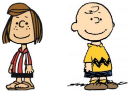
Cloud Storage Space vs. Download Time for Large Files

Emina Soljanin, Rutgers

Collaborators:

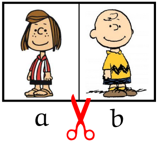
G. Joshi, S. Kadhe, Y. Liu, A. Sprintson, G. Wornell

Demand for Reliable Storage



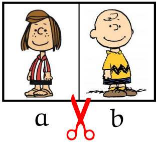
← data

Demand for Reliable Storage



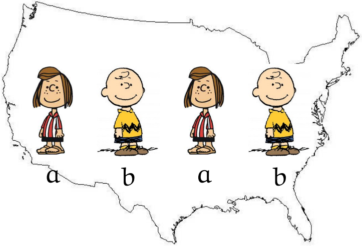
← data ($k = 2$ chunks)

Demand for Reliable Storage

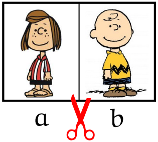


← data ($k = 2$ chunks)

replication

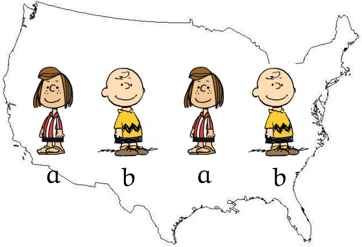


Demand for Reliable Storage

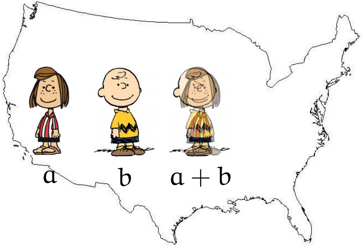


← data ($k = 2$ chunks)

replication



coding



Any k coded pieces (out of n) are sufficient for content recovery.

Coding in Distributed Storage

... because disks fail & data changes

If a disk (node) fails, we want to

1. still be able to recover data from the remaining storage (reliability) &
2. reproduce the lost data (or reliability) on each replacement disk with
   - minimal download from the remaining storage (repair bandwidth), or
   - by downloading (coded) data from only a few other nodes.

If the stored data changes, we must accordingly update the storage.

$\Longrightarrow$

Many new interesting problems in coding theory.

Coding in Distributed Storage

... because disks fail & data changes

If a disk (node) fails, we want to

1. still be able to recover data from the remaining storage (reliability) &
2. reproduce the lost data (or reliability) on each replacement disk with
   - minimal download from the remaining storage (repair bandwidth), or
   - by downloading (coded) data from only a few other nodes.

If the stored data changes, we must accordingly update the storage.

$\Longrightarrow$

Many new interesting problems in coding theory.

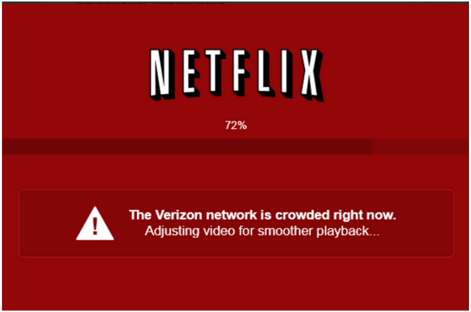
Do new codes for distributed storage affect data retrieval?

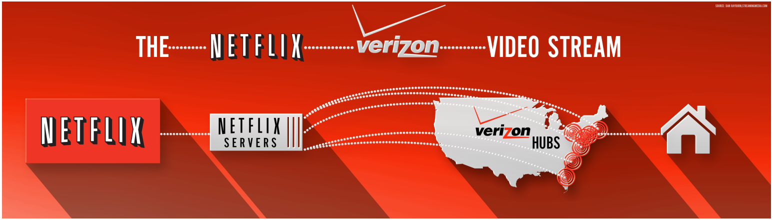
Demand for Low Latency

- Webpage download
  Amazon: 100ms $\sim$ costs 1% sales, Google: 1s $\sim$ page view drops 11%
- Interactive Tasks: 100ms - 150ms
- Online Gaming: 30ms
- Augmented Reality: 7ms - 20ms
- 5G, The Tactile Internet: 1ms

Download Latency

Whose fault was that?





Download Latency

How do we reduce it?

Download Latency

How do we reduce it?

How about rolling out many wheels of fortune?

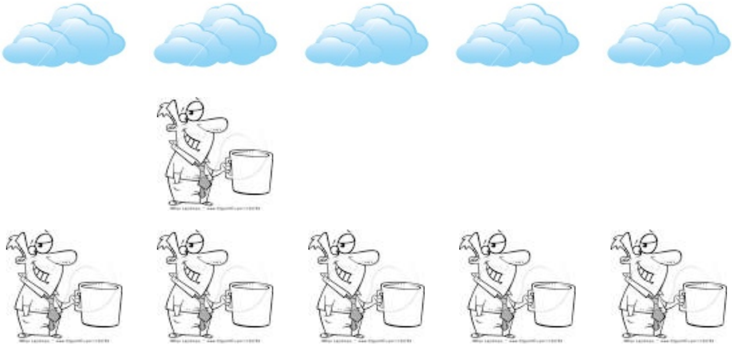
Buying Ground Coffee

Getting Data from the Cloud(s)

Getting Data from the Cloud(s)

Getting Data from the Cloud(s)

Getting Data from the Cloud(s)



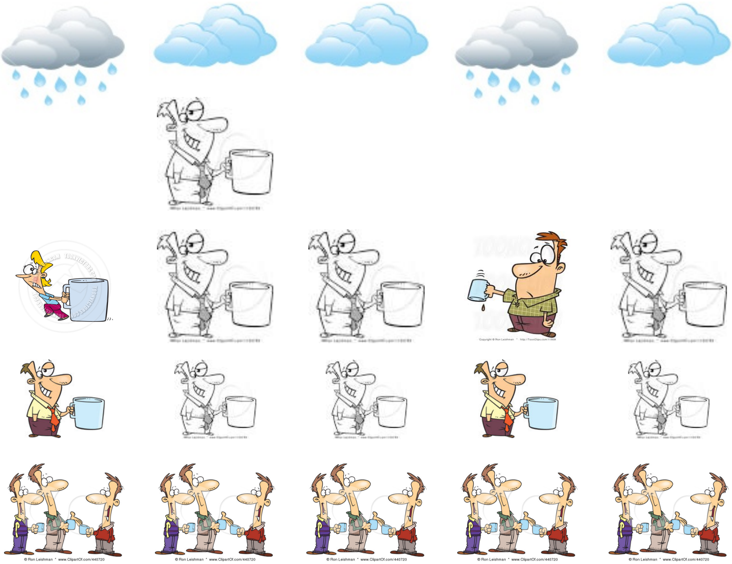
Getting Data from the Cloud(s)

