MODELING AND OPTIMIZATION OF LATENCY IN ERASURE-CODED STORAGE SYSTEMS

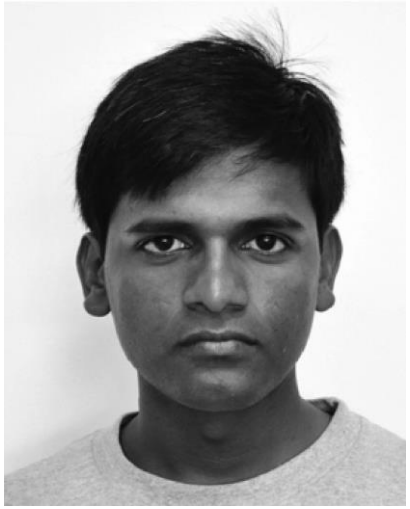
Vaneet Aggarwal, Tian Lan, Parimal Parag

IISc GW UNIVERSITY WASHINGTON DC PURDUE UNIVERSITY

OUR TEAM

Vaneet Aggarwal
Purdue

Tian Lan
GWU



Parimal Parag
IISc

PART 1: INTRODUCTION

CLOUD STORAGE

Source: https://www.google.com/url?sa=i&url=https%3A%2F%2Fmedium.com%2F%40spirosx%2Fthe-beginners-guide-to-the-cloud-d5d6c48a4f46&psig=AOvVaw3CIuhUwfhHpJOPX5Fh66F8&ust=1634791868711000&source=images&cd=vfe&ved=2ahUKEwj4w5m2mNjzAhWIbqwKHSlvBAgQr4kDegQIARAx

Source: https://www.makeuseof.com/cloud-trends-in-2021-and-beyond/

- Demand for storage services increasing rapidly (Backup, photos, videos)
- Companies increasing storage space rapidly (Baidu 2TB, Qihoo 360 36TB)

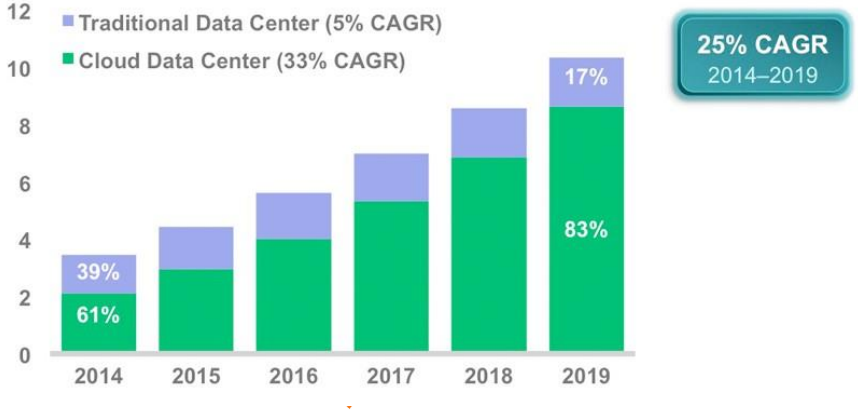
IISc GW UNIVERSITY WASHINGTON DC PURDUE UNIVERSITY

GROWTH IN CLOUD STORAGE

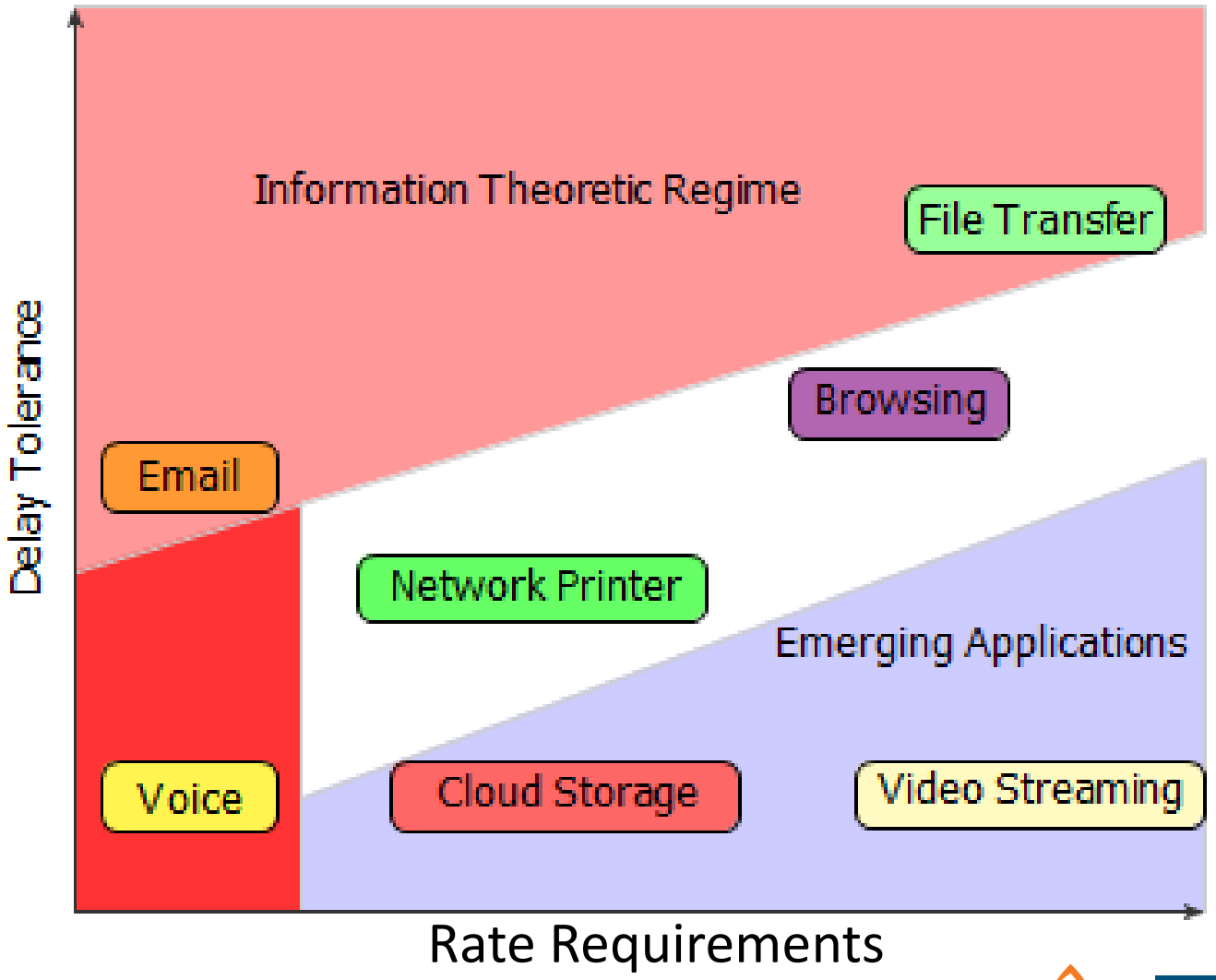
Source: https://time.com/wp-content/uploads/2015/05/cloud_index_white_paper.pdf

- Growth in personal cloud storage and sharing of photos/videos/documents
- The global **cloud storage** market size is projected to grow from USD 50.1 billion in 2020 to USD 137.3 billion by 2025, at a Compound Annual **Growth** Rate (CAGR) of 22.3% during the forecast period (MarketsandMarkets.com).

Source: https://www.forbes.com/sites/gilpress/2015/10/28/cisco-says-cloud-traffic-to-quadruple-by-2019-driven-by-consumers-mobility-and-iot/

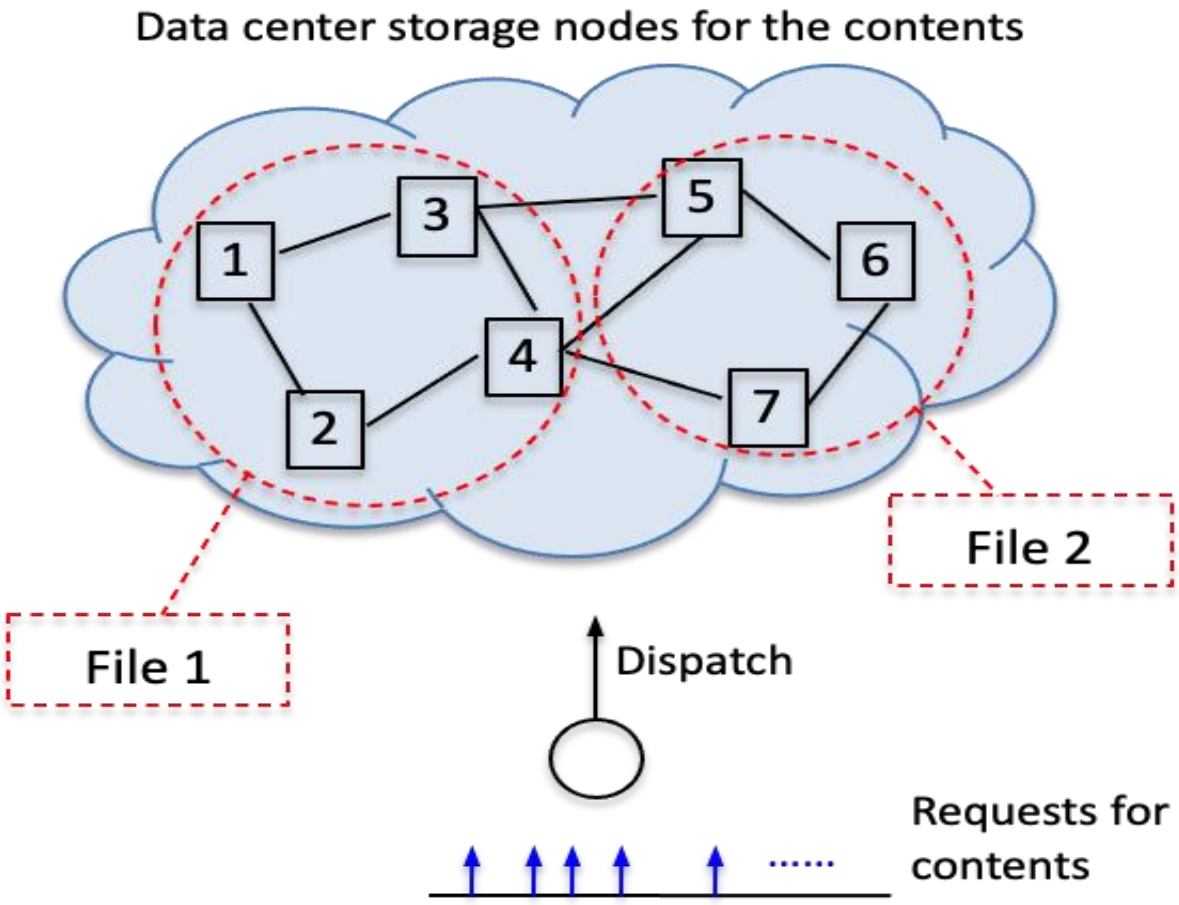


EVOLVING DIGITAL LANDSCAPE



IISc GW UNIVERSITY WASHINGTON DC PURDUE UNIVERSITY

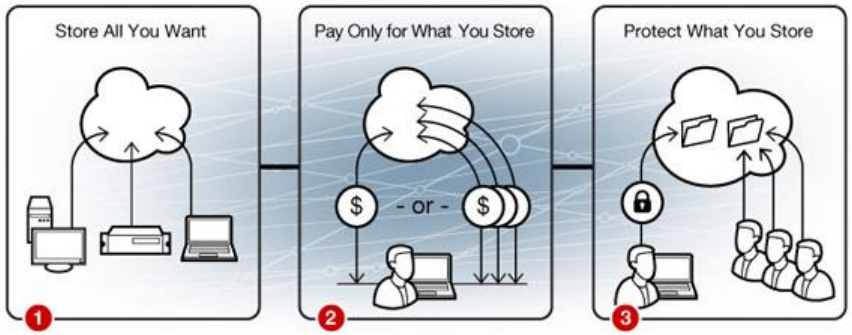
KEY PROBLEM IN THIS TUTORIAL



- Modeling, characterization, and optimization of latency for distributed storage systems

USER REQUIREMENTS

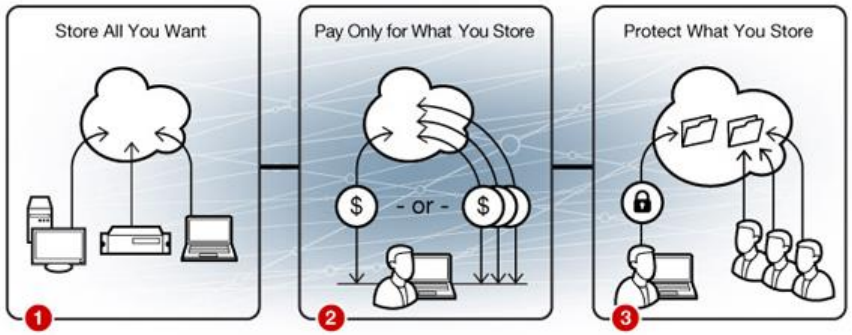
- Latency: Need content within certain seconds
- Reliability: Data should not be lost
- Cost: Less cost to store
- Security: Data should not go in wrong hands



Efficient distributed storage that works on the cloud with limited bandwidth links

USER REQUIREMENTS

- Latency: Need content within certain seconds
- Reliability: Data should not be lost
- Cost: Less cost to store
- Security: Data should not go in wrong hands



Efficient distributed storage that works on the cloud with limited bandwidth links

Conflicting Requirements

Need redundancy

IISc GW UNIVERSITY WASHINGTON DC PURDUE UNIVERSITY

REDUNDANCY WITH ERASURE CODES

- Triple Replication – 2x storage overhead
- Erasure codes: same reliability with less overhead as compared to replication
- All companies moving towards erasure coded storage systems
- Cleversafe: 60% more capacity, 100 million X reliable, 80-90% less cost

Facebook's advanced erasure codes

by ROBIN HARRIS on FRIDAY, 21 JUNE, 2013

We want our data protected from device failures. When there is a failure we want to get our data back quickly. And we want to pay as little as possible for the protection and the restore. How?

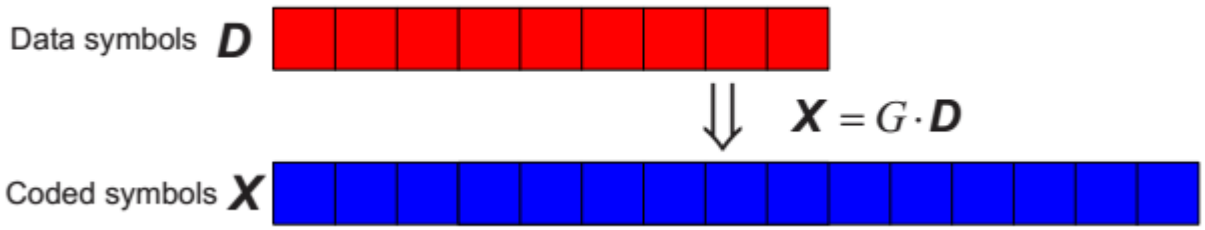
Data Series: USENIX Best Paper Award - Erasure Coding in Windows Azure Storage

13 Jun 2012 9:56 AM BradCalder 0

We just published a paper describing how we Erasure Code data in Windows Azure Storage that won a Best Paper Award at the June 2012 USENIX Annual Technical Conference. This was joint work between Microsoft Research and the Windows Azure Storage team.

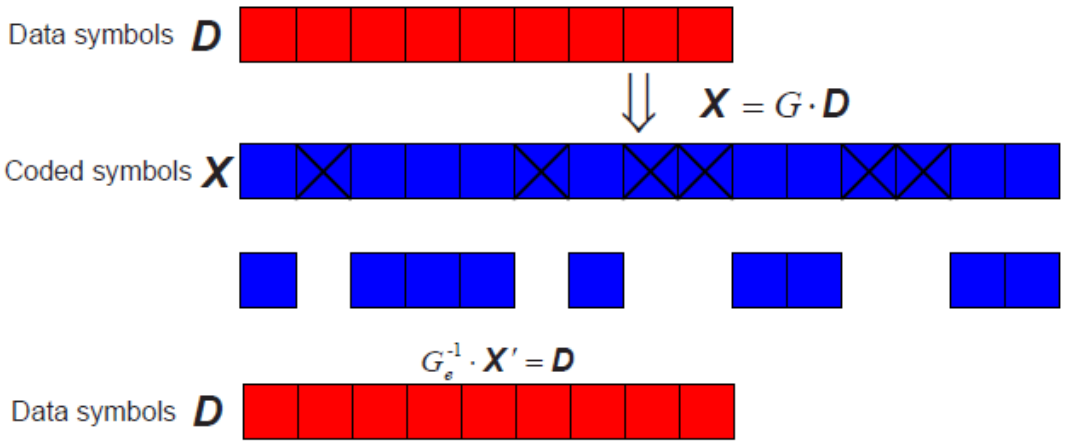
WHAT IS AN ERASURE CODE?

- Erasure Code (EC) involves encoding the message in a redundant manner
- EC transforms message of k symbols to n symbols

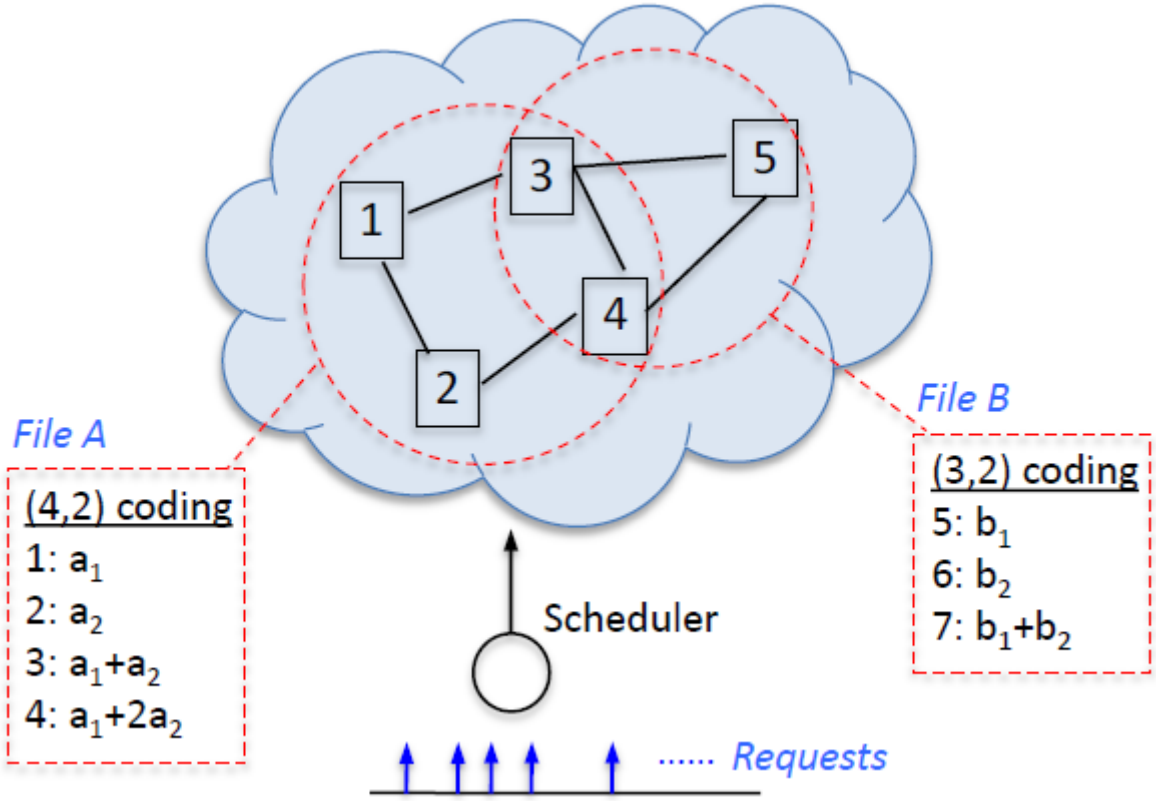


WHAT IS AN ERASURE CODE?

- Erasure Code (EC) involves encoding the message in a redundant manner
- EC transforms message of k symbols to n symbols
- There exists a set of k un-erased symbols for recovery
- For MDS codes, any k un-erased symbols suffice (e.g., Reed-Solomon codes)

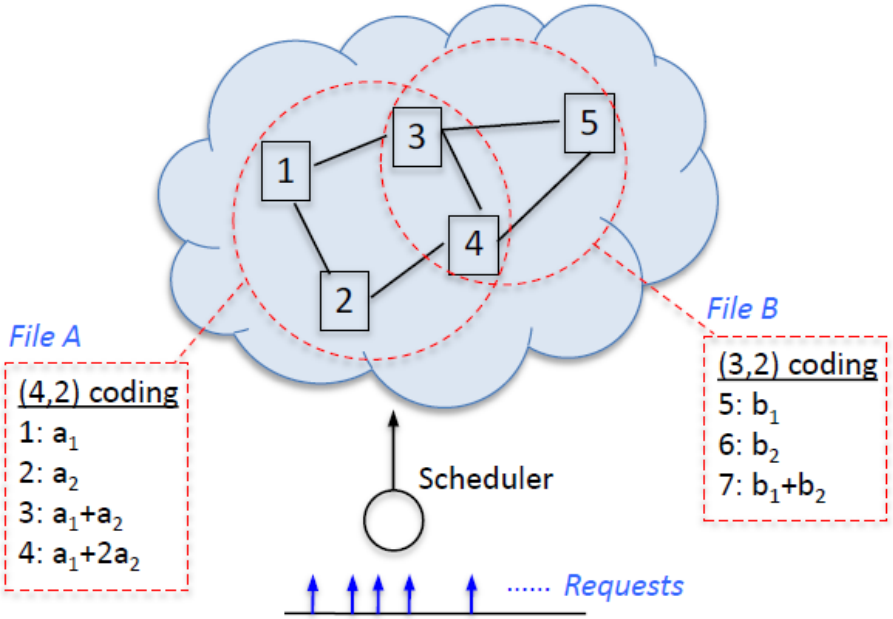


FILE REQUEST IN CLOUD STORAGE SYSTEMS



- Request for files, each file has multiple chunks
- Given placement, latency=?

FILE REQUEST IN CLOUD STORAGE SYSTEMS



- Key Questions:
  - What is the choice of scheduling strategy?
  - How to characterize different measures of latency?
  - How much redundancy to add?
  - What is the optimal placement for coded chunks?
  - How to exploit the tradeoff between latency and cost?

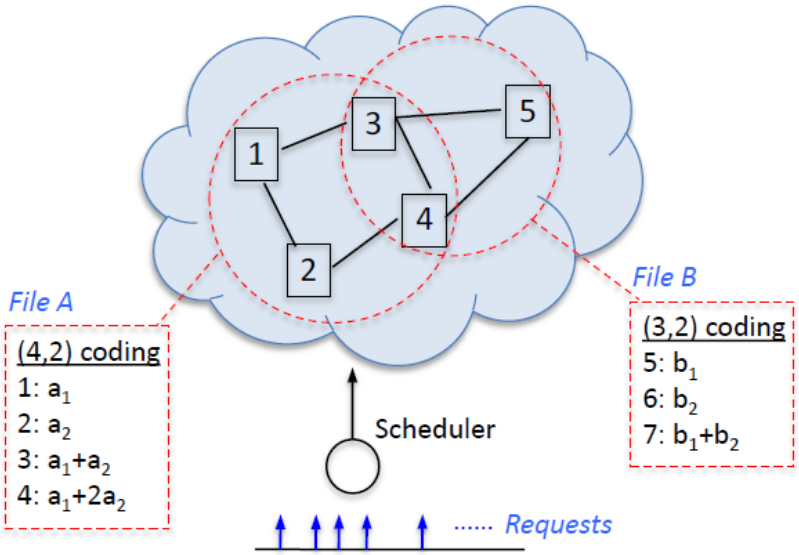
ASSUMPTIONS

- File Request is Poisson (Each file request means get any k of n chunks)
- There are multiple servers (say m)
- Each server has a service rate given by some distribution

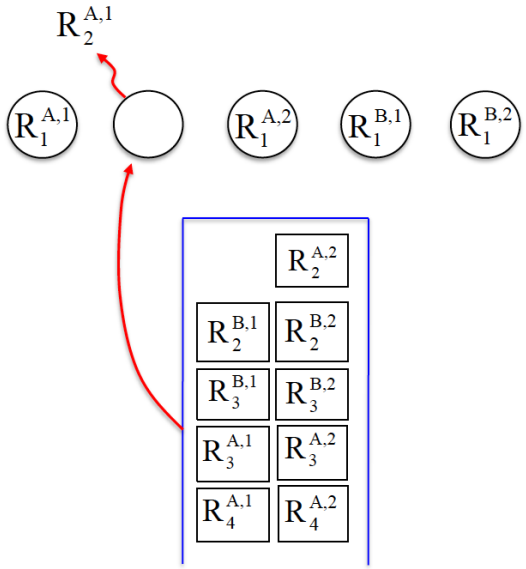
ASSUMPTIONS

- File Request is Poisson (Each file request means get any k of n chunks)
- There are multiple servers (say m)
- Each server has a service rate given by some distribution
- Single file, k=n=1, this is M/G/1 queue model
- In general, open problem
- Queuing theory techniques don't account for simultaneous arrivals at different servers

OPTIMAL SCHEDULING IS HARD



Erasure-coded storage.



Scheduling problem.

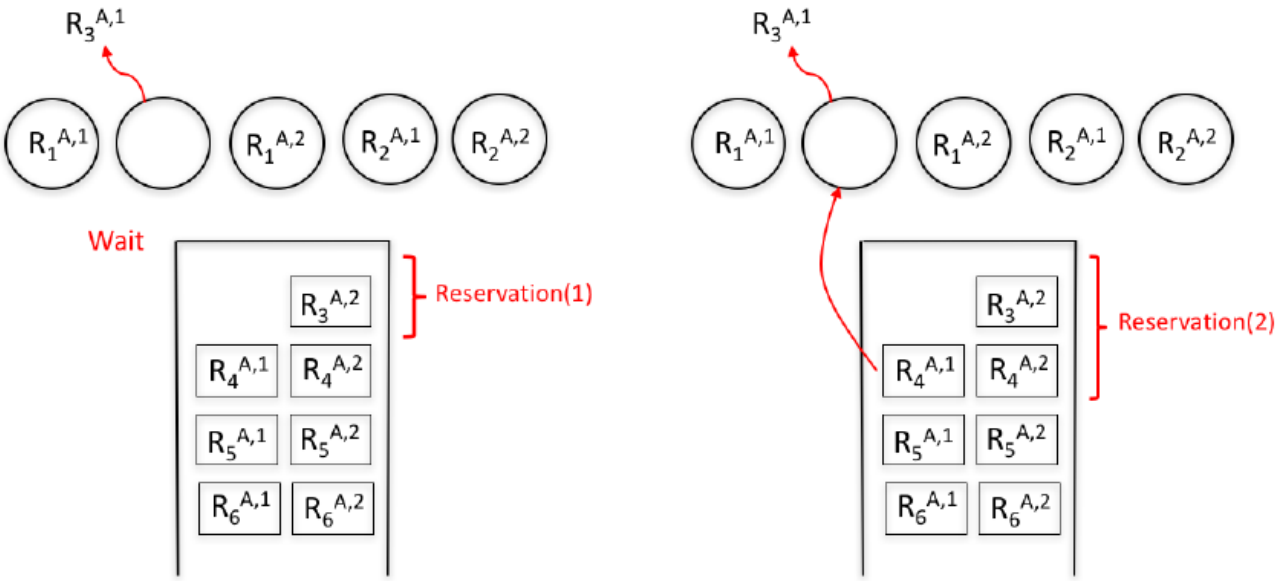
OPTIMAL SCHEDULING IS HARD

- Very large state space due to synchronous arrival to multiple servers, and need to keep track of their partial service
- Latency characterizations are difficult, since the underlying processes are multi-dimensional
- Even when the process evolutions are Markov, equilibrium distributions are unknown as it is equivalent to finding eigenfunctions of multi-dimensional operators
- Finding the latency optimal scheduling policy is still open

KEY SCHEDULING STRATEGIES

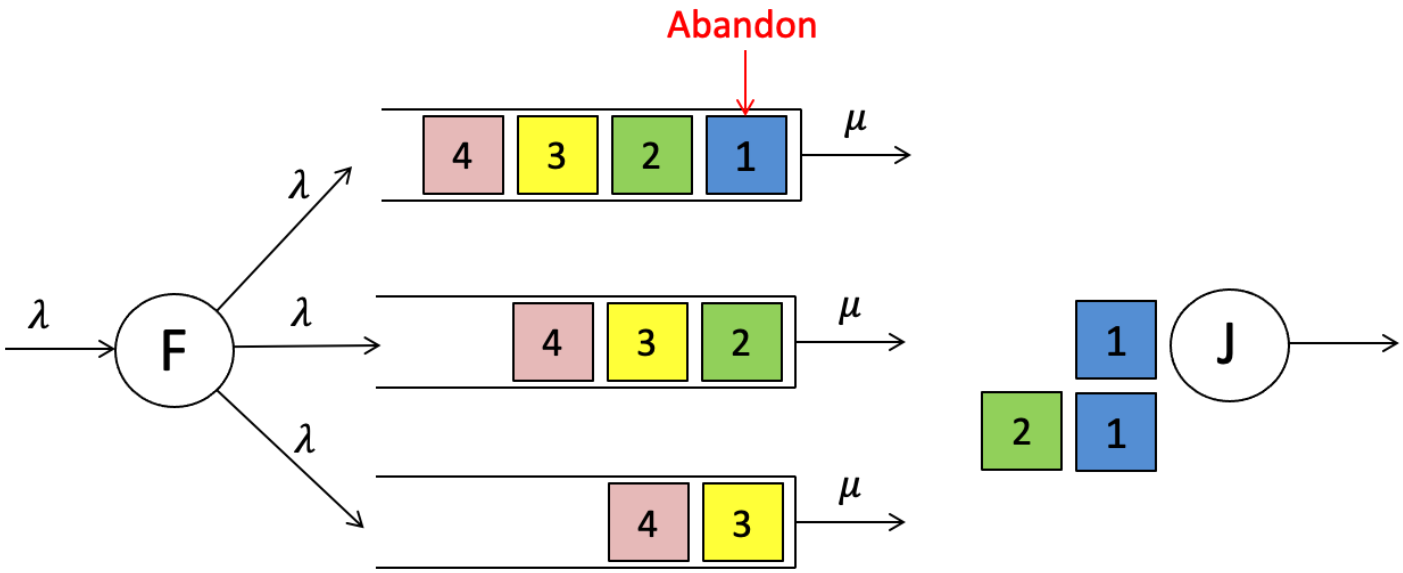
- Reservation scheduling
  - Huang, Pawar, Zhang, Ramchandran (2012), Lee, Shah, Huang, Ramchandran (2017)
- Fork-join scheduling
  - Joshi, Liu, Soljanin (2014), Joshi, Soljanin, Wornell (2017), Kumar, Tandon, Clancy (2017), Badita, Parag, Chamberland (2019)
- Probabilistic scheduling
  - Xiang, Lan, Aggarwal, Chen (2014, 2016), Aggarwal, Fan, Lan (2017), Alabbasi, Aggarwal, Lan (2019), Wang, Harchol-Balter, Jiang, Scheller-Wolf, Srikant (2019)
- Delayed-relaunch scheduling
  - Badita, Parag, Aggarwal (2020, 2021)

