Modeling of Request Cloning in Cloud Server Systems using Processor Sharing

Tommi Nylander, Johan Ruuskanen, Karl-Erik Årzén, Martina Maggio

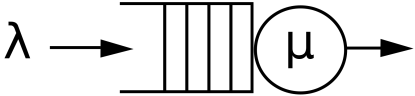
Department of Automatic Control,
Lund Univeristy

11th ACM/SPEC International Conference on Performance Engineering
April 20-24, 2020

Cloud Applications

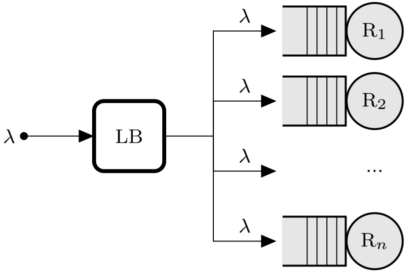
- Compute capacity and memory for sale
- Virtual machines
- Docker containers
- Cloud users host applications
  - Example: Online store
  - Compute capacity costs
  - End-users want low latency
- **Use resources efficiently!**

Queuing Theory: Single Server



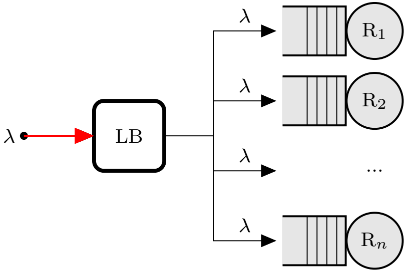
- Represents e.g. compute capacity of a virtual machine
- Modeled using statistics and queuing theory
- Request arrival rate $\lambda \in F_{arr}$
  - Captures user behavior
- Request service rate $\mu \in F_{ser}$
  - Captures request size and server behavior
- Queuing disciplines:
  - First Come First Serve (FCFS)
  - **Processor Sharing (PS)**
- Allows us to analyze response times and utilizations

Request Cloning in Cloud Data Centers



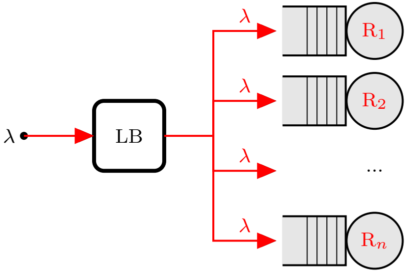
- Each request is sent to n servers simultaneously
- Fastest response is used – all other requests are canceled

Request Cloning in Cloud Data Centers



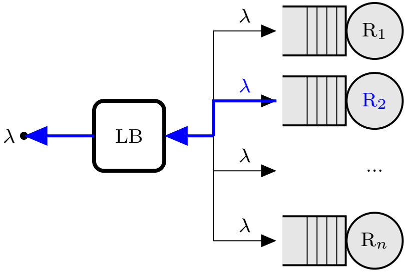
- Each request is sent to n servers simultaneously
- Fastest response is used – all other requests are canceled

Request Cloning in Cloud Data Centers



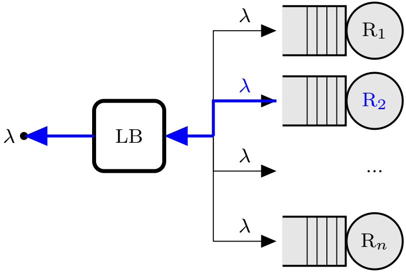
- Each request is sent to n servers simultaneously
- Fastest response is used – all other requests are canceled

Request Cloning in Cloud Data Centers



- Each request is sent to n servers simultaneously
- Fastest response is used – all other requests are canceled

Request Cloning in Cloud Data Centers



- Each request is sent to n servers simultaneously
- Fastest response is used – all other requests are canceled
  - Cancel-on-Complete cloning

Why?

