Routing, Scheduling, and Networking in Data Centers

R. Srikant and Weina Wang
University of Illinois at Urbana-Champaign

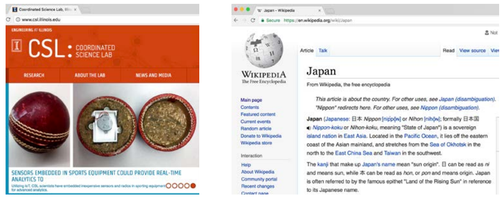
Data Centers

Large-scale data processing

Search engines

Cloud computing

Webpages



Data analytics

Cloud storage

Tutorial

- Resource Allocation Problems in Data Centers and Cloud Computing
  - Recent results and open problems

- Mean-Field Approximation
  - A very brief and high-level introduction

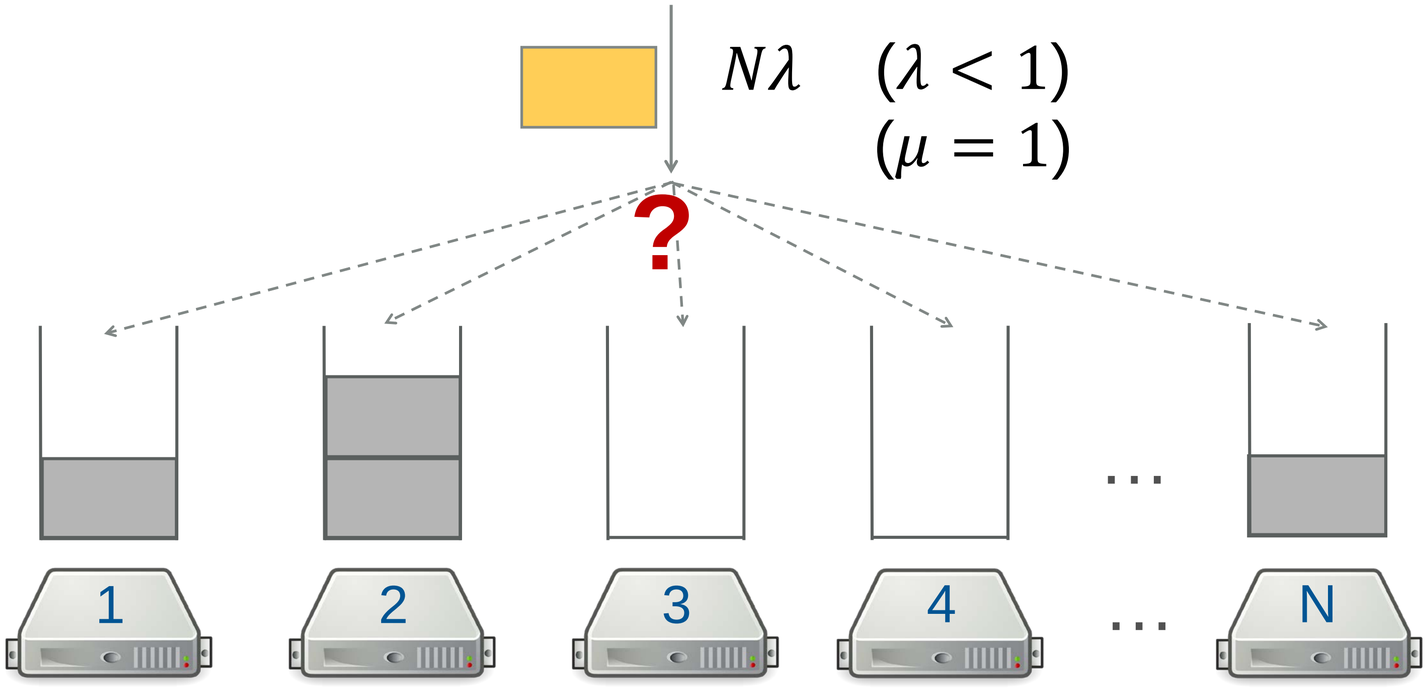
- Heavy-Traffic Approximation
  - A very brief and high-level introduction

