On Delay-Storage Trade-offs in Content Download from Coded Distributed Storage Systems

Gauri Joshi (MIT)

joint work with

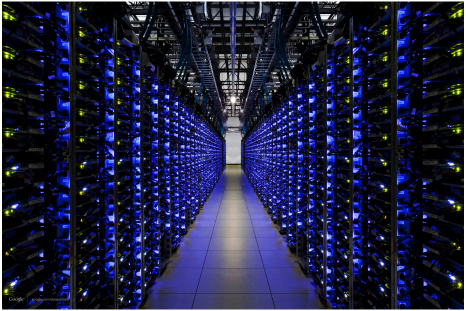
Yanpei Liu (UW-Madison)
Emina Soljanin (Bell Labs)

DIMACS Workshop on Algorithms for Green Data Storage

Why Use Coding in Distributed Storage

Data Centers

- Server clusters that store and process all the data in the Internet
- More than 500000 data centers worldwide
- Consume vast amounts of energy - more than 2% of US electricity
  - Power to run and repair servers, and for cooling systems



Trade-offs in Coding for Distributed Storage

Reliability vs. Storage

- Replication is the most commonly used redundancy
- (n, k) MDS Codes - any k out of n sufficient for data recovery

Repair Bandwidth vs. Storage

- Locally Repairable Codes[Dimakis, IT-Tran '10]
- Regenerative codes for storage [Rashmi, IT-Tran '12]

Trade-offs in Coding for Distributed Storage

Accessibility vs. Storage

- Lower blocking probability than replication for the same storage (Energy Cost) [Ferner, Allerton '12]

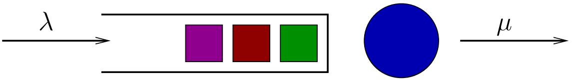
Delay vs. Storage

- Our work - k out of n fork-join queues
- Packet Routing Diversity [Maxemchuk, 1991], [Kabatiansky, 2005] – do not consider queueing
- Redundant requests, MDS queue [Shah, Lee, 2013]

How Coding Reduces Download Time

Single M/M/1 Queue

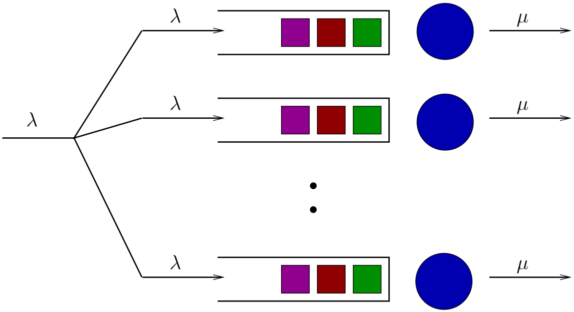
- Requests arrive at rate λ and served at rate μ
- Mean response time $T_{1,1} = \frac{1}{\mu - \lambda}$ for Poisson arrivals and departures



How Coding Reduces Download Time

Multiple Copies give Diversity, but with More Storage

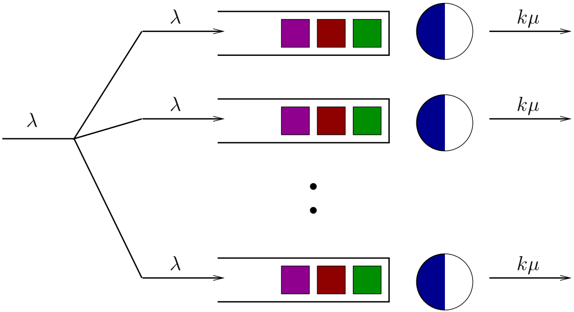
- Requests is sent to n disks storing copies of content
- Need to wait only for download of only one n copies
- Mean response time $T_{n,1} = \frac{1}{n\mu - \lambda}$, but storage increases n-fold



How Coding Reduces Download Time

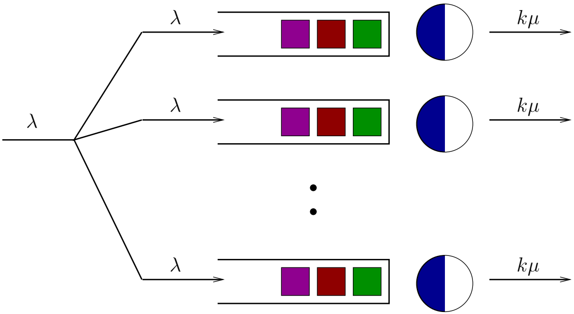
Coding Gives Diversity with Lower Storage

- Content divided into k blocks and encoded to n blocks
- Each disk stores $1/k$ units, so service rate becomes $\mu' = k\mu$
- Downloading any k blocks is sufficient to decode the file