- Cloud server systems inherently stochastic
- Reduce mean and tail latencies

Previous Work

- Shown to work well in practice (2010 –)
- First theoretical analysis in 2015 – for exponential distributions
- Some properties for more general service time distributions
- Still, all publications so far have very restrictive assumptions
  - Independent and identically distributed service times
  - Only for queuing discipline FCFS
- In-depth related work available in our paper

Contributions

- We generalize the current cloning models to allow for
  - Heterogeneity
  - Dependencies
  - Any queuing discipline
- We formalize the enabling *synchronized service* criterion
- We analyze synchronized server systems under PS
- We relax the synchronized service assumption:
  - Bounds for arrival and cancellation delays
  - Approximate model for Join-Shortest-Queue (JSQ)
- We evaluate our claims using a discrete-event simulator
  - Artifact including simulator code available at `https://doi.org/10.5281/zenodo.3635905`

The Distribution of the Minimum

- 2 samples drawn from distributions X and Y
- The minimum is always chosen
- Then, the CDF of the minimum F_{min} is obtained by:

$$F_{\text{min}}(x) = F_X(x) + F_Y(x) - F_{X,Y}(x)$$

The Distribution of the Minimum

- 2 samples drawn from distributions X and Y
- The minimum is always chosen
- Then, the CDF of the minimum F_{min} is obtained by:

$$F_{\text{min}}(x) = F_X(x) + F_Y(x) - F_{X,Y}(x)$$

- If X and Y are independent, then

$$F_{\text{min}}(x) = 1 - (1 - F_X(x))(1 - F_Y(x))$$

The Distribution of the Minimum

- 2 samples drawn from distributions X and Y
- The minimum is always chosen
- Then, the CDF of the minimum F_{min} is obtained by:

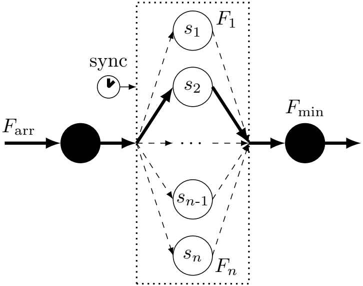
$$F_{\text{min}}(x) = F_X(x) + F_Y(x) - F_{X,Y}(x)$$

- If X and Y are independent, then

$$F_{\text{min}}(x) = 1 - (1 - F_X(x))\,(1 - F_Y(x))$$

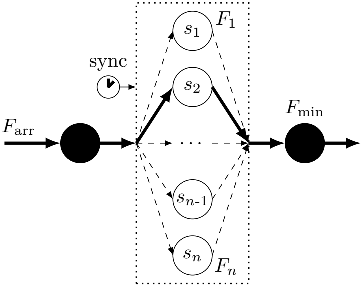
- Can be generalized for any n number of samples

Our Cloning Model I



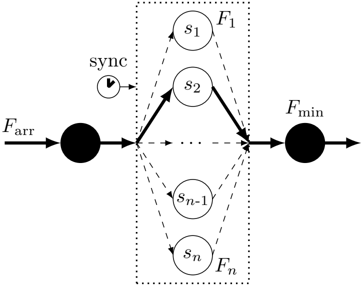
- Assumption: All clones have to receive *synchronized service*
  - Requests enter service simultaneously
  - Requests leave service simultaneously
  - Requires perfect cancellation
- Fastest response ⇔ shortest service time

Our Cloning Model I



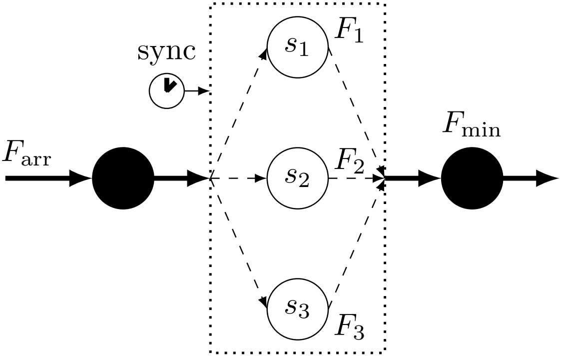
- Assumption: All clones have to receive *synchronized service*
  - Requests enter service simultaneously
  - Requests leave service simultaneously
  - Requires perfect cancellation
- Fastest response ⇔ shortest service time
- **Allows us to use minimum distribution theorem!**