Getting Data from the Cloud(s)

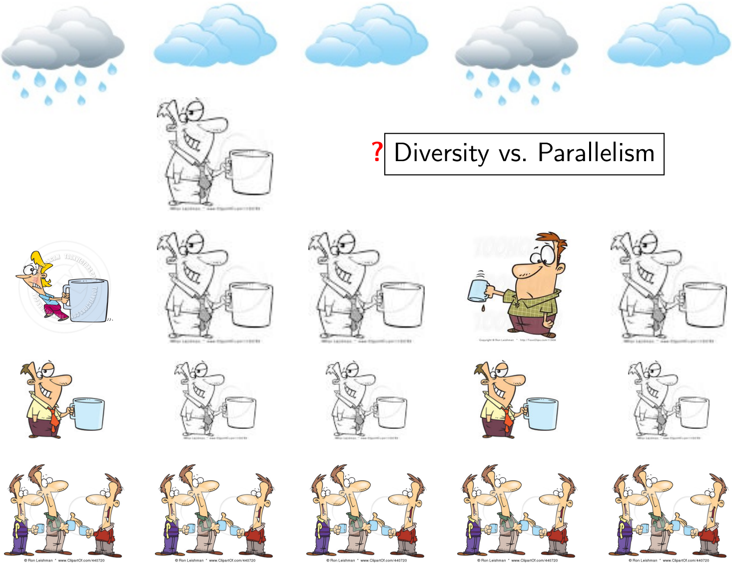
Getting Data from the Cloud(s)

Getting Data from the Cloud(s)

Getting Data from the Cloud(s)



Getting Data from the Cloud(s)



(n, k) Multiple Broadcasts Data Access Model

- Users request the same content (file F), stored in the cloud.
- Upon receiving a request, server s
  1. acquires F from the cloud at the time $W_s \sim \exp(W)$
  2. delivers F by broadcast to the users in time $D_s \sim D$

$\Rightarrow$ File F download time from server s is $W_s + D_s$.

(n, k) Multiple Broadcasts Data Access Model

- Users request the same content (file F), stored in the cloud.
- Upon receiving a request, server s
  1. acquires F from the cloud at the time $W_s \sim \exp(W)$
  2. delivers F by broadcast to the users in time $D_s \sim D$

  $\Rightarrow$ File F download time from server s is $W_s + D_s$.

- F is split into k blocks and encoded into n blocks s.t. any k out of n blocks are sufficient for content reconstruction.
- Each user's request is sent to n servers, for one block each.

(n, k) Multiple Broadcasts Data Access Model

- Users request the same content (file F), stored in the cloud.
- Upon receiving a request, server s
  1. acquires F from the cloud at the time $W_s \sim \exp(W)$
  2. delivers F by broadcast to the users in time $D_s \sim D$

  $\Rightarrow$ File F download time from server s is $W_s + D_s$.

- F is split into k blocks and encoded into n blocks s.t. any k out of n blocks are sufficient for content reconstruction.
- Each user's request is sent to n servers, for one block each.

When do k out of n servers delver their F/k-size blocks?

Order Statistics

Order Statistics

For iid RVs $\{X_1, X_2, \cdots, X_n\}$, the k^{th} smallest among them is an RV, known as the k^{th} order statistics $X_{k:n}$.

Order Statistics

For iid RVs $\{X_1, X_2, \cdots, X_n\}$, the k^{th} smallest among them is an RV, known as the k^{th} order statistics $X_{k:n}$.

When X_i's are $\exp(W)$, the mean and variance of $X_{k,n}$ are

$$E[X_{k,n}] = W(H_n - H_{n-k}) \text{ and } V[X_{k,n}] = W^2(H_{n^2} - H_{(n-k)^2}),$$

where H_n and H_{n^2} are (generalized) harmonic numbers

$$H_n = \sum_{j=1}^{n} \frac{1}{j} \text{ and } H_{n^2} = \sum_{j=1}^{n} \frac{1}{j^2}.$$

Order Statistics

For iid RVs $\{X_1, X_2, \cdots, X_n\}$, the k^{th} smallest among them is an RV, known as the k^{th} order statistics $X_{k:n}$.

When X_i's are $\exp(W)$, the mean and variance of $X_{k,n}$ are

$$\mathsf{E}[X_{k,n}] = W(H_n - H_{n-k}) \ \text{ and } \ \mathsf{V}[X_{k,n}] = W^2(H_{n^2} - H_{(n-k)^2}),$$

where H_n and H_{n^2} are (generalized) harmonic numbers

$$H_n = \sum_{j=1}^{n} \frac{1}{j} \ \text{ and } \ H_{n^2} = \sum_{j=1}^{n} \frac{1}{j^2}.$$

When do k out of n servers delver their F/k-size blocks?