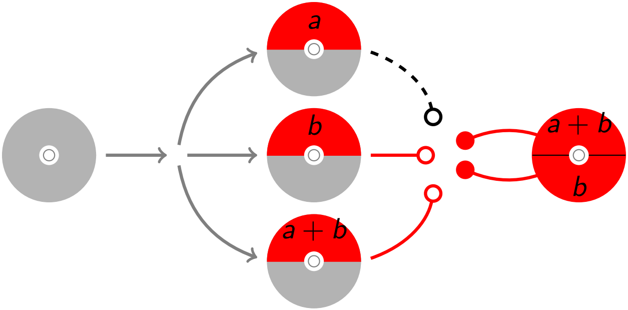
Definition: (n, k) Fork-Join System

- Requests arrivals are Poisson with rate λ
- A request forked into n tasks $\rightarrow$ enter FCFS queues at the n disks
- Time to download one block of content $\sim \exp(\mu')$, where $\mu' = k\mu$
- Load factor $\rho = \lambda/\mu'$ for each queue.



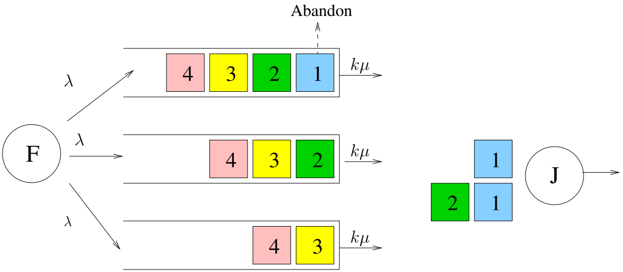
Fork-Join Queues: Example

- A content file of unit size is divided into $k = 2$ blocks, a and b
- Encoded into 3 blocks, a, b and $a + b$
- Downloading any 2 blocks is sufficient to decode the entire file
- Storage is 50% higher, but response time is reduced.



Fork-Join Queues: Example

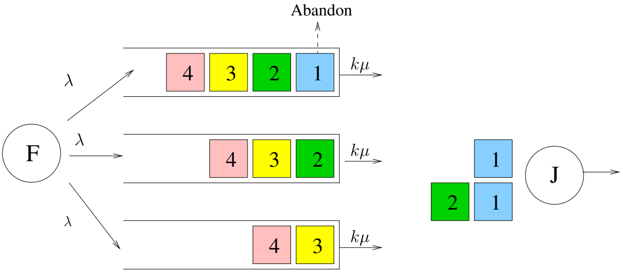
- A content file of unit size is divided into $k = 2$ blocks, a and b
- Encoded into 3 blocks, a, b and $a + b$
- Downloading any 2 blocks is sufficient to decode the entire file
- Storage is 50% higher, but response time is reduced.



Mean Response Time

Challenges

- Arrivals to the n queues are perfectly synchronized.
- Hence it is not the k^{th} order statistic of exponential
- Previous work has attempted finding $T_{n,n}$, but only bounds are known



Our Contributions

- Bounds on mean response time of the (n, k) fork-join system
- Delay-Storage Trade-offs
  - Fixed storage expansion k/n what is the best n?
  - Fixed n disks what is the best k?
- Extensions to correlated service times, (m, n, k) fork-join etc.

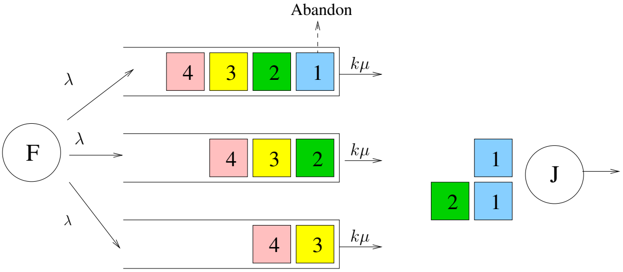
[1] G. Joshi, Y. Liu, E. Soljanin, "Coding for Fast Content Download", Allerton Conference 2012

[2] G. Joshi, Y. Liu, E. Soljanin, "On Delay-Storage Trade-offs in Content Download from Coded Distributed Storage Systems", to appear in JSAC 2014

Upper Bound on Response Time

Comparison with a split-merge system

- Split-merge system - All n queues are blocked until k tasks finish
- Response time of split-merge is always greater than fork-join



Upper Bound on Response Time

- Equivalent to an $M/G/1$ queue
  - Arrivals are Poisson with rate λ
  - Departures according to S, k^{th} order statistic of $\exp(\mu')$

$$\mathrm{E}[S] = \frac{H_n - H_{n-k}}{\mu'}$$

$$\mathrm{V}[S] = \frac{H_{n^2} - H_{(n-k)^2}}{\mu'^2}.$$

- Mean Response time given by the Pollaczek-Khinchin formula,

$$T_{n,k} \leq \mathrm{E}[S] + \frac{\lambda\left(\mathrm{V}[S] + \mathrm{E}[S]^2\right)}{2(1 - \lambda \mathrm{E}[S])}$$

Lower Bound on Response Time

Stages of Processing of a Job

- A job goes through k stages of processing, at stage j, $0 \leq j \leq k-1$
- At stage j, the job has completed j tasks and waiting for the remaining $k-j$
- The service rate of a job in stage j stage is at most $(n-j)\mu'$ [Varki].