Outline of Part I

- Load Balancing
- Load-balancing in large data-storage systems
- Scheduling with Data Locality
- Job vs Task Scheduling
- Minimizing data transfer delay in data center networks

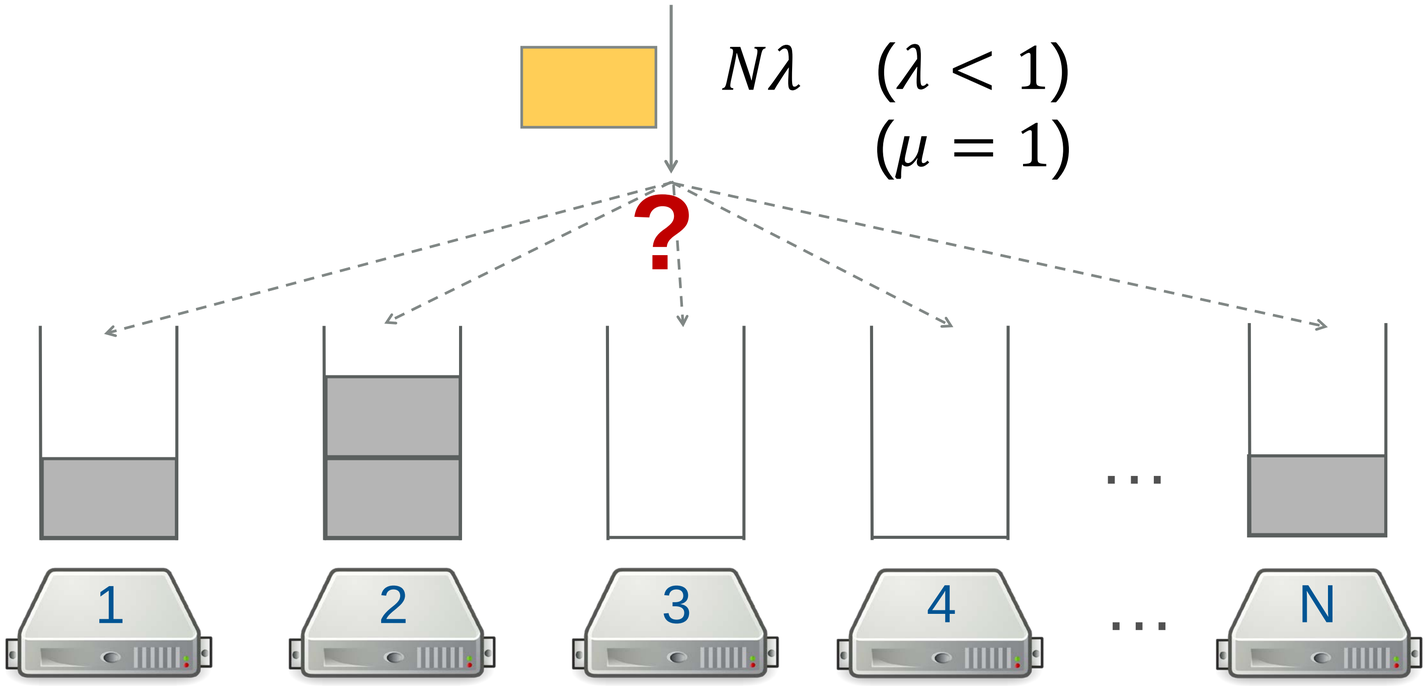
Load Balancing

- Which queue to join?



Load Balancing

- Join-the-Shortest Queue (Overhead?)
- Power-of-two choices: sample two at random, join the shortest of the two
- ….