(n, k) Multiple Broadcasts Response Time

Allerton'12, with G. Joshi, MIT, and Y. Liu, WISC

Theorem:

The mean download completion time is given by

$$T_{n,k} = \mathcal{W}(H_n - H_{n-k}) + \frac{D}{k} \qquad H_\ell = \sum_{i=1}^{\ell} \frac{1}{i}$$

(n, k) Multiple Broadcasts Response Time

Allerton'12, with G. Joshi, MIT, and Y. Liu, WISC

Theorem:

The mean download completion time is given by

$$T_{n,k} = W(H_n - H_{n-k}) + \frac{D}{k} \qquad H_\ell = \sum_{i=1}^{\ell} \frac{1}{i}$$

- k-th order statistics

(n, k) Multiple Broadcasts Response Time

Allerton'12, with G. Joshi, MIT, and Y. Liu, WISC

Theorem:

The mean download completion time is given by

$$T_{n,k} = W(H_n - H_{n-k}) + \frac{D}{k} \qquad H_\ell = \sum_{i=1}^{\ell} \frac{1}{i}$$

- k-th order statistics → $W(H_n - H_{n-k})$
- one k-th of data → $\frac{D}{k}$

(n, k) Multiple Broadcasts Response Time

Allerton'12, with G. Joshi, MIT, and Y. Liu, WISC

Theorem:

The mean download completion time is given by

$$T_{n,k} = W(H_n - H_{n-k}) + \frac{D}{k} \qquad H_\ell = \sum_{i=1}^{\ell} \frac{1}{i}$$

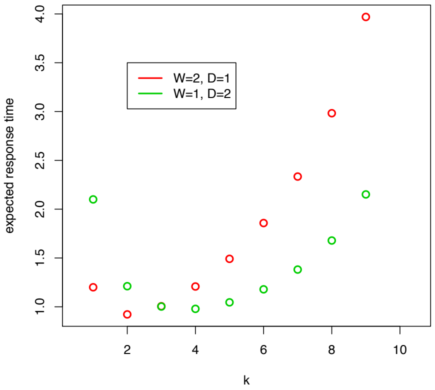
- k-th order statistics
- one k-th of data

$\Longrightarrow$

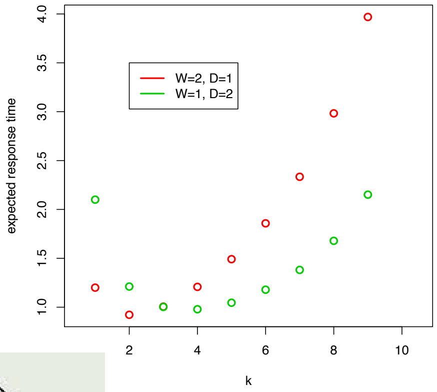
k that minimizes $T_{n,k}$ is

$$k \approx \frac{-D + \sqrt{D^2 + 4nWD}}{2W}$$

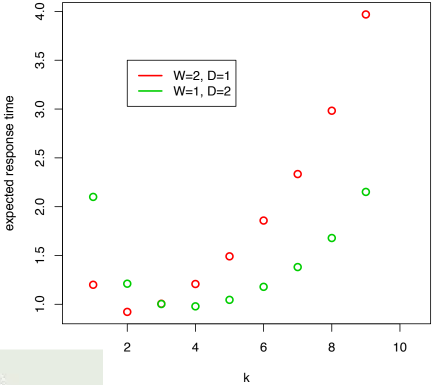
What is Really the Optimal k?



What is Really the Optimal k?

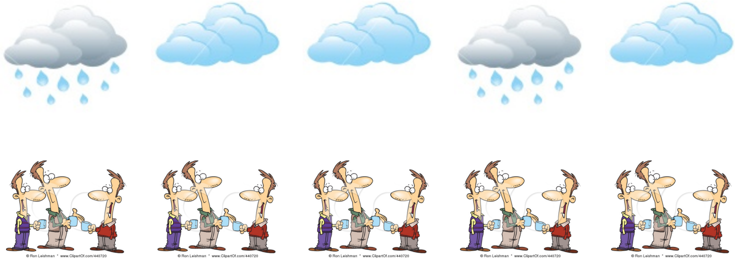


What is Really the Optimal k?



VS.

Queueing for Content



Queueing for Content

Queueing for Content

Single Queues and Server Farms

Single M/M/1 Queue:

- Requests arrive at rate λ according to a Poisson process.
- Job service times have an exponential distribution with rate μ.
- Many metrics of interest are well understood for this model, e.g. the response time is exponential with rate $\mu - \lambda$.

Single Queues and Server Farms

Single M/M/1 Queue:

- Requests arrive at rate λ according to a Poisson process.
- Job service times have an exponential distribution with rate μ.
- Many metrics of interest are well understood for this model, e.g. the response time is exponential with rate $\mu - \lambda$.

A Fork-Join n-Server Queue Model:

- requests arrive at rate λ according to a Poisson process, & are split on arrival and must be joined before departure.
- At each queue, service times are exponential with rate μ.

Single Queues and Server Farms

Single M/M/1 Queue:

- Requests arrive at rate λ according to a Poisson process.
- Job service times have an exponential distribution with rate μ.
- Many metrics of interest are well understood for this model, e.g. the response time is exponential with rate $\mu - \lambda$.

A Fork-Join n-Server Queue Model:

- requests arrive at rate λ according to a Poisson process, & are split on arrival and must be joined before departure.
- At each queue, service times are exponential with rate μ.

- It is seen as a key model for parallel/distributed systems, e.g., RAID.
- There is a renewed interest in the problem (e.g., map-reduce).
- Few analytical results exist, but various approximations are known.

The (n, k) Fork-Join System

Allerton'12, with G. Joshi, MIT, and Y. Liu, WISC

Architecture:

- F is split into k blocks and encoded into n blocks s.t. any k out of n blocks are sufficient for content reconstruction.
- The n coded blocks are stored on n disks.

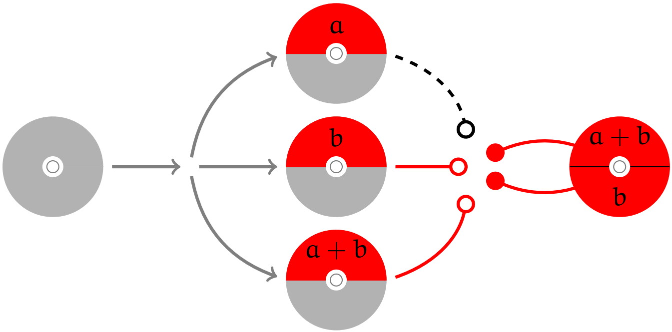
Operation:

- User request for F are forked to all n disks.
- Downloads from any k disks jointly enables reconstruction of F.