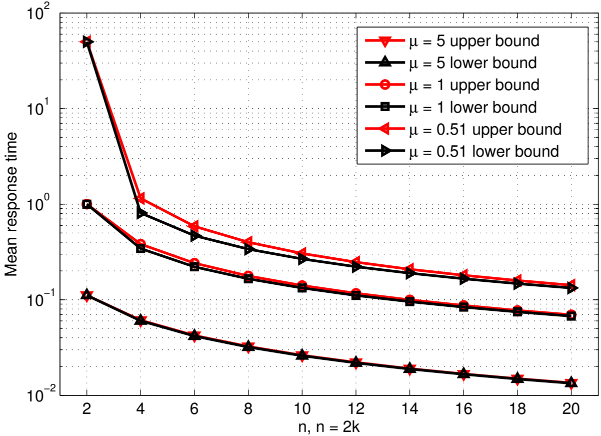
$$T_{n,k} \geq \sum_{j=0}^{k-1} \frac{1}{(n-j)\mu' - \lambda} \quad \text{Sum of response times of } k \text{ stages}$$

$$= \frac{1}{\mu'} \sum_{j=0}^{k-1} \left[\frac{1}{n-j} + \frac{\rho}{(n-j)(n-j-\rho)} \right]$$

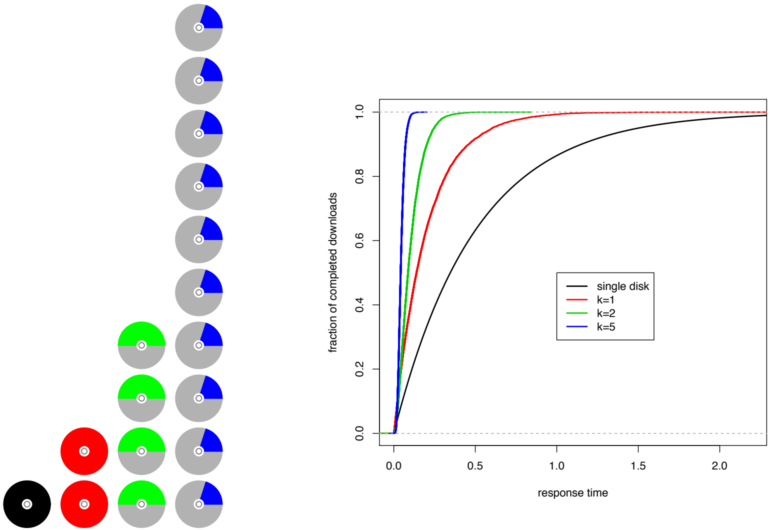
$$= \frac{1}{\mu'} \left[H_n - H_{n-k} + \rho \cdot (H_{n(n-\rho)} - H_{(n-k)(n-k-\rho)}) \right]$$

Flexible Disks, Fixed Storage Expansion

- Parameters: Expansion $k/n = 1/2$, $\lambda = 1$
- **More diversity $\rightarrow$ Lower Response Time**

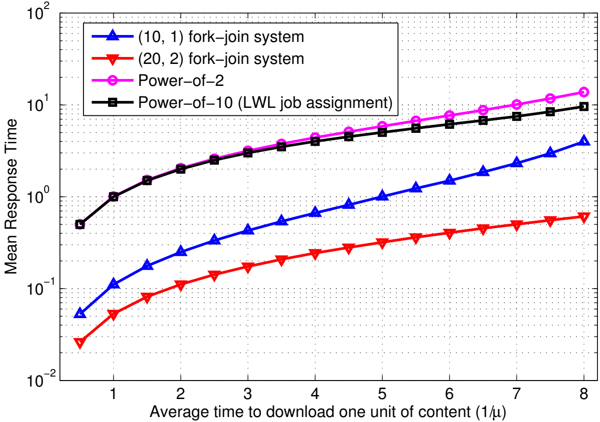


How Much Can Double Storage Improve Completion Time?