RESERVATION SCHEDULING



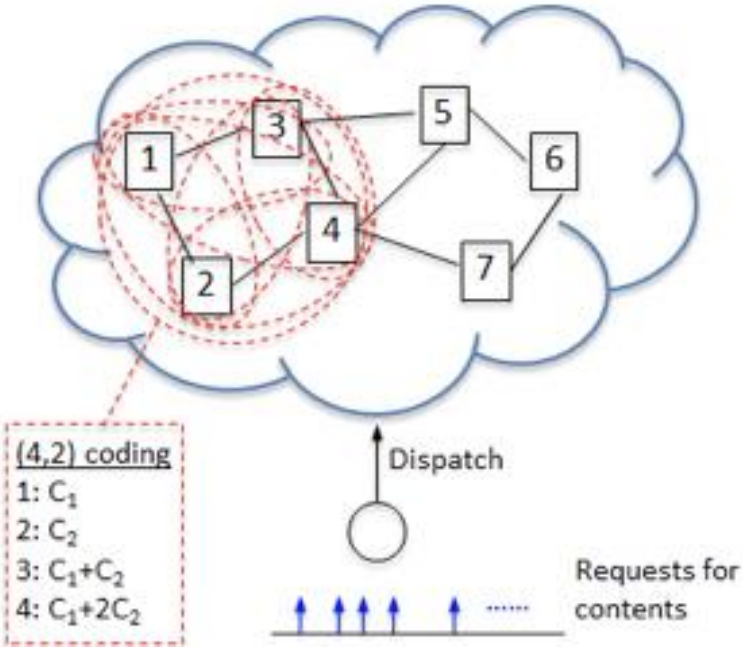
- Reservation scheduling
  - Huang, Pawar, Zhang, Ramchandran (2012), Lee, Shah, Huang, Ramchandran (2017)

FORK-JOIN SCHEDULING



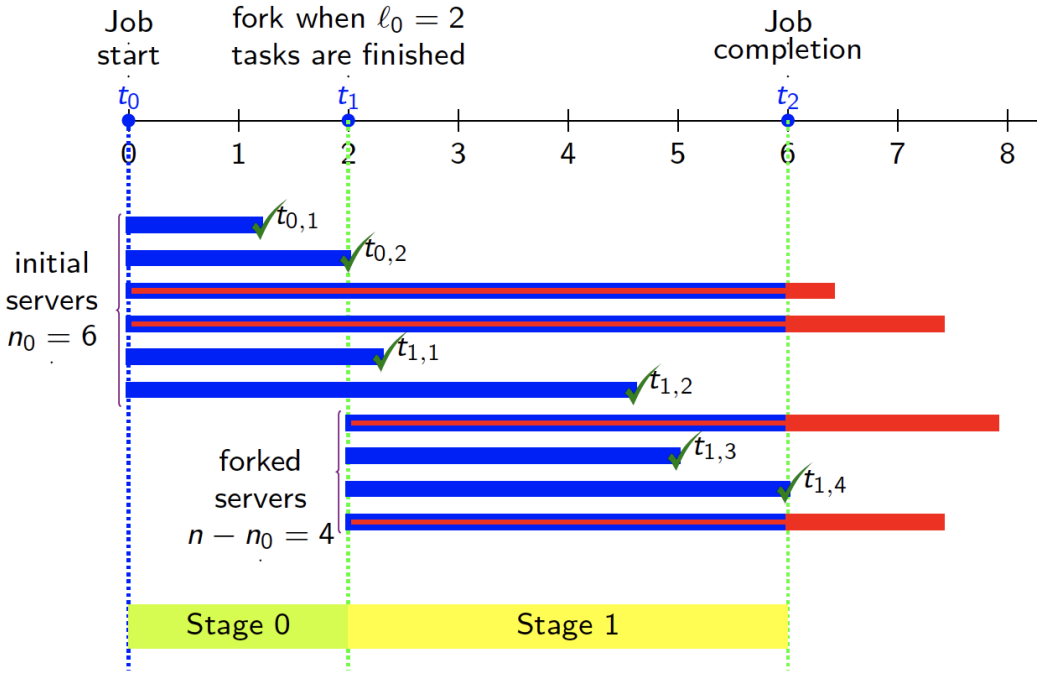
- (n,k) fork-join queue:
  - Joshi, Liu, Soljanin (2014), Joshi, Soljanin, Wornell (2017), Kumar, Tandon, Clancy (2017), Badita, Parag, Chamberland (2019).

PROBABILISTIC SCHEDULING



- Probabilistic scheduling chooses different k-subsets with some probability
  - Xiang, Lan, Aggarwal, Chen (2014, 2016), Aggarwal, Fan, Lan (2017), Alabbasi, Aggarwal, Lan (2019), Wang, Harchol-Balter, Jiang, Scheller-Wolf, Srikant (2019)

DELAYED-RELAUNCH SCHEDULING



- Delayed-Relaunch scheduling: Job at some servers are started with a delay based on completion of some tasks.
  - Badita, Parag, Aggarwal (2020, 2021)

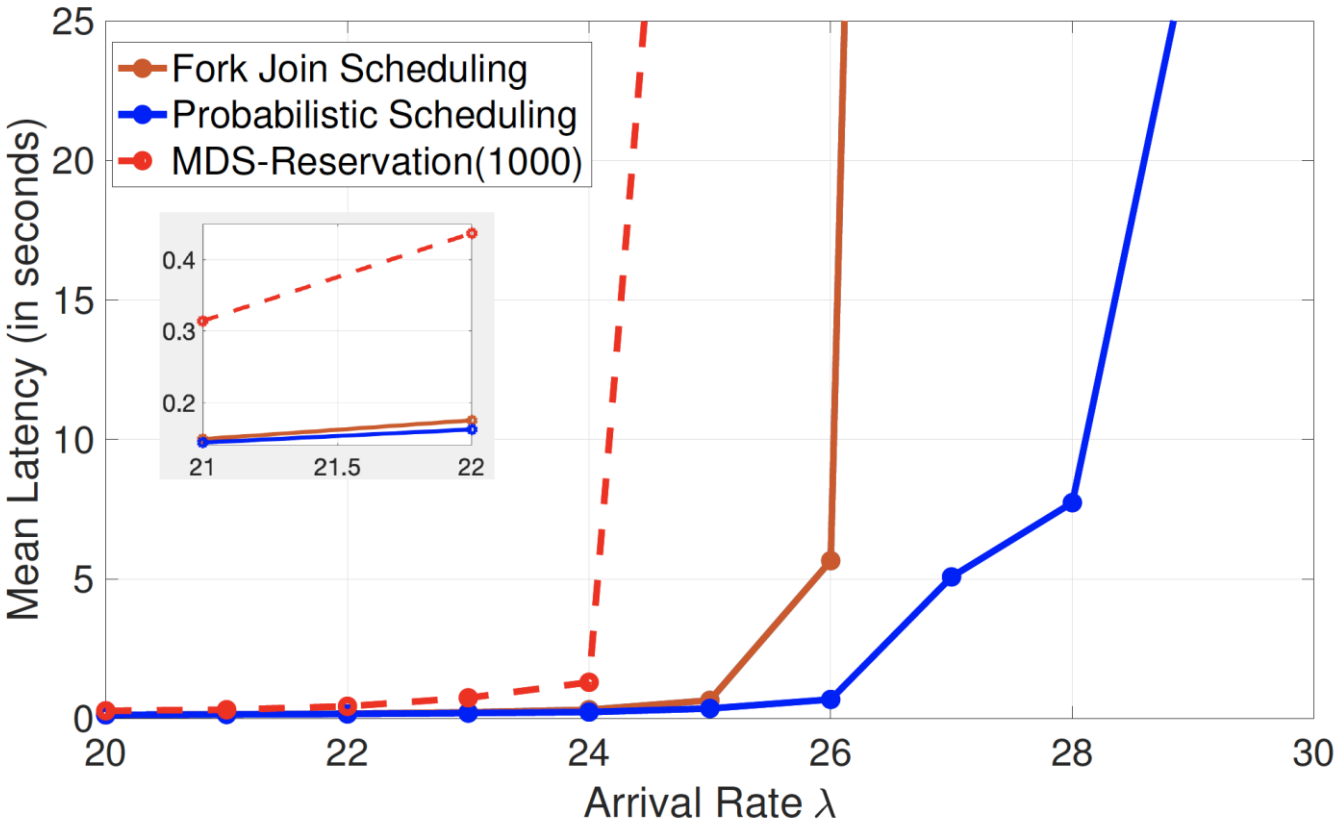
COMPARISON OF STATE-OF-ART: ASSUMPTIONS

	MDS-reservation	Fork-Join	Probabilistic	Delayed relaunch
Homogenous Files	Yes	No	No	Yes
Homogenous Placement	Yes	Yes	No	Yes
Homogeneous Servers	Yes	Yes	No	Yes
Exponential Service Time	Yes	No	No	No[3]

COMPARISON OF STATE-OF-ART: ANALYSIS RESULTS

	MDS-Reservation	Fork-Join	Probabilitic	Delayed Relaunch
Optimal Homogenous Stability Region	No	Exponential	General	General
Queuing Analysis	Yes	Yes	Yes	No
Analysis for general distribution	No	Yes	Yes	No
Closed Form Expressions	No	Yes	Yes	N/A[5]
Asymptotic Optimality	No	No	Yes	Yes
Tail Characterization	No	No	Yes	No

NUMERICAL COMPARISON OF KEY SCHEDULING STRATEGIES



- Shifted Exponential Service Times, 12 servers
- Homogenous files with (12,7) code
- Hyperparameter search for probabilistic scheduling by choosing best of 100 random selections.

- MDS-Reservation and Fork-Join strategies do not achieve the optimal stability region
- Probabilistic scheduling outperforms Fork-Join scheduling for all arrival rates in this simulation

OUTLINE

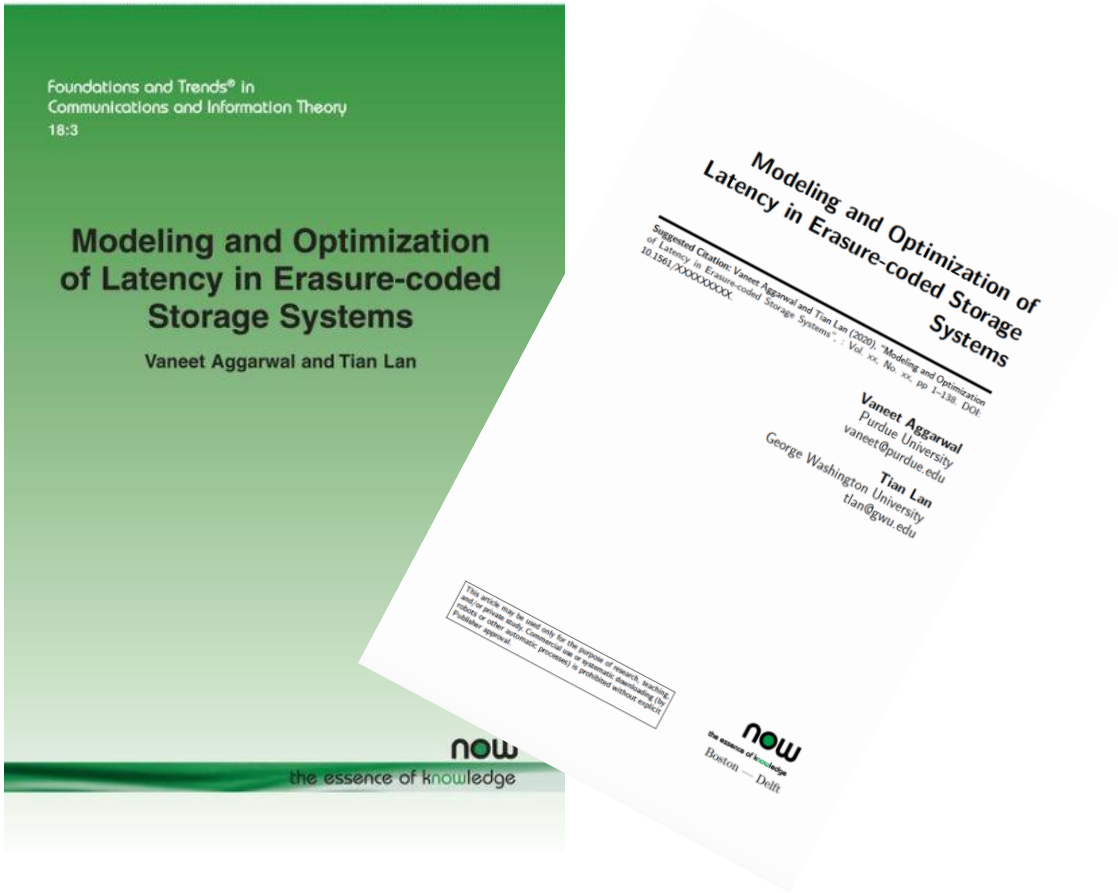
- Introduction
- Fork-join scheduling
- Probabilistic scheduling
- Delayed-Relaunch scheduling
- Evaluations and other applications

ACKNOWLEDGEMENTS

- AT&T Research: Yih-Farn Robin Chen (now retired), Moo-Ryong Ra (now at Amazon), Vinay Vaishampayan (now at City University of NY), Chao Tian (now at Texas A&M University)
- Purdue University: Abubakr Al-Abbasi (now at Qualcomm), Jingxian Fan (now at Google), and Ciyuan Zhang
- George Washington University: Yu Xiang (now at AT&T)
- IISc Bangalore: Ajay Badita (now at IOTA), Rooji Jinan, Saraswathy Ramanathan, Vikram Srinivasan
- IIT Madras: Pradeep Sarvepalli
- Rutgers University: Rawad Bitar (now at TUM), Salim El Rouayheb
- Texas A&M University: Jean-Francois Chamberland
- University of Illinois, Chicago: Balajee Vamanan

- Funding:
  - NSF CNS 1618628
  - ONR N00014-20-1-2146
  - CISCO Systems, Inc.
  - AT&T Research
  - Department of Science and Technology, SERB ECR DSTO1677
  - Department of Telecomm, Govt of India, DoTC 0001
  - Robert Bosch Centre for Cyber Physical Systems
  - Centre for Networked Intelligence (a CISCO CSR initiative)

STORAGE BOOK



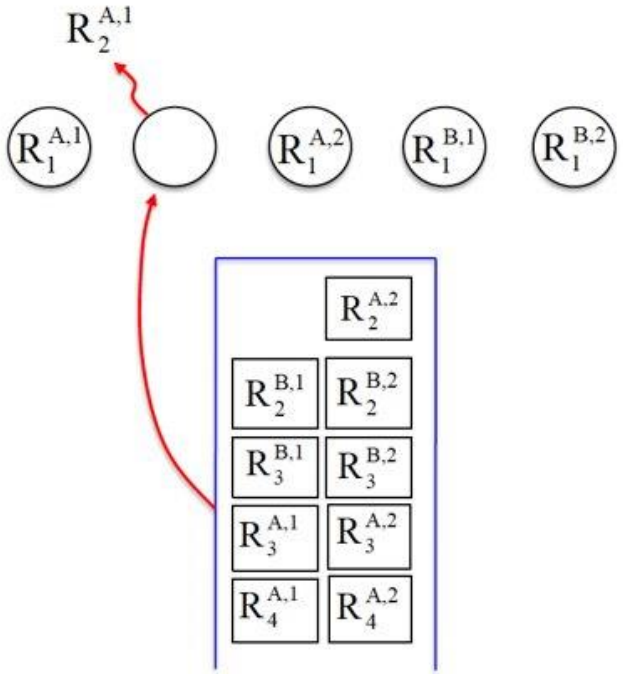
- Promotion Code: 994513

PART 2: FORK-JOIN SCHEDULING

Tian Lan, ECE@GWU

IISc
GW UNIVERSITY WASHINGTON DC
PURDUE UNIVERSITY

RECAP

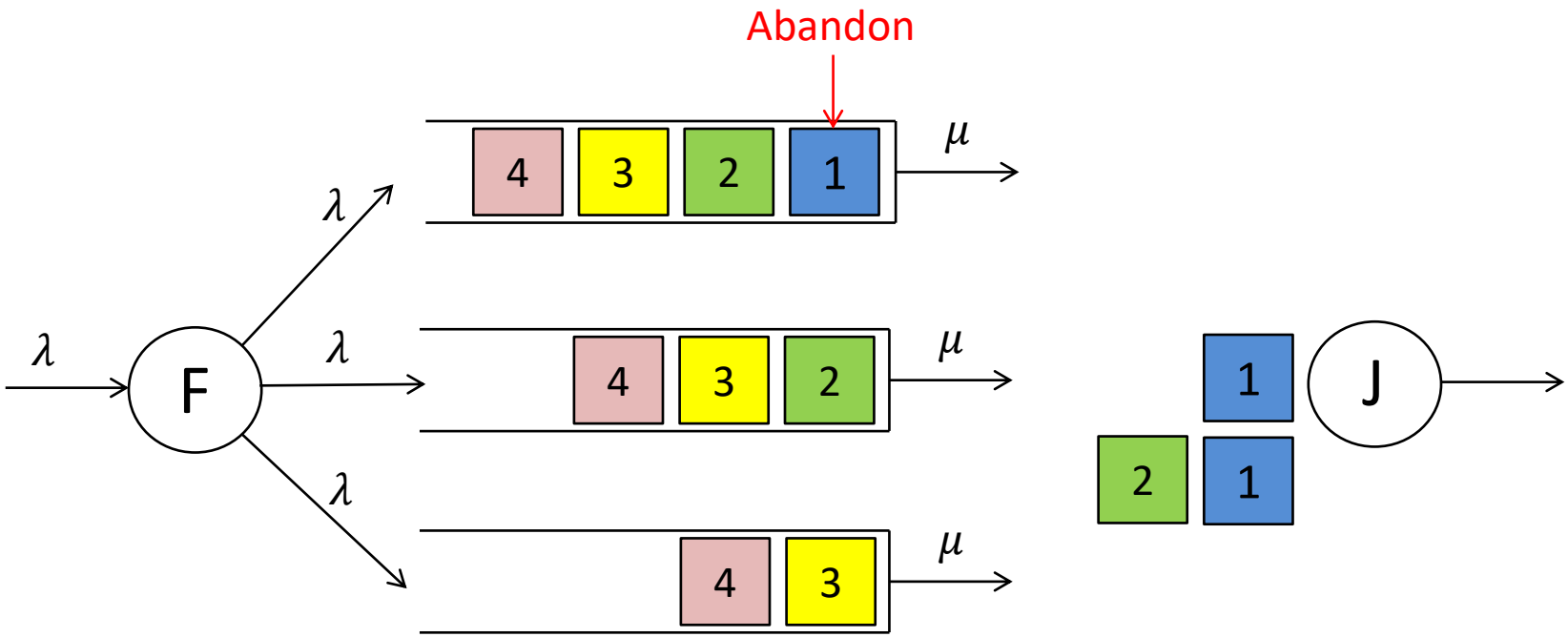


Scheduling problem in erasure-coded storage.

SYSTEM MODEL

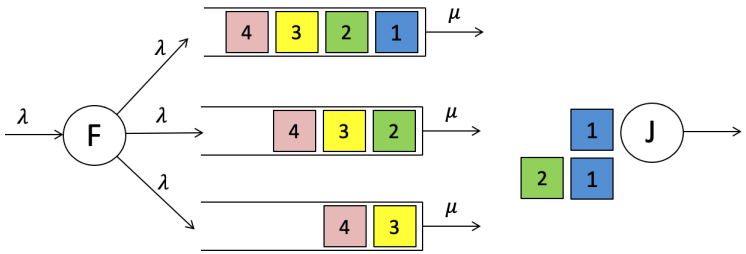
- File storage
  - Each file divided into k chunks
  - Chunks encoded and stored on n servers
- Arrival of requests
  - Each request wants entire file
  - Poisson arrival requests with rate λ
- Time in the system
  - Until the recipient of entire file
- Service at each server
  - i.i.d. exponential service time with rate μ

FORK-JOIN SYSTEM



- (n,k) fork-join queue:
  - Joshi, Liu, Soljanin (2014), Joshi, Soljanin, Wornell (2017), Kumar, Tandon, Clancy (2017), Badita, Parag, Chamberland (2019).

STABILITY OF FORK-JOIN SYSTEM



Theorem

For the (n, k) fork-join system to be stable, the Poisson arrival rate λ and the service rate μ per server must satisfy $\lambda < n\mu/k$.

- Proof outline:
  - When k out of the n tasks finish service, the remaining $n - k$ tasks abandon their queues
  - A task can be one of the abandoning tasks with probability $(n - k)/n$.
  - The effective arrival rate to each queue is λ minus abandonment $\lambda(n - k)/n$.
  - $\lambda - \lambda(n - k)/n < \mu$ gives the condition.

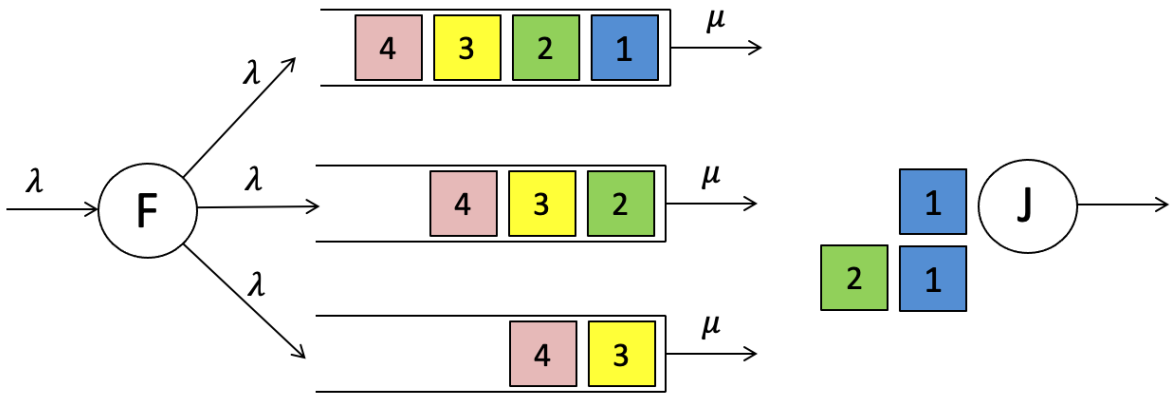
PRIOR WORK AND KEY CONTRIBUTIONS

- Kannan et al: join k queues for replication and MDS codes
  - Numerical bounds using block Markov chains
  - Trade-off between numerical accuracy and computational effort
- Soljanin, Wornell et al: fork-join (n; k) queues for MDS codes
  - Closed-form upper and lower bounds
  - Loose bounds for most of the rate region
- Parag et al: fork-join (n; k) queues for all symmetric codes
  - Tight closed-form approximations for all rate regions
  - Stability region for all symmetric codes

IISc GW UNIVERSITY WASHINGTON DC PURDUE UNIVERSITY

CHALLENGES OF ANALYZING FORK-JOIN

Fork-join:

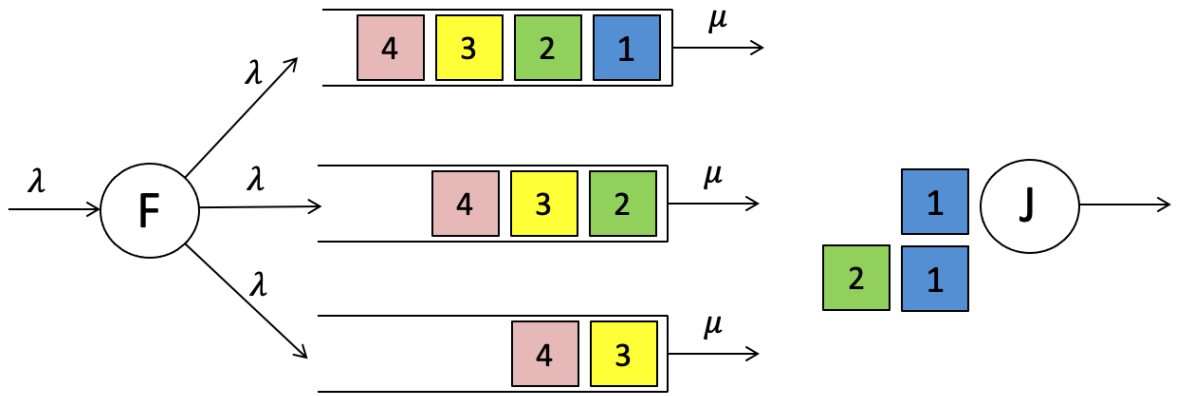


- Recall: Latency is defined as the average time spent in the fork-join system.
- Analyzing the waiting time using Markov Chains requires:
  - Modeling individual queue evolutions that are dependent
  - Encapsulating the execution history in MC

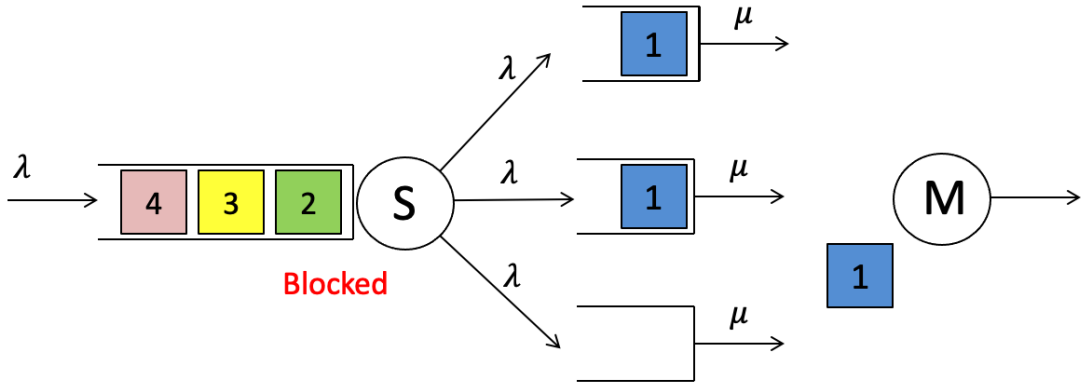
IISc GW UNIVERSITY WASHINGTON DC PURDUE UNIVERSITY

LATENCY ANALYSIS USING SPLIT-MERGE QUEUES

Fork-join:



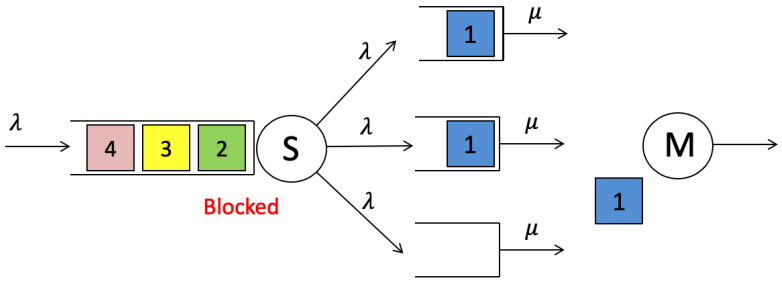
Split-merge:



Split-merge queues provide an upper bound on fork-join.

LATENCY UPPER BOUND

- (n, k) split-merge is equivalent to an $M/G/1$ queue.
  - Arrivals are Poison with rate λ.
  - Service time S is the kth order statistic.



- Find $\mathrm{E}[S]$ and $\mathrm{var}[S]$:
  - Independent services times at the servers.
  - Analyze the kth order statistic of exponential distributions of $1/\mu$.
- Compute the average latency:
  - Use the Pollaczek-Khinchin formula for $M/G/1$ queue.
- It gives an upper bound on the latency of fork-join system.

LATENCY UPPER BOUND

- Given i.i.d. service times $X_1, X_2 \ldots, X_n$.
- Equivalent service time $S = X_{(k)}$, i.e., the kth smallest of $X_1, X_2 \ldots, X_n$.
- Distribution for kth order statistic:
  - $f_{x_{(k)}}(x_k) = \frac{n!}{(n-k)!(k-1)!}[F(x_k)]^{k-1} f(x_k)[1 - F(x_k)]^{n-k}$.
- Applying exponential service time distribution:
  - $E[S] = H^1_{n-k,n}/\mu$ and $\text{var}[S] = H^2_{n-k,n}/\mu^2$,
  - where $H^z_{x,y} = \sum_{j=x+1}^{y} \frac{1}{j^z}$ is the generalized harmonic number of order z.

LATENCY UPPER BOUND

- The Pollaczek-Khinchin formula for $M/G/1$ queue with service time S:

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

- Substituting the values of $\mathrm{E}[S]$ and $\mathrm{var}[S]$, we find an upper bound on the latency of fork-join systems.

LATENCY UPPER BOUND

Theorem

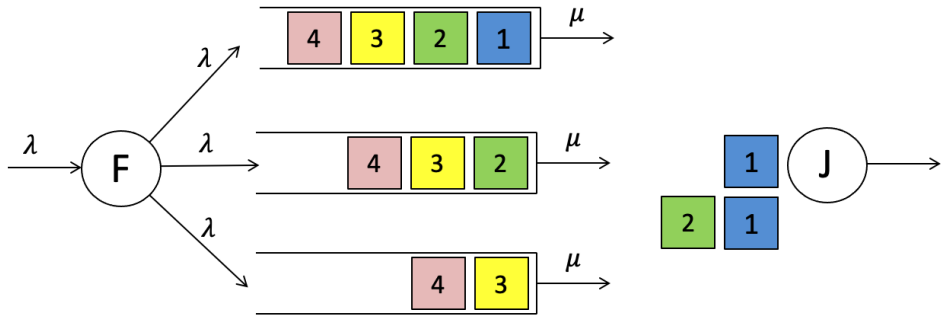
The expected latency $T_{(n,k)}$ for an (n,k) fork-join system satisfies

$$T_{(n,k)} \leq \frac{H^1_{n-k,n}}{\mu} + \frac{\lambda\left[H^2_{n-k,n} + \left(H^1_{n-k,n}\right)^2\right]}{2\mu^2(1-\rho H^1_{n-k,n})}$$

- The upper bound is valid only when $1 - \rho H^1_{n-k,n} > 0$.
  - $H^1_{n-k,n} = \sum_{j=n-k+1}^{n} 1/j$ is the generalized harmonic number.
- This stability condition is loose compared to $\lambda < n\mu/k$ that was derived for fork-join systems.