$\Rightarrow$ Arrival rate at each of the n queues is λ and service rate is $k\mu$.

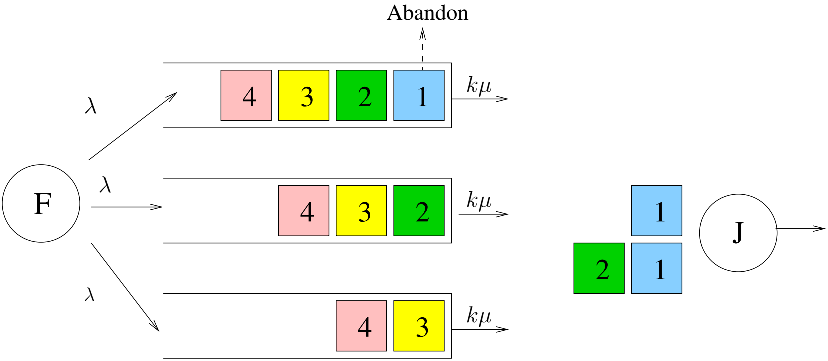
$(3, 2)$ Fork-Join System Architecture

- Content F is split into equal parts a and b, and stored on 3 disks as a, b, and $a + b \Rightarrow$ each disk stores half the size of F.
- User request for F are forked to all 3 disks.
- Downloads from any 2 disks jointly enables reconstruction of F.



Storage is 50% higher, but download time (per disk & overall) is reduced.

(3, 2) Fork-Join System Operation



Stability of (n, k) Fork-Join FHW System

The rate of arrivals λ and the service rate $k\mu$ per node must satisfy

$$\lambda - \frac{\lambda(n-k)}{n} < k\mu$$

$\Rightarrow \lambda < n\mu$

Some Related Work on (n, k)-Type Sytems

Fork-Join Queues (1980's, 1990's, 2000's)

Baccelli, Makowski, Shwartz, Flatto, Boxma, Koole, Kim, Agrawala, Nelson, Tantawi, Xia, Liu, Towsley, Lelarge, ...

Codes and Queues (2012 –)

Joshi, Liu, Soljanin, Liang, Kozat, Kumar, Tandon, Clancy, Ziang, Lang, Agawall, Chen, Shah, Lee, Ramchandran, Huang, Pawar, Zhang, ...

Replication and Queues (2012 –)

Vulimiri, Godfrey, Mittal, Sherry, Ratnasamy, Shenker, Gardner, Zbarsky, Doroudi, Harchol-Balter, Scheller-Wolf, Hyytiä, ...

Codes and Blocking (2012 –)

Ferner, Médard, Soljanin

The $(n, 1)$ Fork-Join System

The system behaves as an $M/M/1$ queue with service rate $n\mu$
$\Rightarrow$ the system response time is $\exp(1/(n\mu - \lambda))$.

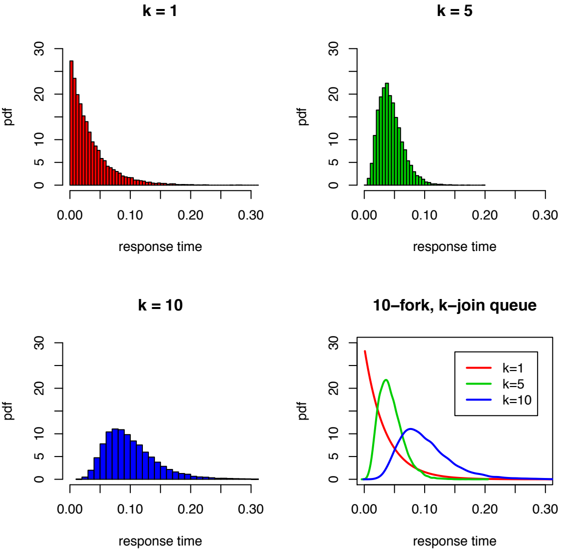
The $(n, 1)$ Fork-Join System

The system behaves as an $M/M/1$ queue with service rate $n\mu$
$\Rightarrow$ the system response time is $\exp(1/(n\mu - \lambda))$.

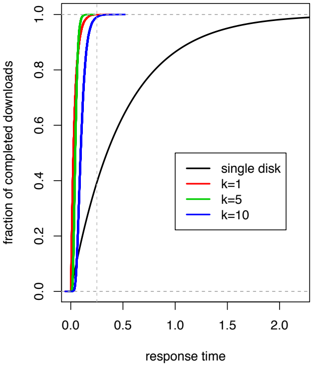
A model "superimposing" multiple $(n_\ell, 1), \mu_\ell, \lambda_\ell$ systems:
"Queueing with Redundant Requests: First Exact Analysis,"
at Sigmetrics 2015 by
Gardner, Zbarsky, Doroudi, Harchol-Balter, Scheller-Wolf, Hyytiä

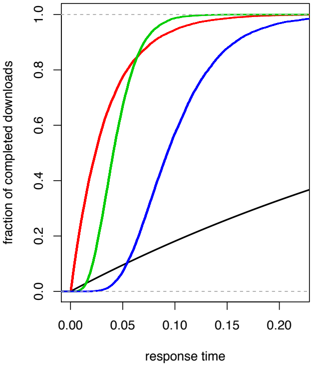
Response Time Histogram for 10^4 Downloads