Our Cloning Model II



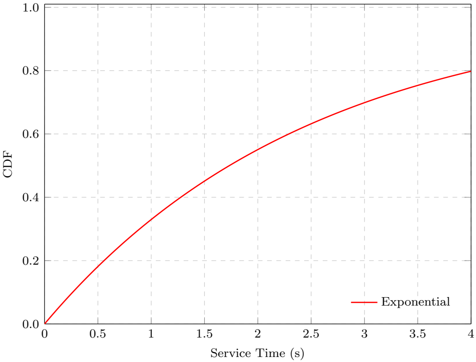
- Cloned system equivalent to **one** server with
  - Arrival distribution F_{arr}
  - Service time distribution $F_{\text{min}}(x)$
- No assumptions on service time or arrival distributions!
  - Allows for heterogeneity and dependencies
- No assumptions on queuing discipline!
  - But we focus on processor sharing (PS)

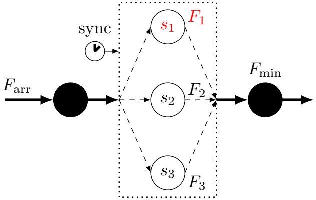
Model Example



- Synchronized service is assumed
- 3 independent heterogeneous servers
- All requests cloned to all servers

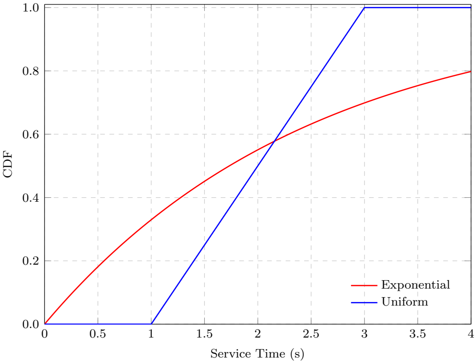
Model Example

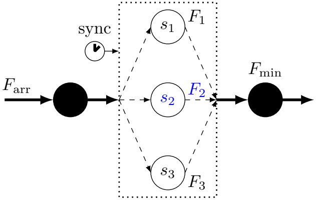




- Exponential: $\text{CDF}_{\text{exp}} = 1 - e^{-x/2.5}$

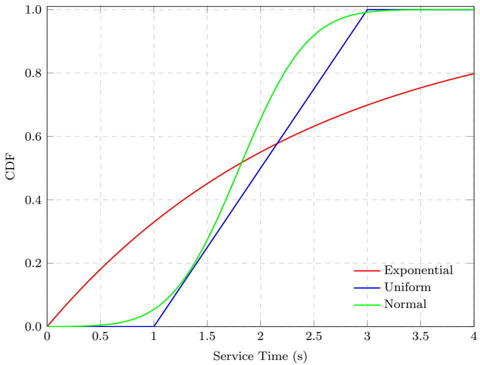
Model Example

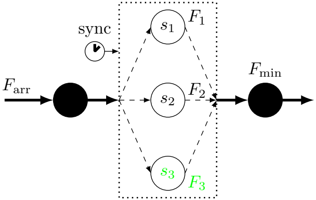




- Uniform: $\text{CDF}_{\text{uni}} = \frac{x-1}{2}, 1 \leq x \leq 3$

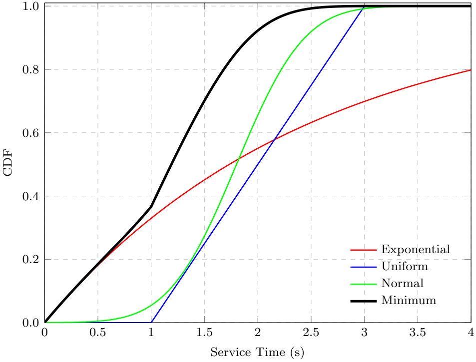
Model Example

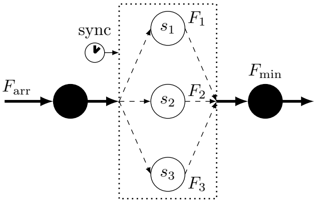




- Normal: $\mathrm{CDF}_{\mathrm{norm}} = \frac{1}{2}\left(1 + \mathrm{erf}\left(\frac{x - 1.8}{0.5\sqrt{2}}\right)\right), \quad x \geq 0$