LATENCY LOWER BOUND

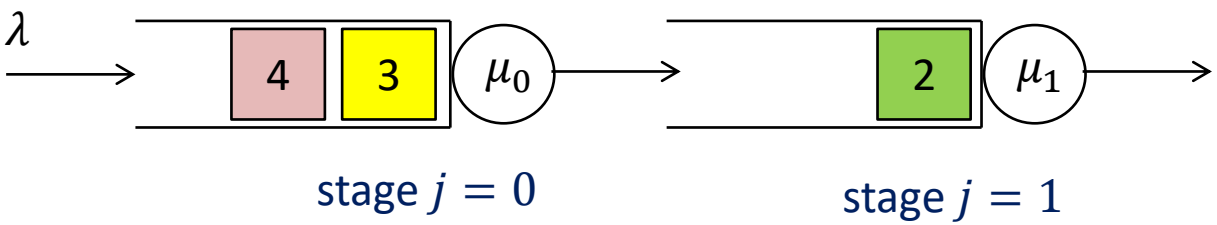
Fork-join:



- For an (n, k) fork-join system, each request goes through a sequence of k stages of processing.
- In the jth stage, where $0 \leq j \leq k-1$, j chunk tasks have been served, and the request will depart when $k - j$ more finish service.
- This results in a tandem-queue model!

LATENCY LOWER BOUND

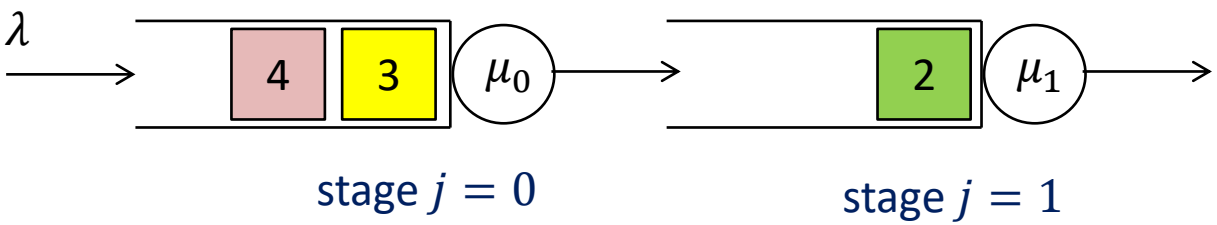
Tandem queue with $(n = 3, k = 2)$



- The service rate in the jth stage is at most $\mu_j = (n - j)\mu$.
  - In the jth stage, at most $(n - j)$ chunks are actively processed.
  - The service time is the minimum of $(n - j)$ exponential service times.
- The actual service rate is indeed lower because:
  - As a request moves from the jth to the $(j + 1)$th stage, there may be other tasks at the head of its respective queues.

LATENCY LOWER BOUND

Tandem queue with $(n = 3, k = 2)$



- Each stage is an $M/M/1$ queue with arrival rate λ (Ross, 2019) and service rate $\mu_j = (n - j)\mu$.
  - The time for a request to move from the jth to the $(j + 1)$th stage is bounded by $1/(\mu_j - \lambda)$.
- A lower bound is given by the sum of mean service time in each of the k stages: $T_{n,k} \geq \sum_{j=0}^{k-1} 1/(\mu_j - \lambda)$.

LATENCY LOWER BOUND

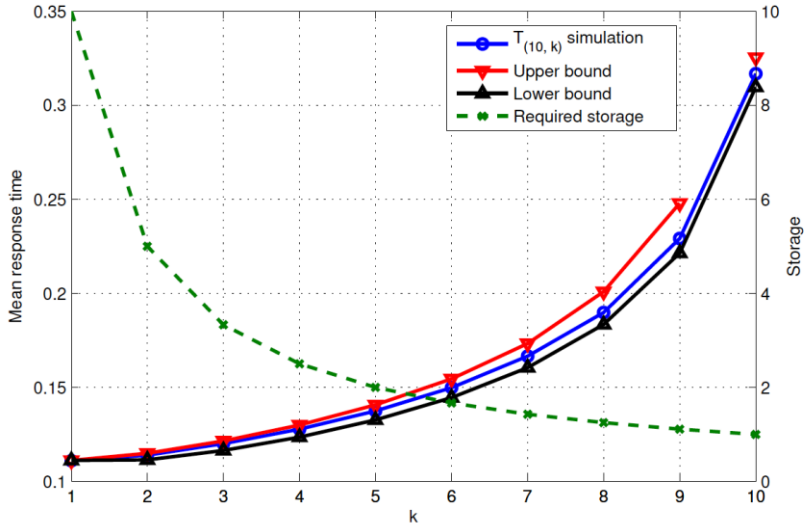
Theorem

The expected latency $T_{(n,k)}$ for an (n,k) fork-join system satisfies

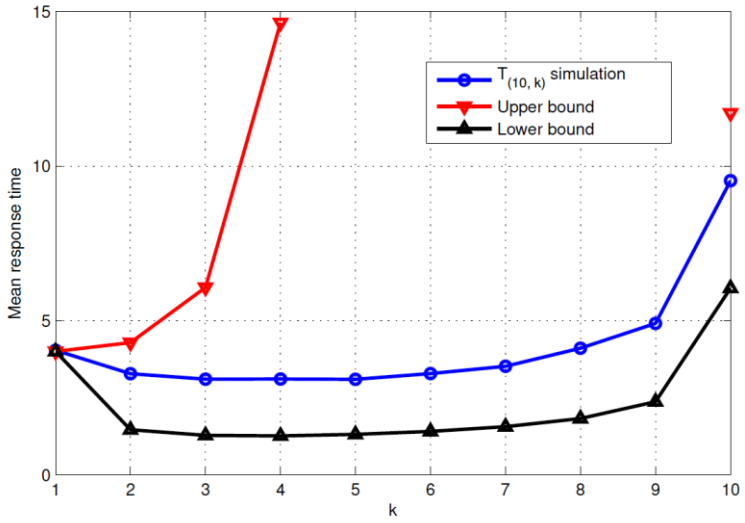
$$T_{(n,k)} \geq \sum_{j=0}^{k-1} \frac{1}{(n-j)\mu - \lambda}.$$

- The lower bound is valid only when $\lambda < (n-k+1)\mu$.
- This stability condition is loose compared to $\lambda < n\mu/k$ that was derived for fork-join systems.

NUMERICAL EXAMPLES



Arrival rate λ=1 and service rate μ=10.

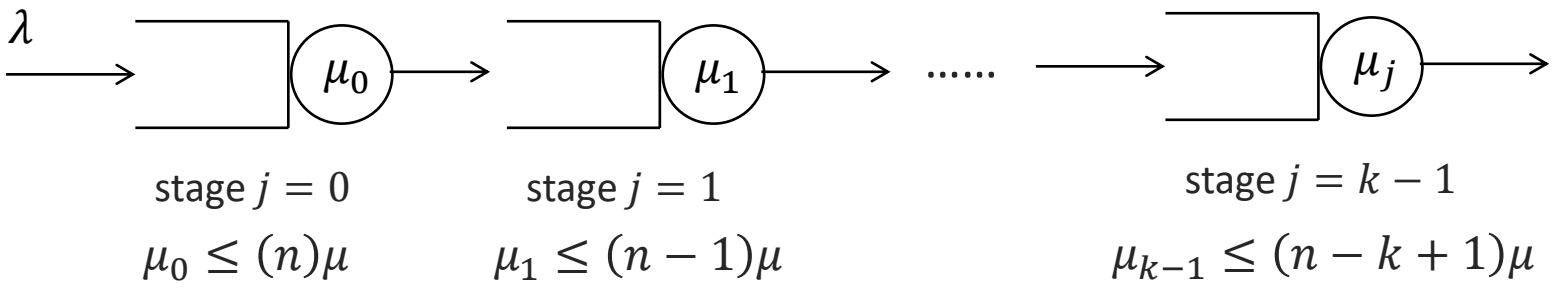


Arrival rate λ=1 and service rate μ=1.25.

[1] Joshi, Soljanin, Wornell (2015).

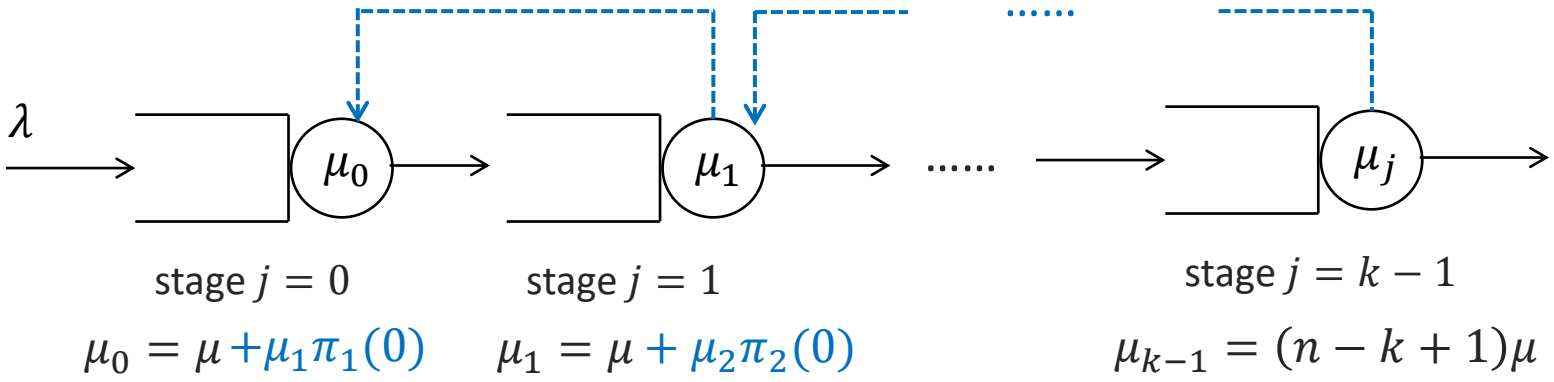
FINDING A BETTER APPROXIMATION

- Recall that the tandem-queue model provides a loose bound by setting the service rate in the jth stage with $\mu_j \leq (n-j)\mu$.
  - However, the maximum cannot be achieved due to only n servers available.



- An improved approximation can be obtained through a better estimate of service rate μ_j in the jth stage.

LATENCY ANALYSIS USING TANDEM QUEUES



- Consider a tandem queue with resource pooling:
  - Service rate is μ except for the last server.
  - Use free resource (with probability $\pi_j(0)$) at queue $j + 1$ to help with j.
- The service rate at each stage j can be solved backwards.

LATENCY ANALYSIS USING TANDEM QUEUES

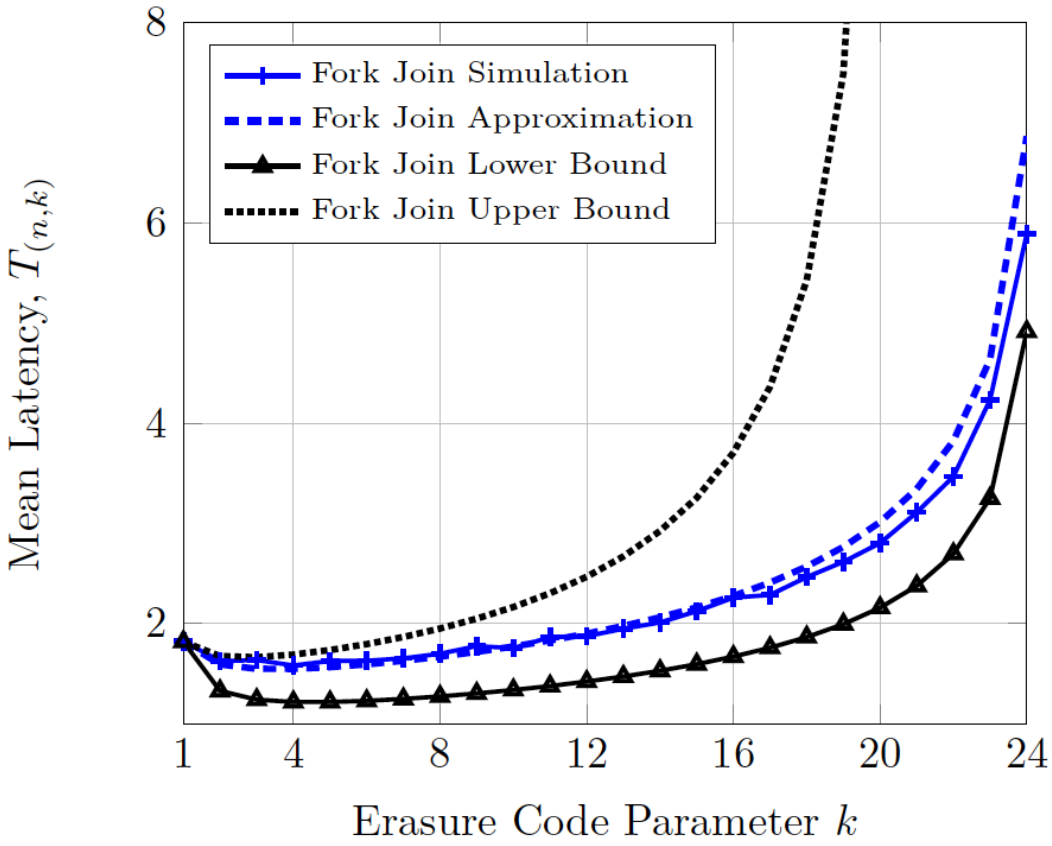
Theorem

The expected latency $T_{(n,k)}$ for an (n,k) fork-join system can be approximated by:

$$T_{(n,k)} \approx \sum_{j=0}^{k-1} \frac{1}{(n-j)\mu - (k-j)\lambda}.$$

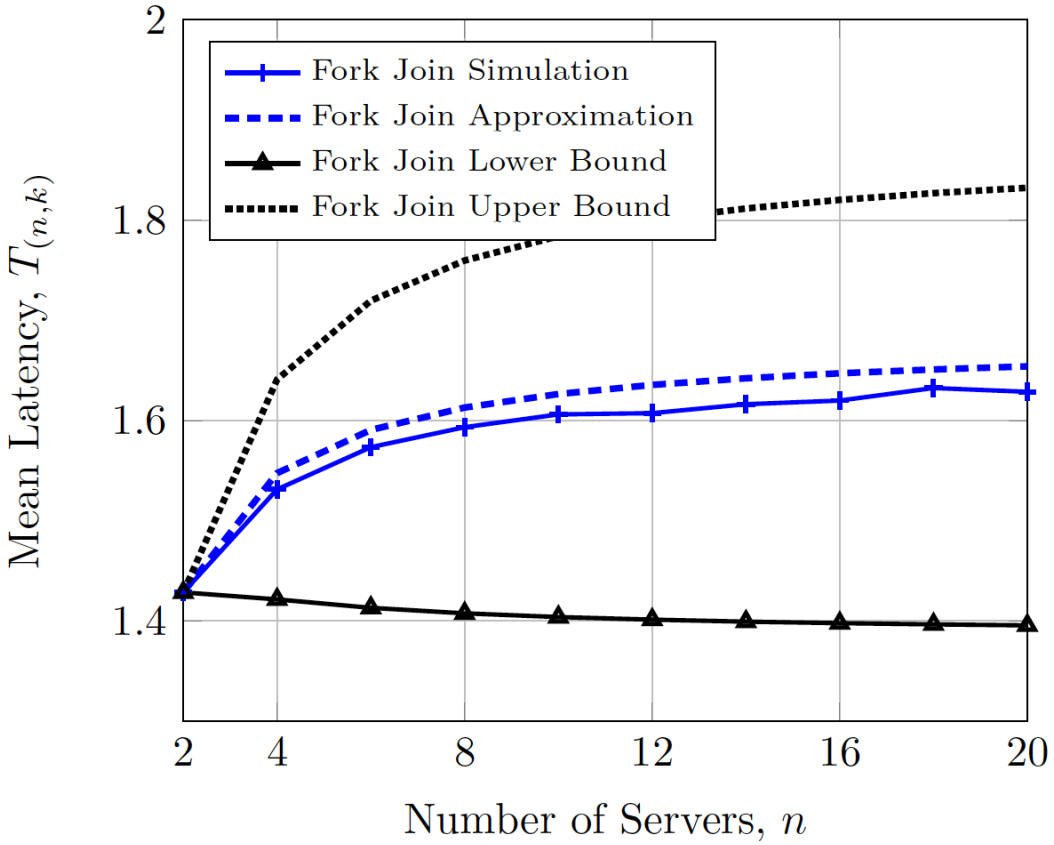
- The lower bound is valid only when $k\lambda < n\mu$.
- This stability condition is the same as that of fork-join systems.

NUMERICAL EXAMPLES



We choose $n = 24$, arrival rate $\lambda = 0.45$, and service rate $\mu = k/n$.

NUMERICAL EXAMPLES



We choose arrival rate $\lambda = 0.3$, and service rate $\mu = k/n = 0.5$.

GENERALIZATIONS

- i.i.d. and general service times:
  - *Joshi et al., 2014, Joshi et al., 2017.*
- Each file i encoded using an (n, k_i) code and has arrival rate λ_i :
  - *Kumar et al., 2017.*

SUMMARY

- Fork-join systems provide an analytical framework for the study of erasure-coded storage, e.g.,
  - minimizing file access latency.
  - optimizing coding strategy.
- Upper and lower bounds to analyze the latency of general codes.
- A tight closed-form approximation of average latency.
- Average latency is better for MDS codes for all code-rates.
- It works with homogeneous file placement and service rates.

OPEN PROBLEMS FOR FORK-JOIN SYSTEMS

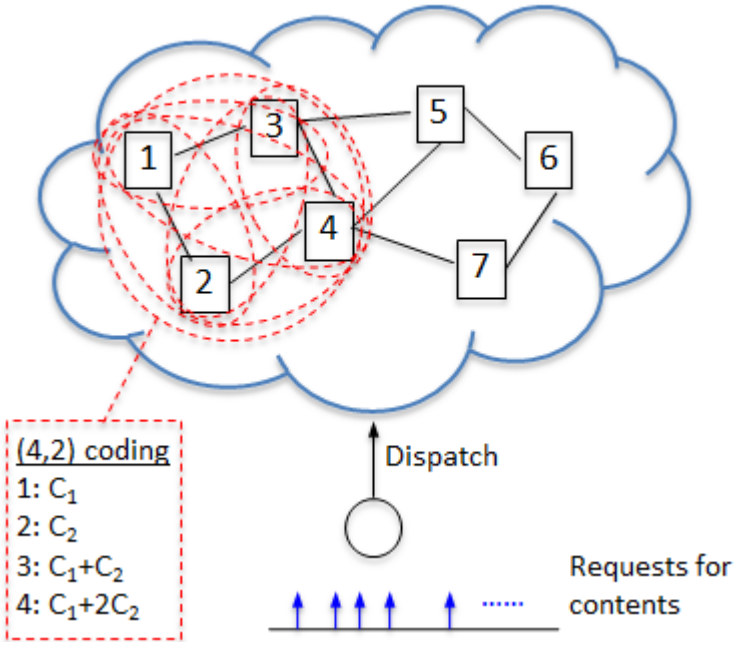
- Tight upper bound:
  - There is still a large gap between the upper bound and the optimal stability conditions even for exponential service times.
- General file placement:
  - When each file is placed on a subset of the servers, no latency result is available for this general setting.
- Heterogeneous servers:
  - Analyzing the latency for heterogeneous servers with different service time distribution is still an open problem.
- Approximations and guarantees:
  - In the asymptotic regime?

PART 3: PROBABILISTIC SCHEDULING

PROBABILISTIC SCHEDULING

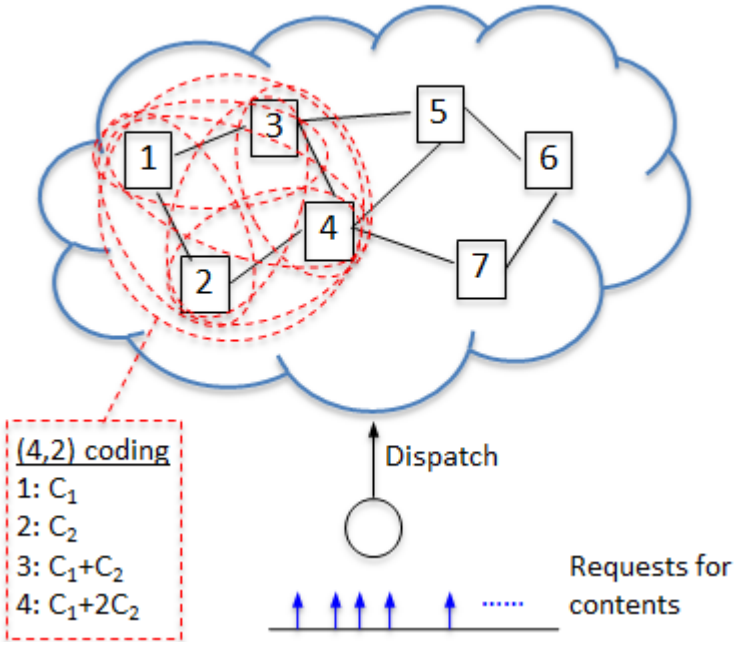
- Question: Which k subsets to choose

- Probabilistic scheduling: Choose all possible (n choose k) subsets with certain probabilities



PROBABILISTIC SCHEDULING

- Probabilistic scheduling: Choose all possible (n choose k) subsets with certain probabilities
- Since this is a scheme, it upper bounds the latency of the optimal scheme
- Number of probability terms to optimize: (n choose k) – hard problem
- Question: Can reduce terms?



PROBABILITY OVER SUBSETS -> PROBABILITY OVER NODES

Theorem [Xiang Lan Aggarwal Chen 2014]

Given node selection probabilities $\prod_{ij}$, there exists a scheme with feasible load balancing $P(A_i)$, where A_i are k-subsets, if and only if

$$\sum_{j=1}^{m} \pi_{i,j} = k_i \ \forall i$$

PROBABILITY OVER SUBSETS -> PROBABILITY OVER NODES

Theorem [Xiang Lan Aggarwal Chen 2014]

Given node selection probabilities $\prod_{ij}$, there exists a scheme with feasible load balancing $P(A_i)$, where A_i are k-subsets, if and only if

$$\sum_{j=1}^{m} \pi_{i,j} = k_i \ \forall i$$

- Necessity: Given the set probability, we can find node probability.
- This is because when set is chosen, all nodes are chosen.
- Thus, node probability is the sum of all set probabilities such that the node is part of the set.

PROBABILITY OVER SUBSETS -> PROBABILITY OVER NODES

Theorem [Xiang Lan Aggarwal Chen 2014]

Given node selection probabilities $\prod_{ij}$, there exists a scheme with feasible load balancing $P(A_i)$, where A_i are k-subsets, if and only if

$$\sum_{j=1}^{m} \pi_{i,j} = k_i \ \forall i$$

- Sufficiency: Given node probabilities, there exist set probabilities.
- We use Farkas-Minkowski Theorem to show that the linear equations from node probability to set probability has a valid solution for set probabilities given node probabilities.

PROBABILITY OVER SUBSETS -> PROBABILITY OVER NODES

Theorem [Xiang Lan Aggarwal Chen 2014]

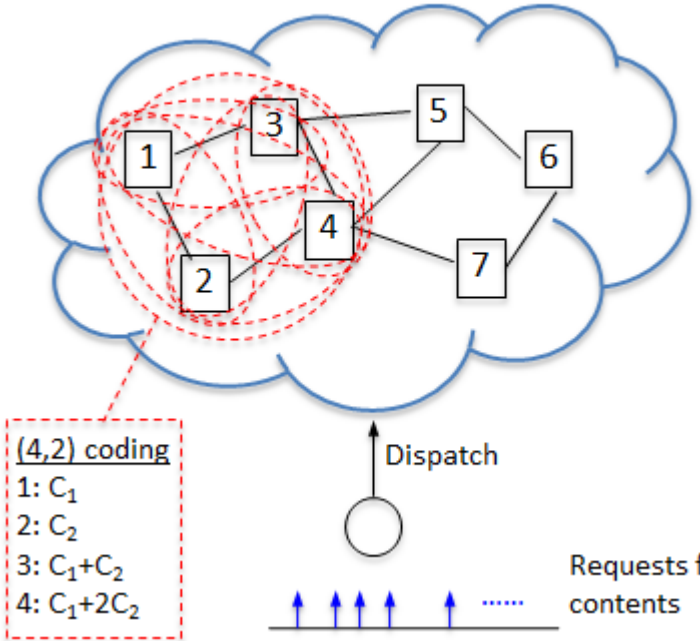
Given node selection probabilities $\prod_{ij}$, there exists a scheme with feasible load balancing $P(A_i)$, where A_i are k-subsets, if and only if

$$\sum_{j=1}^{m} \pi_{i,j} = k_i \ \forall i$$

- This result demonstrates that independent node selection is sufficient.

LATENCY BOUND WITH PROBABILISTIC SCHEDULING

- Probabilistic Scheduling: Choose all possible (n choose k) subsets with certain probabilities
- Probability over independent servers is equivalent
- Now, request at each server with certain probability and thus Poisson.
- Can characterize mean and variance of delay at each server as described next.



USE OF POLLACZEK-KHINCHIN THEOREM FOR M/G/1

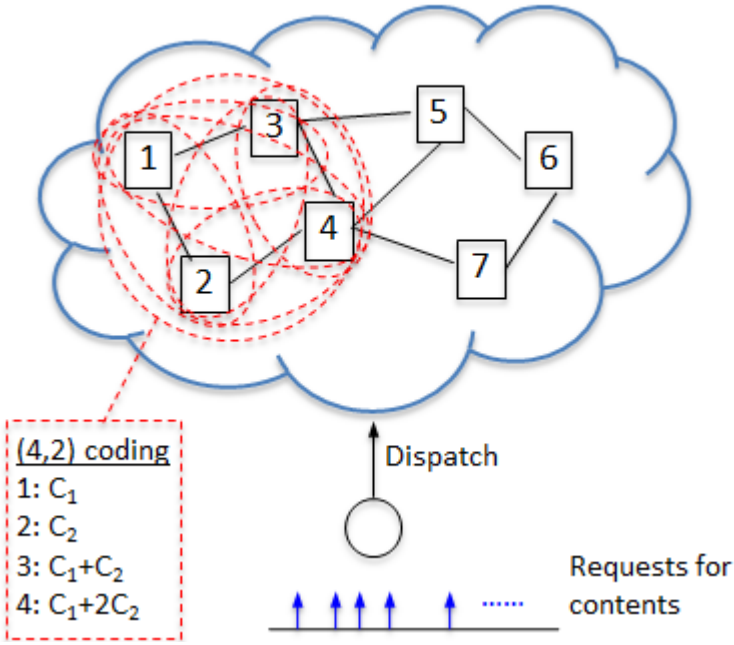
- Let $\mathbb{Z}_j(t_i)$ be the moment generating function of service at server j.
- Let Λ_j be the arrival rate at server j
- The moment generating of the time chunk of file i spends in the queue (including waiting in queue and service) is given as:

$$\mathbb{E}\left[e^{t_i \mathbf{W}_{i,j}}\right] = \frac{(1-\rho_j)\, t_i \mathbb{Z}_j(t_i)}{t_i - \Lambda_j\left(\mathbb{Z}_j(t_i) - 1\right)}$$

$$\rho_j = \Lambda_j \mathbb{E}\left[\mathbb{X}_j\right] = \Lambda_j \left[\frac{d}{dt}\mathbb{Z}_j(t_i)\,|_{t_i=0}\right]$$

LATENCY BOUND WITH PROBABILISTIC SCHEDULING

- Can characterize mean and variance of delay at each server (M/G/1) queue (Pollaczek-Khinchin Theorem)
- How about overall delay?



OVERALL FILE DELAY- ORDERED STATISTICS

- Overall delay is the maximum over the choice of servers which are selected using probabilistic scheduling

$$\mathbb{E}[\mathbf{Q}_i] \triangleq \mathbb{E}_{\mathbf{W}_{i,j}}\left[\mathbb{E}_{\mathcal{A}_i}\left[\max_{j\in\mathcal{A}_i}\mathbf{W}_{i,j}\right]\right]$$

- The moment generating function of the overall delay is given as:

$$\begin{aligned}\mathbb{E}\left[e^{t_i\mathbf{Q}_i}\right] &= \mathbb{E}_{\mathcal{A}_i,\mathbf{W}_{i,j}}\left[\max_{j\in\mathcal{A}_i}e^{t_i\mathbf{W}_{i,j}}\right]\\ &= \mathbb{E}_{\mathcal{A}_i}\left[\mathbb{E}_{\mathbf{W}_{i,j}}\left[\max_{j\in\mathcal{A}_i}e^{t_i\mathbf{W}_{i,j}}\,|\mathcal{A}_i\right]\right]\end{aligned}$$

- Bounding max by sum would only give a logarithmic gap in latency (due to the use of moment generating functions).

$$\mathbb{E}_{\mathcal{A}_i}\left[\sum_{j\in\mathcal{A}_i}\mathbb{E}_{\mathbf{W}_{i,j}}\left[e^{t_i\mathbf{W}_{i,j}}\right]\right]$$

OVERALL FILE DELAY- ORDERED STATISTICS

- Overall delay can be bounded as:

$$\begin{aligned}
\mathbb{E}\left[e^{t_i \mathbf{Q}_i}\right] &\leq \mathbb{E}_{\mathcal{A}_i}\left[\sum_{j\in\mathcal{A}_i}\mathbb{E}_{\mathbf{W}_{i,j}}\left[e^{t_i \mathbf{W}_{i,j}}\right]\right] \\
&= \mathbb{E}_{\mathcal{A}_i}\left[\sum_{j}\mathbb{E}_{\mathbf{W}_{i,j}}\left[e^{t_i \mathbf{W}_{i,j}}\right]\mathbf{1}_{(j\in\mathcal{A}_i)}\right] \\
&= \sum_{j}\mathbb{E}_{\mathbf{W}_{i,j}}\left[e^{t_i \mathbf{W}_{i,j}}\right]\mathbb{E}_{\mathcal{A}_i}\left[\mathbf{1}_{(j\in\mathcal{A}_i)}\right] \\
&= \sum_{j}\mathbb{E}_{\mathbf{W}_{i,j}}\left[e^{t_i \mathbf{W}_{i,j}}\right]\mathbb{P}(j\in\mathcal{A}_i) \\
&= \sum_{j}\pi_{i,j}\mathbb{E}_{\mathbf{W}_{i,j}}\left[e^{t_i \mathbf{W}_{i,j}}\right]
\end{aligned}$$

- Further, Jensen's inequality gives bound on the average latency

$$e^{t_i \mathbb{E}[\mathbf{Q}_i]} \leq \mathbb{E}\left[e^{t_i \mathbf{Q}_i}\right]$$

OVERALL FILE DELAY

- Bounding max by sum in moment generating function would only give a logarithmic gap in latency.