$(10, k)$ Fork-Join Queue, FHW Model, $\lambda = 1$, $\mu = 3$



Storage Space vs. Download Time in (10, k) Systems

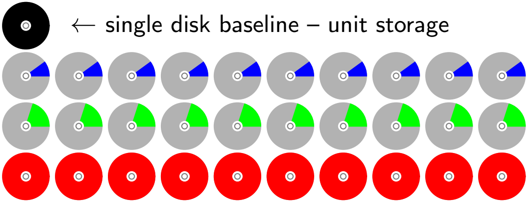




M/M/1

request rate $\lambda = 1$

$\mu = 3$ per unit-download



← single disk baseline – unit storage

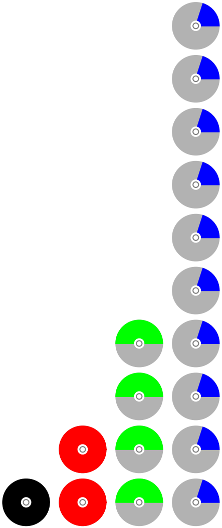
← the same total storage

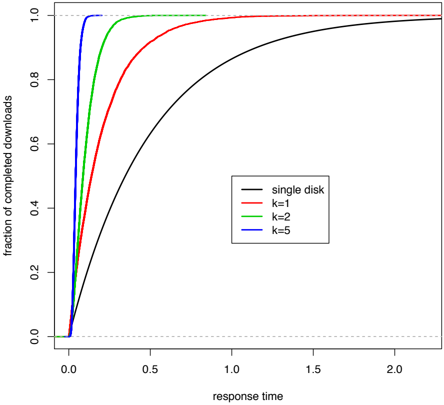
← double total storage

← 10× increase in storage

Doubling Storage Space Shortens Download Time

2k-fork, k-join Queue, $M/M/1$, request rate $\lambda = 1$, $\mu = 3$ per unit-download





FHW (n, k) Fork-Join System

- $k = n$ means there is no redundancy $\Rightarrow$ fork-join FHW queue.

FHW (n, k) Fork-Join System

- $k = n$ means there is no redundancy $\Rightarrow$ fork-join FHW queue.
- $k = 1$ means replication $\Rightarrow$ n independent M/M/1 queues.

FHW (n, k) Fork-Join System

- $k = n$ means there is no redundancy $\Rightarrow$ fork-join FHW queue.
- $k = 1$ means replication $\Rightarrow$ n independent M/M/1 queues.
- $1 < k < n$ means coding $\Rightarrow$
  1. there is no independence and the system is not memoryless $\Rightarrow$ hard to derive analytical results,
  2. but there is enough independence to benefit from diversity.

We are interested in the mean response time $T_{n,k}$.

Previous work has attempted finding $T_{n,n}$, but only bounds are known.

Upper Bound on Response Time $T_{n,k}$

Consider a modified (n, k) fork-join system in which a completed task does not exit its queue until k tasks of the same job are completed. (cf. split-merge system)

The (n, k) split-merge system

- has response time greater than its fork-join counterpart, and
- is equivalent to an $M/G/1$ queue with service time S_k, the k^{th} order statistics of exp($k\mu$), with the mean and variance

$$\mathsf{E}[S_k] = \frac{H_n - H_{n-k}}{k\mu} \qquad \mathsf{V}[S_k] = \frac{H_{n^2} - H_{(n-k)^2}}{k\mu^2}.$$

$\Rightarrow$ An upper bound on $T_{n,k}$ is given by the Pollaczek-Khinchin formula:

$$T_{n,k} \leqslant \mathsf{E}[S_k] + \frac{\lambda\left(\mathsf{V}[S_k] + \mathsf{E}[S_k]^2\right)}{2(1 - \lambda\mathsf{E}[S_k])}$$

Remarks on the Upper Bound

Stability Condition:

$$\frac{1}{\lambda} > \mathsf{E}[S_k] \quad \Rightarrow \quad \frac{\lambda}{\mu} \cdot \frac{H_n - H_{n-k}}{k} < 1$$

Remarks on the Upper Bound

Stability Condition:

$$\frac{1}{\lambda} > \mathsf{E}[S_k] \quad \Rightarrow \quad \frac{\lambda}{\mu} \cdot \frac{H_n - H_{n-k}}{k} < 1$$

The Nelson & Tantawi approach upper bound on $T_{n,n}$:

- the response times of the n queues form a set of associated RVs
- the expected maximum of associated RVs is smaller than that of independent RVs with identical marginal distributions.

$$T_{n,n} \leqslant \frac{H_n}{n\mu - \lambda}$$

This does not hold for the k^{th} order statistics when $k < n$.

Lower Bound on Response Time $T_{n,k}$

Stages of Job Processing:

(Varki et al. approach)

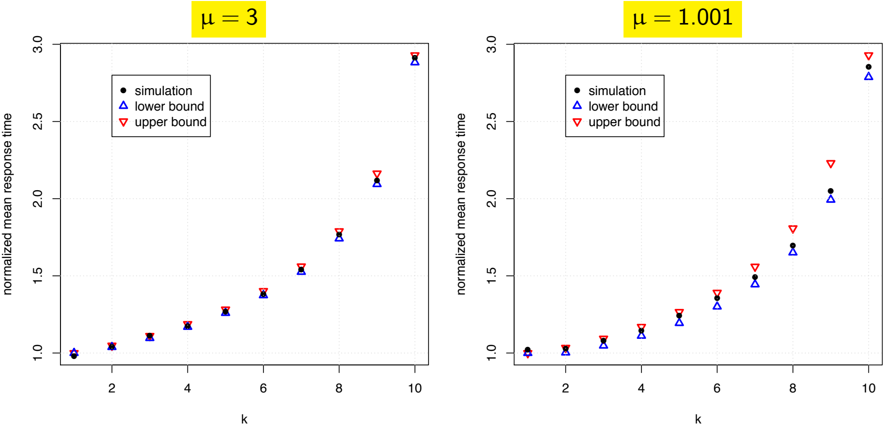
- A job goes through k stages of processing, one for each task.
- At stage j, $0 \leqslant j \leqslant k-1$, the job has completed j tasks.
- The service rate of a job in stage j stage is at most $(n-j)k\mu$.

$$T_{n,k} \geqslant \sum_{j=0}^{k-1} \frac{1}{(n-j)k\mu - \lambda} \quad \leftarrow \text{sum of response times of } k \text{ stages}$$

