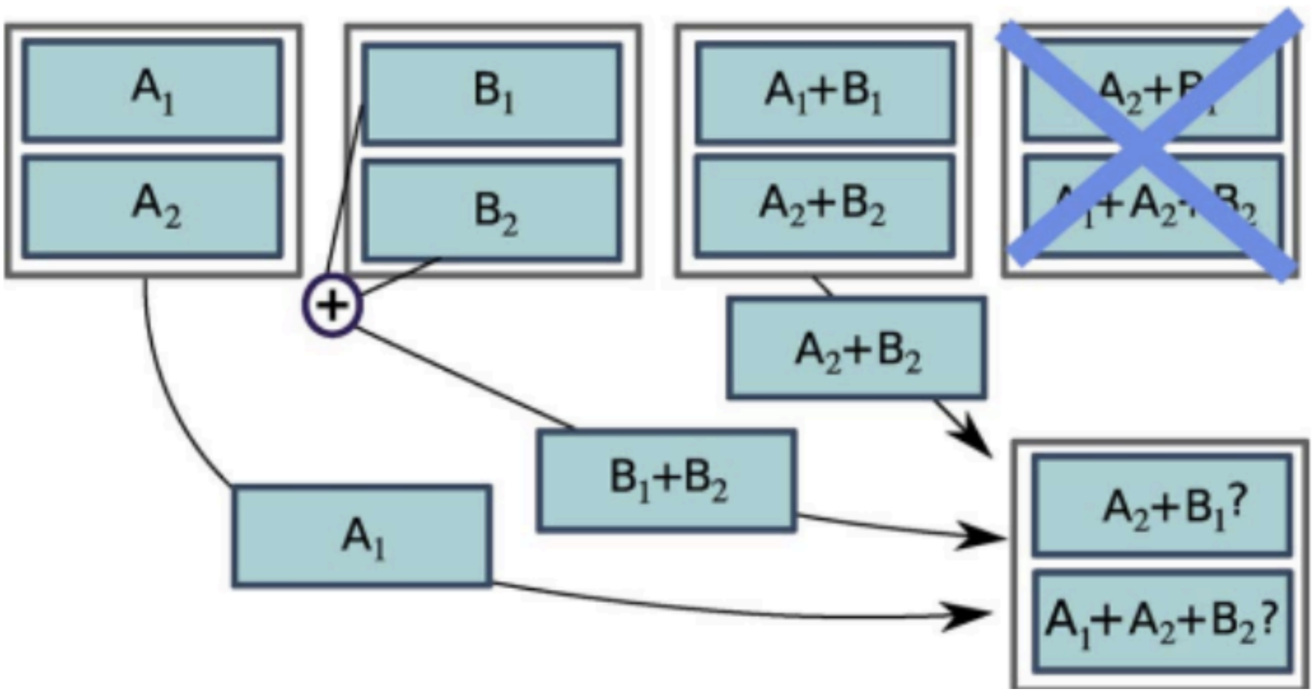
- Repairing failed nodes is hard with Reed-Solomon Codes..