- This result allows multiple contents, state of the art has single file. Even for single file, our bound is better for general distribution.

Theorem

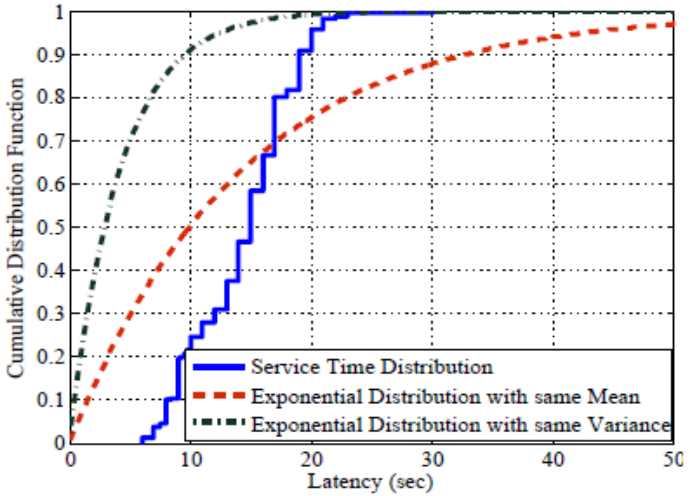
Given mean and variance of delay at each server, the expected latency of content i is upper bounded as follows

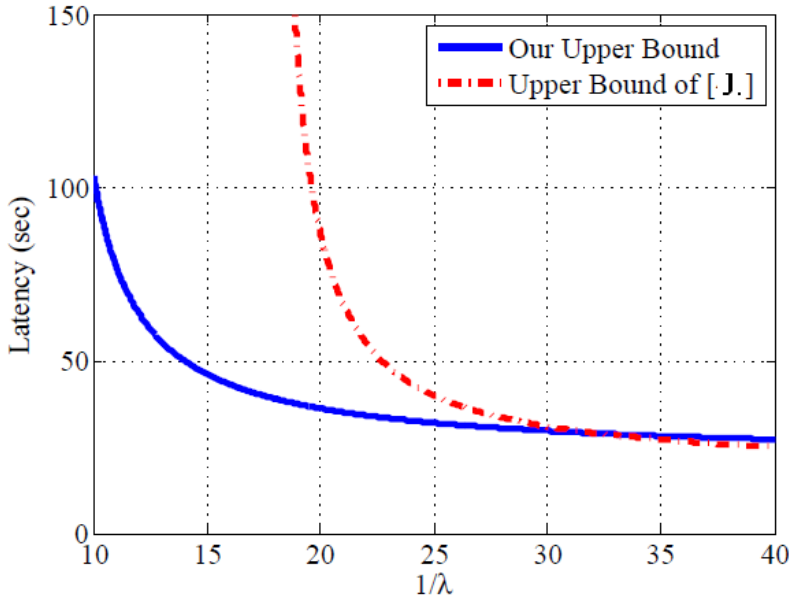
$$\mathbb{E}[\mathbf{Q}_i] \leq \frac{1}{t_i} \log \left(\sum_{j=1}^{m} \pi_{i,j} \frac{(1-\rho_j) t_i \mathbb{Z}_j(t_i)}{t_i - \Lambda_j \left(\mathbb{Z}_j(t_i) - 1\right)} \right)$$

for any $t_i > 0$, $\rho_j = \Lambda_j \left[\frac{d}{dt} \mathbb{Z}_j(t_i) \mid_{t_i=0} \right]$, $\rho_j < 1$, and $\Lambda_j \left(\mathbb{Z}_j(t_i) - 1\right) < t_i$.

ADVANTAGES OF THE PROPOSED BOUND

- First approach for multiple files
- Works for any service distribution
- Improvement even for single file [Shah et al, 2012, Joshi et al, 2013]
- Further improvement: Rather thar k out of n, can do d out of n – less content from each helps latency, but d^{th} best hurts





TAIL LATENCY

- Users are impatient.
- Increase in delay of web traffic leads to loss of customers, significantly affecting revenues.

	Distinct Queries/User	Query Refinement	Revenue/User	Any Clicks	Satisfaction	Time to Click (increase in ms)
50ms	-	-	-	-	-	-
200ms	-	-	-	-0.3%	-0.4%	500
500ms	-	-0.6%	-1.2%	-1.0%	-0.9%	1200
1000ms	-0.7%	-0.9%	-2.8%	-1.9%	-1.6%	1900
2000ms	-1.8%	-2.1%	-4.3%	-4.4%	-3.8%	3100

- Long tail of latency is of particular concern, with 99.9th percentile response times that are orders of magnitude worse than the mean
- We can use probabilistic scheduling to characterize the tail latency of a file too. [Infocom 2017, TNSM 2019]

TAIL LATENCY CALCULATIONS

- Tail Latency of a file from a server is given as

$$\Pr(\mathbf{W}_{i,j} \geq \sigma) \leq \frac{\mathbb{E}[e^{t_{i,j}\mathbf{W}_{i,j}}]}{e^{t_{i,j}\sigma}}$$
$$= \frac{1}{e^{t_{i,j}\sigma}} \frac{(1-\rho_j)\, t_{i,j} \mathbb{Z}_j(t_{i,j})}{t_{i,j} - \Lambda_j \left(\mathbb{Z}_j(t_{i,j}) - 1\right)}$$

- Overall tail latency can be computed using ordered statistics

TAIL LATENCY CALCULATIONS

- Overall tail latency can be bounded as:

$$
\begin{aligned}
\Pr\left(\mathbf{Q}_i \geq \sigma\right) &= \Pr\left(\max_{j \in \mathcal{A}_i} \mathbf{W}_{i,j} \geq \sigma\right) \\
&= \mathbb{E}_{\mathcal{A}_i, \mathbf{W}_{i,j}}\left[\max_{j \in \mathcal{A}_i} \mathbf{1}_{(\mathbf{W}_{i,j} \geq \sigma)}\right] \\
&\leq \mathbb{E}_{\mathcal{A}_i, \mathbf{W}_{i,j}}\left[\sum_{j \in \mathcal{A}_i}\left[1_{(\mathbf{W}_{i,j} \geq \sigma)}\right]\right] \\
&= \mathbb{E}_{\mathcal{A}_i}\left[\sum_{j \in \mathcal{A}_i}\left[\Pr(\mathbf{W}_{i,j} \geq \sigma)\right]\right] \\
&= \sum_{j} \pi_{i,j}\left[\Pr(\mathbf{W}_{i,j} \geq \sigma)\right]
\end{aligned}
$$

PURDUE UNIVERSITY

TAIL LATENCY CALCULATIONS

- Using tail latency of the individual W, overall tail latency can be bounded as:

$$\Pr\left(\mathbf{Q}_i \geq \sigma\right) \leq \sum_j \frac{\pi_{i,j}}{e^{t_{i,j}\sigma}} \frac{(1-\rho_j)t_{i,j}\mathbb{Z}_j(t_{i,j})}{t_{i,j} - \Lambda_j(\mathbb{Z}_j(t_{i,j}) - 1)}$$

for any $t_{i,j} > 0$, $\rho_j = \Lambda_j \left[\frac{d}{dt}\mathbb{Z}_j(t_{i,j}) \Big|_{t_{i,j}=0}\right]$, $\rho_j < 1$, and $\Lambda_j(\mathbb{Z}_j(t_{i,j})-1) < t_{i,j}$.

TAIL LATENCY INDEX

- File-sizes are heavy tailed [**Aggarwal** et al., ICC3, 2013].

- Cdf of chunk size is given as Pareto Distribution with index α

$$\Pr(L_i > x) = \begin{cases} (x_m/x)^\alpha & x \geq x_m \\ 0 & x < x_m \end{cases}$$

- What is tail index of Latency?

TAIL LATENCY INDEX

- File-sizes are heavy tailed [**Aggarwal** et al., ICC3, 2013].
- Cdf of chunk size is given as Pareto Distribution with index α

$$\Pr(L_i > x) = \begin{cases} (x_m/x)^\alpha & x \geq x_m \\ 0 & x < x_m \end{cases}$$

- What is tail index of Latency?
- Ans: ceil(α-1)
- Probabilistic scheduling is optimal for tail index.

SUMMARY

- Probabilistic Scheduling is proposed
- It allows for efficient bounds on mean and tail latency
- Probabilistic scheduling is optimal for tail index.

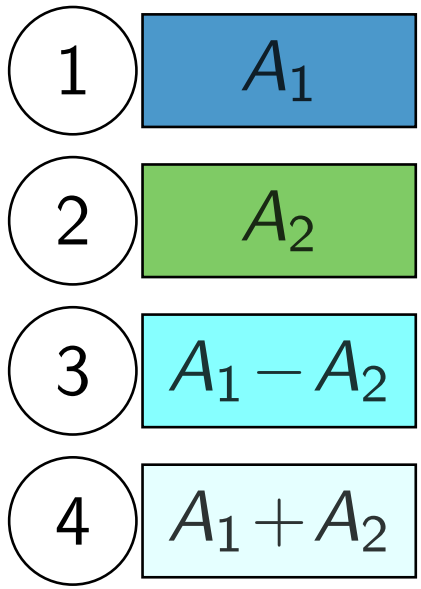
- Xiang, Lan, Aggarwal, and Chen, "Joint Latency and Cost Optimization for Erasure-coded Data Center Storage," IEEE/ACM Transactions on Networking, vol. 24, no. 4, pp. 2443-2457, Aug. 2016 (previous version in Sigmetrics Performance Evaluation Review 2014).
- Aggarwal, Fan, and Lan, "Taming Tail Latency for Erasure-coded, Distributed Storage Systems," in Proc. IEEE Infocom, May 2017
- Alabbasi, Aggarwal, and Lan, "TTLoC: Taming Tail Latency for Erasure-coded Cloud Storage Systems," IEEE Transactions on Network and Service Management, vol. 16, no. 4, pp. 1609-1623, Dec. 2019.
- Alabbasi and Aggarwal, "TTLCache: Taming Latency in Erasure-Coded Storage Through TTL Caching," IEEE Transactions on Network and Service Management, vol. 17, no. 3, pp. 1582-1596, Sept. 2020

OPEN PROBLEMS FOR PROBABILISTIC SCHEDULING SYSTEMS

- Sub-packetization:
  - Sub-packetization can be used to access data from more servers with a smaller part accessed from each server. For same size content from each server, it is simple corollary, how about scheduling approach to determine size of content from each server?
- Approximations and guarantees in Asymptotic Regime:
  - Is it possible to extend the approximation technique to heterogenous files and general service time distributions in the asymptotic regime? Does asymptotic independence hold?
- General data center topology:
  - Data center may have hierarchical storage (fog/edge storage). Further, some locality properties and multiple chunks on same server may be there in placement. Impact of such placement and its impacts is open.

PART 4: DELAYED-RELAUNCH SCHEDULING

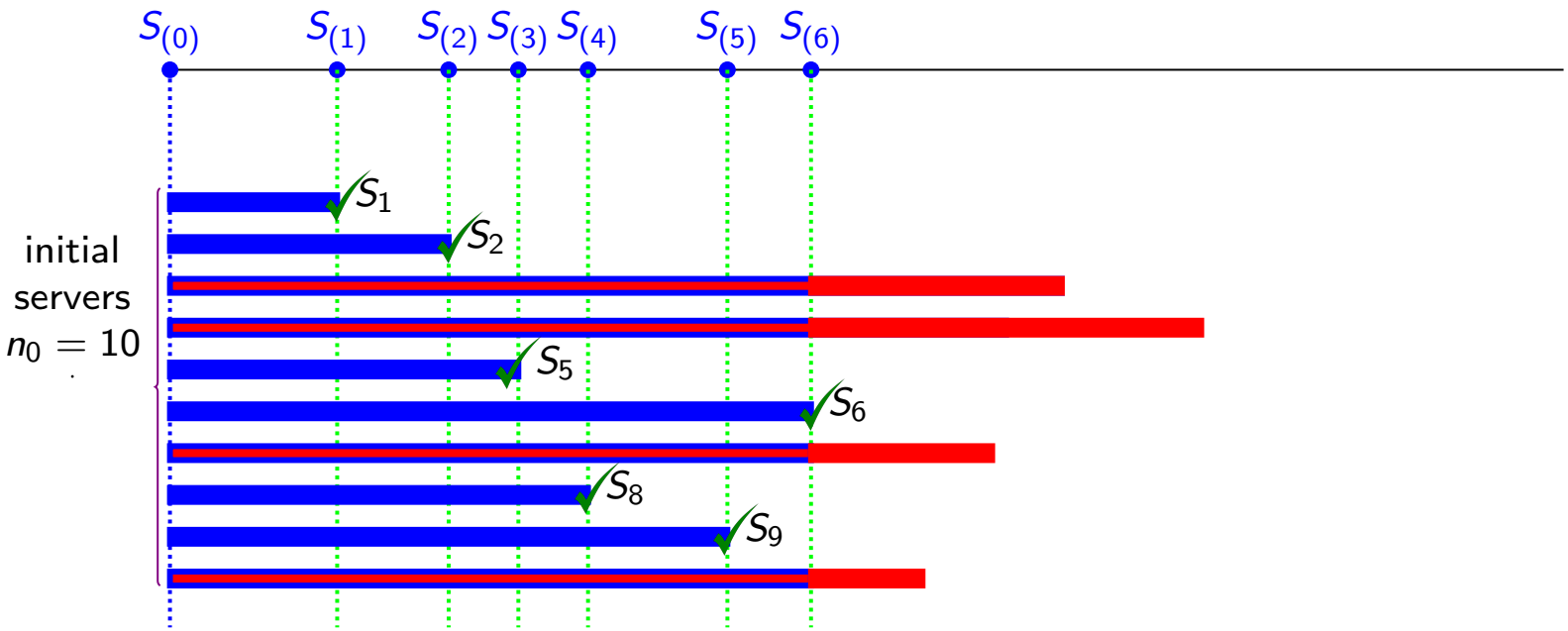
Coded access model



Latency energy tradeoff

- Parallelization leads to download speedup
- Redundancy leads to increased energy consumption

Coded access model

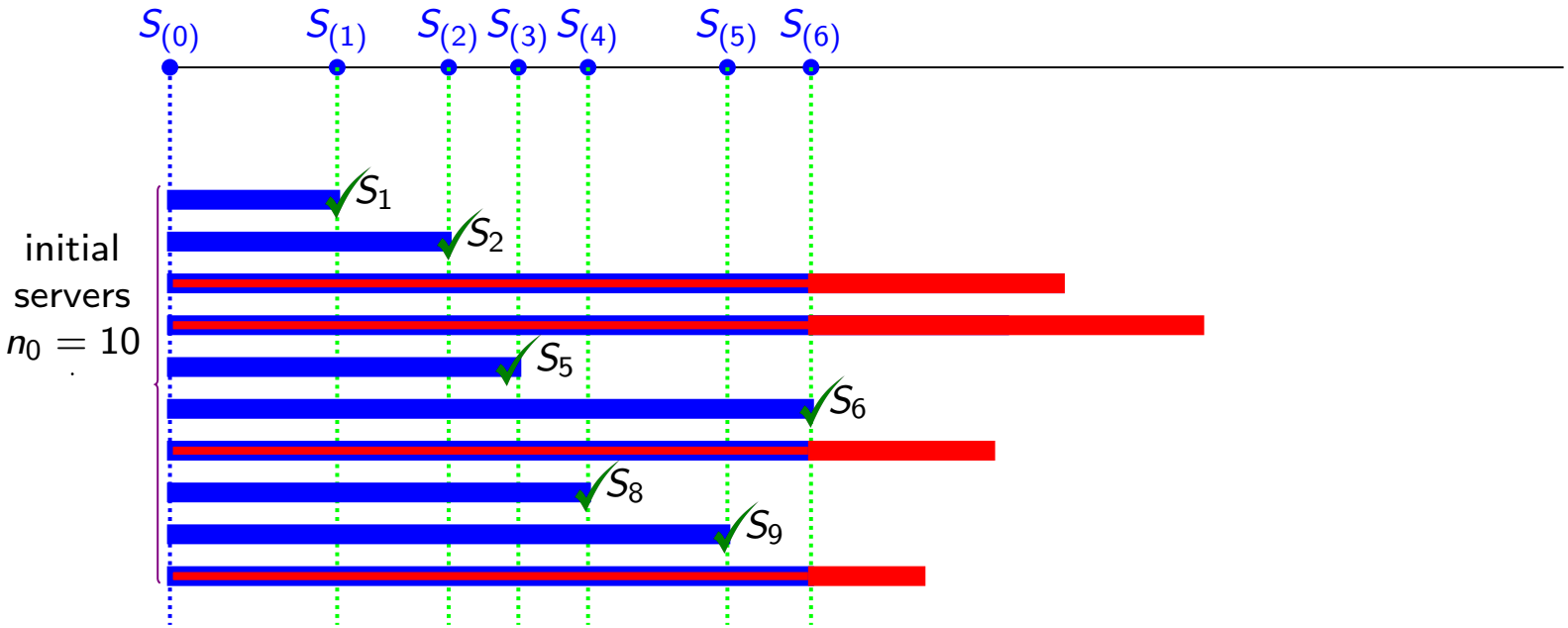


Performance metrics for (n, k) coded system

- Completion time: kth order statistic $S_{(k)}$ of download times
- Server utilization time: $\sum_{\ell=0}^{k-1}(n-\ell)(S_{(\ell+1)} - S_{(\ell)})$

Coded access model

c-shifted unit-rate exponential download times

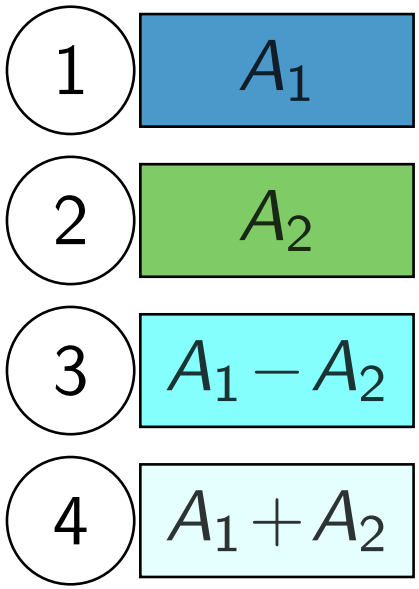


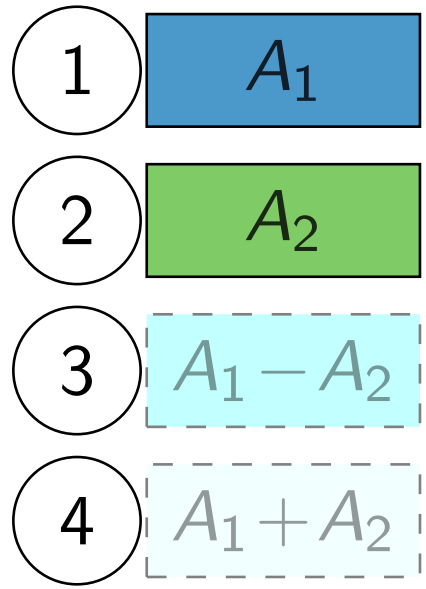
Download times $(S_1, \ldots, S_n)$ *i.i.d.* $(c, 1)$ shifted exponential

- Minimum $S_{(1)}$ is c-shifted n-rate exponential
- The difference $S_{(\ell+1)} - S_{(\ell)}$ is $(n-\ell)$-rate exponential
- Mean completion time: $c + \sum_{\ell=0}^{k-1} \frac{1}{n-\ell}$
- Mean server utilization cost: $nc + k$

To code or not code?

Shifted exponential download times





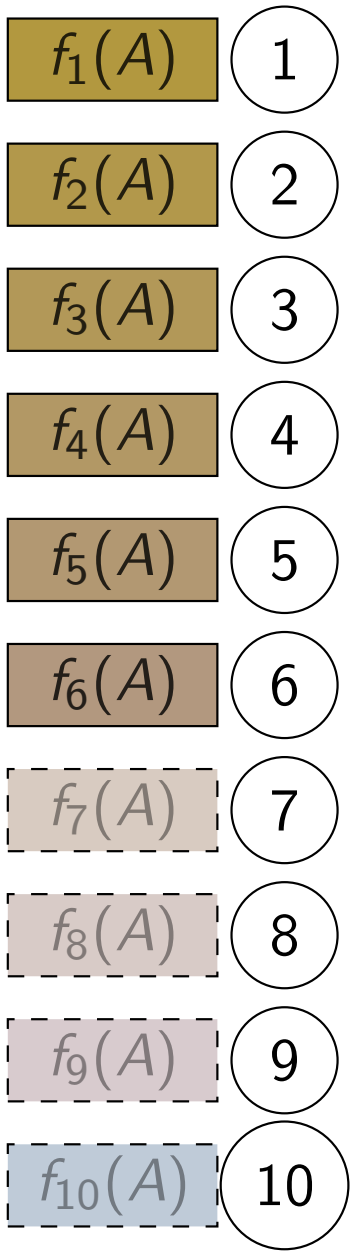
(n, k) coded system

- Completion time: $c + \sum_{\ell=0}^{k-1} \frac{1}{n-\ell}$
- Server utilization time: $nc + k$

(k, k) uncoded system

- Completion time: $c + \sum_{\ell=0}^{k-1} \frac{1}{k-\ell}$
- Server utilization time: $kc + k$

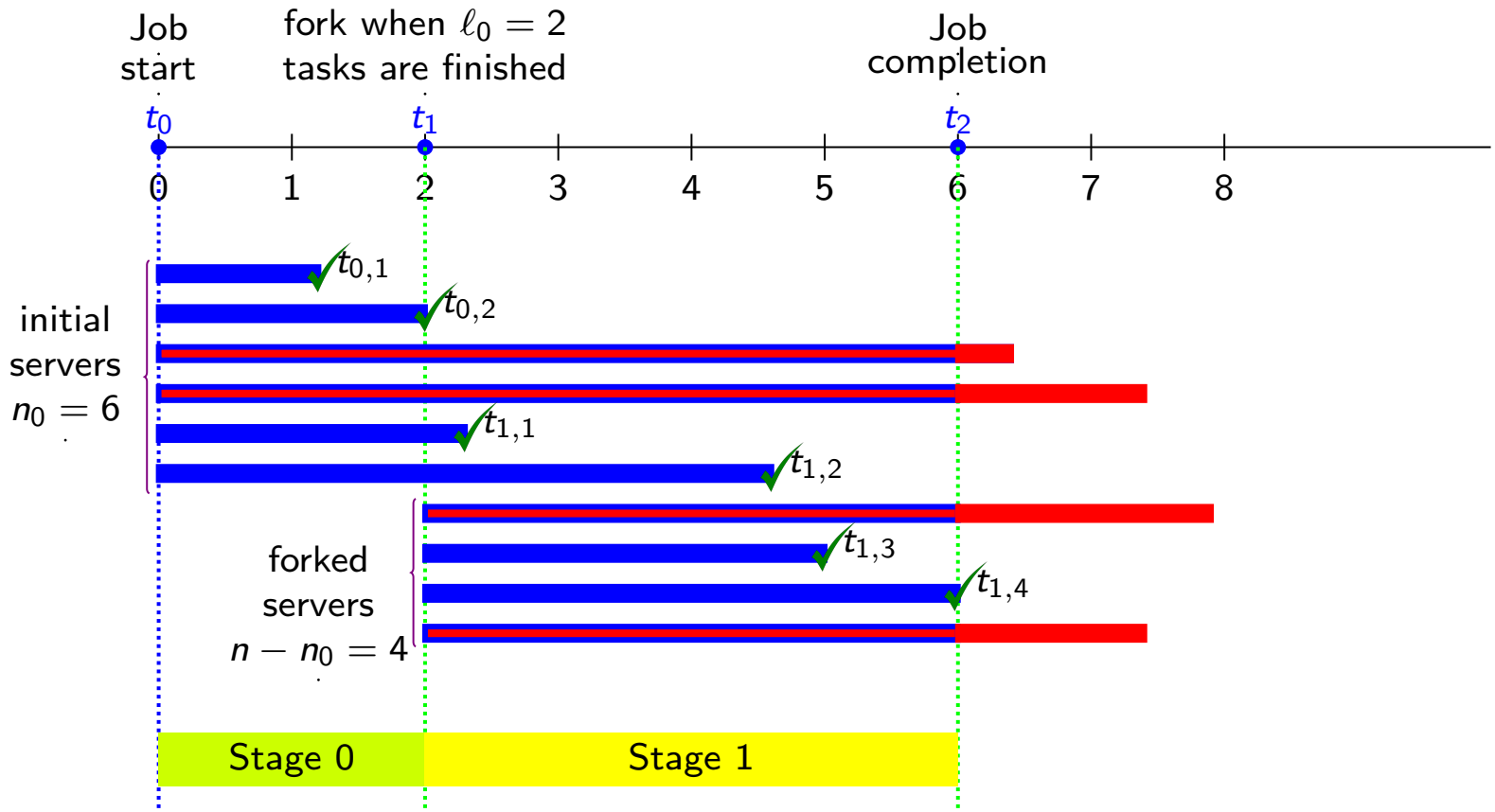
Forking additional servers



Delayed start of requests in multiple stages

- Stage i starts with download from additional n_i servers
- Stage i ends when downloaded from ℓ_i servers
- Design variables are (n_i, ℓ_i) for each stage i

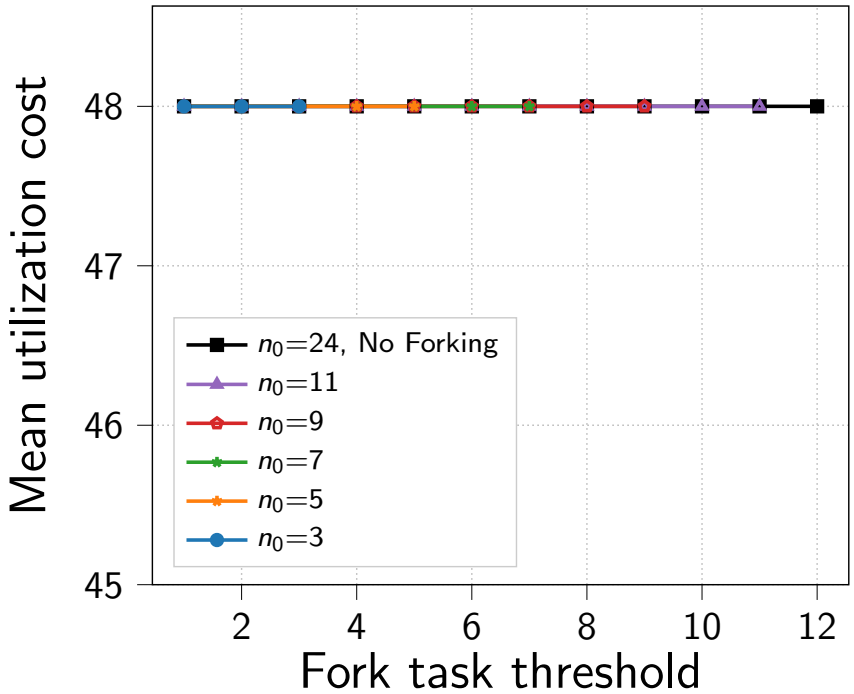
Performance Metric Computation



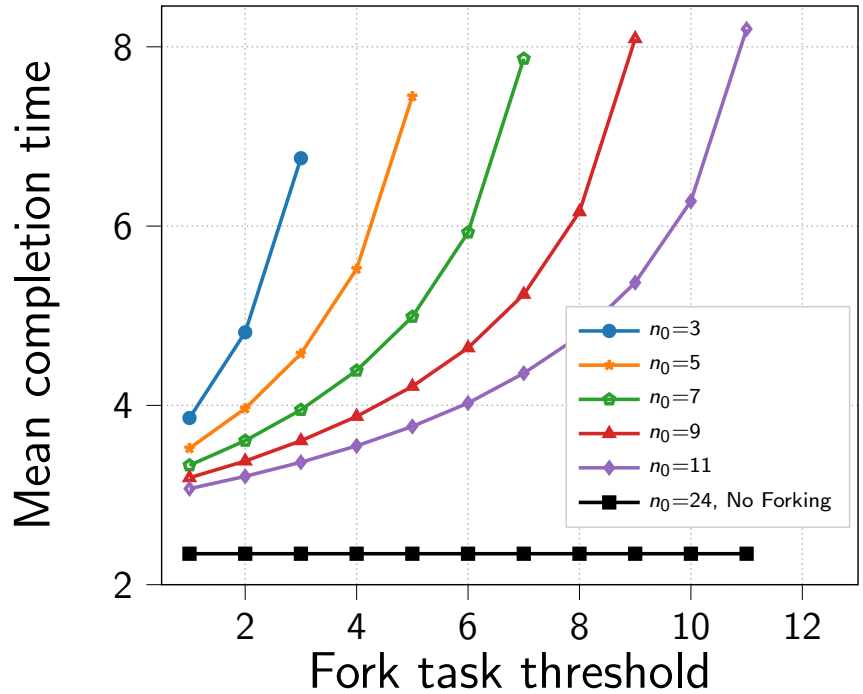
Server utilization

- Stage 0: $\sum_{j=0}^{\ell_0-1}(n_0 - j)(t_{i,j+1} - t_{i,j})$
- Stage 1: $\sum_{j=0}^{\ell_1-1}(n - \ell_0 - j)(t_{i,j+1} - t_{i,j})$