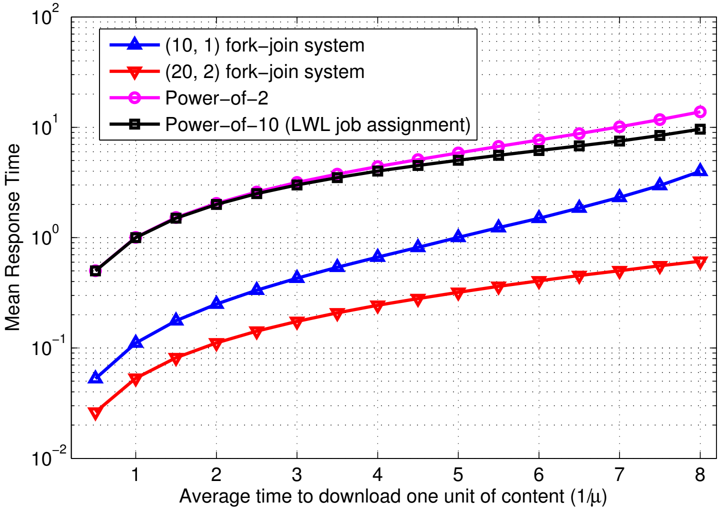
$$= \frac{1}{k\mu}\left[H_n - H_{n-k} + \rho \cdot (H_{n(n-\rho)} - H_{(n-k)(n-k-\rho)})\right] \quad (\rho = \frac{\lambda}{\mu})$$

Tightness of the Bounds

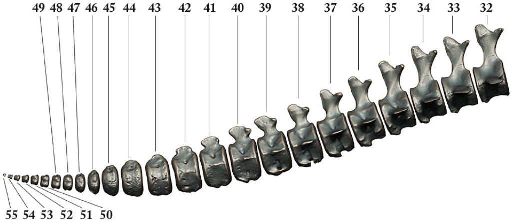
$(10, k)$ Fork-Join Queue, FHW Model, $\lambda = 1$



Diversity – the Power of Choosing All

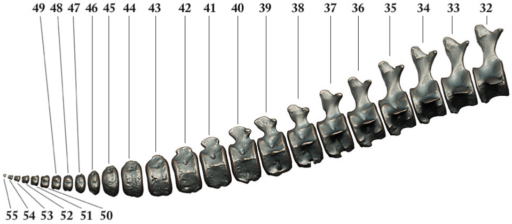


A Coding Tale of a Tail at Scale



Splitting jobs into smaller tasks allows parallel task execution, but increases randomness in the system, **hence the tail.**

A Coding Tale of a Tail at Scale



Splitting jobs into smaller tasks allows parallel task execution, but increases randomness in the system, **hence the tail.**

Coding cuts the tail.

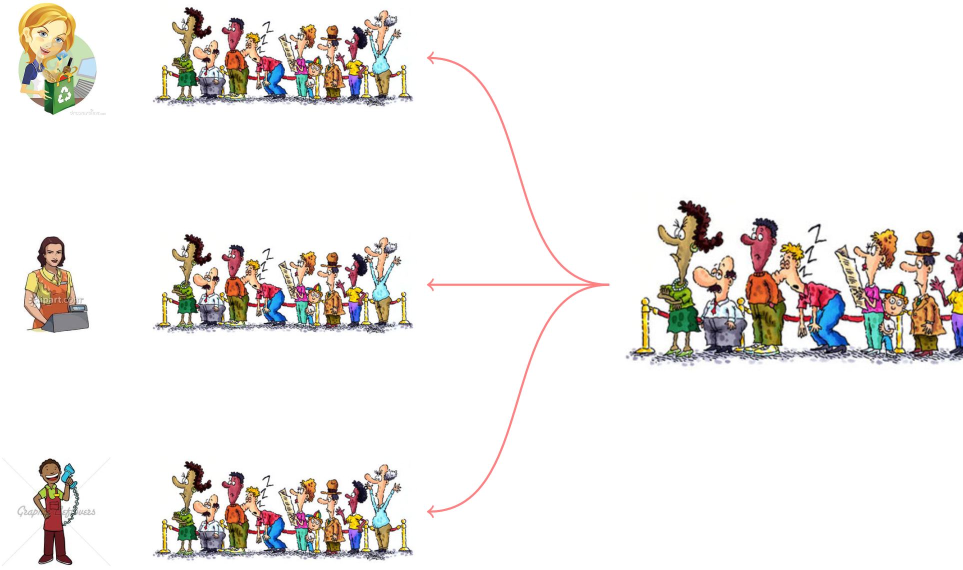
Which jobs permit cutting the tail?

When are the Bounds Valid, Tight, Applicable?

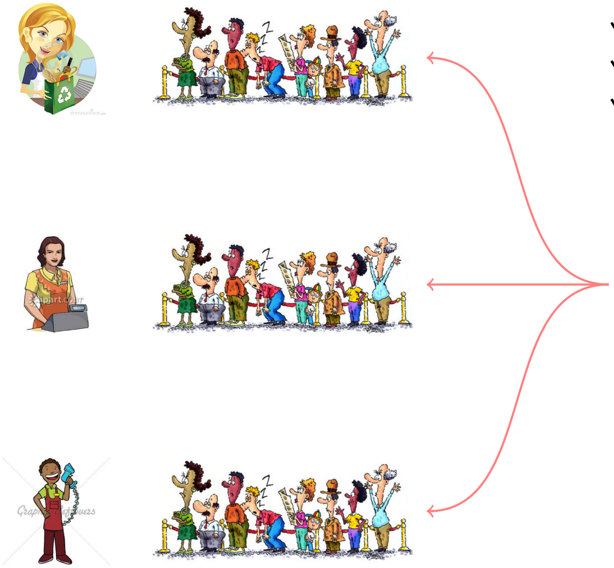
What is a Good Model for Service Time?

What is a Good Model for Service Time?

What is a Good Model for Service Time?



What is a Good Model for Service Time?