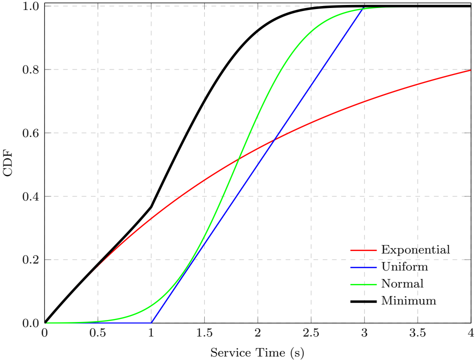
Model Example

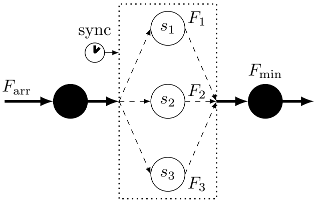




- Min: $\mathsf{CDF}_{\mathsf{min}} = 1 - \left(1 - \mathsf{CDF}_{\mathsf{exp}}\right)\left(1 - \mathsf{CDF}_{\mathsf{uni}}\right)\left(1 - \mathsf{CDF}_{\mathsf{norm}}\right)$

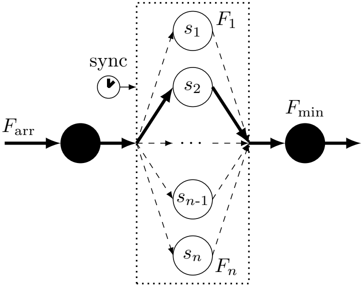
Model Example





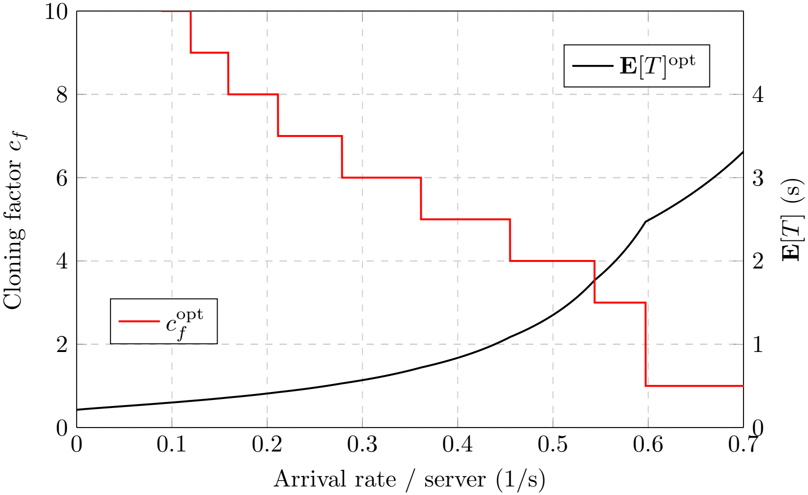
- Min: $\text{CDF}_{\text{min}} = 1 - \left(1 - \text{CDF}_{\text{exp}}\right)\left(1 - \text{CDF}_{\text{uni}}\right)\left(1 - \text{CDF}_{\text{norm}}\right)$
- **But what can this be used for?**