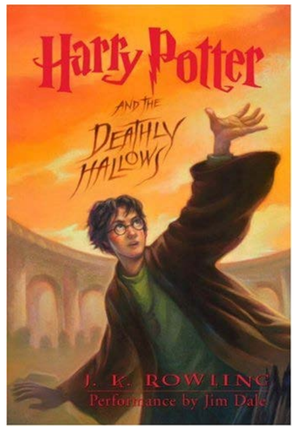
Outline of Part I

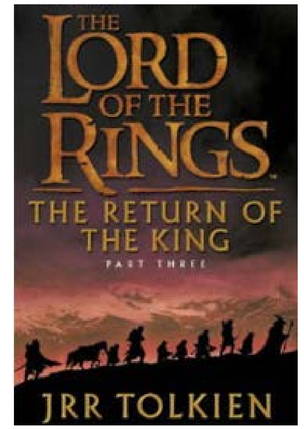
- Load-balancing in large data-storage systems
- Scheduling with Data Locality
- Job vs Task Scheduling
- Minimizing data transfer delay in data center networks

Reliability and Load Balancing

- Servers which store files fail occasionally
- Each file is stored in multiple servers to protect against such failures
- This provides a load balancing opportunity
  - Which server or servers should we fetch a file from?
  - How do different schemes to improve reliability help in terms of load-balancing?
  - Well studied problem, one point of view here