Initial servers n_0 smaller than sub-tasks k



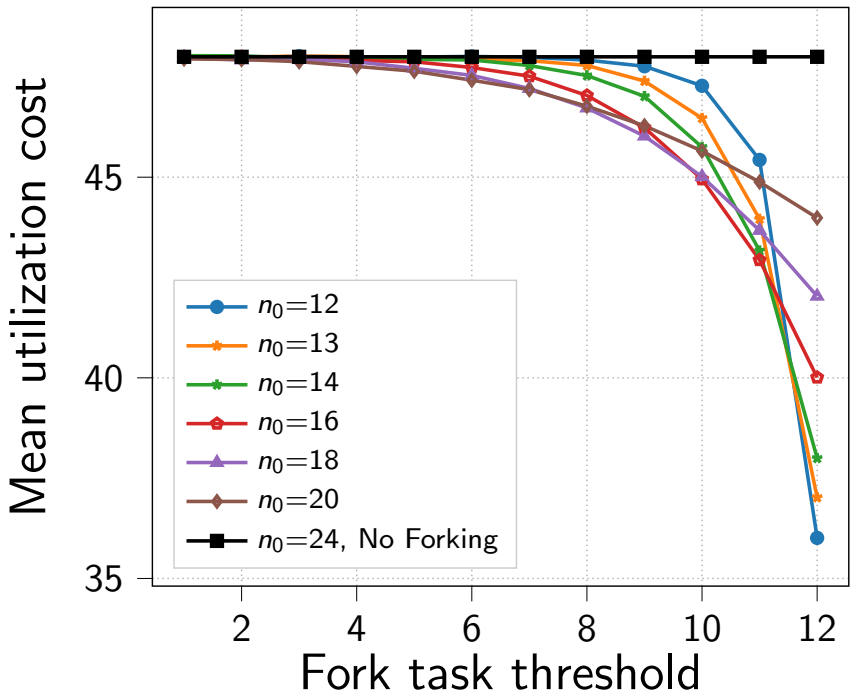
Constant for any choice



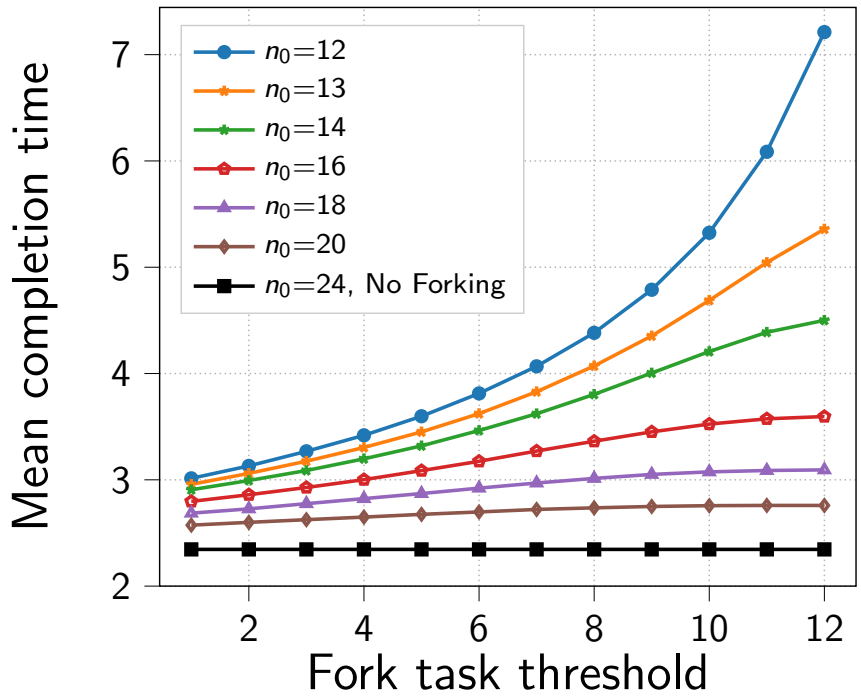
Increasing in fork-task threshold

- Mean utilization is identical to that of no-forking case

Initial servers n_0 greater than or equal to sub-tasks k

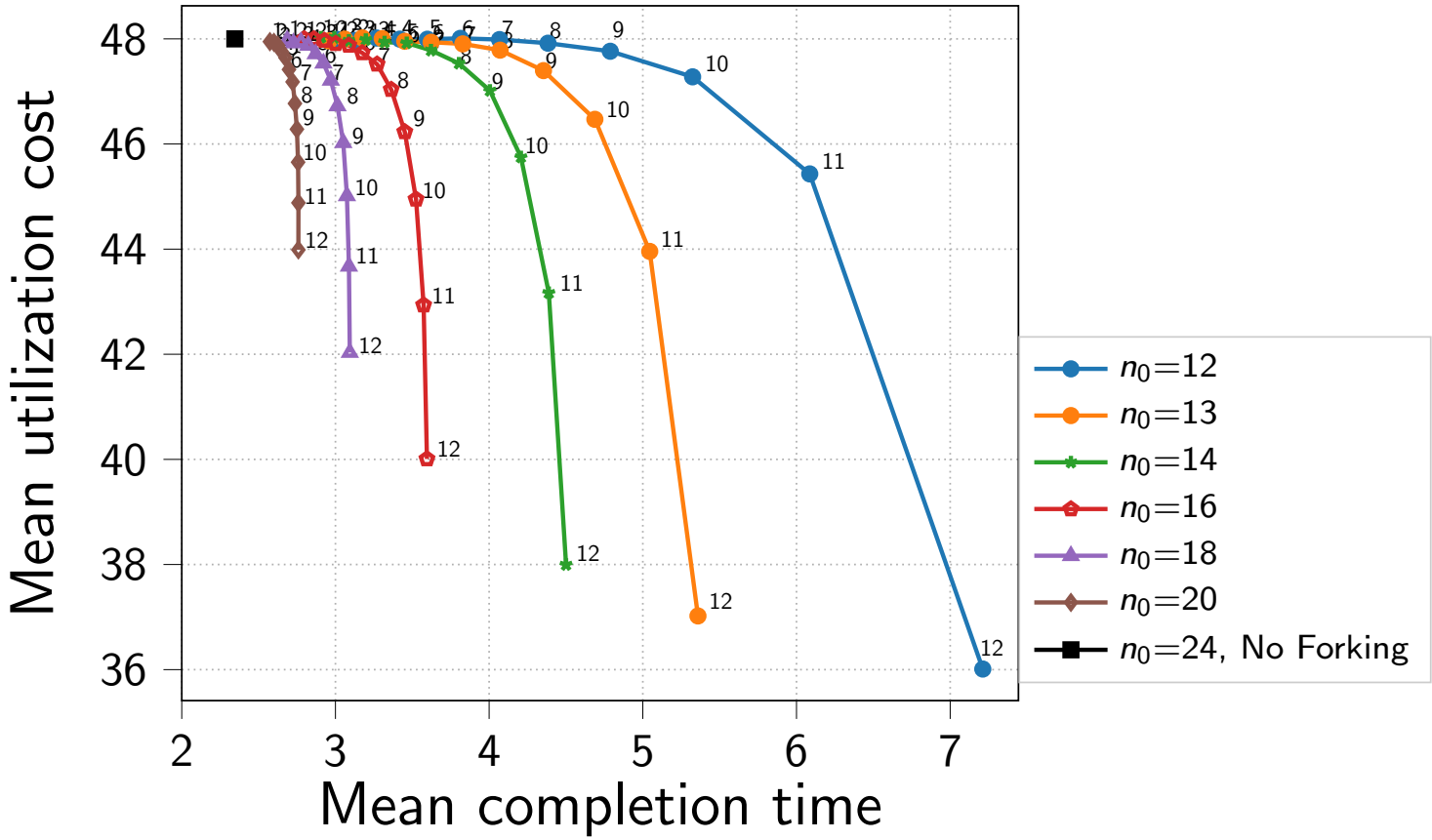


Decreasing in fork-task threshold



Increasing in fork-task threshold

Tradeoff when $n_0 \geqslant k$

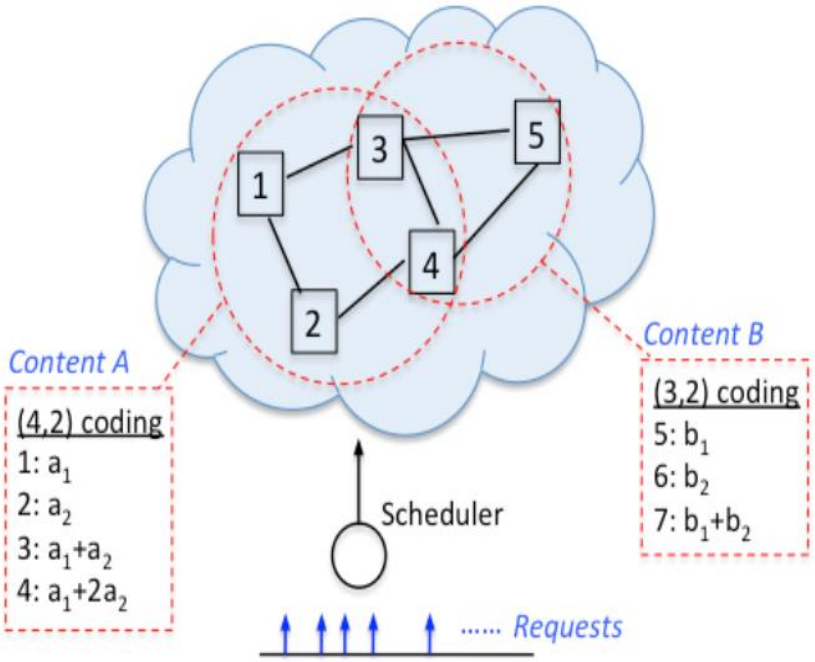


- Choice of initial servers matters
- Fork-task threshold gives a true tradeoff when $n_0 \geqslant k$
- Performance improvement!!

PART 5: EVALUATIONS AND OTHER APPLICATIONS

REQUIREMENTS FOR A DISTRIBUTED STORAGE SYSTEM

- Where to place content?
- What code parameters to choose?
- Which disks to choose for access when the content is requested?



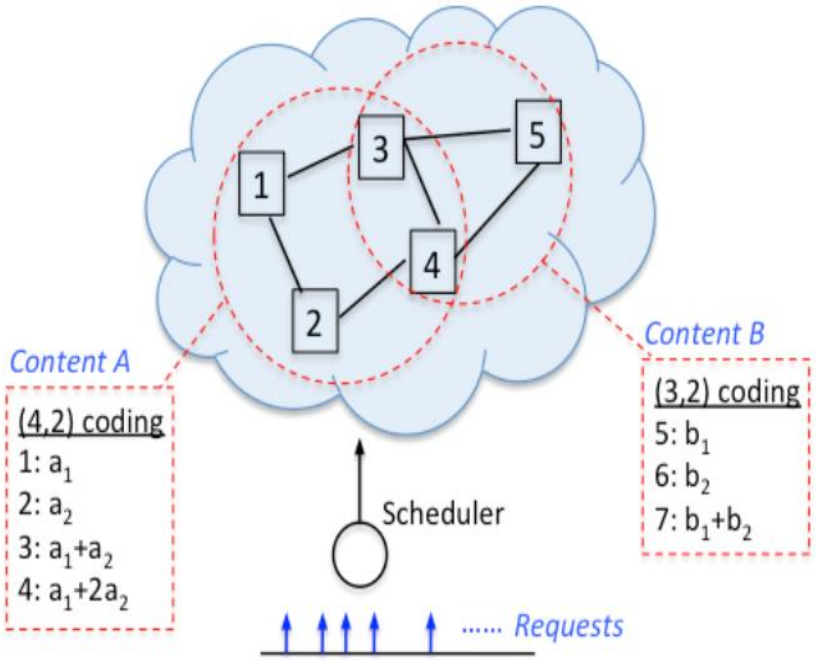
- Baseline:
  - where to place contents: Random
  - what code to use: Fixed
  - from where should content be served: Lowest queue servers

IISc GW UNIVERSITY WASHINGTON DC PURDUE UNIVERSITY

LATENCY AND COST

- Where to place content?
- What code parameters to choose?
- Which disks to choose for access when the content is requested?

- Optimization Variables:
  - Code Parameters
  - Content Placement Servers
  - Access Probabilities from different servers (Latency bound as described before)



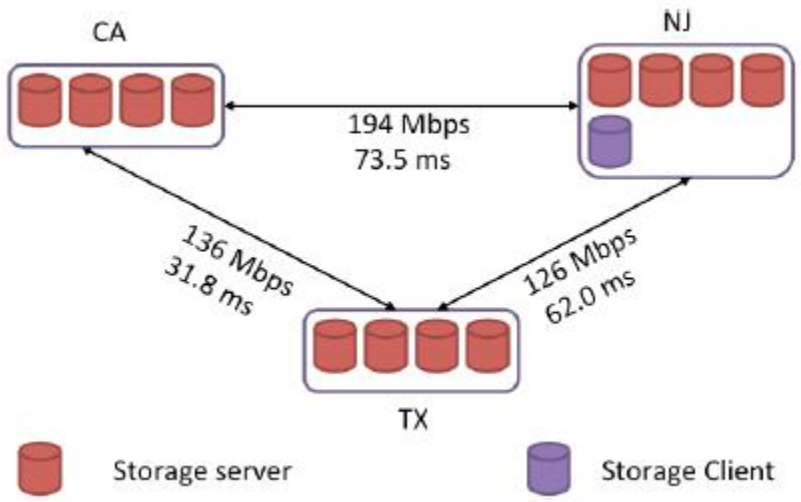
- Latency
  - Connection delay
  - Queuing delay

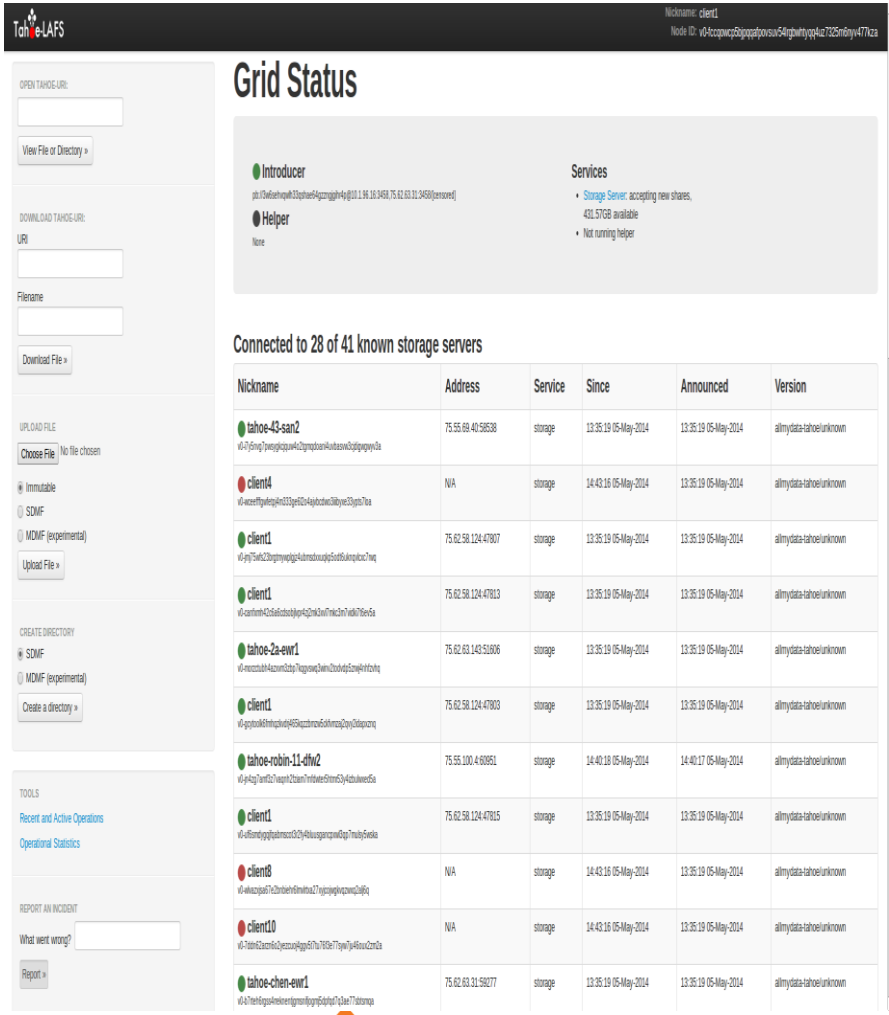
- Cost
  - Storage Cost

VALIDATION ON OPEN SOURCE STORAGE SYSTEM

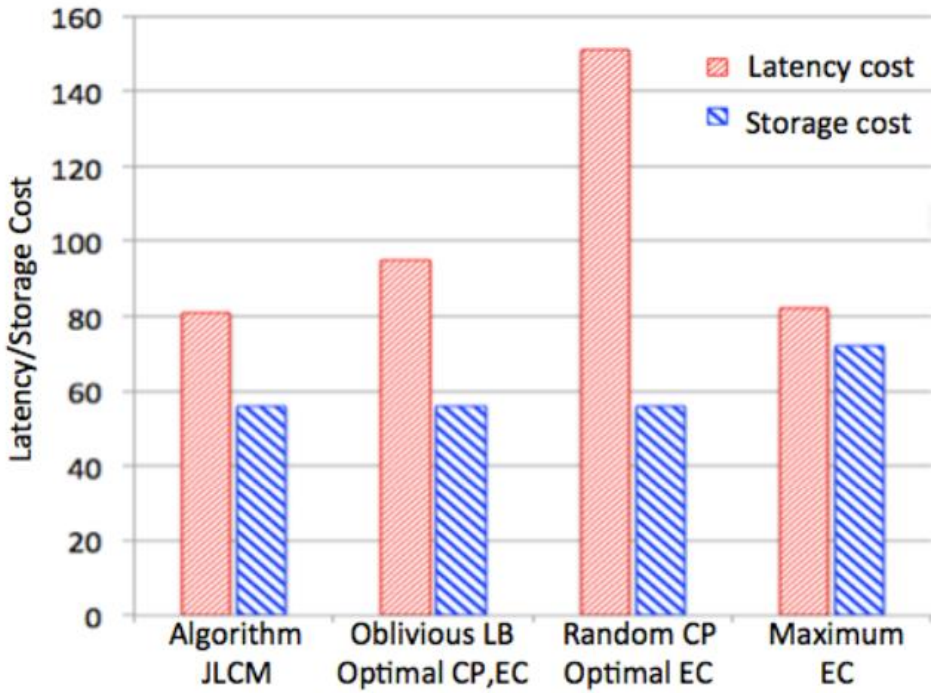
Tahoe-LAFS (**Tahoe Least-Authority Filesystem**) is an open source, secure, decentralized, fault-tolerant, peer-to-peer distributed data store and distributed file system.[4][5] It can be used as an online backup system, or to serve as a file or web host similar to Freenet,[6] depending on the front-end used to insert and access files in the Tahoe system. Tahoe can also be used in a RAID-like manner to use multiple disks to make a single large RAIN pool of reliable data storage.

SETUP OF STORAGE SERVERS FOR VALIDATION





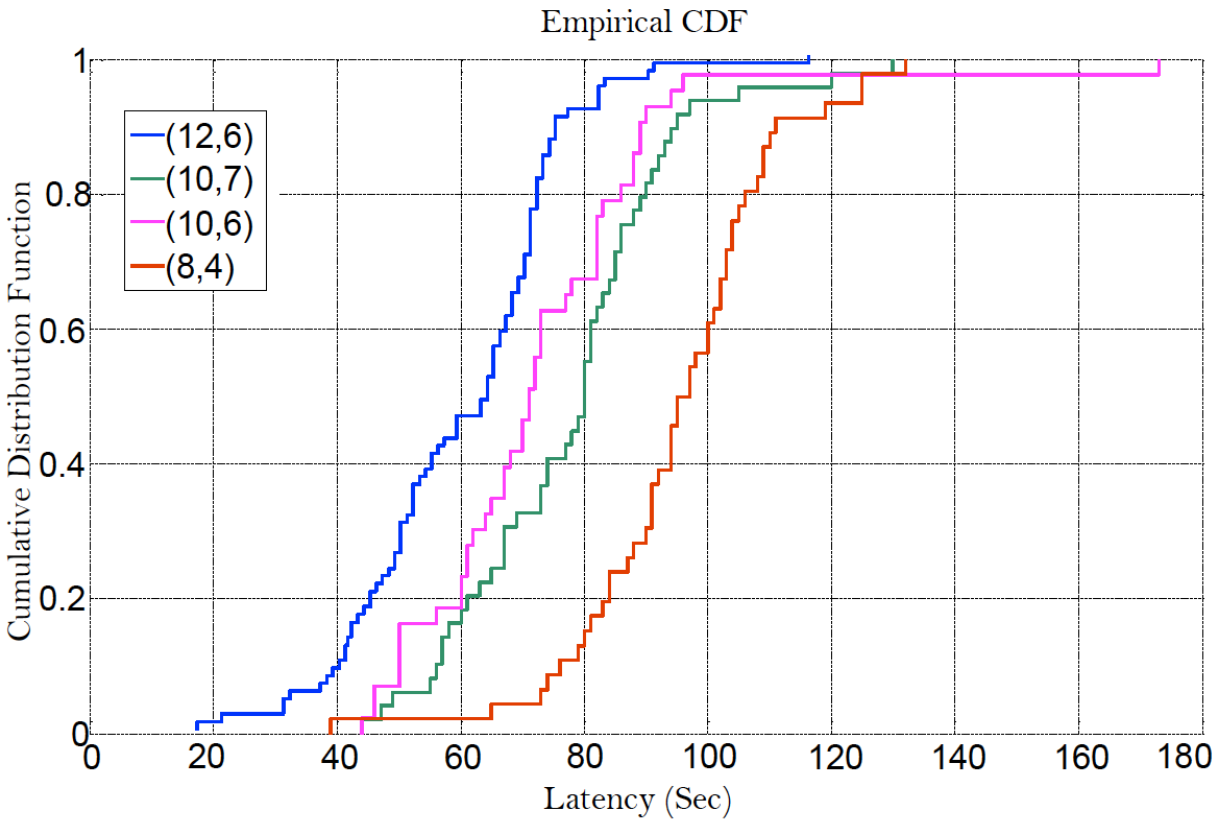
JOINT OPTIMIZATION (CODE, PLACEMENT, ACCESS) IS NEEDED



- 1000 files, size 150MB. Cost: $1 for 25MB, tradeoff factor of 200 sec/dollar, chunk size 25MB
- Oblivious LB: Select nodes with probability proportional to service rate
- Random placement: Chooses best outcome of 100 random runs

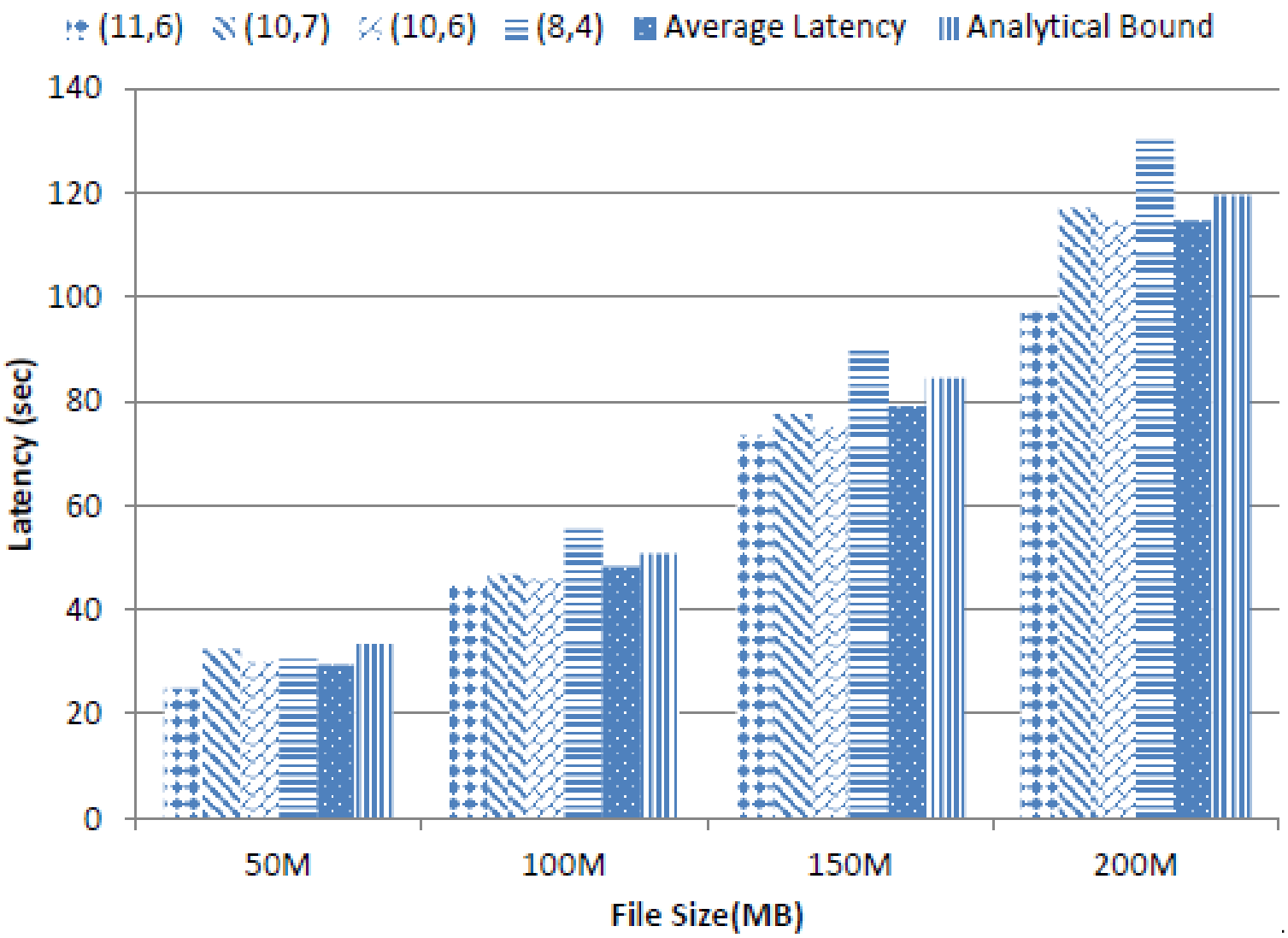
IISc GW UNIVERSITY WASHINGTON DC PURDUE UNIVERSITY

LATENCY DISTRIBUTION

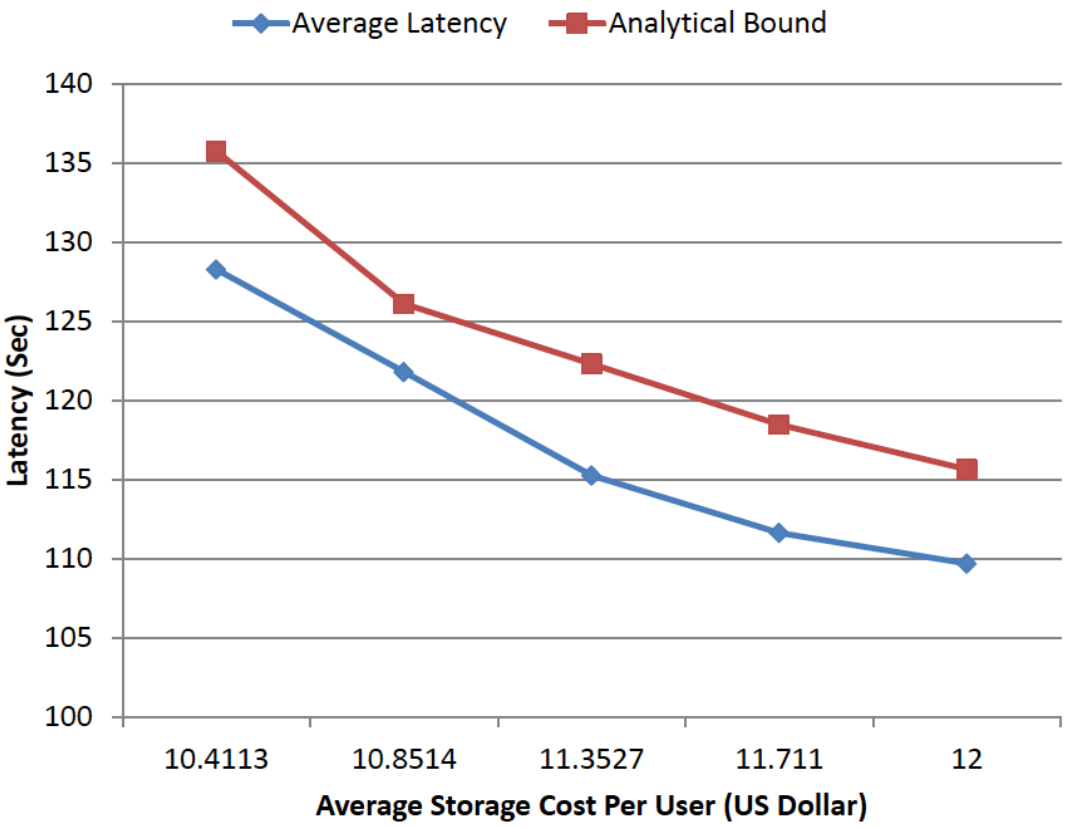


- 1000 files of size 150 MB, using erasure codes (12, 6), (10, 7), (10, 6), and (8, 4), aggregate rate at 0.118/s.

LATENCY INCREASES SUPER-LINEARLY WITH FILE SIZE



TRADEOFF CURVES



- Visualization of latency and cost tradeoff for file size of (150, 150, 100)MB and arrival rates 1/(30 sec), 1/(30sec), 1/(40 sec).