Clone-to-All: Performance Analysis



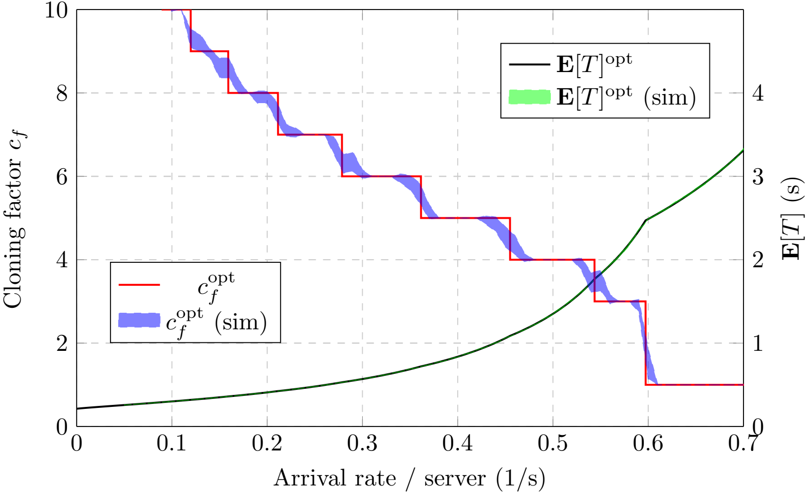
- Build G/G/1 model with F_{arr} and $F_{\min}$ from any cloning factor
- Analyze stability: $\dfrac{\lambda}{\mu} < 1$
- Analyze performance: $\mathbf{E}(T) = \dfrac{1}{\mu - \lambda}$ (For PS and Poisson arr.)
- Determine **optimal cloning factor** under any system load

Clone-to-All: Example



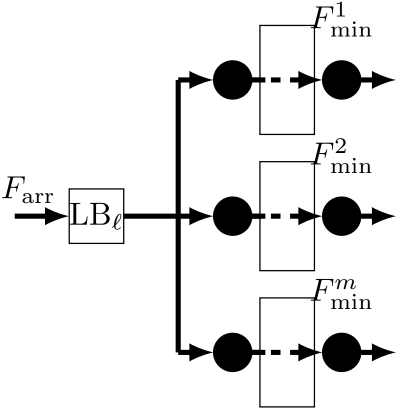
- Theoretical results for an example distribution

Clone-to-All: Example



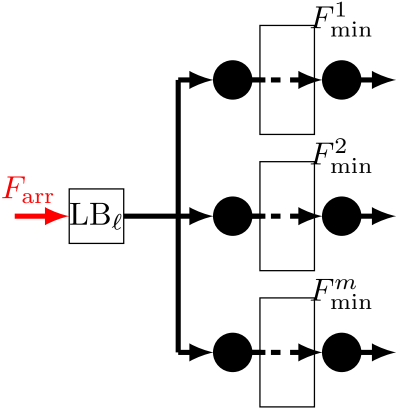
- Results comparison for an example distribution

Clone-to-Clusters



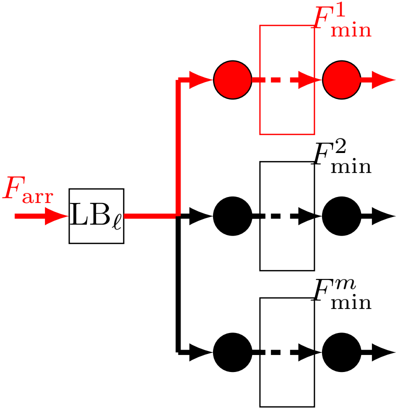
- Requests cloned within each cluster with cloning factor c_f
- LB strategy ℓ chooses the cluster to send original request to
- Co-design of ℓ and c_f necessary!

Clone-to-Clusters



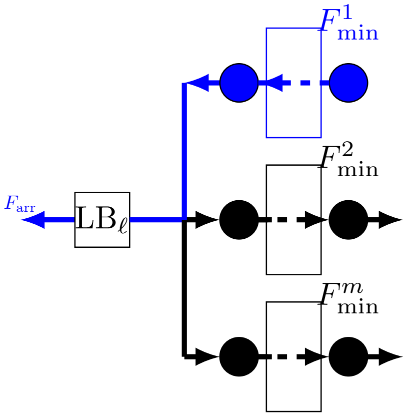
- Requests cloned within each cluster with cloning factor c_f
- LB strategy ℓ chooses the cluster to send original request to
- Co-design of ℓ and c_f necessary!