Replication





Server 1

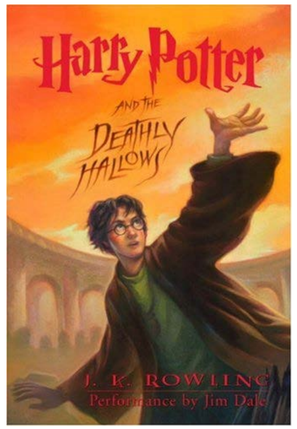
Server 2

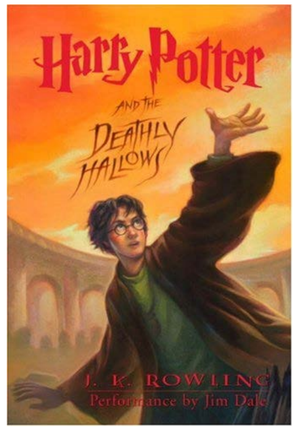
Server 3

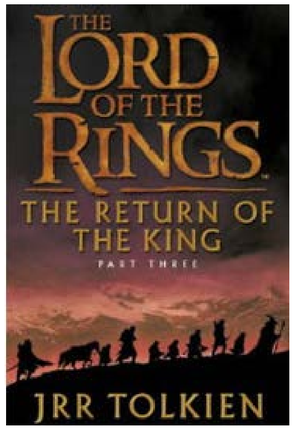
Server 4

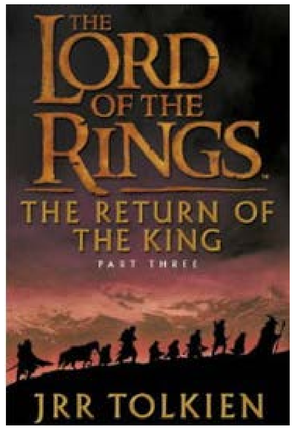
