Minimizing delay in distributed service systems with redundancy and random slowdowns

Martin Zubeldia
TU/e & UvA

October 17th, 2019
YEQT XIII
Eindhoven, Netherlands

Motivation: Parallelizable jobs

Example: Matrix-vector multiplication

$$M.v = \begin{bmatrix} — & M_1 & — \\ — & M_2 & — \\ — & M_3 & — \\ — & M_4 & — \end{bmatrix} \cdot v$$

Parallelization: Compute each $M_i.v$ separately!

Motivation: Parallelizable jobs

Example: Matrix-vector multiplication

$$M.v = \begin{bmatrix} — & M_1 & — \\ — & M_2 & — \\ — & M_3 & — \\ — & M_4 & — \end{bmatrix} \cdot v$$

Parallelization: Compute each $M_i.v$ separately!

Problem: Computations may take random time...

Motivation: Parallelizable jobs

Example: Matrix-vector multiplication

$$M.v = \begin{bmatrix} — & M_1 & — \\ — & M_2 & — \\ — & M_3 & — \\ — & M_4 & — \end{bmatrix} \cdot v$$

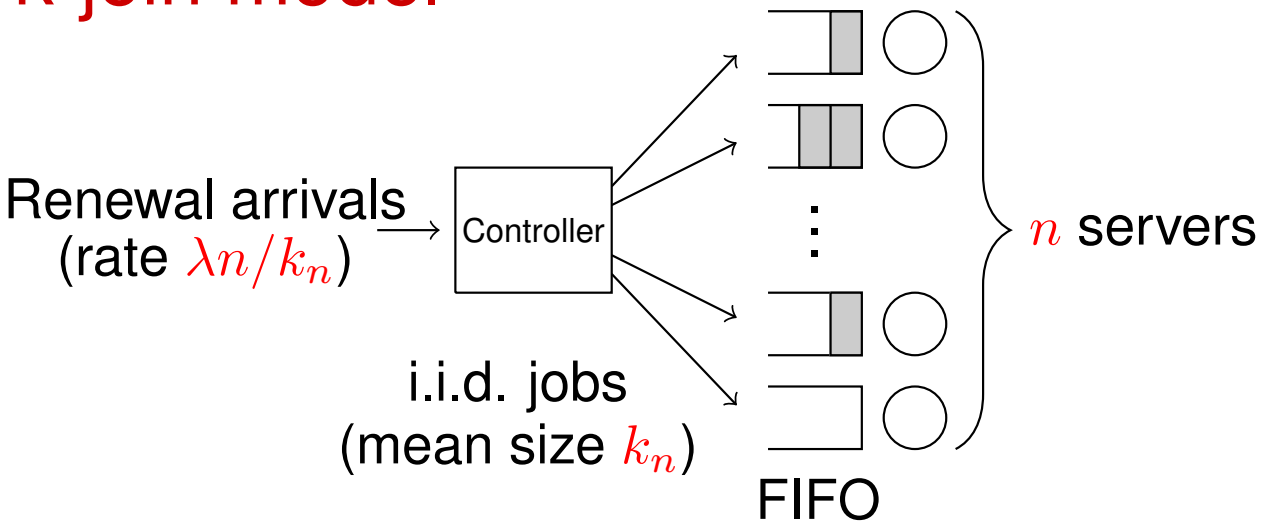
Parallelization: Compute each $M_i.v$ separately!

Problem: Computations may take random time...

Accelarating with redundancy:
We can recover $M.v$ with **any** 4 out of the 5:

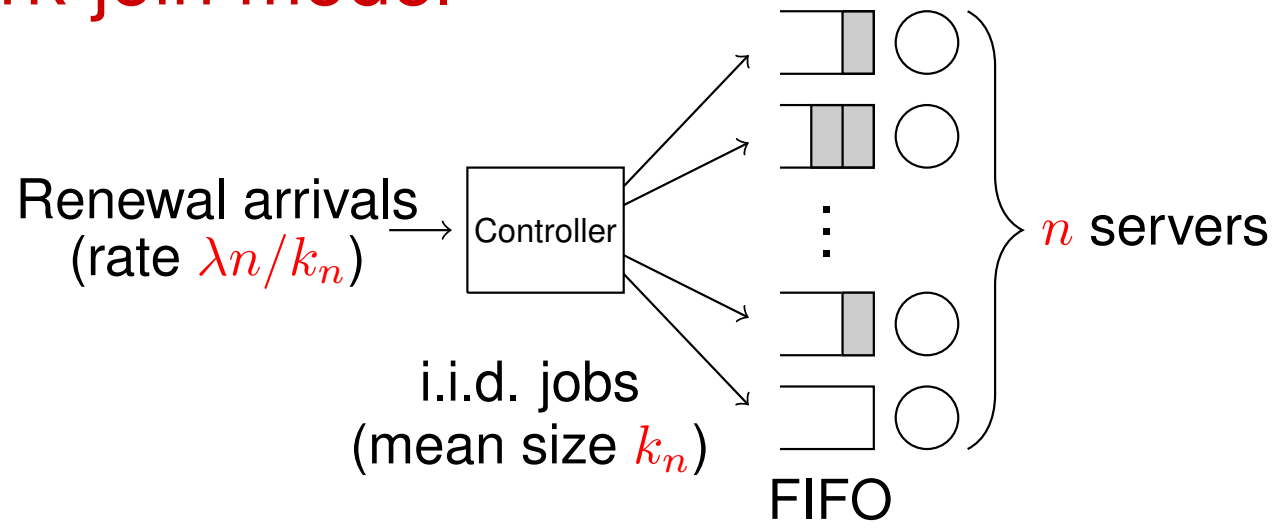
$$M_1.v, \qquad M_2.v, \qquad M_3.v, \qquad M_4.v, \qquad (M_1 + M_2 + M_3 + M_4).v$$

Partial fork-join model



- Jobs consist of k_n tasks of equal size
- A job is done when all task are processed

Partial fork-join model



- Jobs consist of k_n tasks of equal size
- A job is done when all task are processed

Redundancy:

- The k_n tasks can be encoded into any number $r \geq k_n$ of "replicas" so that job is done when **any** k_n of these "replicas" finish
- When any k_n finish, the rest are cancelled inmmedaitely

Server slowdowns

Idea: Servers processing rate fluctuates over time

Server slowdowns

Idea: Servers processing rate fluctuates over time

Service time of i-th replica from j-th job $= X_j\left(1 + S_{j,i}\right)$

- X_j: Size of tasks in job j
- $S_{j,i}$: Random slowdown

Server slowdowns

Idea: Servers processing rate fluctuates over time

Service time of i-th replica from j-th job $= X_j\left(1 + S_{j,i}\right)$

- X_j: Size of tasks in job j
- $S_{j,i}$: Random slowdown

Basic assumptions:

- Conditioned on $X_j = x$, $\{S_{j,i}\}_{i \geq 1}$ are i.i.d.

Server slowdowns

Idea: Servers processing rate fluctuates over time

Service time of i-th replica from j-th job $= X_j\big(1 + S_{j,i}\big)$

- X_j: Size of tasks in job j
- $S_{j,i}$: Random slowdown

Basic assumptions:

- Conditioned on $X_j = x$, $\{S_{j,i}\}_{i\geq 1}$ are i.i.d.
- $\{(S_{j,i} : i \geq 1)\}_{j\geq 1}$ are i.i.d.

Problem and intuition

Objective: Find policies that **minimize the delay of a job** as $n \to \infty$

Problem and intuition

Objective: Find policies that **minimize the delay of a job** as $n \to \infty$

Delay of a job:
Time between arrival, and when **any** k_n of its replicas finish service

Problem and intuition

Objective: Find policies that **minimize the delay of a job** as $n \to \infty$

Delay of a job:
Time between arrival, and when **any** k_n of its replicas finish service
$\Rightarrow$ Is the k_n-th order statistic of the delay of its replicas

Problem and intuition

Objective: Find policies that **minimize the delay of a job** as $n \to \infty$

Delay of a job:
Time between arrival, and when **any** k_n of its replicas finish service
$\Rightarrow$ Is the k_n-th order statistic of the delay of its replicas

Myopic intuition:

- More replicas $\Rightarrow$ less queueing delay

Problem and intuition

Objective: Find policies that **minimize the delay of a job** as $n \to \infty$

Delay of a job:
Time between arrival, and when **any** k_n of its replicas finish service
$\Rightarrow$ Is the k_n-th order statistic of the delay of its replicas