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

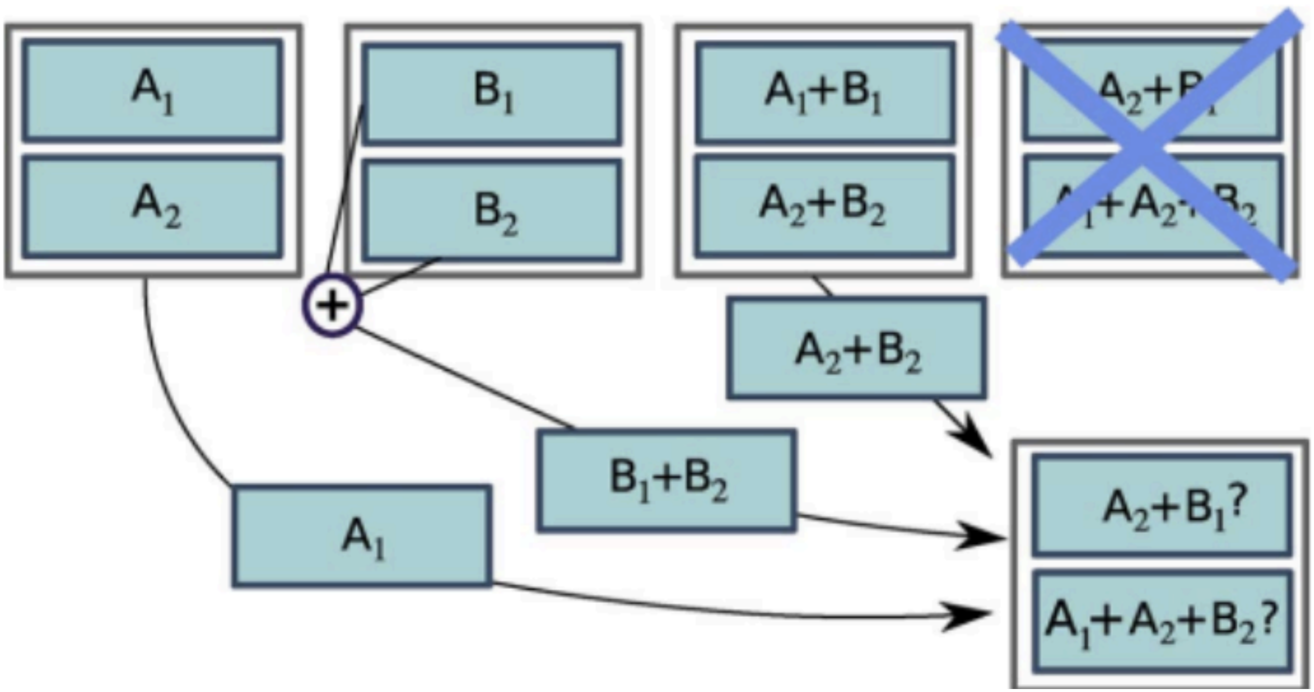
Solution: Locally Repairable Codes

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



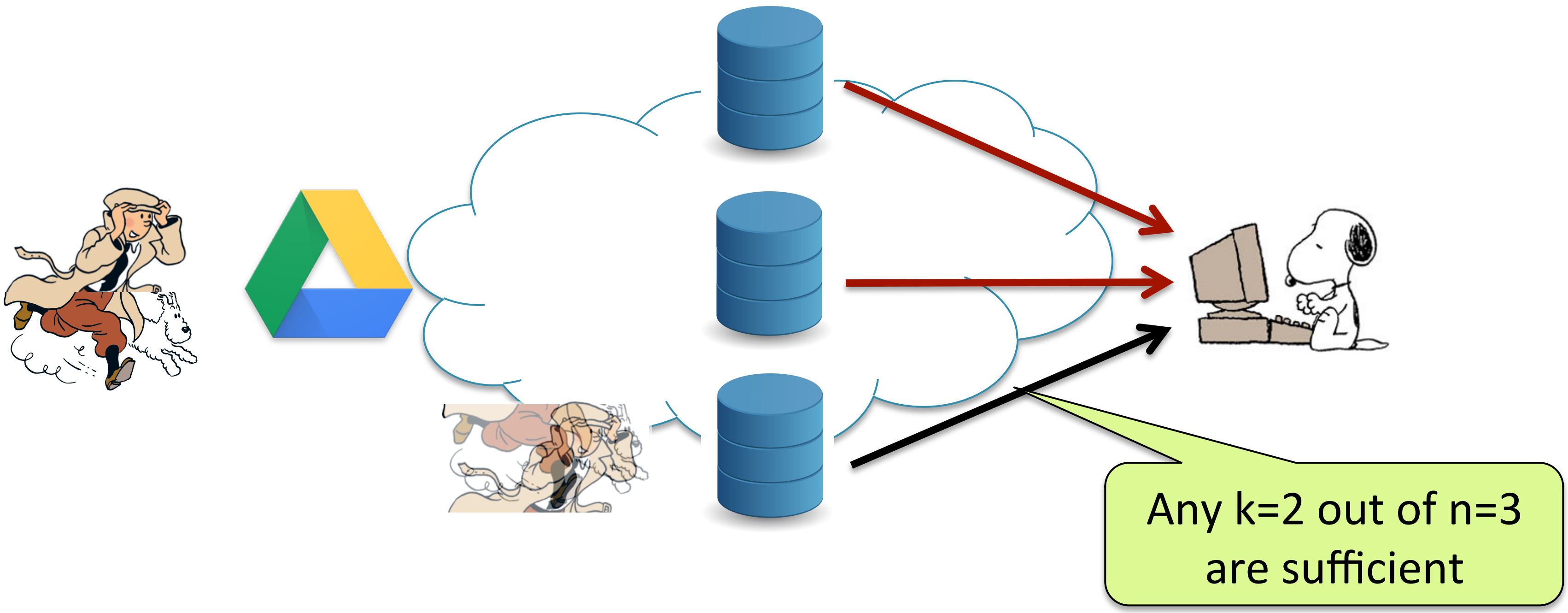
Guest Lecture: Prof. Rashmi Vinayak

- Systems Implementation of ‘piggybacking’ erasure codes in Apache Hadoop