Replication

- (2,1) code









Server 1

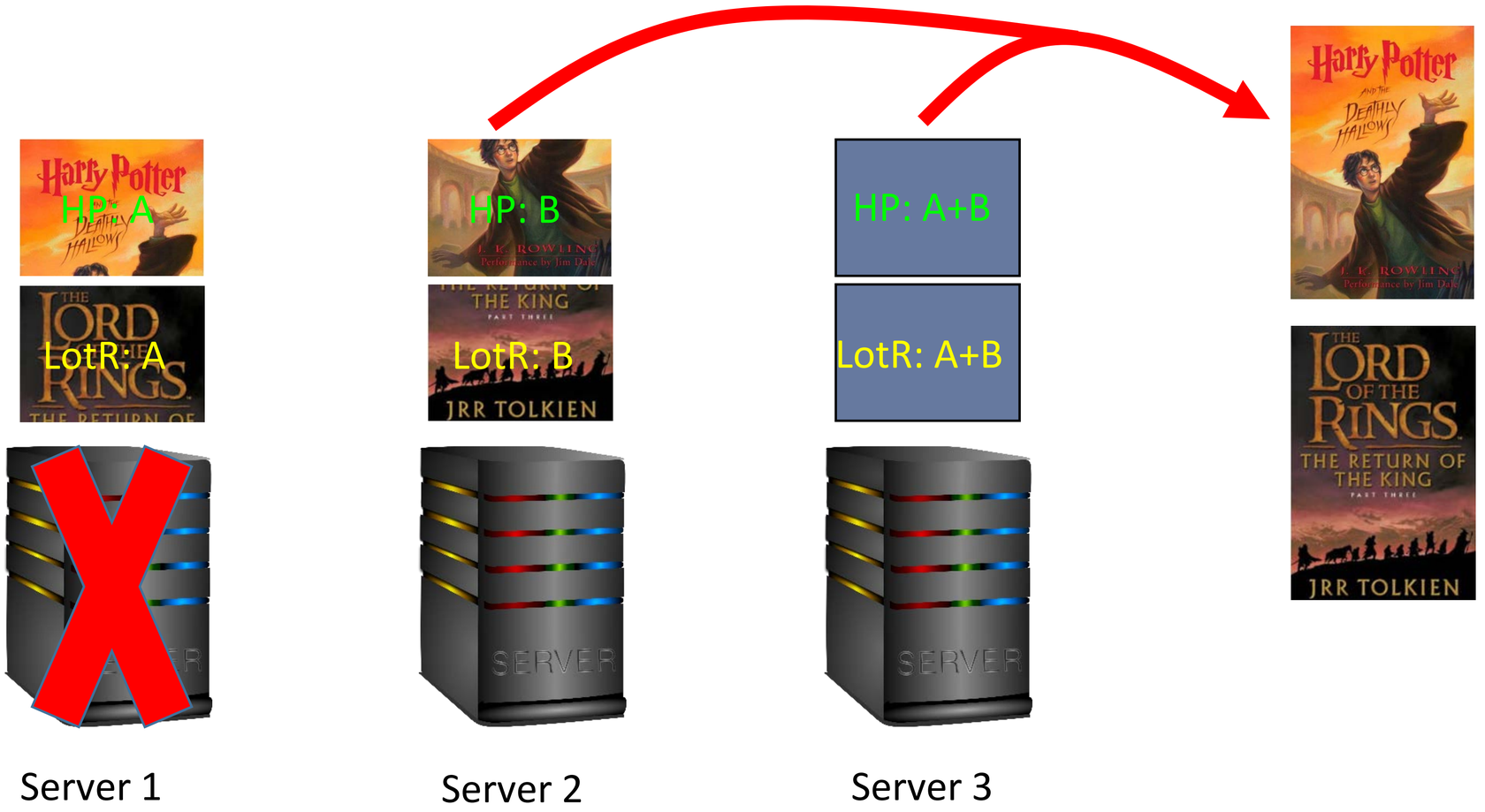
Server 2

Server 3

Server 4

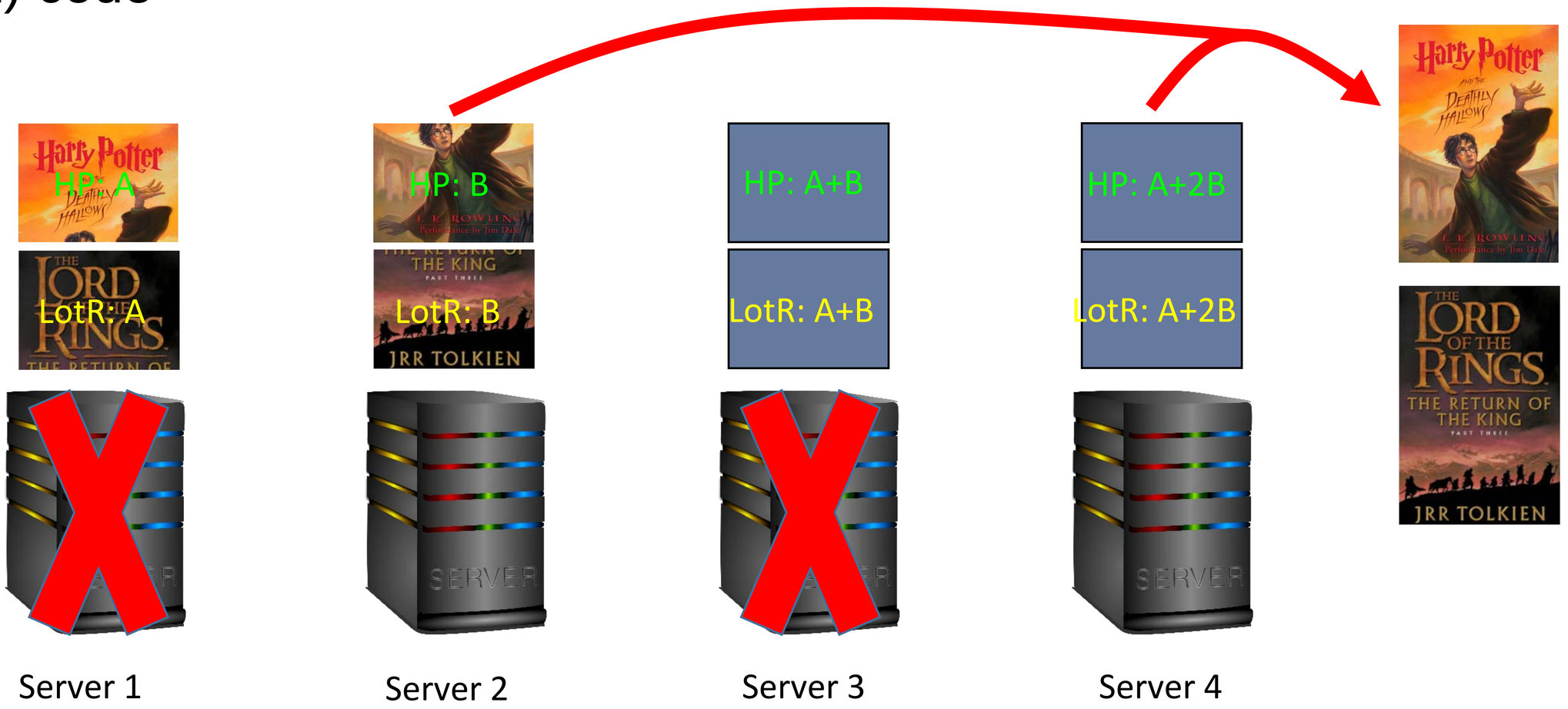
Coding: Reduce Storage, Maintain Reliability