OTHER APPLICATIONS

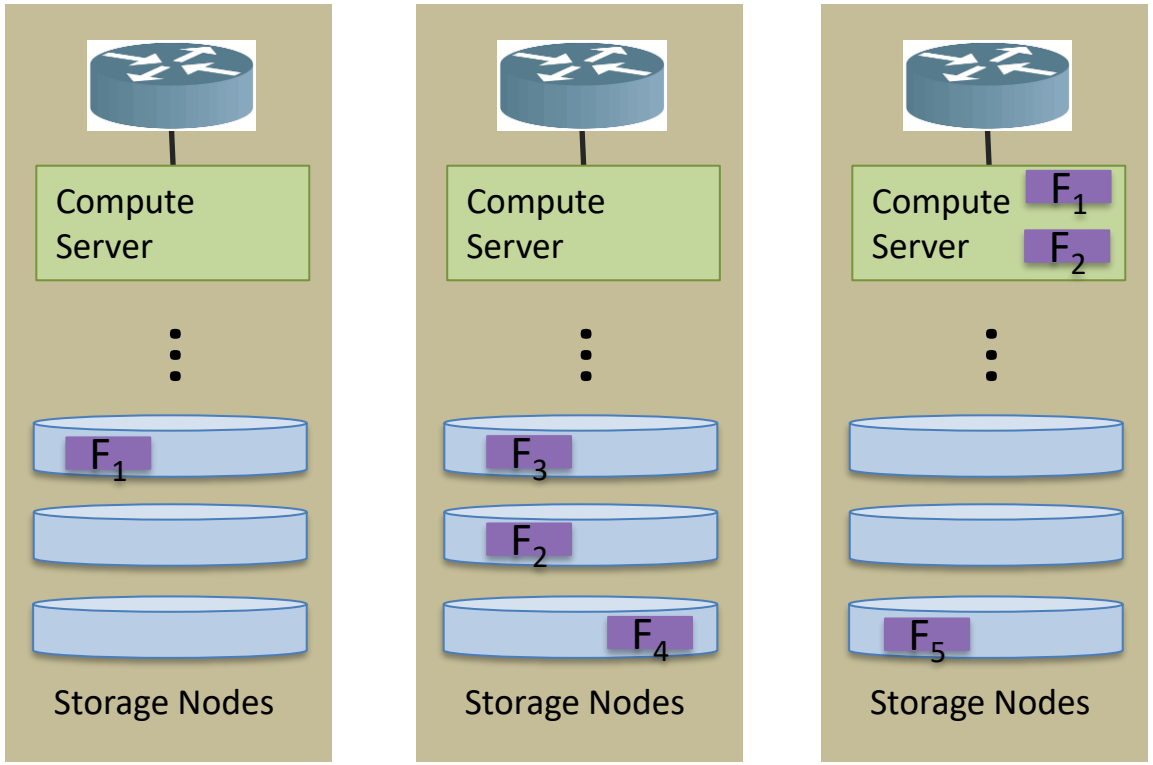
- Caching
- Video streaming over Cloud
- Memory-constrained system
- Coded Computing

OTHER APPLICATIONS

- Caching
- Video streaming over Cloud
- Memory-constrained system
- Coded Computing

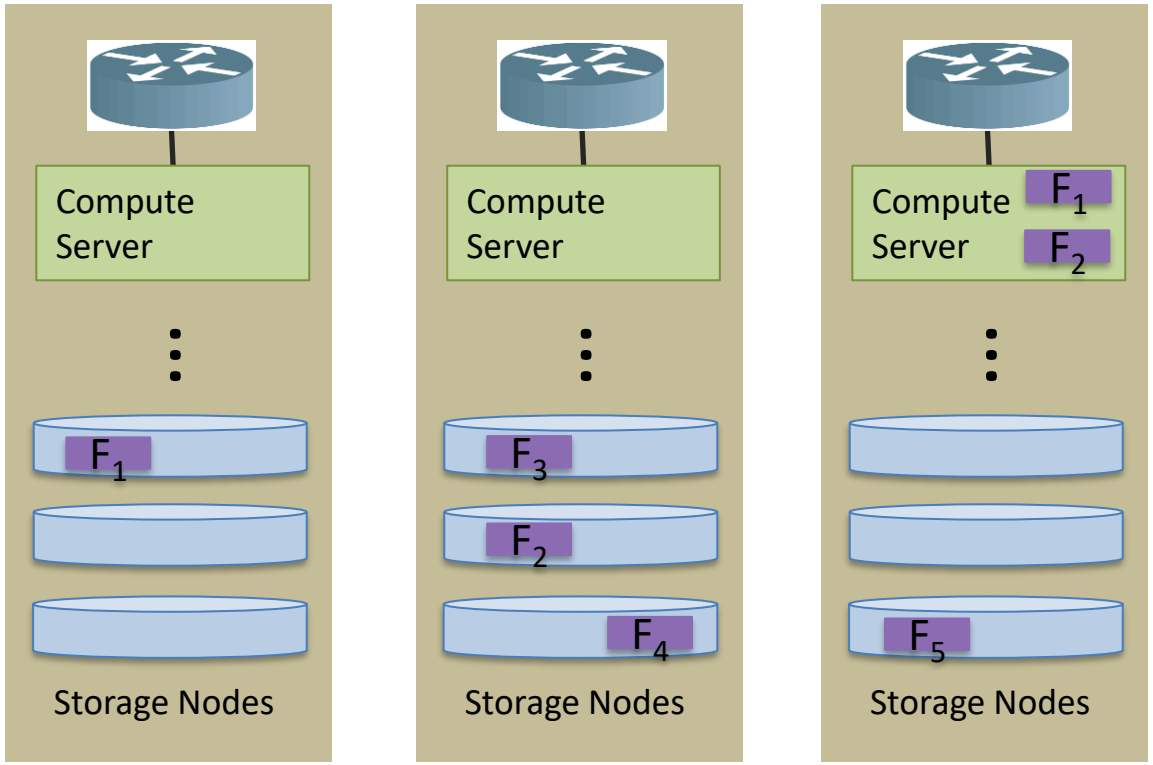
CACHING IN ERASURE CODED SYSTEMS

- Caching is used to reduce network congestion and improve service delay
- What files to cache?
- One approach: Least Recently Used (LRU) – add current file in cache and remove least recently used file.



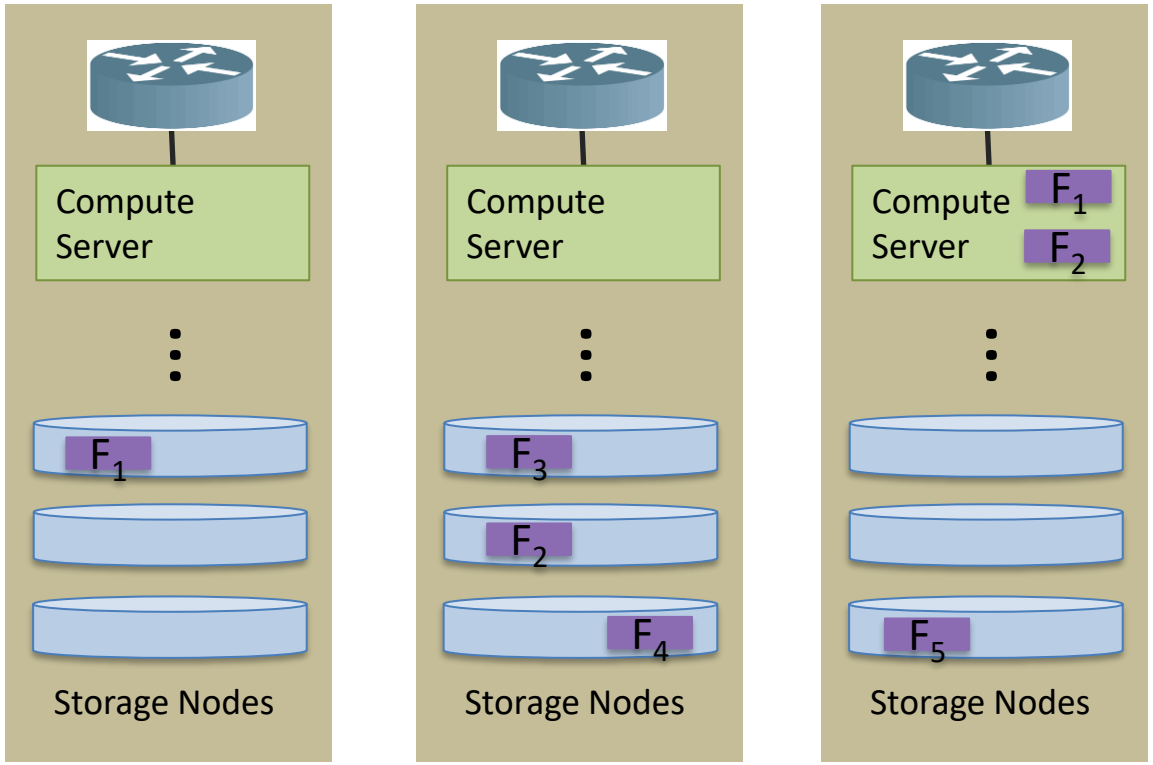
CACHING IN ERASURE CODED SYSTEMS

- Caching is used to reduce network congestion and improve service delay
- What files to cache?
- One approach: Least Recently Used (LRU) – add current file in cache and remove least recently used file.
- Issue: All chunks are stored in cache. Partial chunks?



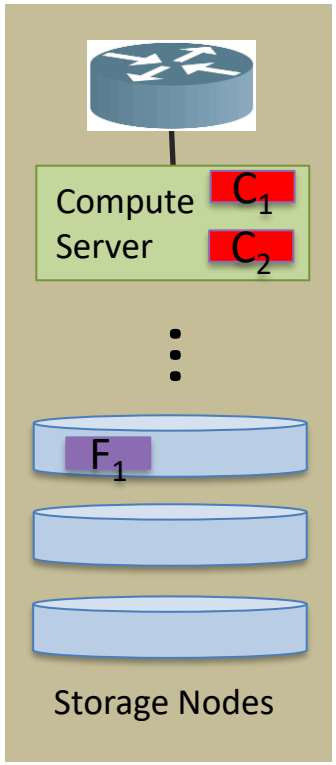
CACHING IN ERASURE CODED SYSTEMS

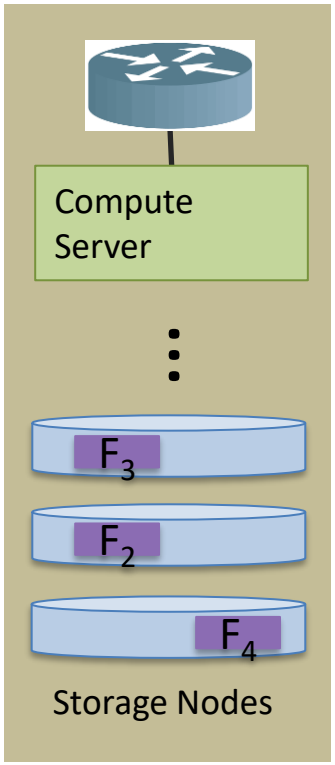
- Issue: All chunks are stored in cache. Partial chunks?
- Let a file has (n,k) coding
- Let d<k of file chunks are in cache
- On file access, k-d out of n-d file chunks will be requested using probabilistic scheduling.
- Is this the best??

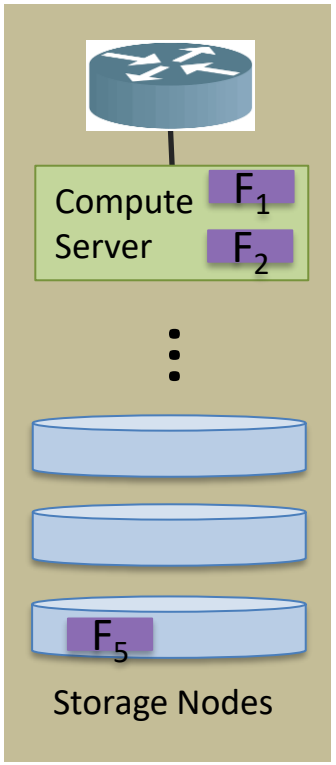


FUNCTIONAL CACHING IN ERASURE CODED SYSTEMS

- Erasure Coded Systems allow for functional caching
- Rather than exact chunks, place functionally equivalent chunks. Have file encoded as (n+k,k), where n are in the servers
- To place d in cache, use d of the residual k chunks.
- On access, k-d out of n can be requested.







Functional Cache

$C_1=A_1+2A_2+3A_3+4A_4$

$C_2=4A_1+3A_2+2A_3+A_4$

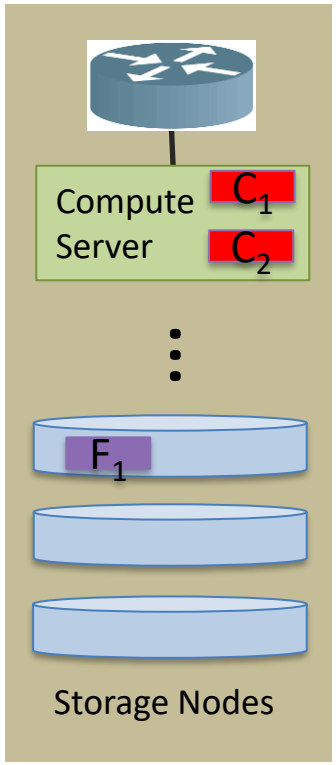
Exact Cache

$C_1=F_1$

$C_2=F_2$

LATENCY CALCULATION WITH FUNCTIONAL CACHING

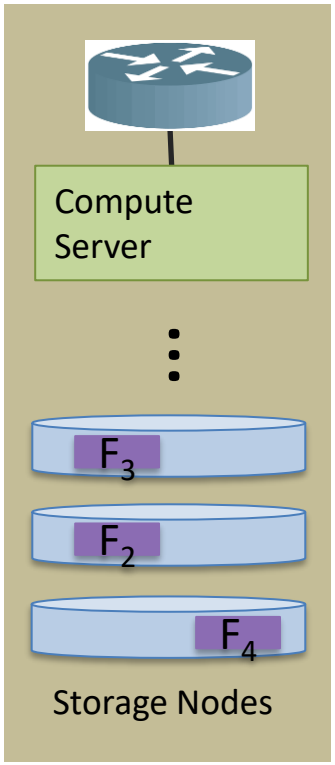
- The latency calculations remain the same as before except that the number of servers to access changes from k to k-d.
- This helps reduce the latency with caching.
- Specific choice of d chunks in the cache will have also change the possibility of accessed servers, while functional caching is more flexible due to using (n+k,k) rather than (n,k) code.

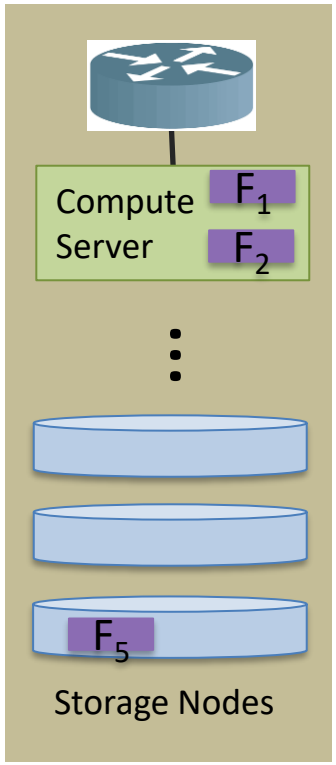


Functional Cache

$C_1=A_1+2A_2+3A_3+4A_4$

$C_2=4A_1+3A_2+2A_3+A_4$



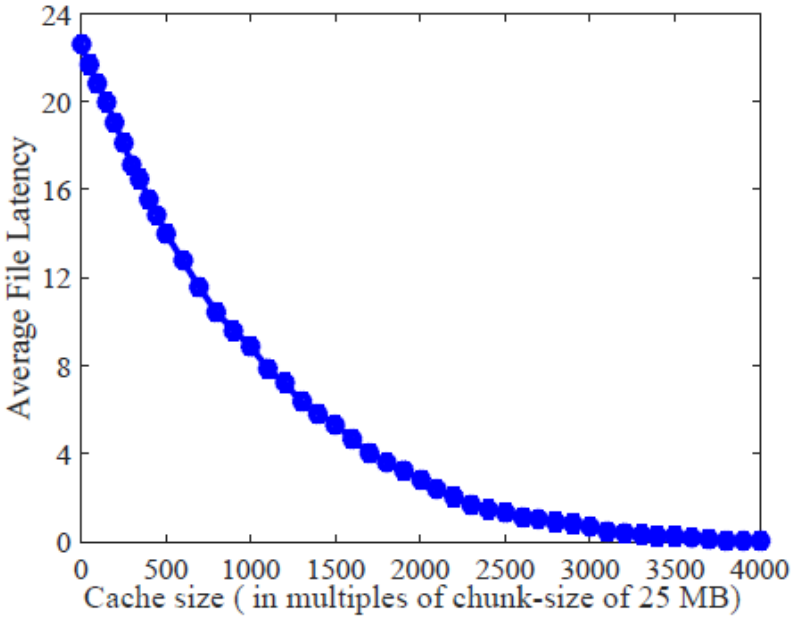


Exact Cache

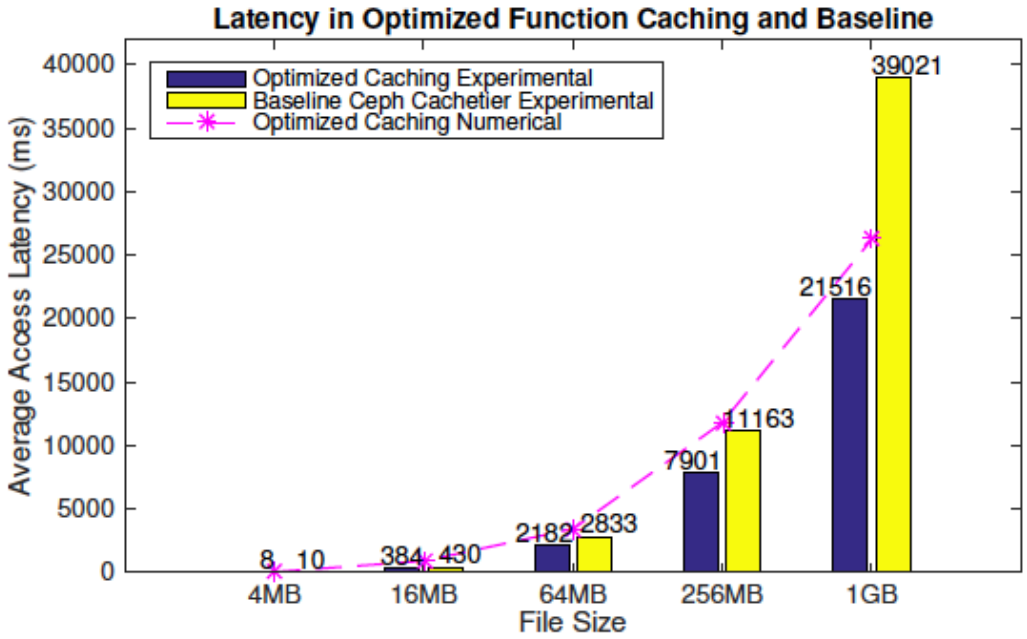
$C_1=F_1$

$C_2=F_2$

IMPACT OF FUNCTIONAL CACHING



1000 files 100 MB each, (n=7,k=4)



1000 files, (n=7,k=4), cache size 10GB

SUMMARY

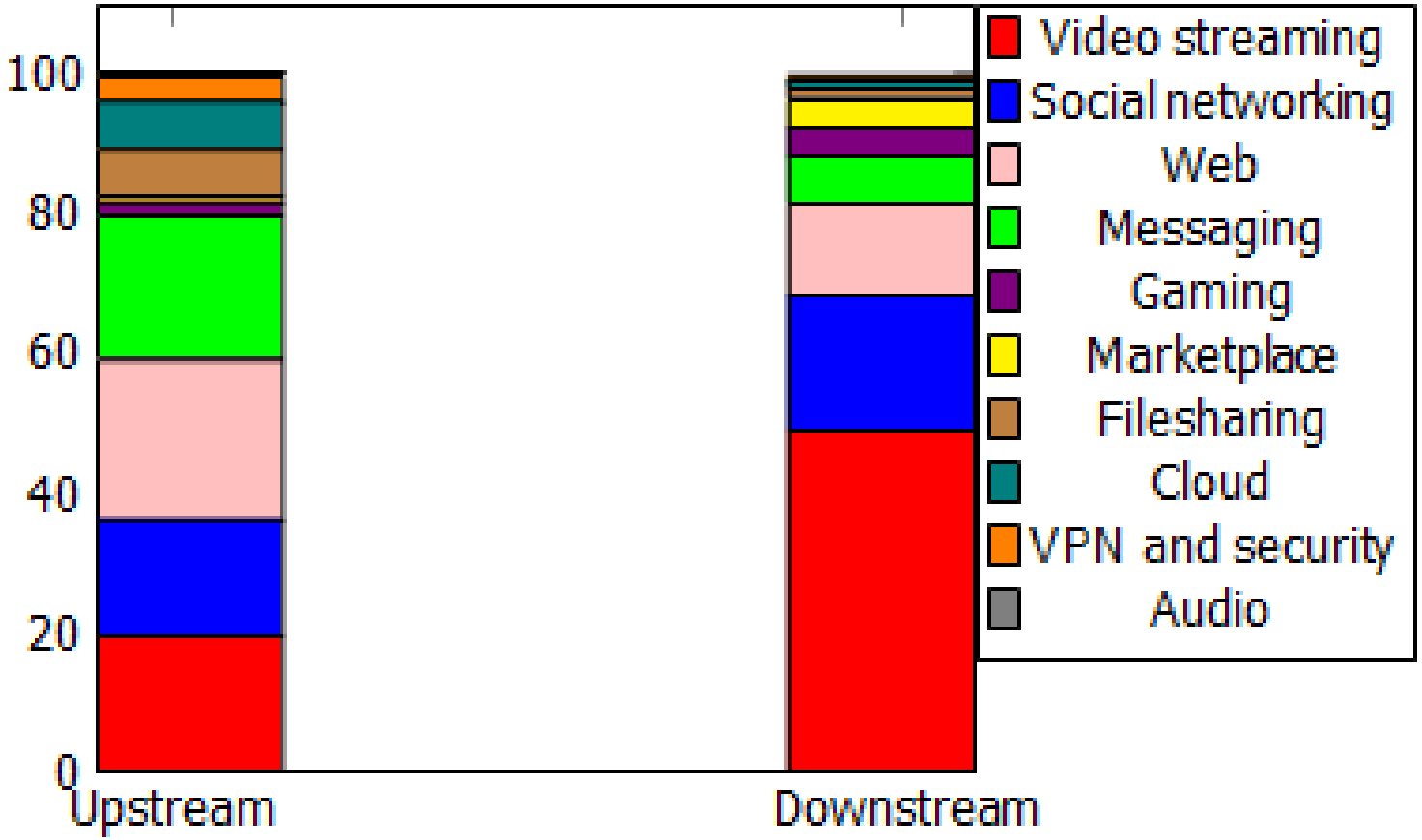
- Distributed Erasure-coded storage allows for improved caching strategies.
- Placing coded segments of the files help provide improved performance.
- The caching implemented on Ceph demonstrate improved performance metrics

- Key Reference
  - Aggarwal, Chen, Lan, and Xiang, "Sprout: A functional caching approach to minimize service latency in erasure-coded storage," IEEE/ACM Transactions on Networking, vol. 25, no. 6, pp. 3683-3694, Dec 2017 (earlier version in ICDCS 2016).
- Erasure-coded caching with complete file in the cache, better than LRU
  - Abubakr O. Alabbasi and Vaneet Aggarwal, "TTLCache: Taming Latency in Erasure-Coded Storage Through TTL Caching," IEEE Transactions on Network and Service Management, vol. 17, no. 3, pp. 1582-1596, Sept. 2020,
- Other possible caching for Cloud Storage Systems
  - Friedlander and Aggarwal, "Generalization of LRU Cache Replacement Policy with Applications to Video Streaming," ACM Tompecs, Volume 4 Issue 3, August 2019.

OTHER APPLICATIONS

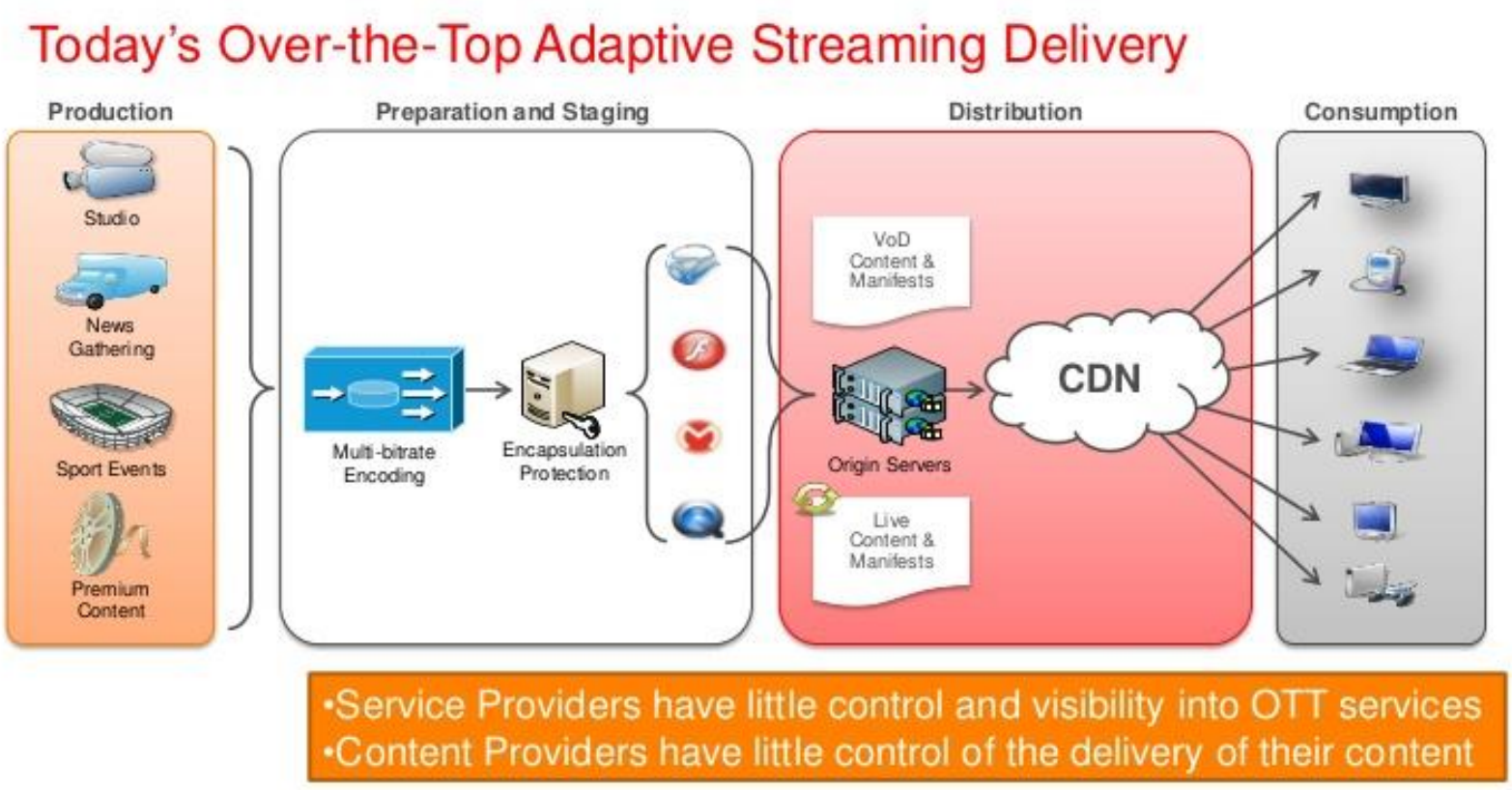
- Caching
- Video streaming over Cloud
- Memory-constrained system
- Coded Computing

GOLBAL APPLICATION TRAFFIC SHARE 2021



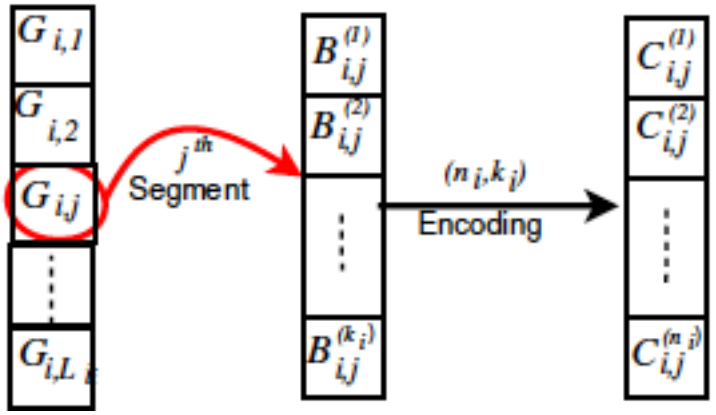
MOTIVATION

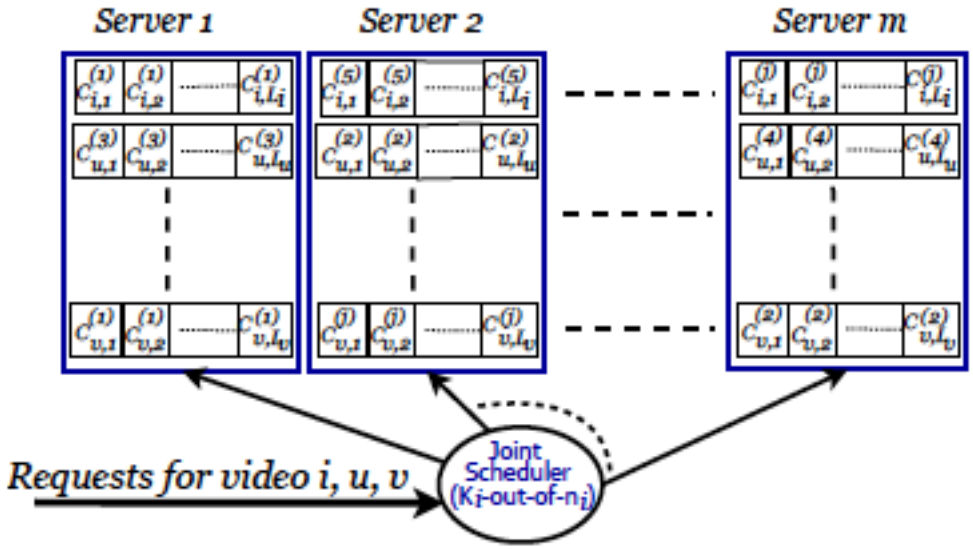
- Video streaming applications represents 62% of the Internet traffic in US
- More than 50% of over-the-top video traffic is now delivered through CDNs



VIDEO STREAMING

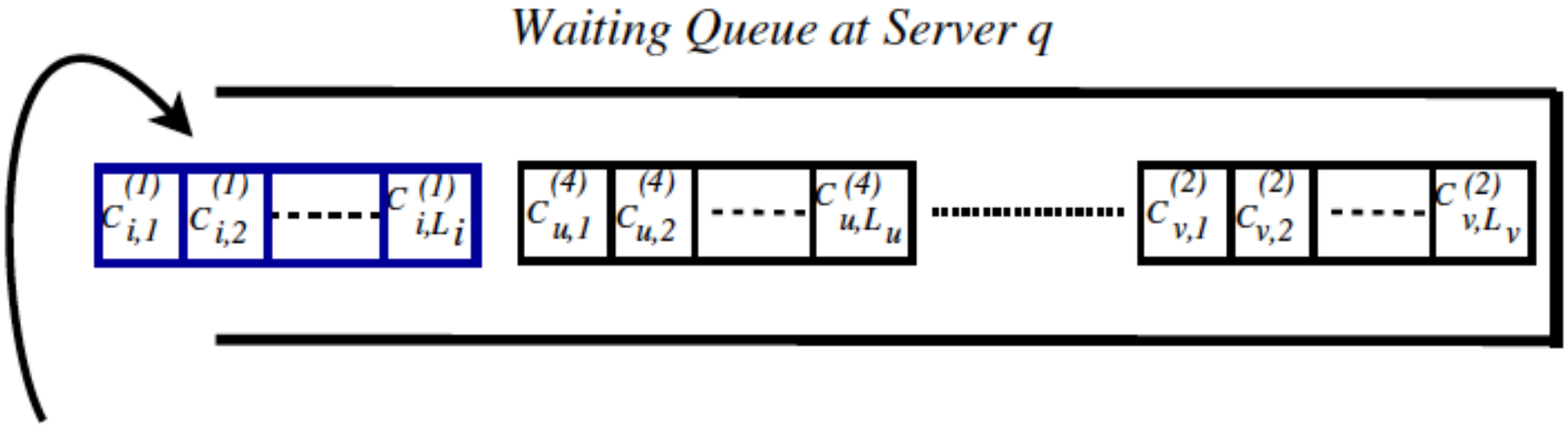
- Video Streaming rather than file download.
- Each chunk is erasure-coded
- Coded chunks on server
- Ques: How does servers stream video?