Myopic intuition:

- More replicas $\Rightarrow$ less queueing delay
- More replicas (even without queueing) $\Rightarrow$ smaller service times

Problem and intuition

Objective: Find policies that **minimize the delay of a job** as $n \to \infty$

Delay of a job:
Time between arrival, and when **any** k_n of its replicas finish service
$\Rightarrow$ Is the k_n-th order statistic of the delay of its replicas

Myopic intuition:

- More replicas $\Rightarrow$ less queueing delay
- More replicas (even without queueing) $\Rightarrow$ smaller service times

BUT: More replicas $\Rightarrow$ higher load?

Practical issues

- How many replicas are too many?

Practical issues

- How many replicas are too many?
- Adapt replication level to congestion?

Practical issues

- How many replicas are too many?
- Adapt replication level to congestion?
- More replicas if it's taking too long?

Practical issues

- How many replicas are too many?
- Adapt replication level to congestion?
- More replicas if it's taking too long?
- Bigger jobs replicated more or less than smaller ones?

Practical issues

- How many replicas are too many?
- Adapt replication level to congestion?
- More replicas if it's taking too long?
- Bigger jobs replicated more or less than smaller ones?

Today: Answers for two different models for the slowdowns

Independent exponential slowdowns

Independent exponential slowdowns

What the data says:

Download times from AWS' servers are shifted exponential r.v.

Independent exponential slowdowns

What the data says:

Download times from AWS' servers are shifted exponential r.v.

Recall: Service time of i-th replica from j-th job $= X_j\left(1 + S_{j,i}\right)$

- X_j: Size of tasks in job j
- $S_{j,i}$: Random slowdown

Independent exponential slowdowns

What the data says:

Download times from AWS' servers are shifted exponential r.v.

Recall: Service time of i-th replica from j-th job $= X_j\left(1 + S_{j,i}\right)$

- X_j: Size of tasks in job j
- $S_{j,i}$: Random slowdown

Assumptions:

- $\{S_{j,i}\}_{j,i \geq 1}$ are i.i.d.

Independent exponential slowdowns

What the data says:

Download times from AWS' servers are shifted exponential r.v.

Recall: Service time of i-th replica from j-th job $= X_j\big(1 + S_{j,i}\big)$

- X_j: Size of tasks in job j
- $S_{j,i}$: Random slowdown

Assumptions:

- $\{S_{j,i}\}_{j,i\geq 1}$ are i.i.d.
- They are exponential with mean $1/\mu$

Independent exponential slowdowns

What the data says:
Download times from AWS' servers are shifted exponential r.v.

Recall: Service time of i-th replica from j-th job $= X_j\big(1 + S_{j,i}\big)$

- X_j: Size of tasks in job j
- $S_{j,i}$: Random slowdown

Assumptions:

- $\{S_{j,i}\}_{j,i\geq 1}$ are i.i.d.
- They are exponential with mean $1/\mu$

Note: This is a special case of the S&X model [Gardner et al. '16]

Universal delay lower bound

Controller:

- Knows full queue state (with elapsed times)

Universal delay lower bound

Controller:

- Knows full queue state (with elapsed times)
- Can create “replicas” at any time

Universal delay lower bound

Controller:

- Knows full queue state (with elapsed times)
- Can create “replicas” at any time
- Cannot (prematurely) cancel replicas in service

Universal delay lower bound

Controller:

- Knows full queue state (with elapsed times)
- Can create “replicas” at any time
- Cannot (prematurely) cancel replicas in service

Theorem

There exists a stable policy $\Leftrightarrow r^* \triangleq \frac{1}{\lambda} - \frac{1}{\mu} > 1$

Universal delay lower bound

Controller:

- Knows full queue state (with elapsed times)
- Can create “replicas” at any time
- Cannot (prematurely) cancel replicas in service

Theorem

There exists a stable policy $\Leftrightarrow r^* \triangleq \frac{1}{\lambda} - \frac{1}{\mu} > 1$

Theorem

For any stable policy:

$$\mathbb{E}[\text{Delay}(n)] \geq 1 + \frac{1}{\mu} \sum_{i=1}^{k_n} \frac{1}{k_n r^* - i + 1}, \qquad \forall\, n$$