- (3,2) code



Coding: Improve Reliability, Maintain Storage

- (4,2) code

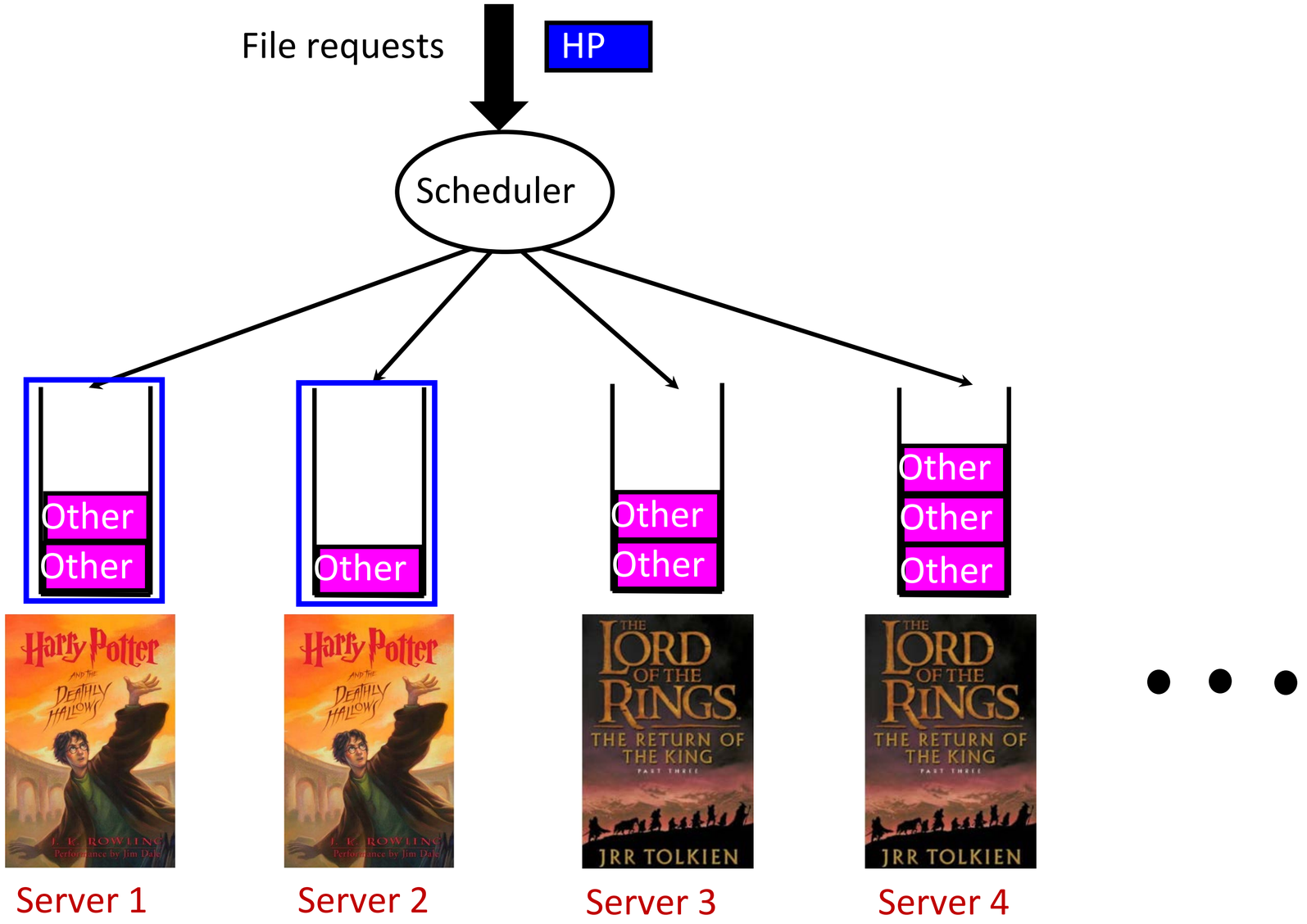


Focus on Load Balancing

- For the rest of this part, we will not model server failures
  - These will be assumed to occur at a slower timescale than the rate at which files are accessed from the servers