STREAMING APPROACH

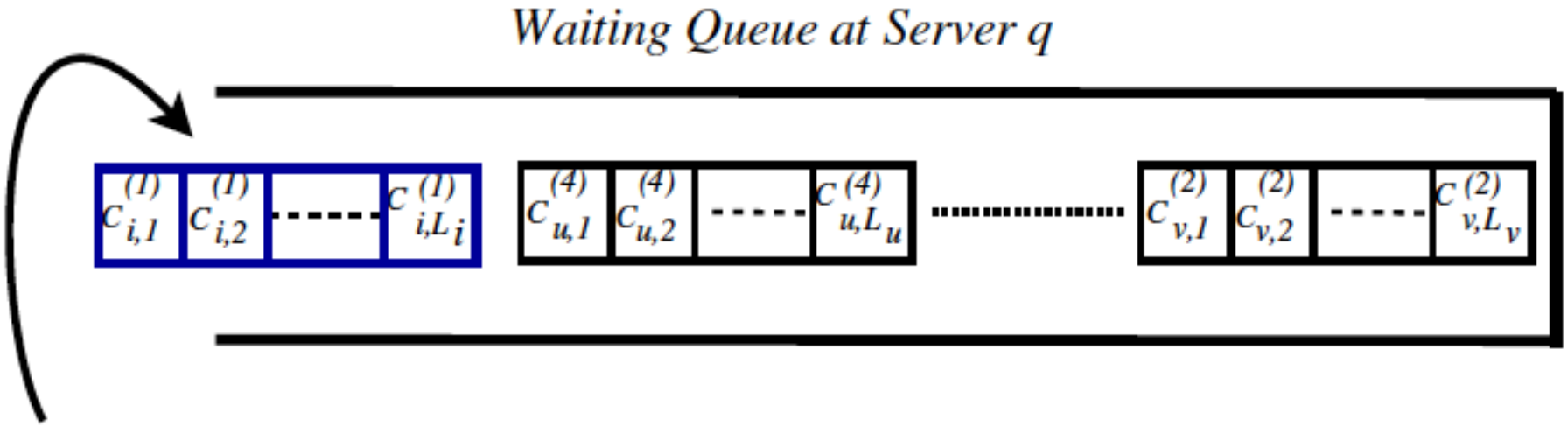
- Video Streaming rather than file download.
- Ques: How does servers stream video?
- Approach:



- Issue: A complete video behind another.
- Resolution: Have multiple virtual queues, and each video can be decided among these virtual queues

STALL DURATION

- Video Streaming rather than file download.
- Ques: How does servers stream video?
- Approach

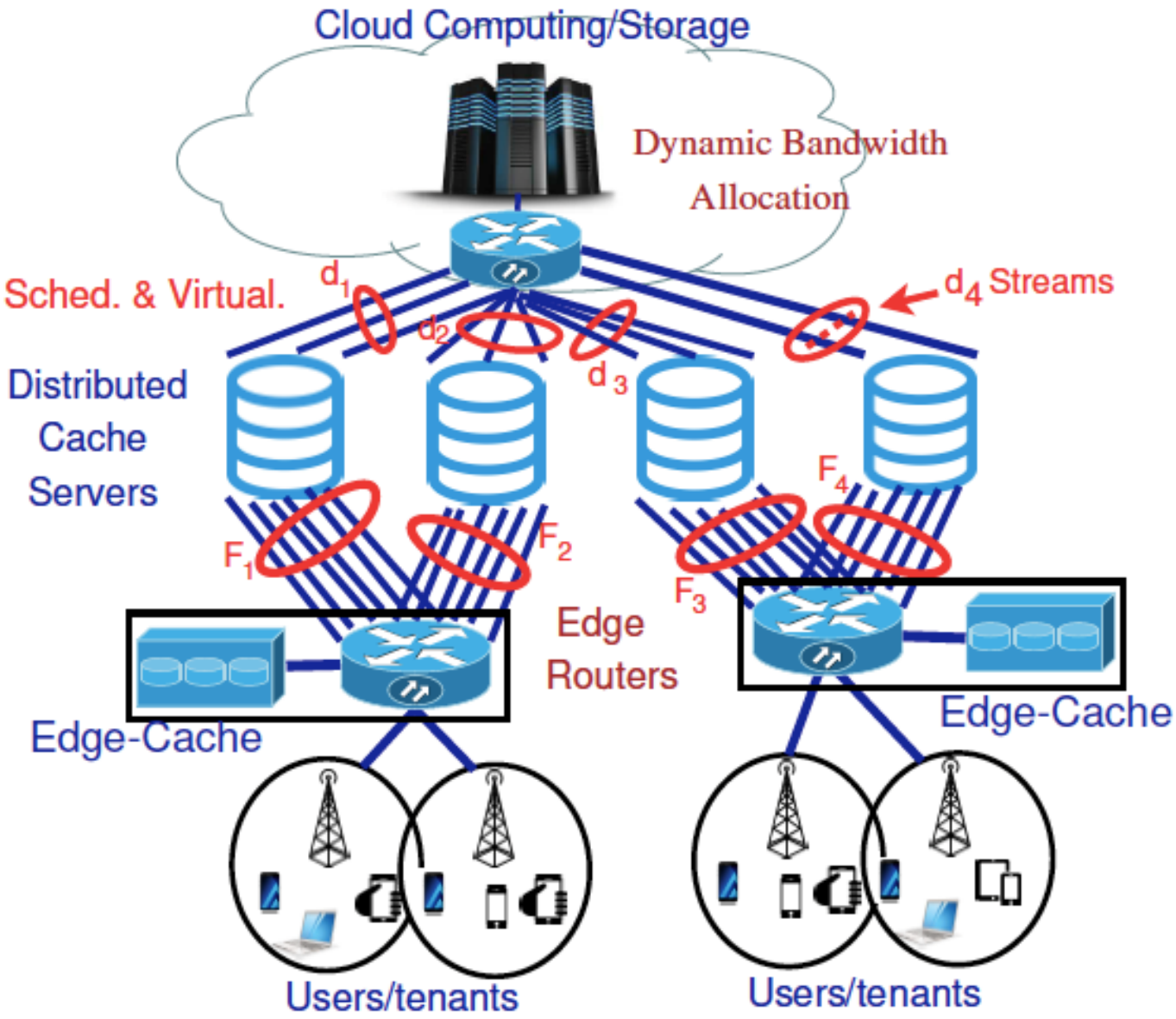


- Metric: Stall Duration. Very different from download time since stalls happen anywhere, and all correlated segments need to be accounted.
- Characterized mean and tail of stall durations for this model.

KEY STEPS FOR STALL DURATION

- Compute the time in the queue for each server. Consider the entire data of a file in server j, the requests are still Poisson.
- The start of service with additional of coded-chunk service times will give the service times of the different coded-chunks.
- The ordered statistics are used to obtain the receipt of video segments at the end user.
- From the download times, the play times are found
  - Play time for first segment is max of startup delay and download time of 1st segment
  - Play time for segment k is max of play time of segment k-1+play time of segment and download time of segment k
- From play time, stall duration is calculated as actual play time of last segment minus expected play time of last segment.
- As before, max are changed with sum in m.g.f.

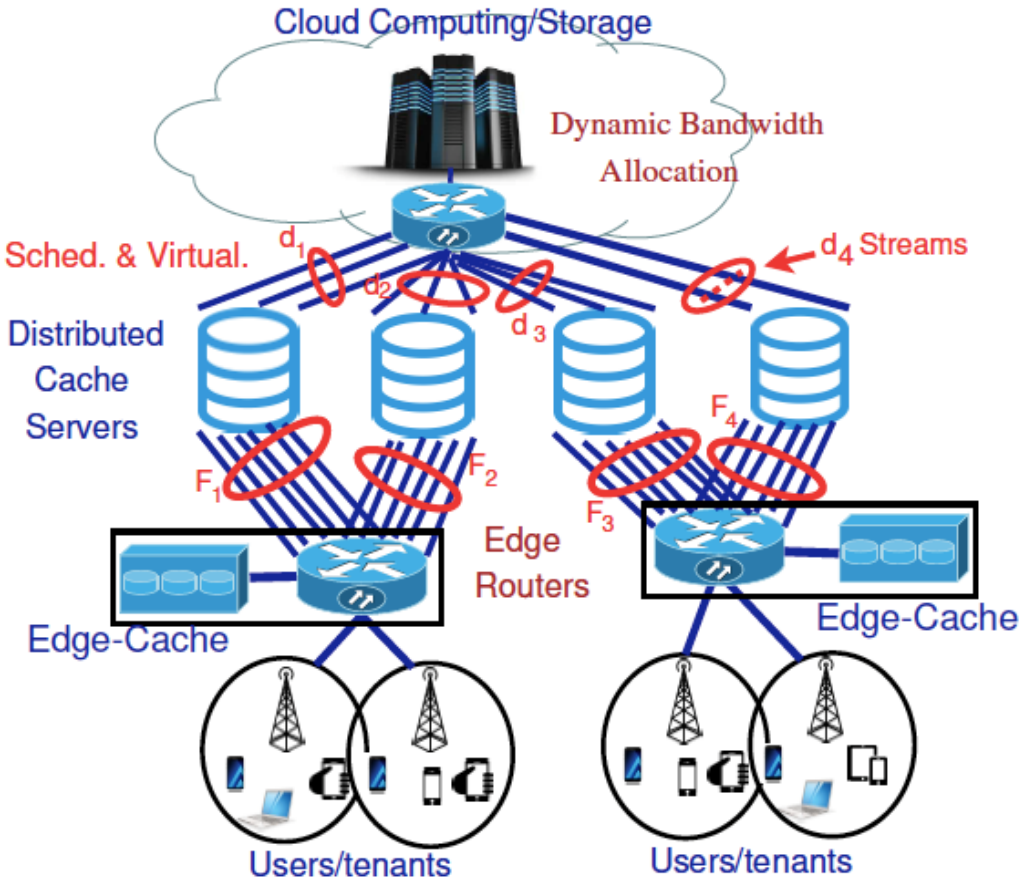
BEYOND SINGLE TIER



PURDUE UNIVERSITY

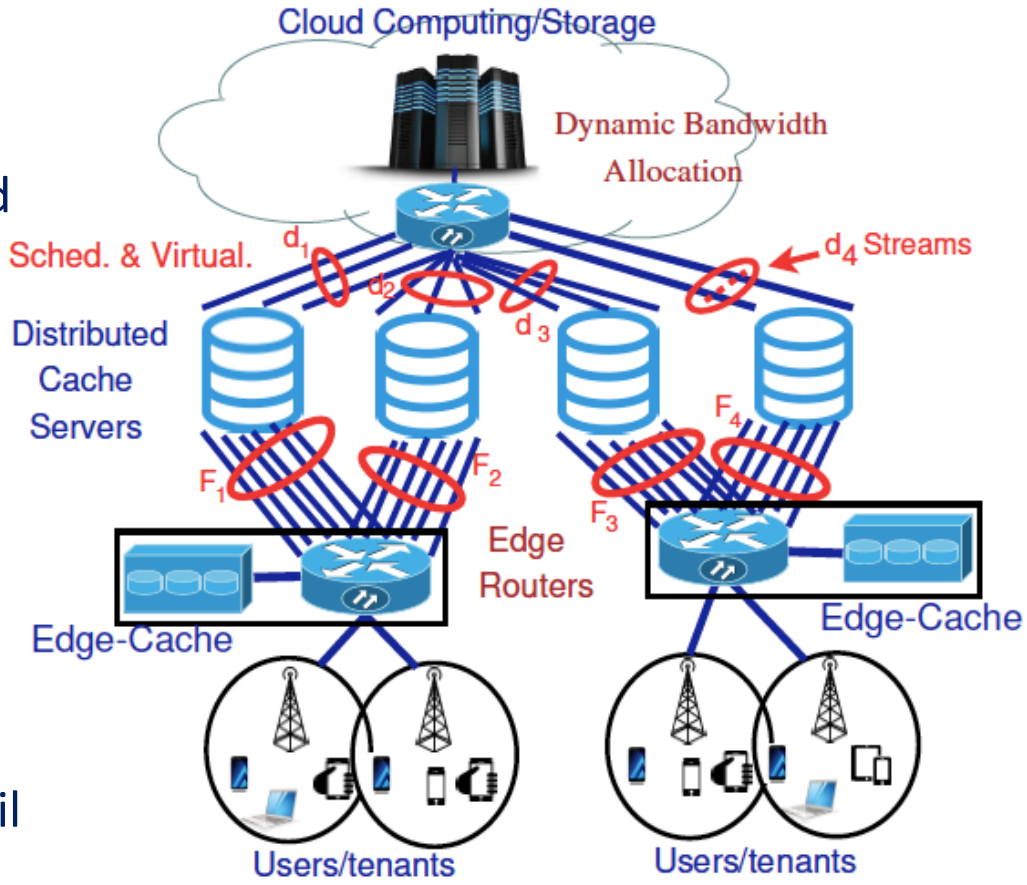
BEYOND SINGLE TIER

- Multiple CDNs
- Caching at CDNs
- Caching in Edge cache
- Edge cache allows for multicast since a later user can get previous content from cache.
- CDN Cache policy: How many initial chunks of each file?
- Edge Cache policy: Each requested file is cached for a certain time, and if not re-requested removed.

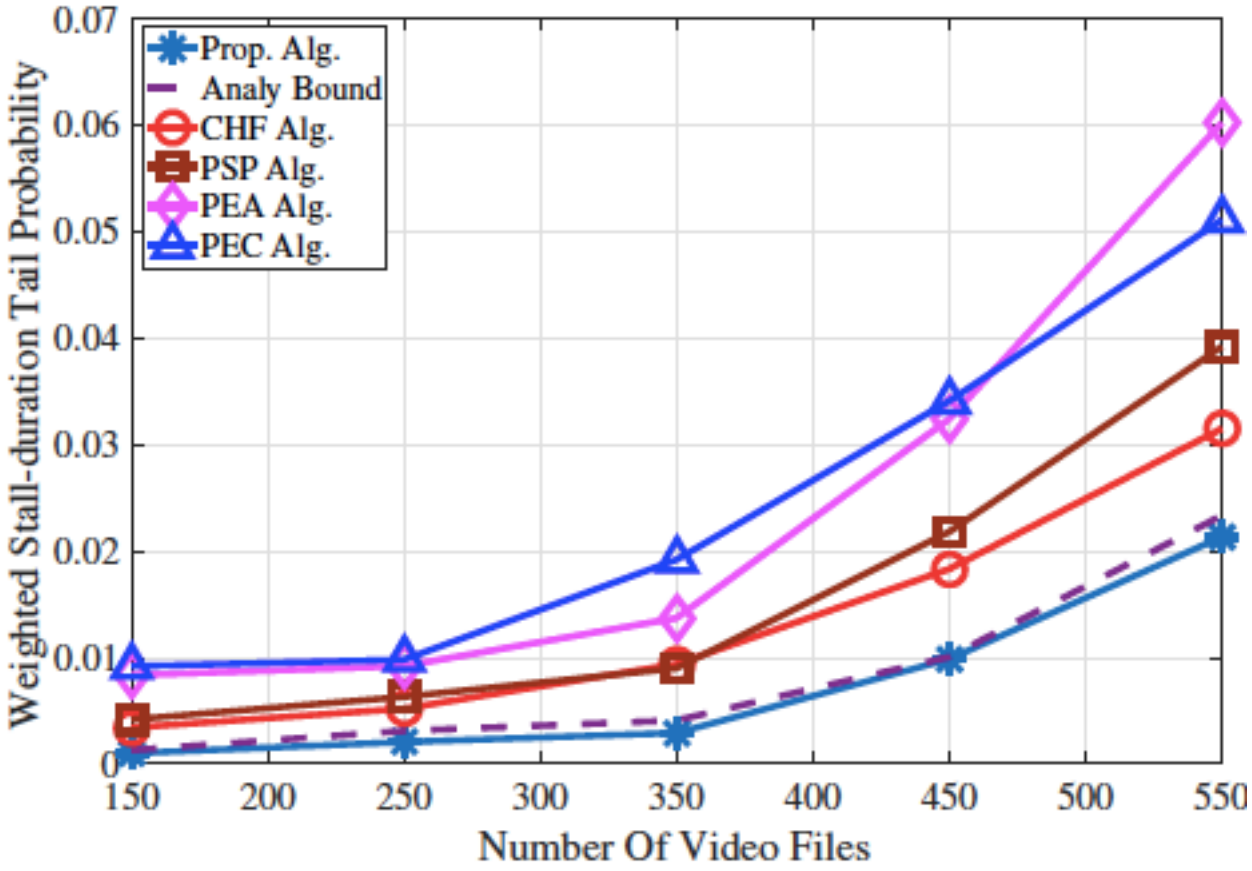


OPTIMIZATION PARAMETERS AND METRIC

- Access probabilities for CDNs, and the different streams from CDN and cloud storage.
- Auxiliary parameters for the bound
- Bandwidth parameters
- Cache placement in CDN
- Edge Cache removal parameter
- Metric: Weighted Stall Duration Tail Probability



OPENSTACK IMPLEMENTATION RESULT



- CHF: Caching hot files, PSP: projected proportional service, PEA: Equal probability access, PEC: Projected Equal Caching.

SUMMARY

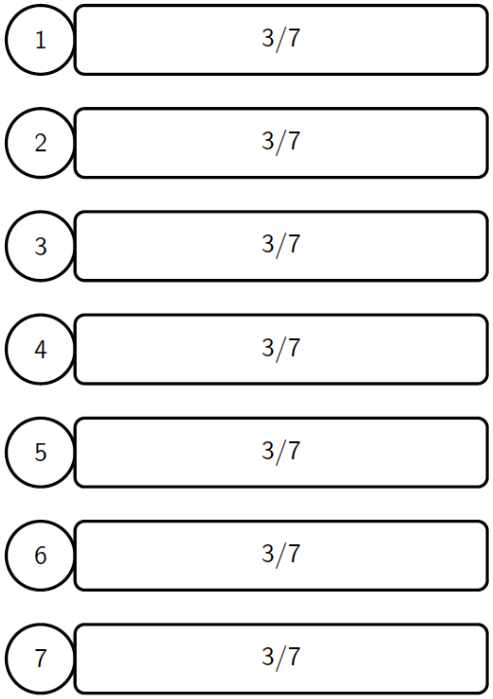
- New framework for video streaming over CDN
- Gave new bounds for stall duration with multiple flexibilities
- The results demonstrate improved performance metrics

- Single Tier
  - Alabassi and Aggarwal, "Video Streaming in Distributed Erasure-coded Storage Systems: Stall Duration Analysis," IEEE/ACM Transactions on Networking, vol. 26, no. 4, pp. 1921-1932, Aug. 2018.
  - Al-Abbasi and Aggarwal, "VidCloud: Joint Stall and Quality Optimization for Video Streaming over Cloud," ACM Transactions on Modeling and Performance Evaluation of Computing Systems, article no. 17, Jan 2021
- Multi-Tier
  - Alabbasi, Aggarwal, Lan, Xiang, Ra, and Chen, "FastTrack: Minimizing Stalls for CDN-based Over-the-top Video Streaming Systems," Accepted to IEEE Transactions on Cloud Computing, Jun 2019.
  - Alabbasi, Aggarwal, and Ra, "Multi-tier Caching Analysis in CDN-based Over-the-top Video Streaming Systems," IEEE/ACM Transactions on Networking, vol. 27, no. 2, pp. 835-847, April 2019.

OTHER APPLICATIONS

- Caching
- Video streaming over Cloud
- Memory-constrained system
- Coded Computing

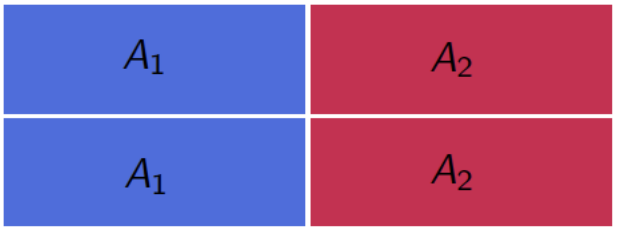
MEMORY CONSTRAINED SYSTEM

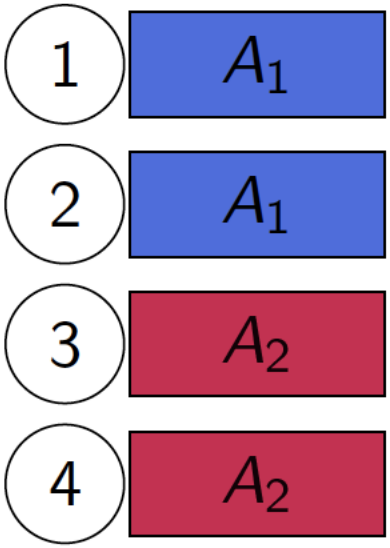


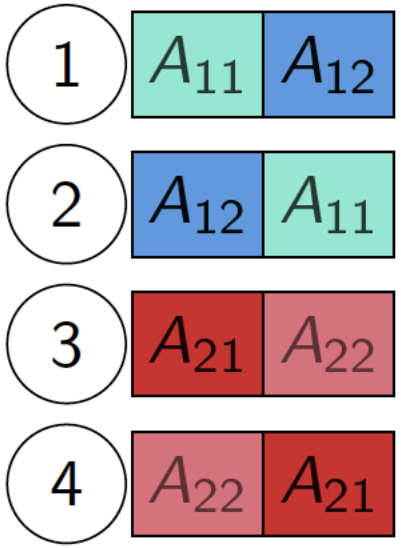
Storing αB size coded messages for a unit size message

- parallel access from all B servers
- α-fragment of message stored at each server

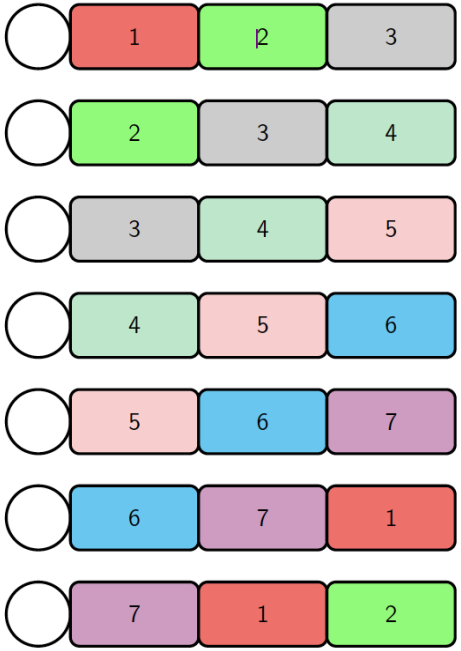
STORAGE MODEL: PLACEMENT







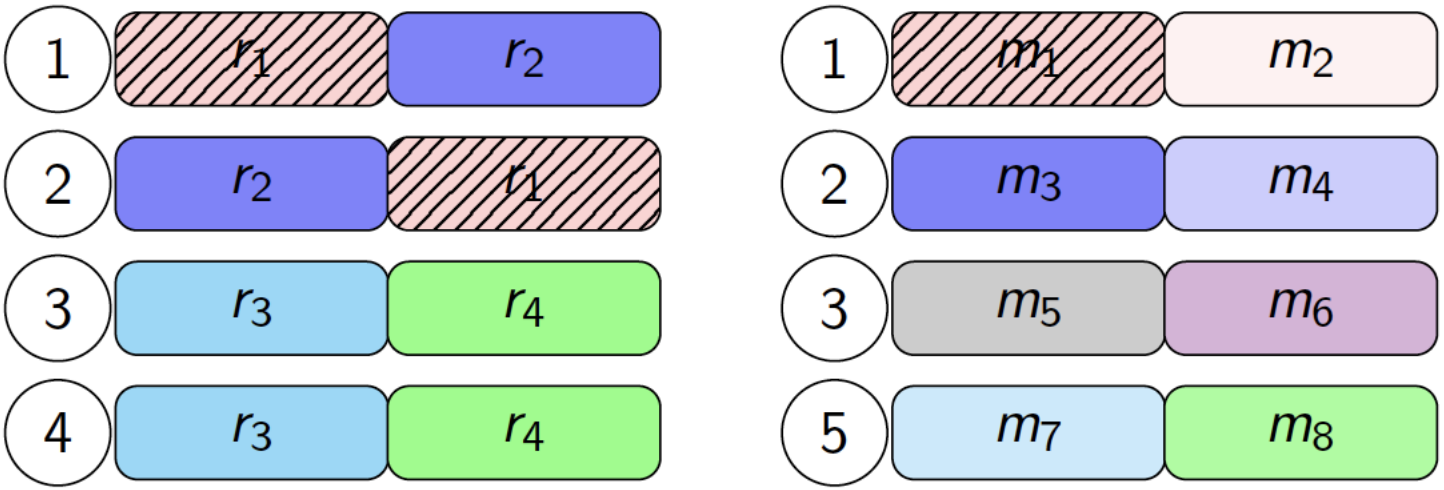
LATENCY OPTIMAL STORAGE AND ACCESS



A unit size divisible message $m = (m_1, \ldots, m_V)$

- replicated $R = \alpha B/V$ times
- **storage:** for each fragment, where to store each replica?
- **access:** for each server, sequence of access for replicas?