Universal delay lower bound

Controller:

- Knows full queue state (with elapsed times)
- Can create "replicas" at any time
- Cannot (prematurely) cancel replicas in service

Theorem

There exists a stable policy $\Leftrightarrow r^* \triangleq \frac{1}{\lambda} - \frac{1}{\mu} > 1$

Theorem

For any stable policy:

$$\mathbb{E}[\text{Delay}(n)] \geq 1 + \frac{1}{\mu} \sum_{i=1}^{k_n} \frac{1}{k_n r^* - i + 1}, \qquad \forall\, n$$

Important observation: If $r^* k_n$ is an integer, the lower bound is the expectation of the k_n-th order statistic of $r^* k_n$ service times of replicas!

Towards an asymptotically optimal policy

Straightforward policy:

- Create $r^* k_n$ replicas per job
- Send them to idle servers

Towards an asymptotically optimal policy

Straightforward policy:

- Create $r^* k_n$ replicas per job
- Send them to idle servers

Not so easy:

- $r^* k_n$ might not be an integer
- Creating $r^* k_n$ replicas per job destabilizes system ($\rho = 1$)

Towards an asymptotically optimal policy

Straightforward policy:

- Create $r^* k_n$ replicas per job
- Send them to idle servers

Not so easy:

- $r^* k_n$ might not be an integer
- Creating $r^* k_n$ replicas per job destabilizes system ($\rho = 1$)

Solution:

- Create a mixture of $\lceil r^* k_n \rceil$ and $\lceil r^* k_n \rceil - 1$ replicas
- Create slightly less than $r^* k_n$ replicas per job (heavy-traffic)
- If possible, send them to idle servers

Delay performance (case $k_n = 1$)

Theorem

If $k_n = 1$ for all n, and $\mathbb{E}\left[X^{2+\epsilon}\right] < \infty$ for some $\epsilon > 0$, then

$$\lim_{n\to\infty} \mathbb{E}[\text{Delay}(n)] = 1 + \frac{1}{\mu}\left(\frac{p}{\lceil r^* \rceil} + \frac{1-p}{\lceil r^* \rceil - 1}\right)$$

where $p \in (0, 1]$ is such that $p\lceil r^* \rceil + (1-p)(\lceil r^* \rceil - 1) = r^*$.

Delay performance (case $k_n = 1$)

Theorem

If $k_n = 1$ for all n, and $\mathbb{E}\left[X^{2+\epsilon}\right] < \infty$ for some $\epsilon > 0$, then

$$\lim_{n\to\infty} \mathbb{E}[\text{Delay}(n)] = 1 + \frac{1}{\mu}\left(\frac{p}{\lceil r^* \rceil} + \frac{1-p}{\lceil r^* \rceil - 1}\right)$$

where $p \in (0, 1]$ is such that $p\lceil r^* \rceil + (1-p)(\lceil r^* \rceil - 1) = r^*$.