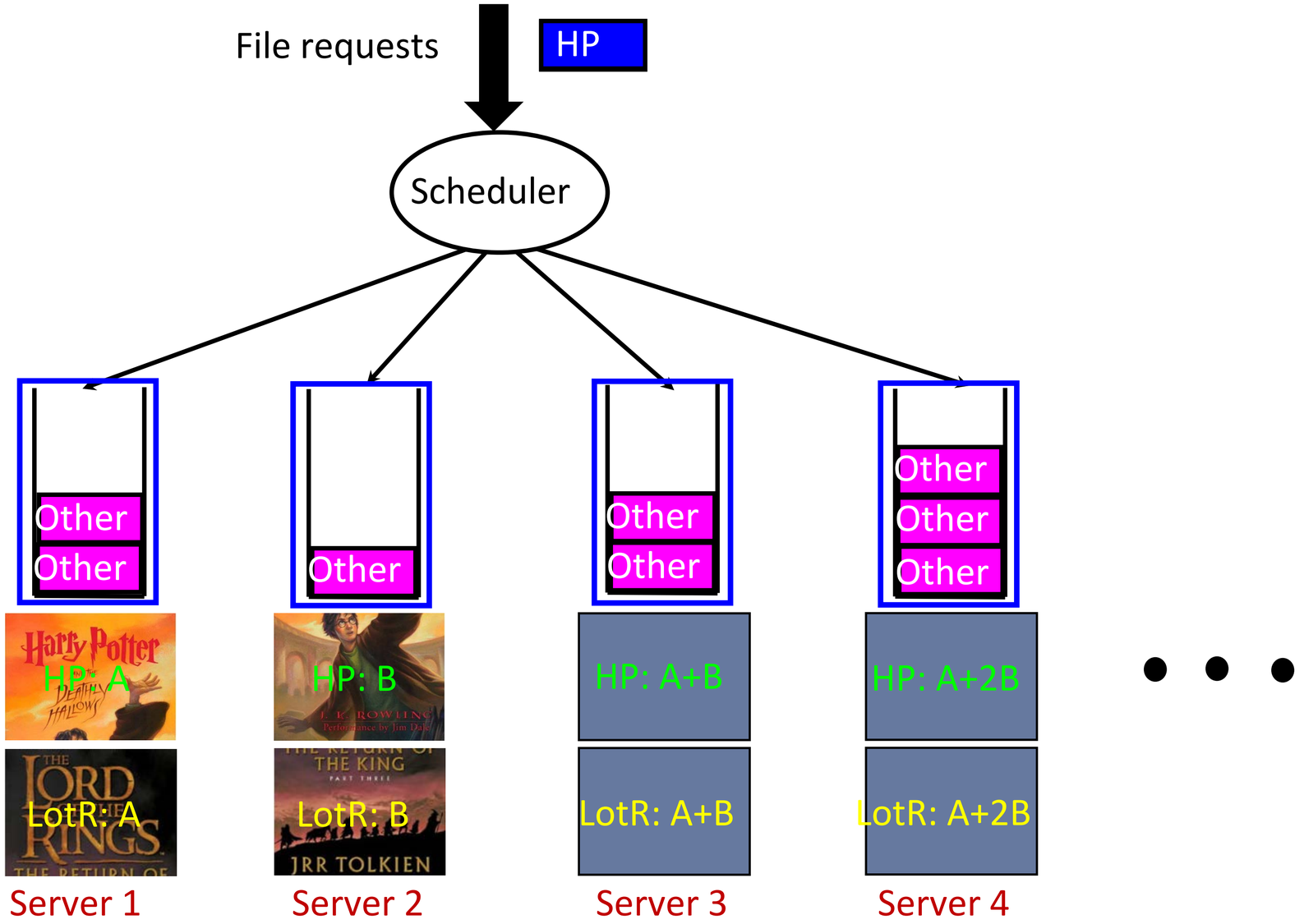
- We will focus on how to exploit redundancy in file storage (through replication or coding) to improve the speed with which we can fetch a file from the servers
  - Using load balancing

Replication and Load Balancing: (2,1) Code



Coding and Load Balancing: (4,2) Code

Need chunks from two servers, but each chunk is half the size of the original photo



In this talk….

- (n,1) code (replication):
  - Each file is replicated at n different servers
  - The size of the each copy of the file is 1

- (nk,k) code
  - Each file is encoded into nk chunks and stored in nk different servers
  - The size of each chunk is $1/k$
  - Any k out of the nk chunks are sufficient to recover the entire file
  - Total storage space is the same as above