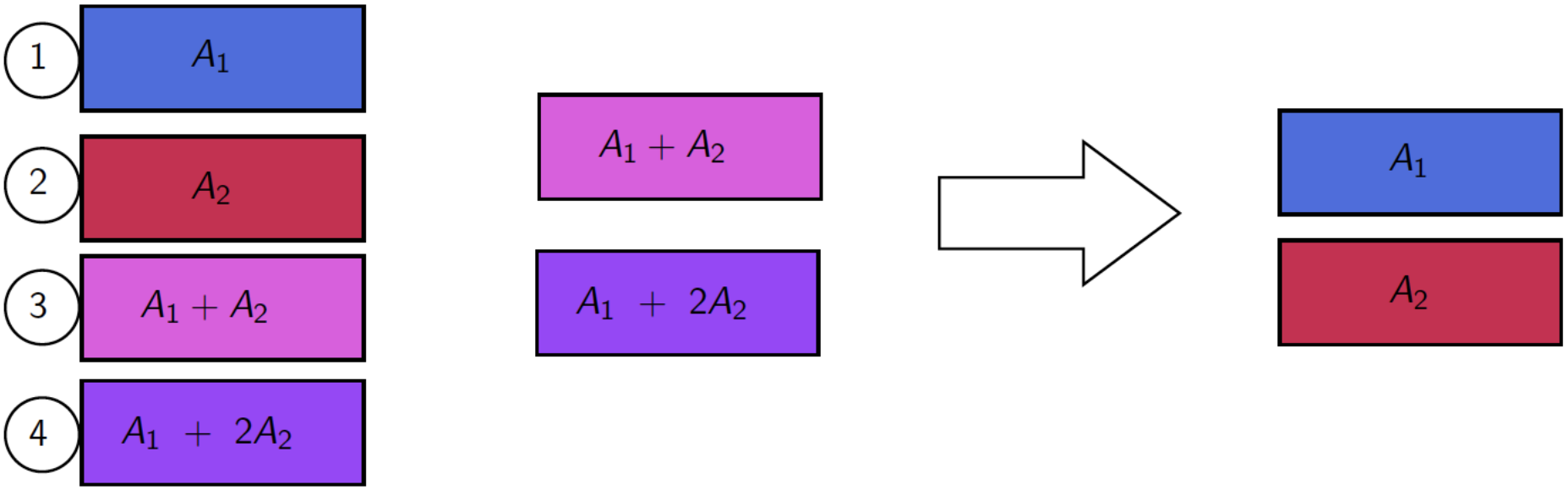
MDS CODED STORAGE



Optimality of MDS coded storage

- Sequence of number of useful servers is the largest
- Latency optimal storage code

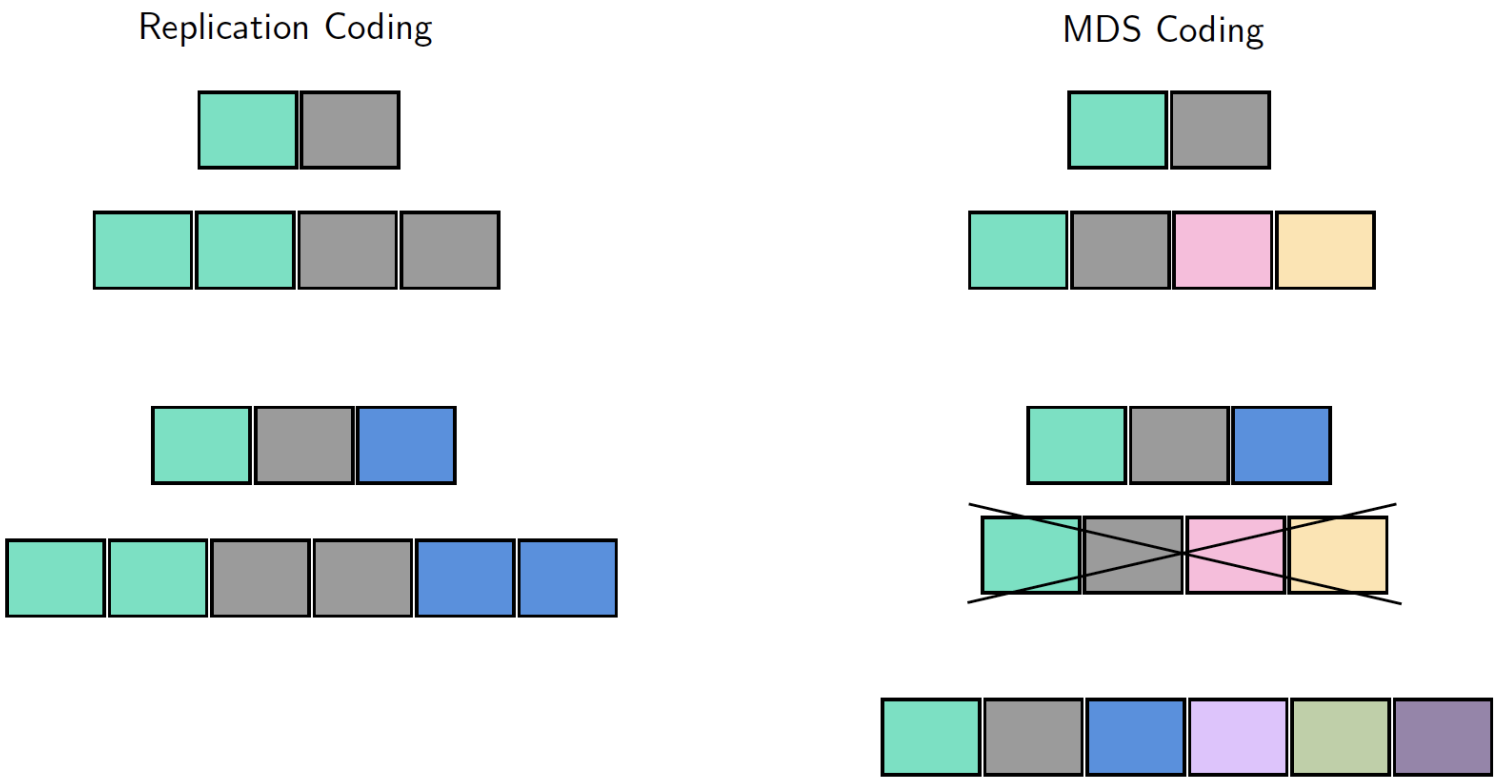
DECODING COMPLEXITY



Implementation challenges

- Requires sufficiently large alphabet or large fragment sizes
- Polynomial decoding complexity that can't be parallelized

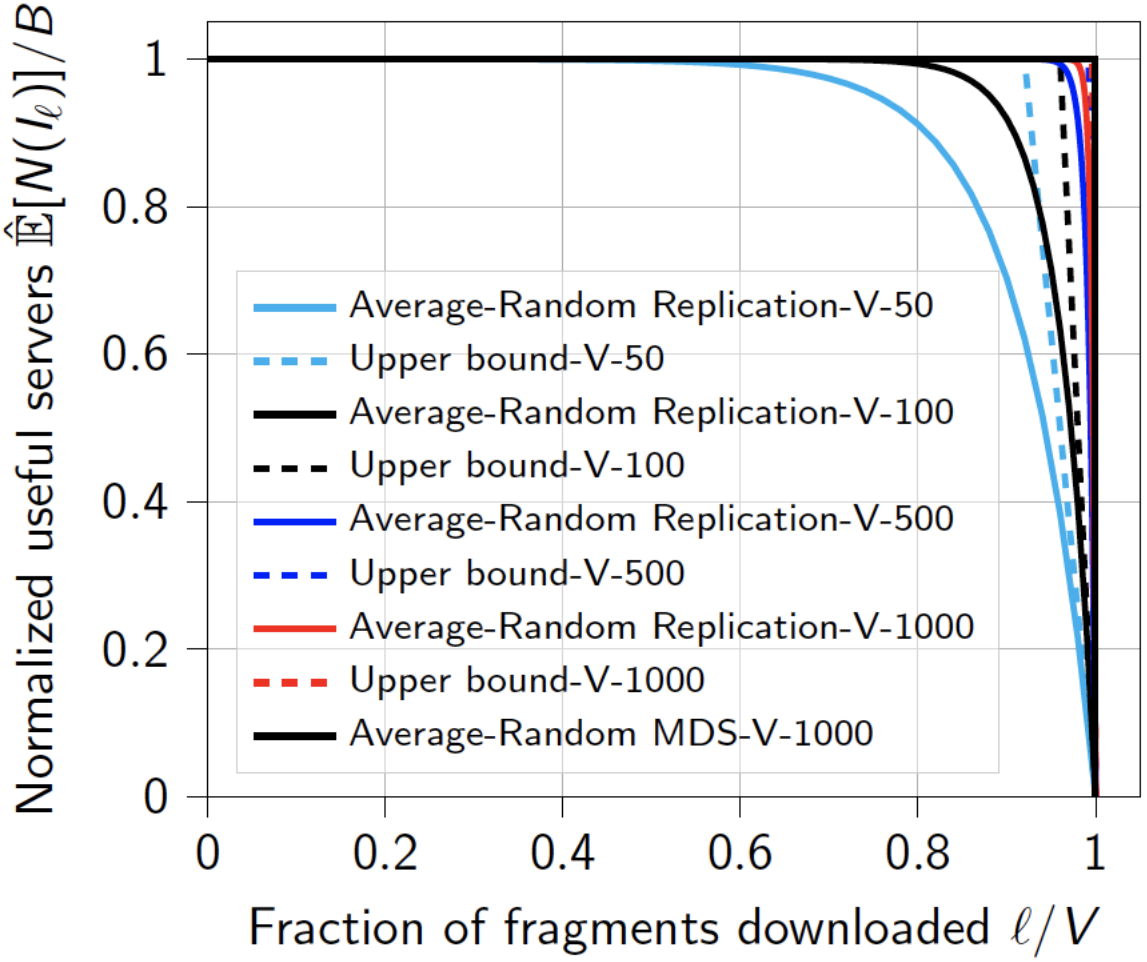
SCALING ISSUES OF MDS CODING



Encoding growing data or redundancy

- Complete re-encoding of data blocks
- Potential data loss waiting for sufficient data blocks

NUMERICAL RESULTS



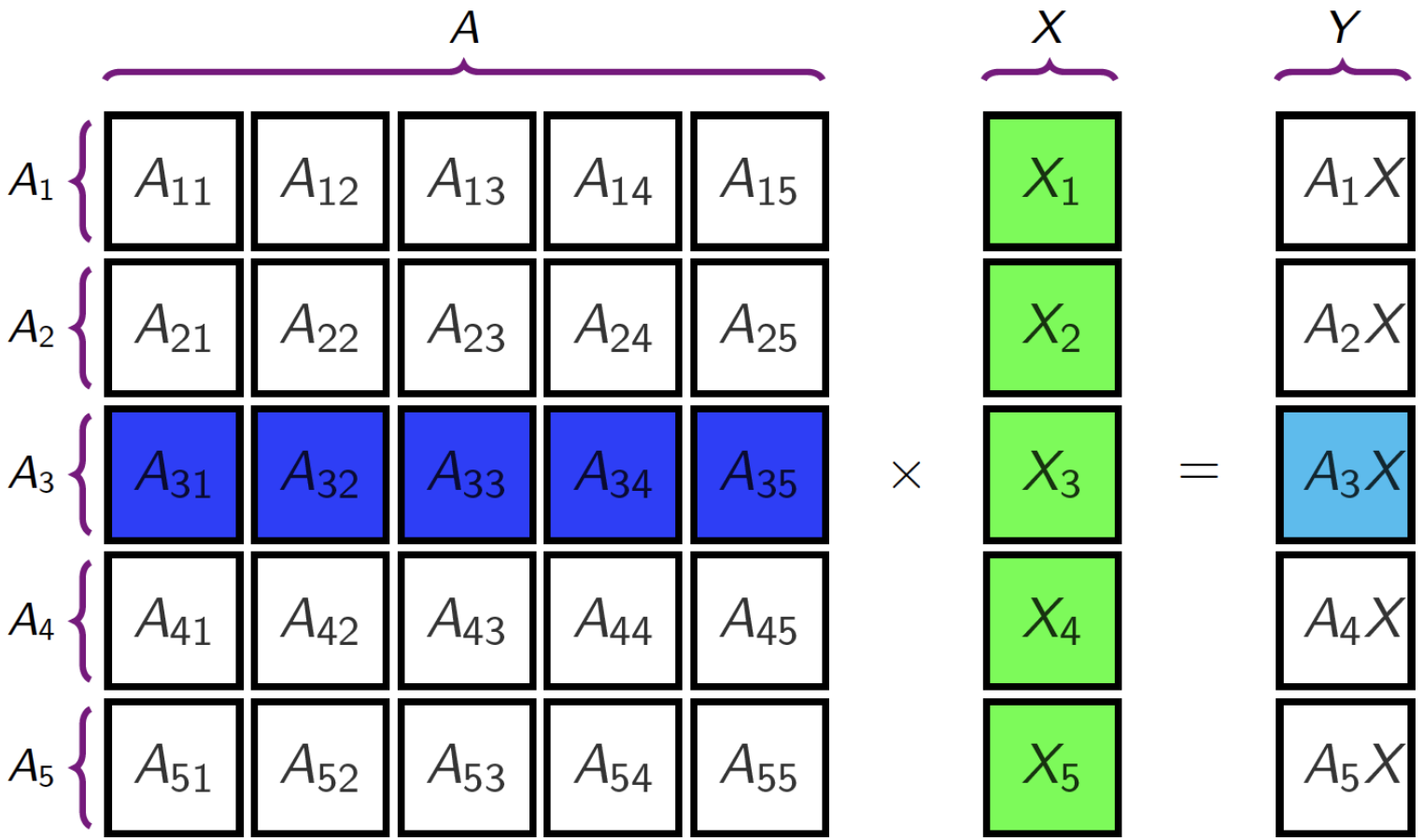
SUMMARY

- MDS coded storage is optimal for subfragmented storage
- Subfragmentation of file can lead to competitive performance of replication coded storage
- When storage nodes have no memory constraints all coded storage have identical latency performance
- Staircase coded storage
  - Bitar, Parag, and Rouayheb, ``Minimizing latency for secure coded computing using secret sharing via staircase codes,'' *IEEE Transactions on Communications*. 68(8):4609–4619, Aug 2020.
- Replication coded storage
  - Jinan, Badita, Sarvepalli, Parag, ``Latency optimal storage and scheduling of replicated fragments for memory-constrained servers,'' preprint, 2021.

OTHER APPLICATIONS

- Caching
- Video streaming over Cloud
- Memory-constrained system
- Coded Computing

MATRIX MULTIPLICATION

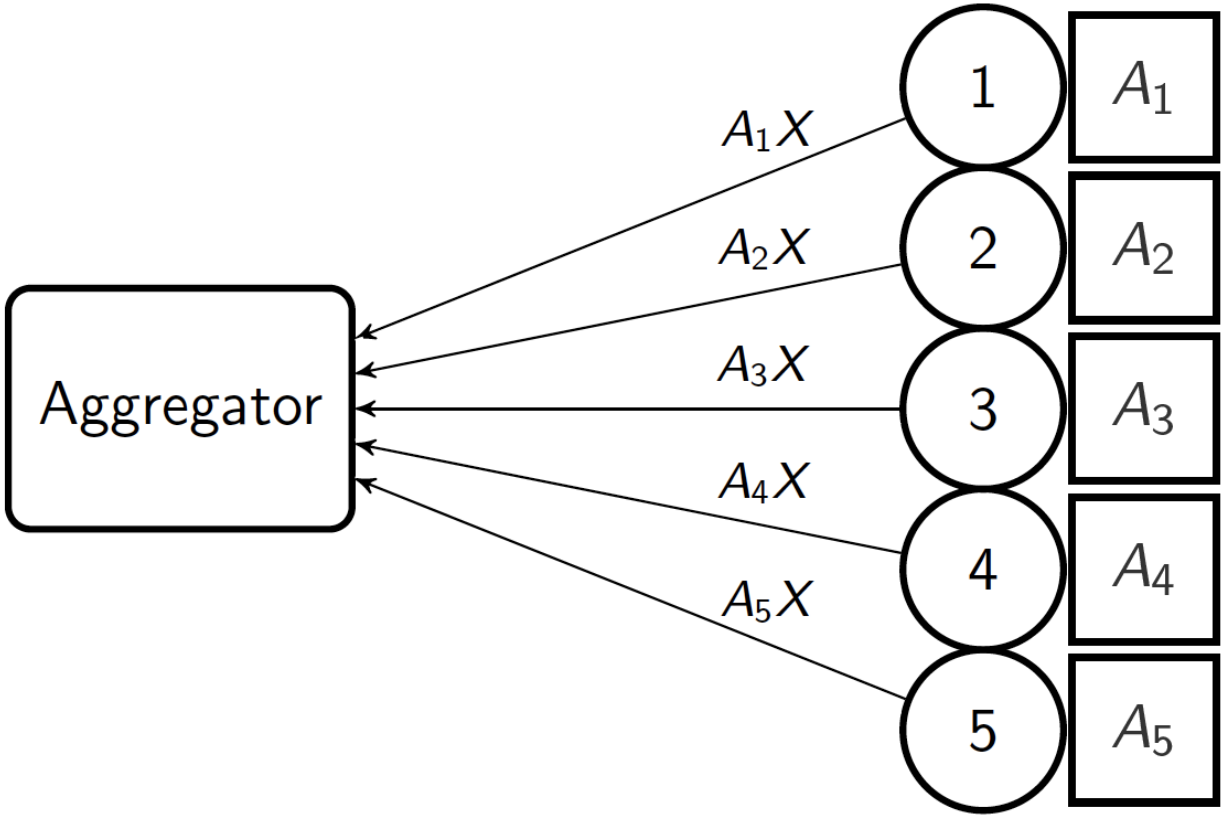


Common computation in many algorithms

- Not scalable as the size of matrix grows
- Output $A_i X = \sum_{j=1}^{K} A_{ij} X_j$

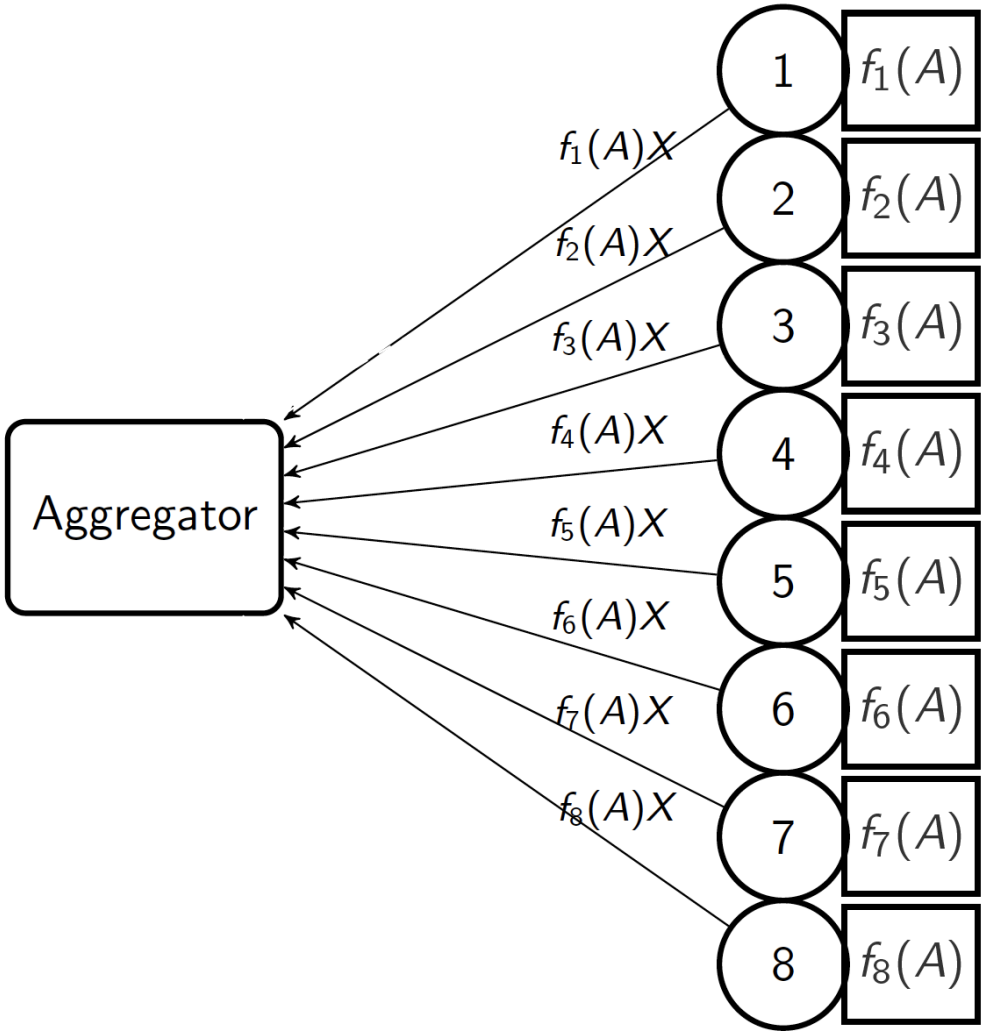
IISc GW UNIVERSITY WASHINGTON DC PURDUE UNIVERSITY

DISTRIBUTED MATRIX MULTIPLICATION

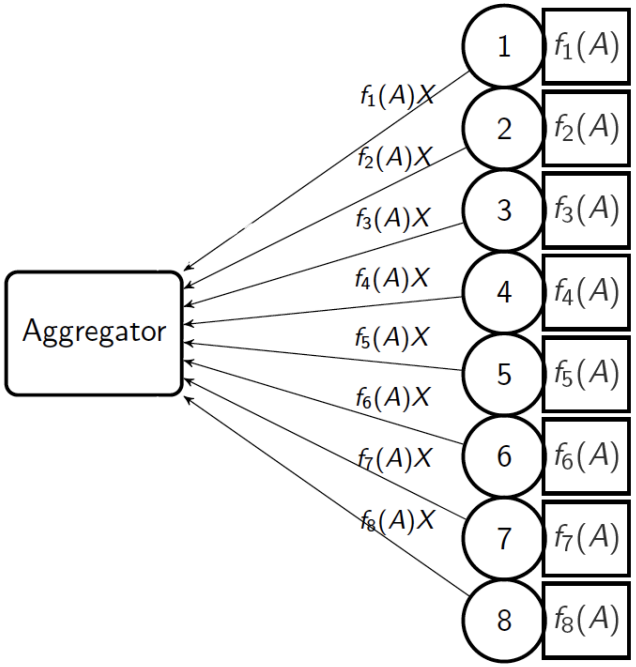


- Uncertainty in compute time at each server
- Total computation time limited by slowest server

REDUNDANCY FOR STRAGGLER MITIGATION



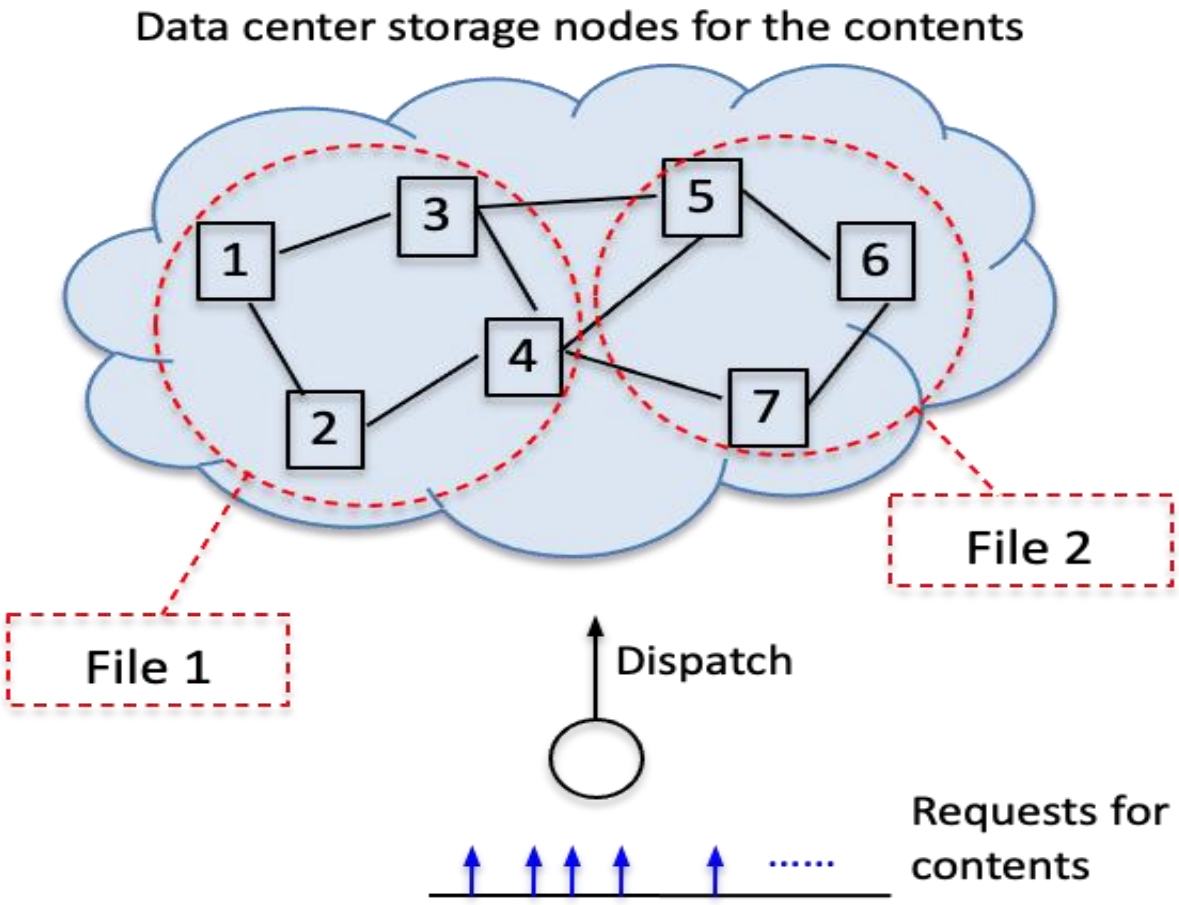
SUMMARY



Storage systems	Compute systems
Fragment download time	Subtask compute time
File download time	Task compute time
Straggling download	Straggling compute
Storage redundancy	Compute redundancy

PART 6: SUMMARY

KEY PROBLEM IN THIS TUTORIAL



- Modeling, characterization, and optimization of latency for distributed storage systems

IISc GW UNIVERSITY WASHINGTON DC PURDUE UNIVERSITY

SUMMARY

- Optimal scheduling is hard
- Multiple scheduling strategies are discussed – Reservation Scheduling, Fork-Join Scheduling, Probabilistic Scheduling, Delayed-Relaunch Scheduling
- The results have different assumptions on server distribution, files, and service times.
- Implementation of scheduling approaches on real servers is demonstrated with a discussion on optimizing storage systems
- Extensions to video streaming over cloud, and sub-packetization are discussed
- The problem is related to cloud computing with stragglers.

- Novel queueing strategies, improved results, different storage server architectures are some possibilities for future research.

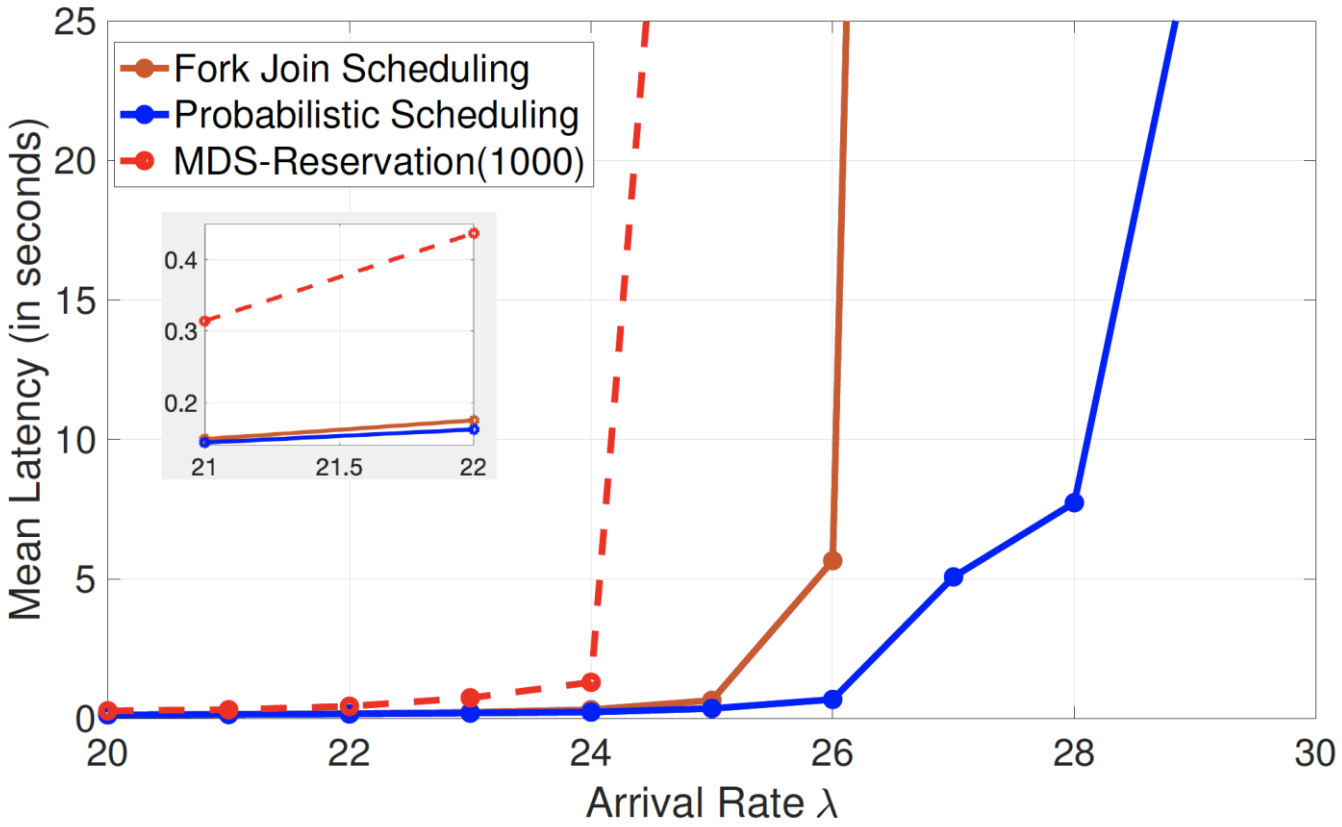
COMPARISON OF STATE-OF-ART: ASSUMPTIONS

	MDS-reservation	Fork-Join	Probabilistic	Delayed relaunch
Homogenous Files	Yes	No	No	Yes
Homogenous Placement	Yes	Yes	No	Yes
Homogeneous Servers	Yes	Yes	No	Yes
Exponential Service Time	Yes	No	No	No[3]

COMPARISON OF KEY SCHEDULING STRATEGIES

	MDS-Reservation	Fork-Join	Probabilitic	Delayed Relaunch
Optimal Homogenous Stability Region	No	Exponential	General	General
Queuing Analysis	Yes	Yes	Yes	No
Analysis for general distribution	No	Yes	Yes	No
Closed Form Expressions	No	Yes	Yes	N/A[5]
Asymptotic Optimality	No	No	Yes	Yes
Tail Characterization	No	No	Yes	No

NUMERICAL COMPARISON OF KEY SCHEDULING STRATEGIES

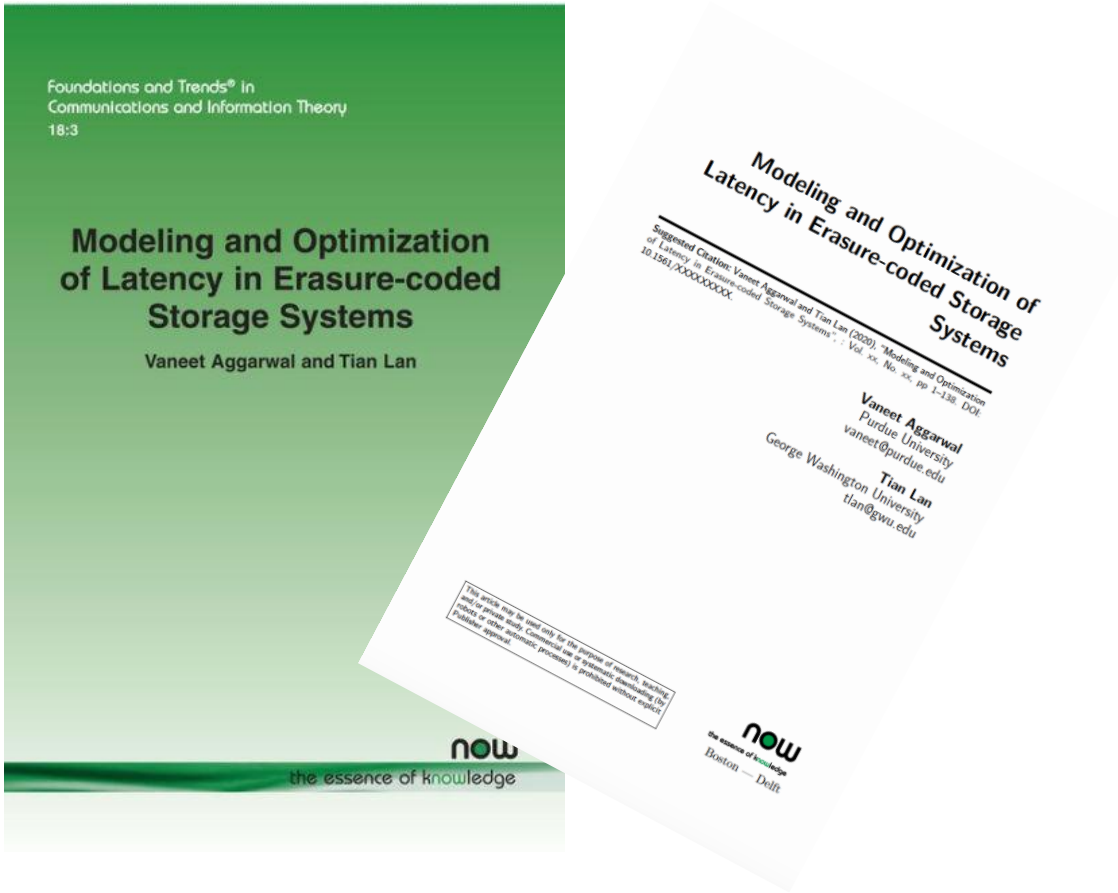


- Shifted Exponential Service Times, 12 servers
- Homogenous files with (12,7) code
- Hyperparameter search for probabilistic scheduling by choosing best of 100 random selections.

- MDS-Reservation and Fork-Join strategies do not achieve the optimal stability region
- Probabilistic scheduling outperforms Fork-Join scheduling for all arrival rates in this simulation

IISc GW UNIVERSITY WASHINGTON DC PURDUE UNIVERSITY

THANK YOU



- Promotion Code: 994513