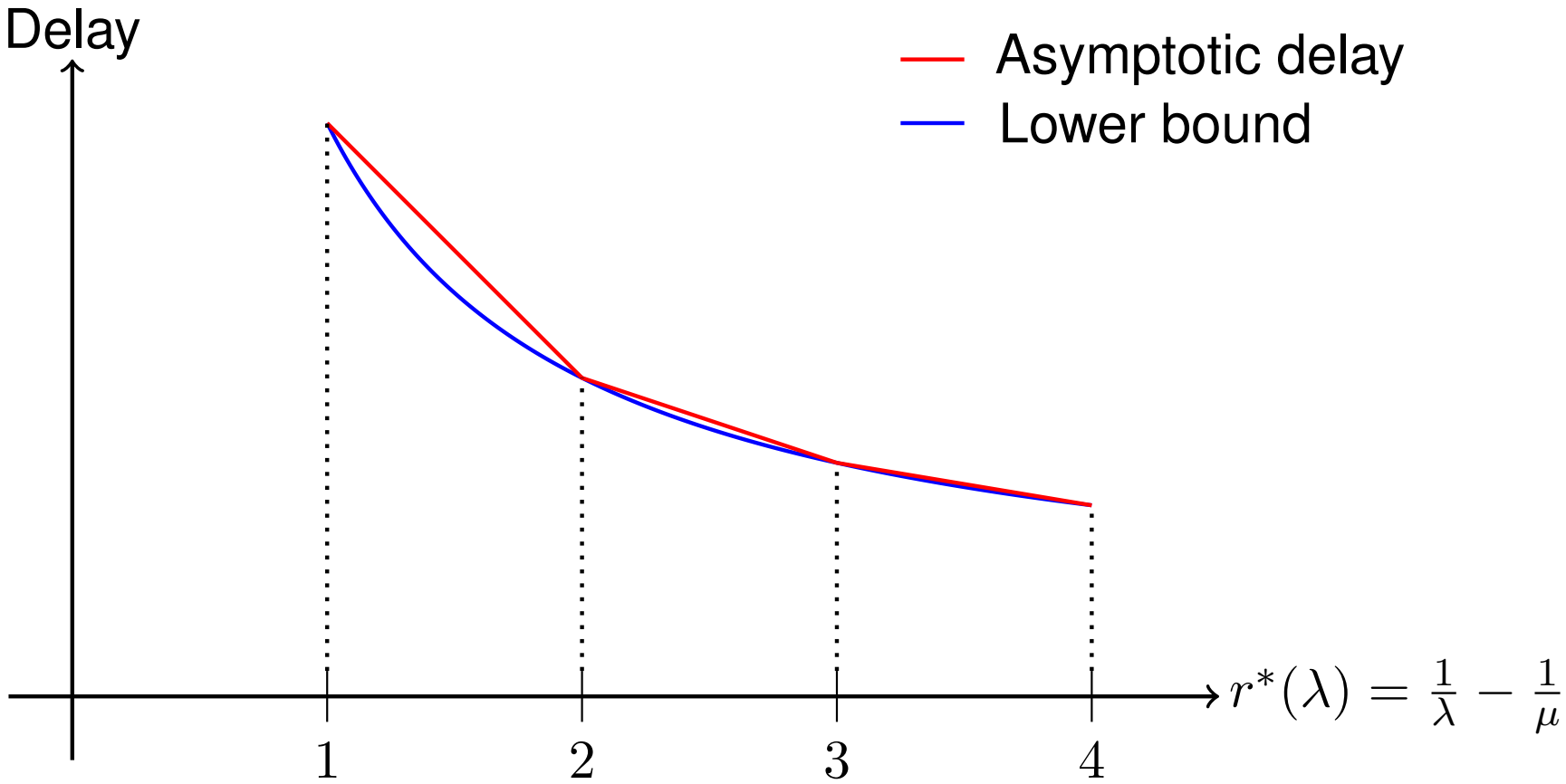
Corollary of lower bound

If $k_n = 1$ for all n, for any stable policy we have

$$\mathbb{E}[\text{Delay}(n)] \geq 1 + \frac{1}{\mu r^*}$$

Note: Only coincides with lower bound if r^* is an integer!

Delay performance (case $k_n = 1$)



Delay performance (case $k_n \to \infty$)

Theorem

If $k_n \in o\left(n^{1/5}\right)$, arrivals are Poisson, and $\mathbb{E}\left[X^{2+\epsilon}\right] < \infty$ for some $\epsilon > 0$, then

$$\lim_{n\to\infty} \mathbb{E}[\text{Service time}(n)] = 1 + \frac{1}{\mu} \log\left(\frac{r^*}{r^* - 1}\right)$$

and

$$\lim_{n\to\infty} \mathbb{P}(\text{Queueing delay}(n) > 0) = 0$$

Delay performance (case $k_n \to \infty$)

Theorem

If $k_n \in o\left(n^{1/5}\right)$, arrivals are Poisson, and $\mathbb{E}\left[X^{2+\epsilon}\right] < \infty$ for some $\epsilon > 0$, then

$$\lim_{n\to\infty} \mathbb{E}[\text{Service time}(n)] = 1 + \frac{1}{\mu} \log\left(\frac{r^*}{r^* - 1}\right)$$

and

$$\lim_{n\to\infty} \mathbb{P}(\text{Queueing delay}(n) > 0) = 0$$

Corollary of lower bound

If $k_n \to \infty$, for any sequence of stable policies we have:

$$\liminf_{n\to\infty} \left(\mathbb{E}[\text{Delay}(n)]\right) \geq 1 + \frac{1}{\mu} \log\left(\frac{r^*}{r^* - 1}\right)$$

Note: Always coincides with lower bound!