Clone-to-Clusters



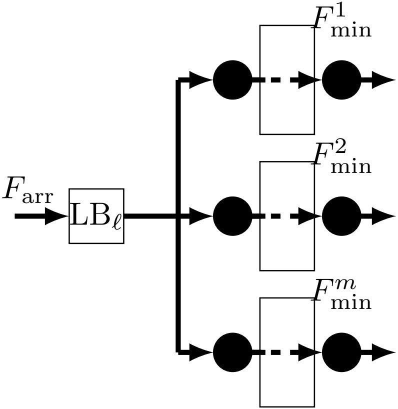
- Requests cloned within each cluster with cloning factor c_f
- LB strategy ℓ chooses the cluster to send original request to
- Co-design of ℓ and c_f necessary!

Clone-to-Clusters



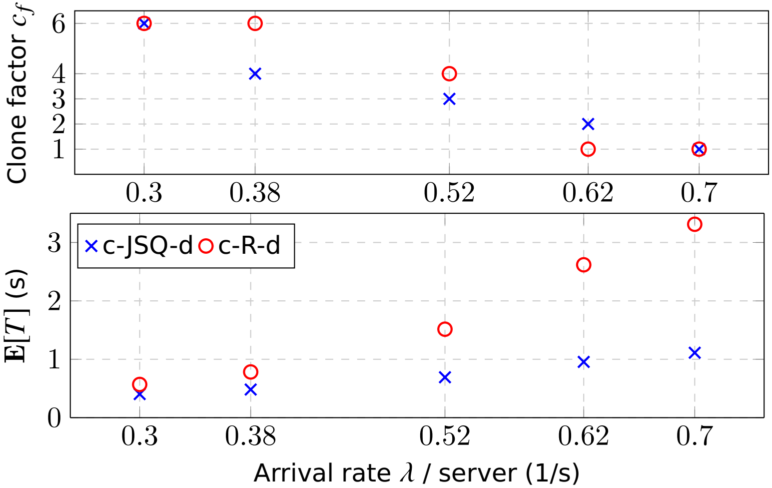
- Requests cloned within each cluster with cloning factor c_f
- LB strategy ℓ chooses the cluster to send original request to
- Co-design of ℓ and c_f necessary!

Clone-to-Clusters



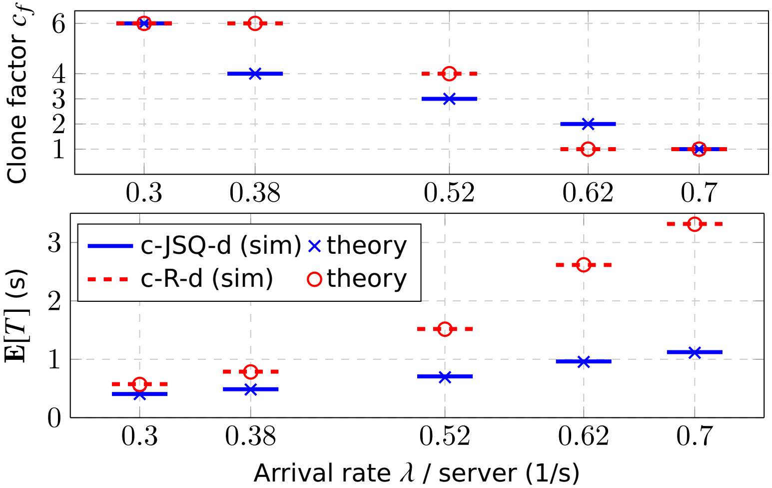
- Requests cloned within each cluster with cloning factor c_f
- LB strategy ℓ chooses the cluster to send original request to
- Co-design of ℓ and c_f necessary!