✓log-concave (convex)
✓job-dependent
✓cancelation time

Cost of Replication

Allerton'15, with G. Joshi and G. Wornell, MIT

- A job is forked in n independent & statistically identical servers.
- If a single server completes the job in time X, then
  - the $(n, 1)$ system completes the job in time $X_{1:n}$ ← delay
  - the system service time spent on the job is $n \cdot X_{1:n}$ ← cost

Cost of Replication

Allerton'15, with G. Joshi and G. Wornell, MIT

- A job is forked in n independent & statistically identical servers.
- If a single server completes the job in time X, then
  - the $(n, 1)$ system completes the job in time $X_{1:n}$ $\leftarrow$ delay
  - the system service time spent on the job is $n \cdot X_{1:n}$ $\leftarrow$ cost

The expected cost of replication:

$nE[X_{1:n}] = E[X]$ if X is exponential,

$nE[X_{1:n}] \leqslant E[X]$ if $\bar{F}_X$ is log-convex,

$nE[X_{1:n}] \geqslant E[X]$ if $\bar{F}_X$ is log-concave.

Cost of Replication

Allerton'15, with G. Joshi and G. Wornell, MIT

Theorem:

If $\bar{F}_X$ log-convex, then $nE[X_{1:n}]$ is non-increasing in n.
If $\bar{F}_X$ log-concave, then $nE[X_{1:n}]$ is non-decreasing in n.

Implications:

- How many redundant requests should be issued and when?
- When is canceling redundant tasks beneficial?

How About Hot Data?

How About Hot Data?

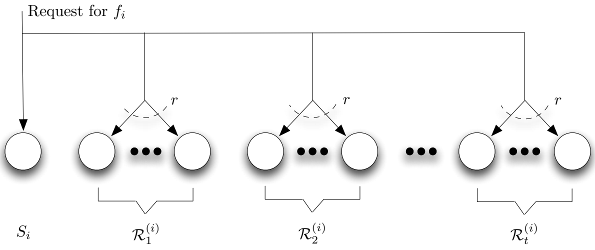
A code has (r, t) availability if

- there are t disjoint repair groups for each data symbol &
- each repair group has at most r symbols.

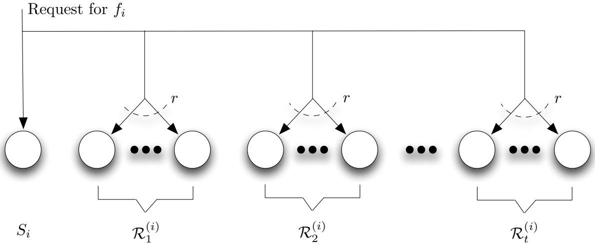
A $(2, 3)$-availability code:

$$\{a, b, c\} \longrightarrow \{\boxed{a}, b, c, a+b, b+c, a+c, a+b+c, \}$$

How Helpful is a Repair Group

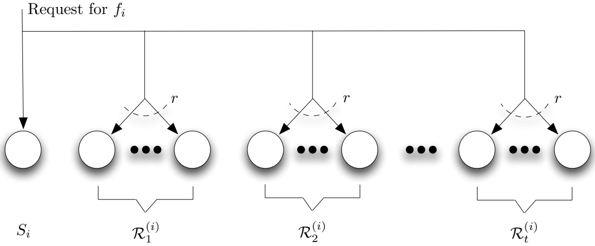


How Helpful is a Repair Group

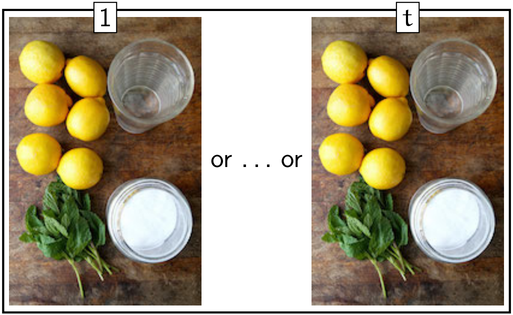


vs.

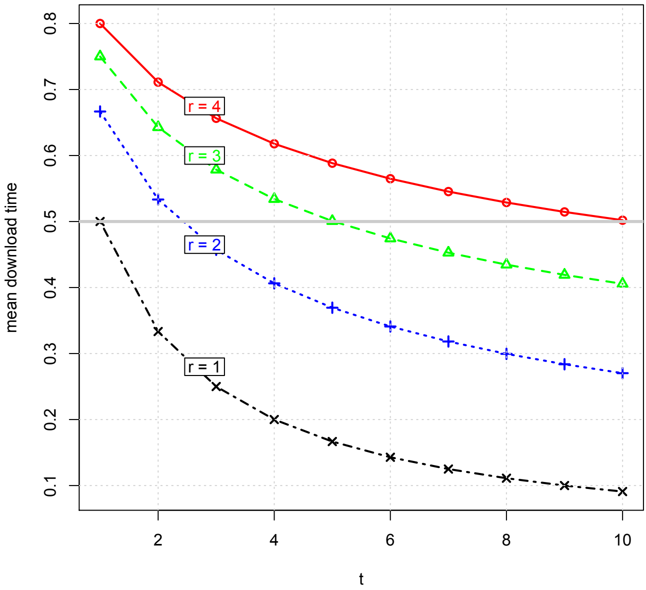
How Helpful is a Repair Group



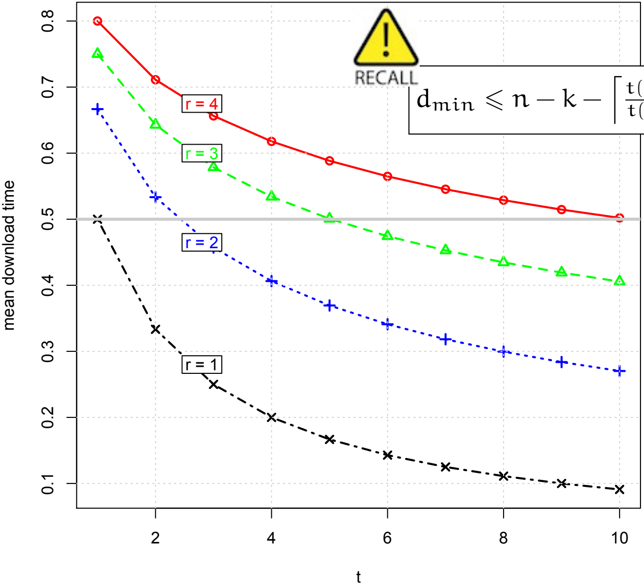
vs.