Practical issues

- How many replicas are too many?
  r^* per task

Practical issues

- How many replicas are too many?
  r^* per task

- Adapt replication level to congestion?
  No

Practical issues

- How many replicas are too many?
  r^* per task
- Adapt replication level to congestion?
  No
- More replicas if it's taking too long?
  No

Practical issues

- How many replicas are too many?
  r^* per task
- Adapt replication level to congestion?
  No
- More replicas if it's taking too long?
  No
- Bigger jobs replicated more or less than smaller ones?
  No

Size-based slowdowns

A different setting

What the data says: Processing times in Google's data centers:

- Big tasks see small variability
- Small tasks see high variability

A different setting

What the data says: Processing times in Google's data centers:

- Big tasks see small variability
- Small tasks see high variability

Recall: Service time of a replica $= X(1 + S)$

- X: Size
- S: Random slowdown

A different setting

What the data says: Processing times in Google's data centers:

- Big tasks see small variability
- Small tasks see high variability

Recall: Service time of a replica $= X(1 + S)$

- X: Size
- S: Random slowdown

$\Rightarrow$ Variance of S should decrease with X

Size-based slowdowns

Recall: Service time of a replica $= X(1 + S)$

Size-based slowdowns

Recall: Service time of a replica $= X(1 + S)$

- Conditioned on $X = x$, slowdowns have mean $1/\mu$

Size-based slowdowns

Recall: Service time of a replica $= X(1+S)$

- Conditioned on $X = x$, slowdowns have mean $1/\mu$
- Averaging effect:

$$\lim_{x\to\infty} \mathbb{E}\left[S_{[i:r]} \middle| X = x\right] = \frac{1}{\mu}$$

for all $i \leq r$

Size-based slowdowns

Recall: Service time of a replica $= X(1+S)$

- Conditioned on $X = x$, slowdowns have mean $1/\mu$
- Averaging effect:

$$\lim_{x\to\infty} \mathbb{E}\left[S_{[i:r]}\middle|X=x\right] = \frac{1}{\mu}$$

  for all $i \leq r$
- Diminishing returns:

$$\mathbb{E}\left[S_{[i:r]} - S_{[i:r+1]}\middle|X=x\right]$$

  is decreasing in x, for all $i \leq r$

Keeping it tractable

Objective: Find policy that minimizes expected delay

Keeping it tractable

Objective: Find policy that minimizes expected delay

Problem:
General slowdowns $+$ Delayed service start times $=$ Intractable!

Keeping it tractable

Objective: Find policy that minimizes expected delay

Problem:
General slowdowns $+$ Delayed service start times $=$ Intractable!

Assumption:
Replicas associated with the same job start service at the same time

Delay lower bound

Consequence: Service time is determined by $\{p(x) : x \geq 0\}$, where

$$p_r(x) \triangleq \mathbb{P}\big(r \text{ replicas start service} \big| X = x\big)$$

Delay lower bound

Consequence: Service time is determined by $\{p(x) : x \geq 0\}$, where

$$p_r(x) \triangleq \mathbb{P}\big(r \text{ replicas start service} \big| X = x\big)$$

Lemma

Expected delay is lower bounded by:

$$\inf_{p} \quad \underbrace{\int_0^\infty x \sum_{r=k_n}^{\infty} p_r(x) \left(1+\mathbb{E}\left[S_{[k_n:r]} \middle| X=x\right]\right) d\mathbb{P}_X(x)}_{\text{EXPECTED SERVICE TIME}}$$

$$s.t. \quad \underbrace{\frac{\lambda n}{k_n} \int_0^\infty x \sum_{r=k_n}^{\infty} p_r(x) \left[r+(r-k_n)\mathbb{E}\left[S_{[k_n:r]} \middle| X=x\right]+\sum_{i=1}^{k_n} \mathbb{E}\left[S_{[i:r]} \middle| X=x\right]\right] d\mathbb{P}_X(x)}_{\text{EXPECTED NUMBER OF BUSY SERVERS}} \leq n$$