Erasure Coded Storage

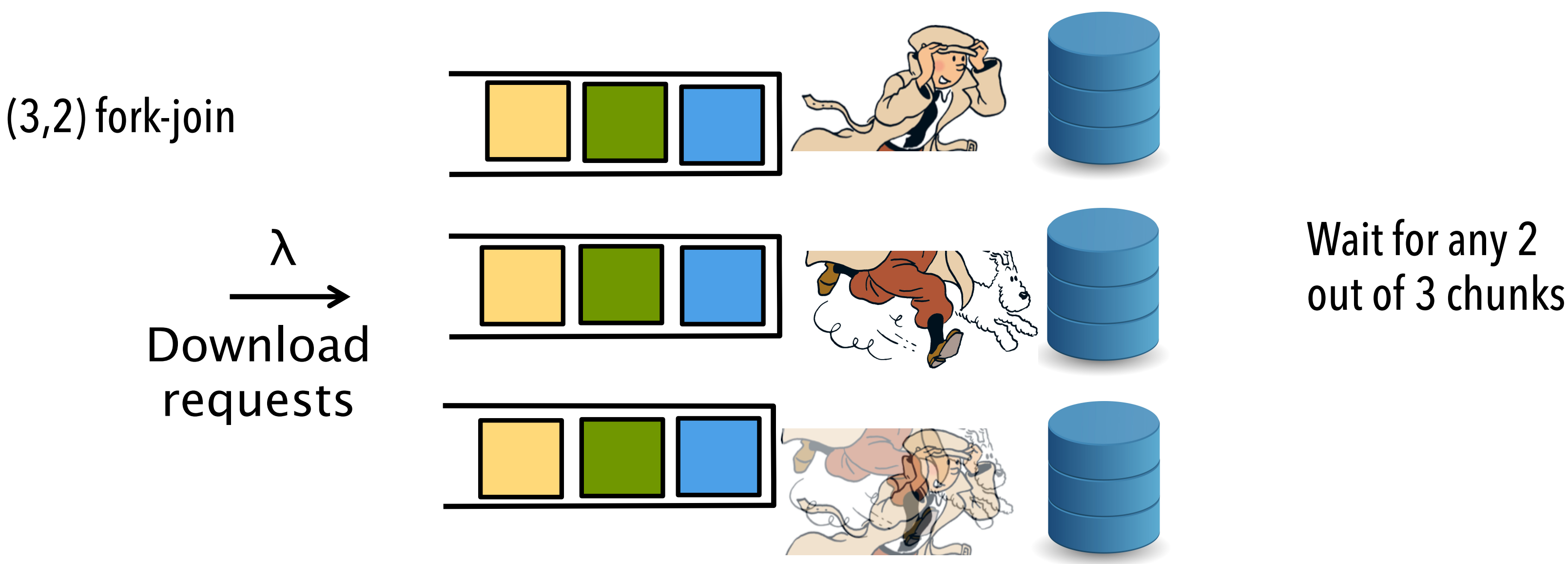
- With an (n,k) MDS code, any k out of n chunks are sufficient
  - Facebook, Google, Microsoft use (14,10) or (7,4) codes
  - Currently used for cold data, increasing for hot data



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

The (n,k) fork-join model
[GJ-Liu-Soljanin 2012,14]