Decrease r or Increase t?



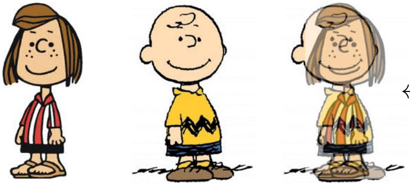
Decrease r or Increase t?



$$d_{min} \leqslant n - k - \left\lceil \frac{t(k-1)+1}{t(r-1)+1} \right\rceil + 2$$

When Data is Changing and/or Expanding ...

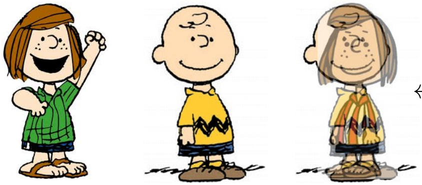
Data Updates



← Update $\boxed{a, b, a+b}$ with new a.

When Data is Changing and/or Expanding ...

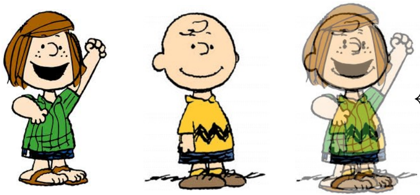
Data Updates



$\leftarrow$ Update $\boxed{a, b, a+b}$ with new a.

When Data is Changing and/or Expanding ...

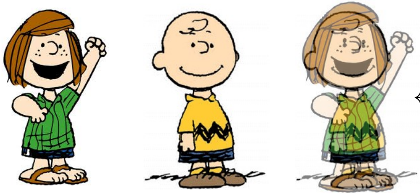
Data Updates



← Update $\boxed{a, b, a+b}$ with new a.

When Data is Changing and/or Expanding ...

Data Updates



$\leftarrow$ Update $\boxed{a, b, a+b}$ with new a.

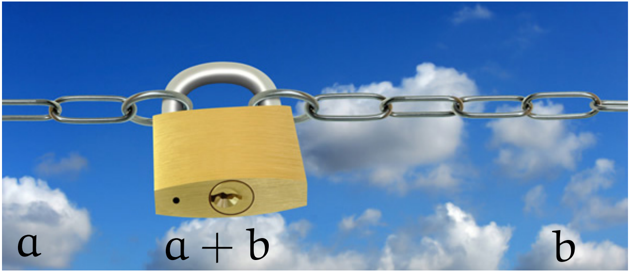
Storage Upgrades

vs.

Expand $\boxed{a, b, a+b}$ to include c and d.

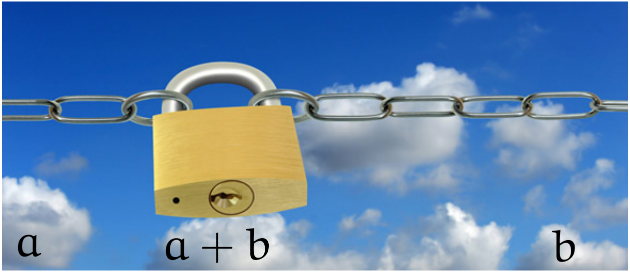
Cloud Data Security

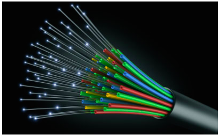
Challenges are posed by coding, distributed storage, independent clouds.



Cloud Data Security

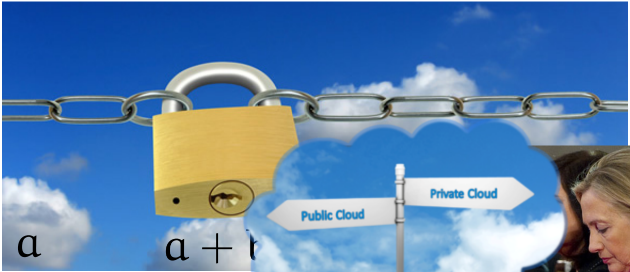
Challenges are posed by coding, distributed storage, independent clouds.

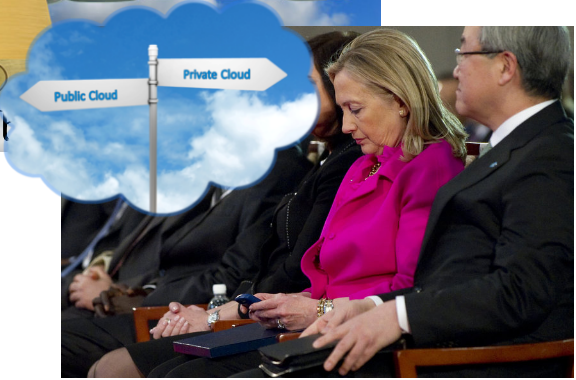


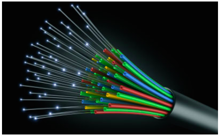


Cloud Data Security

Challenges are posed by coding, distributed storage, independent clouds.







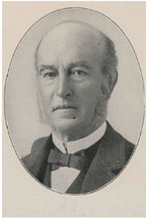
Data Centers, Energy, and Smart (Power) Grid

Data centers electricity usage is
large $\approx 2-3\%$ of the US total electricity,
& growing $\approx 12\%$ (cf. 1% total growth).

Redundant requests reduce storage requirements for a given latency.
But do they introduce some other costs?

Who's Getting Rich in The Big Data Gold Rush?

Who's Getting Rich in The Big Data Gold Rush?



Some Papers

G. Joshi, Y. Liu, and E. Soljanin, "On the delay-storage trade-off in content download from coded distributed storage systems," *IEEE J-SAC Special Issue on Communication Methodologies for the Next-Generation Storage Systems*, pp. 989–997, May 2014.

G. Joshi, E. Soljanin, and G. Wornell, "On the delay-storage trade-off in content download from coded distributed storage systems," *ACM Trans. on Modeling and Performance Evaluation of Computing Systems*, submitted Oct. 2015.

S. Kadhe, E. Soljanin, and A. Sprintson, "Analyzing the download time of availability codes," *2015 IEEE Int. Symp. Inform. Theory (ISIT'15)*, Hong Kong, June 2015.