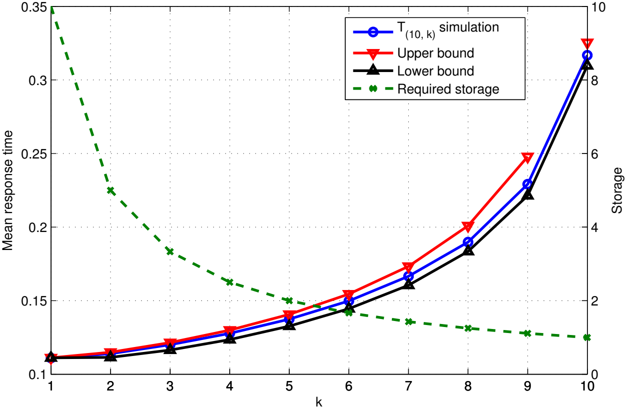
Comparison to Power-of-d

- For **same storage** fork-join gives much faster response

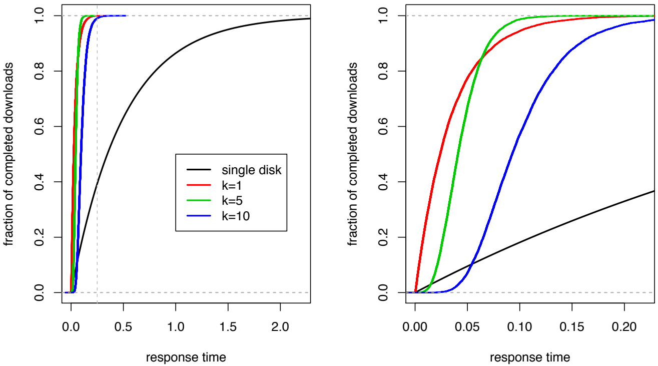


Flexible Storage Expansion, Fixed Disks

- Parameters: $n = 10$, $\lambda = 1$, $\mu = 1$
- **More redundancy → Lower Response Time**



Flexible Storage Expansion, Fixed Disks

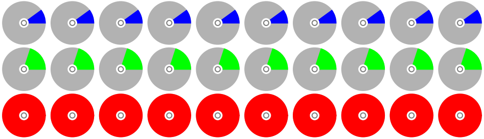


$M/M/1$

$\lambda = 1$

$\mu = 3$

 ← single disk baseline – unit storage



← the same total storage

← double total storage

← $10\times$ increase in storage

Correlated Service Times

- Service time $X = \delta X_d + (1-\delta) X_{r,i}$, for $i = 1, 2, \cdots n$
- More correlation $\rightarrow$ lose the diversity advantage

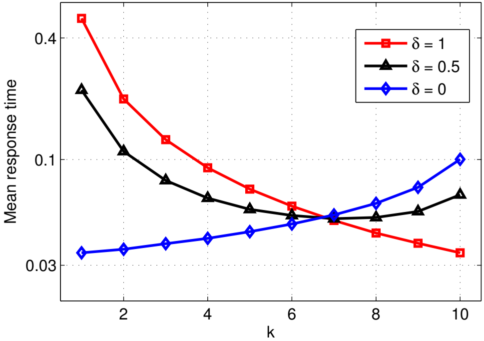
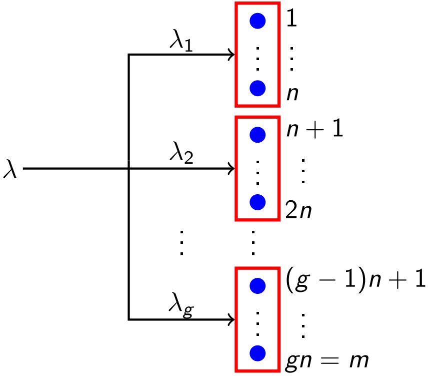


Figure : $\lambda = 1, \mu = 3$

(m,n,k) fork-join system

- Large number of disks $m \gg n$
- Can be divided into $m/n = g$ fork-join systems



(m,n,k) fork-join system

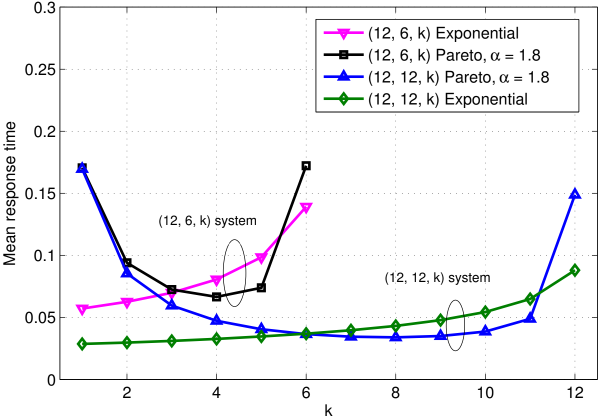


Figure : $\lambda = 1, \mu = 3$

Concluding Remarks

Major Implications

- Investigated the delay-storage trade-off in distributed storage
- Showed that diversity of more disks helps, for same storage space used
- Generalization of (n, n) fork-join systems to the (n, k) fork-join system

Future Perspectives

- Percentile analysis from the CDF of response time
- Extension to parallel **computing** instead of storage