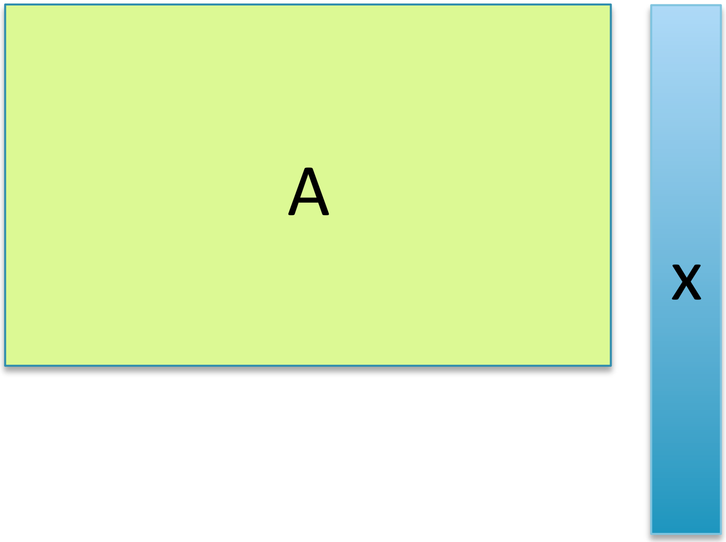
- Request all n chunks, wait for any k to be downloaded
- Each chunk takes service time $X \sim F_X$