Clone-to-Clusters: Example



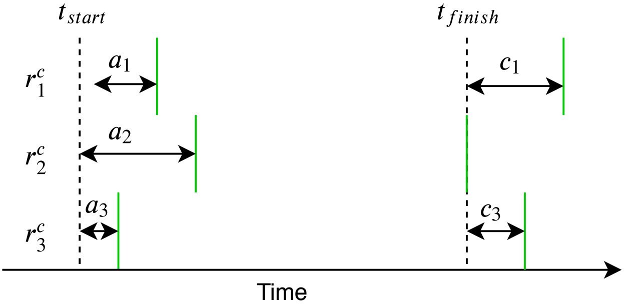
- Theoretical co-design results for an example distribution

Clone-to-Clusters: Example



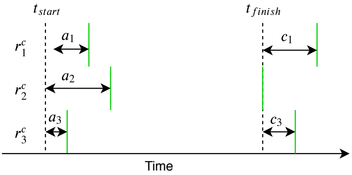
- Co-design results comparison for an example distribution

Arrival and Cancellation Delays I



- Arrival a_i and cancellation c_i delays present
- Synchronized service no longer guaranteed!
- All errors are reset when servers become empty

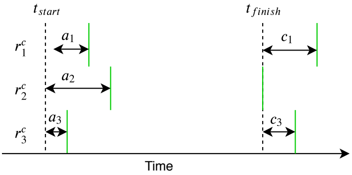
Arrival and Cancellation Delays II



- Arrival delays:

$$\mathbf{E}[T|\mathcal{S}_a] \leq \mathbf{E}[T|\mathcal{S}_0] + \mathbf{E}[a]$$

Arrival and Cancellation Delays II



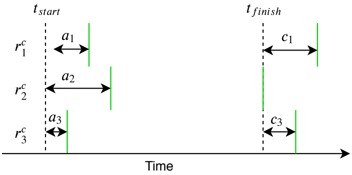
- Arrival delays:

$$\mathbf{E}[T|\mathcal{S}_a] \leq \mathbf{E}[T|\mathcal{S}_0] + \mathbf{E}[a]$$

- Now let $X_i|\mathcal{S}_1 = X_i|\mathcal{S}_0 + \mathbf{E}[c]$
- Cancellation delays:

$$\mathbf{E}[T|\mathcal{S}_c] \leq \mathbf{E}[T|\mathcal{S}_1]$$

Arrival and Cancellation Delays II



- Arrival delays:

$$\mathbf{E}[T|\mathcal{S}_a] \leq \mathbf{E}[T|\mathcal{S}_0] + \mathbf{E}[a]$$

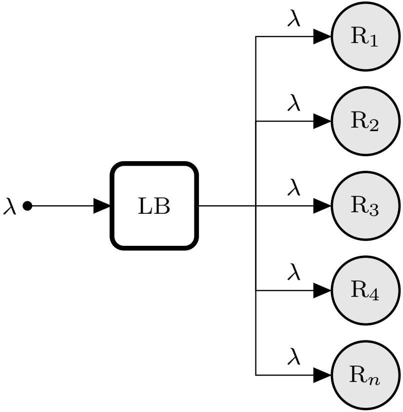
- Now let $X_i|\mathcal{S}_1 = X_i|\mathcal{S}_0 + \mathbf{E}[c]$
- Cancellation delays:

$$\mathbf{E}[T|\mathcal{S}_c] \leq \mathbf{E}[T|\mathcal{S}_1]$$

- Combined delays:

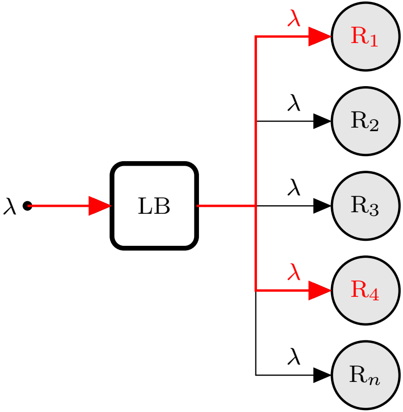
$$\mathbf{E}[T|\mathcal{S}_{ac}] \leq \mathbf{E}[T|\mathcal{S}_1] + \mathbf{E}[a]$$