Asymptotically optimal policy

Lemma

The infimum of the optimization problem is attained for all λ sufficiently small

Asymptotically optimal policy

Lemma

The infimum of the optimization problem is attained for all λ sufficiently small

- Fix $\alpha \in \left(\frac{1}{2}, 1\right)$. Let $p^{(n)}$ be a solution of the optimization problem with up to $n - n^{\alpha}$ busy servers

Asymptotically optimal policy

Lemma

The infimum of the optimization problem is attained for all λ sufficiently small

- Fix $\alpha \in \left(\frac{1}{2}, 1\right)$. Let $p^{(n)}$ be a solution of the optimization problem with up to $n - n^{\alpha}$ busy servers
- If $X = x$, create r replicas with probability $p_r^{(n)}(x)$

Asymptotically optimal policy

Lemma

The infimum of the optimization problem is attained for all λ sufficiently small

- Fix $\alpha \in \left(\frac{1}{2}, 1\right)$. Let $p^{(n)}$ be a solution of the optimization problem with up to $n - n^{\alpha}$ busy servers
- If $X = x$, create r replicas with probability $p_r^{(n)}(x)$
- If possible, send them to idle servers

Asymptotic delay performance

Theorem (asymptotic delay for $k_n = 1$)

If $k_n = 1$ for all n, and $\mathbb{E}[(XS)^{2+\epsilon}] < \infty$ for some $\epsilon > 0$, then

$$\lim_{n\to\infty} \mathbb{E}[\text{Delay}(n)] = \text{Optimal!}$$

Theorem (asymptotic delay for $k_n = k > 1$)

If arrivals are Poisson, and $\mathbb{E}\left[(X \max\{S_1, \ldots, S_k\})^{2+\epsilon}\right] < \infty$ for some $\epsilon > 0$, then

$$\lim_{n\to\infty} \mathbb{E}[\text{Service time}(n)] = \text{Optimal!}$$

and

$$\lim_{n\to\infty} \mathbb{P}(\text{Queueing delay}(n) > 0) = 0$$

Structural properties of solutions

Theorem

If p^* minimizes expected delay, then under p^*:

- Only k_n replicas are made for all x large enough

Structural properties of solutions

Theorem

If p^* minimizes expected delay, then under p^*:

- Only k_n replicas are made for all x large enough
- No more than $\overline{r} \in \Theta(k_n)$ replicas are made

Structural properties of solutions

Theorem

If p^* minimizes expected delay, then under p^*:

- Only k_n replicas are made for all x large enough
- No more than $\overline{r} \in \Theta(k_n)$ replicas are made

Theorem (case $k_n = 1$)

If p^* minimizes expected delay, then:

- $p^*(x)$ is concentrated in up to two consecutive integers

Structural properties of solutions

Theorem

If p^* minimizes expected delay, then under p^*:

- Only k_n replicas are made for all x large enough
- No more than $\overline{r} \in \Theta(k_n)$ replicas are made

Theorem (case $k_n = 1$)

If p^* minimizes expected delay, then:

- $p^*(x)$ is concentrated in up to two consecutive integers
- The expected number of replicas created

$$\sum_{r=k_n}^{\infty} r p_r^*(x)$$

is non-increasing with x

Practical issues

- How many replicas are too many?
  $\Theta(k_n)$ per job, depending on slowdown distributions

Practical issues

- How many replicas are too many?
  $\Theta(k_n)$ per job, depending on slowdown distributions

- Adapt replication level to congestion?
  No

Practical issues

- How many replicas are too many?
  $\Theta(k_n)$ per job, depending on slowdown distributions
- Adapt replication level to congestion?
  No
- More replicas if it's taking too long?
  Maybe

Practical issues

- How many replicas are too many?
  $\Theta(k_n)$ per job, depending on slowdown distributions
- Adapt replication level to congestion?
  No
- More replicas if it's taking too long?
  Maybe
- Bigger jobs replicated more or less than smaller ones?
  Less

Conclusions

- Forced heavy-traffic is the way to go

Conclusions

- Forced heavy-traffic is the way to go
- No need to adapt to congestion

Conclusions

- Forced heavy-traffic is the way to go
- No need to adapt to congestion
- Job size distributions are mostly irrelevant

Conclusions

- Forced heavy-traffic is the way to go
- No need to adapt to congestion
- Job size distributions are mostly irrelevant
- Having the correct assumptions on the slowdowns is key!

Thank you!

Questions?