k = 1: Replicated Case
k = n: Fork-join system actively studied in 90's

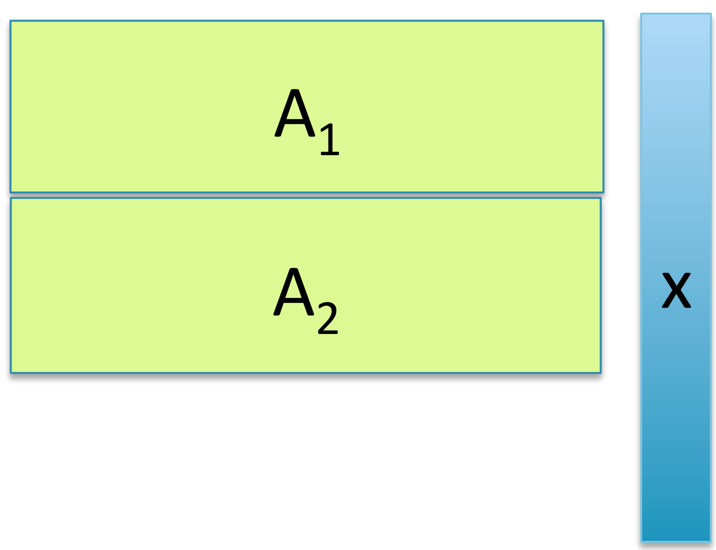
Coded Computing and ML

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



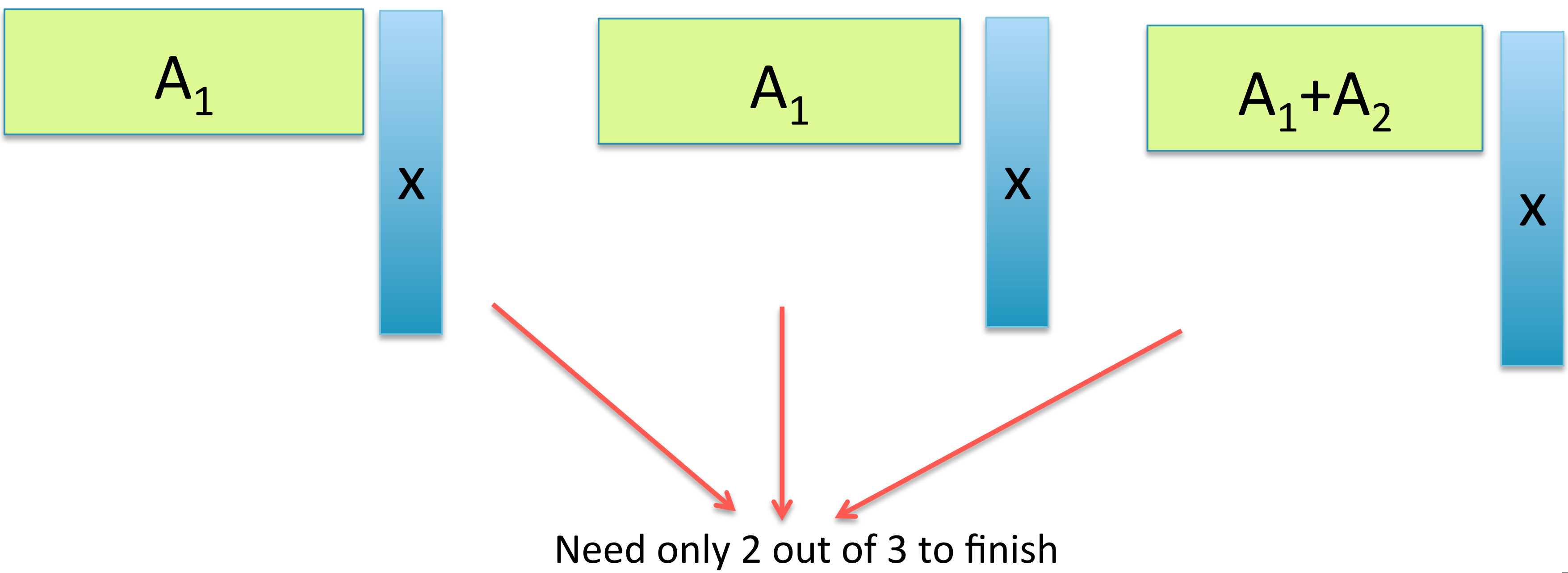
Coded Computing and ML

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



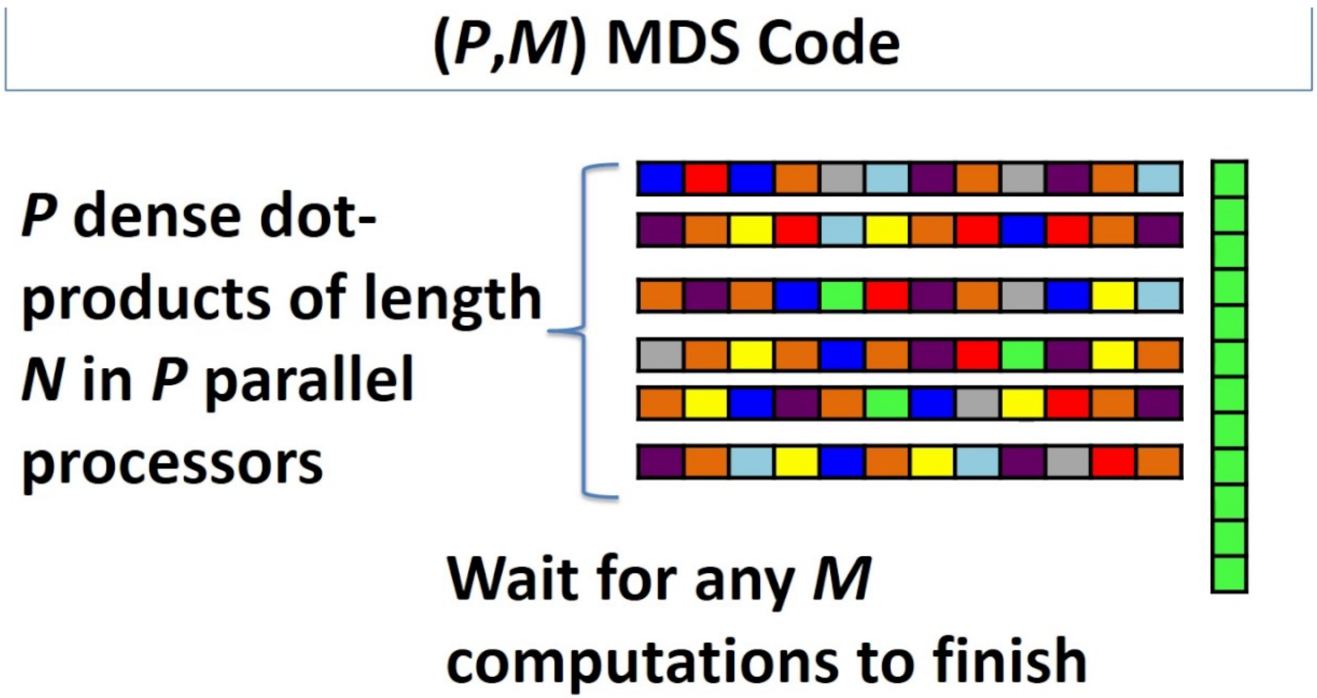
Coded Computing and ML

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



Guest Lecture: Sanghamitra Dutta

Short-dot codes



Second-half of the Class: Machine Learning