Clone-to-Any



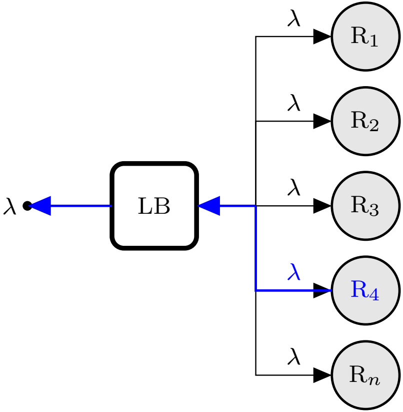
- No clustering – servers are chosen according to ℓ
- Synchronized service not guaranteed!
- Clone error ϵ is a measure of *near-synchronization*

Clone-to-Any



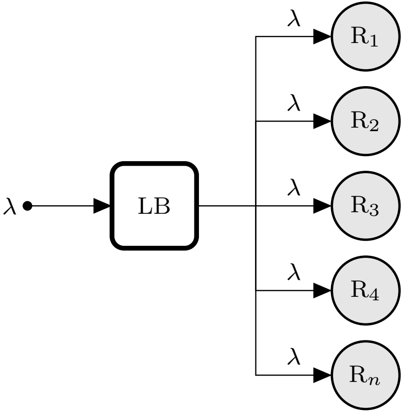
- No clustering – servers are chosen according to ℓ
- Synchronized service not guaranteed!
- Clone error ϵ is a measure of *near-synchronization*

Clone-to-Any



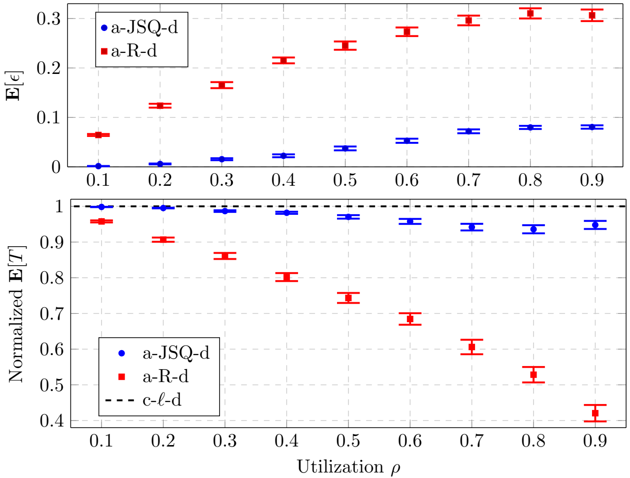
- No clustering – servers are chosen according to ℓ
- Synchronized service not guaranteed!
- Clone error ϵ is a measure of *near-synchronization*

Clone-to-Any



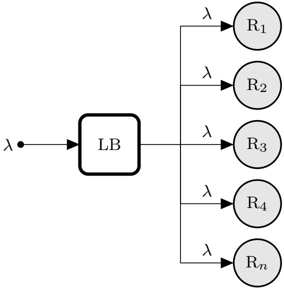
- No clustering – servers are chosen according to ℓ
- Synchronized service not guaranteed!
- Clone error ϵ is a measure of *near-synchronization*

Clone-to-Any vs Clone-to-Clusters



- Clone-to-Any compared to Clone-to-Clusters:
  - Small clone errors ϵ for a-JSQ-d
  - Response time difference less than 10 % for a-JSQ-d

JSQ as a Synchronizer

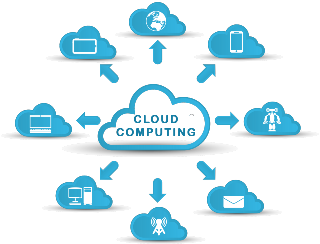


- JSQ chooses server with least requests
- Low utilizations ⇒ All clones r_i^c execute alone!
- High utilizations ⇒ All clones r_i^c execute on almost equally occupied servers!
- Leads to similar processor shares and small clone errors

Conclusions

- We formalized the enabling synchronized service criterion
- We generalized the current cloning models
- We analyzed synchronized server systems under PS
- We relaxed the synchronized service assumption:
  - Bounds for imperfections
  - Clone-to-Any and near-synchronization

Thank you!



Questions?

tommi@control.lth.se