Delay comparison: (n,1) vs. (nk,k) code

- Question: Does the delay improve with coding in the large-system limit (large number of servers, files, arrival rates)?
- "Result" (Li, Ramamoorthy, S., 2016): Under a Poisson arrival/exponential service-time model, the delay decreases by a factor of at least

$$\frac{H(k)}{k}$$

where $H(k)$ is the k^{th} harmonic number:

$$1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{k}$$

Very light traffic (zero queues): (2,1) vs. (4,2) code

- File service time is exponentially distributed
- Mean file access delay under (2,1) code (replication):

$$\bar{W}^{(2,1)} = \mathbf{E}[X] \qquad (X \sim \mathrm{Exp}(1))$$
$$= 1$$

- Mean file access delay under (4,2) code:

$$\bar{W}^{(4,2)} = \mathbf{E}[\max\{Y_1, Y_2\}] \quad (Y_1 \text{ and } Y_2 \text{ are i.i.d. with } \mathrm{Exp}(2))$$

$$= \frac{3}{4}$$

Compared with replication, delay improves by 25% under coding

Very light traffic (zero queues): (*n*,1) vs. (*nk*,*k*) code

- Mean file access delay under (*n*,1) code (replication):

$$\bar{W}^{(n,1)} = \mathbf{E}[X] \qquad (X \sim \mathrm{Exp}(1))$$
$$= 1$$

- Mean file access delay under (*nk*,*k*) code:

$$\bar{W}^{(nk,k)} = \mathbf{E}\left[\max_{i=1,2,\cdots,k} Y_i\right] \qquad (Y_i, \forall i, \text{ are i.i.d. with } \mathrm{Exp}(k))$$

$$= \frac{H(k)}{k}$$

General Traffic Case: (n,1) vs. (nk,k) code

- Question: What happens for general arrival rates?
- Answer: in the many-servers limit,
  - In the heavy-traffic regime

$$\lim_{\lambda \uparrow 1} \frac{\bar{W}^{(nk,k)}}{\bar{W}^{(n,1)}} \leq \frac{H(k)}{k}$$

- Conjecture: in the many-servers limit, for all feasible arrival rates,

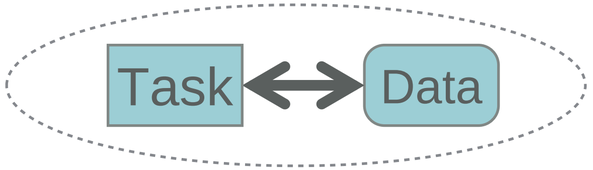
$$\frac{\bar{W}^{(nk,k)}}{\bar{W}^{(n,1)}} \leq \frac{H(k)}{k}$$

Outline of Part I

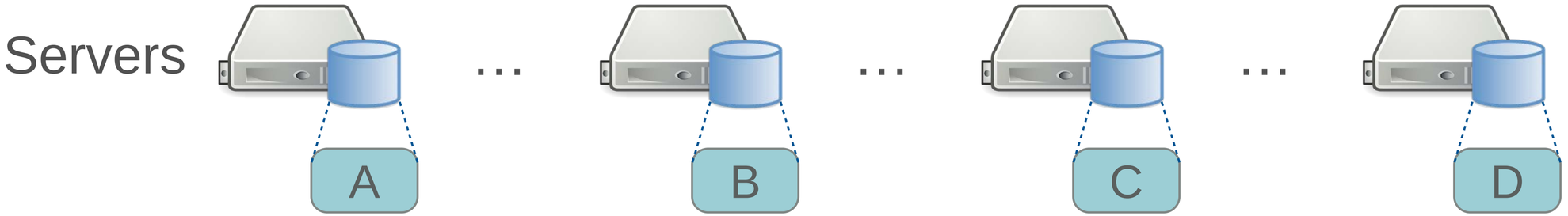
- Load-balancing in large data-storage systems
- Scheduling with Data Locality
- Job vs Task Scheduling
- Minimizing data transfer delay in data center networks

Load-Balancing and Communication

- Each server stores certain data
- Each task is associated with a data chunk

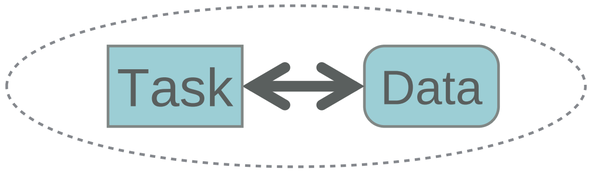


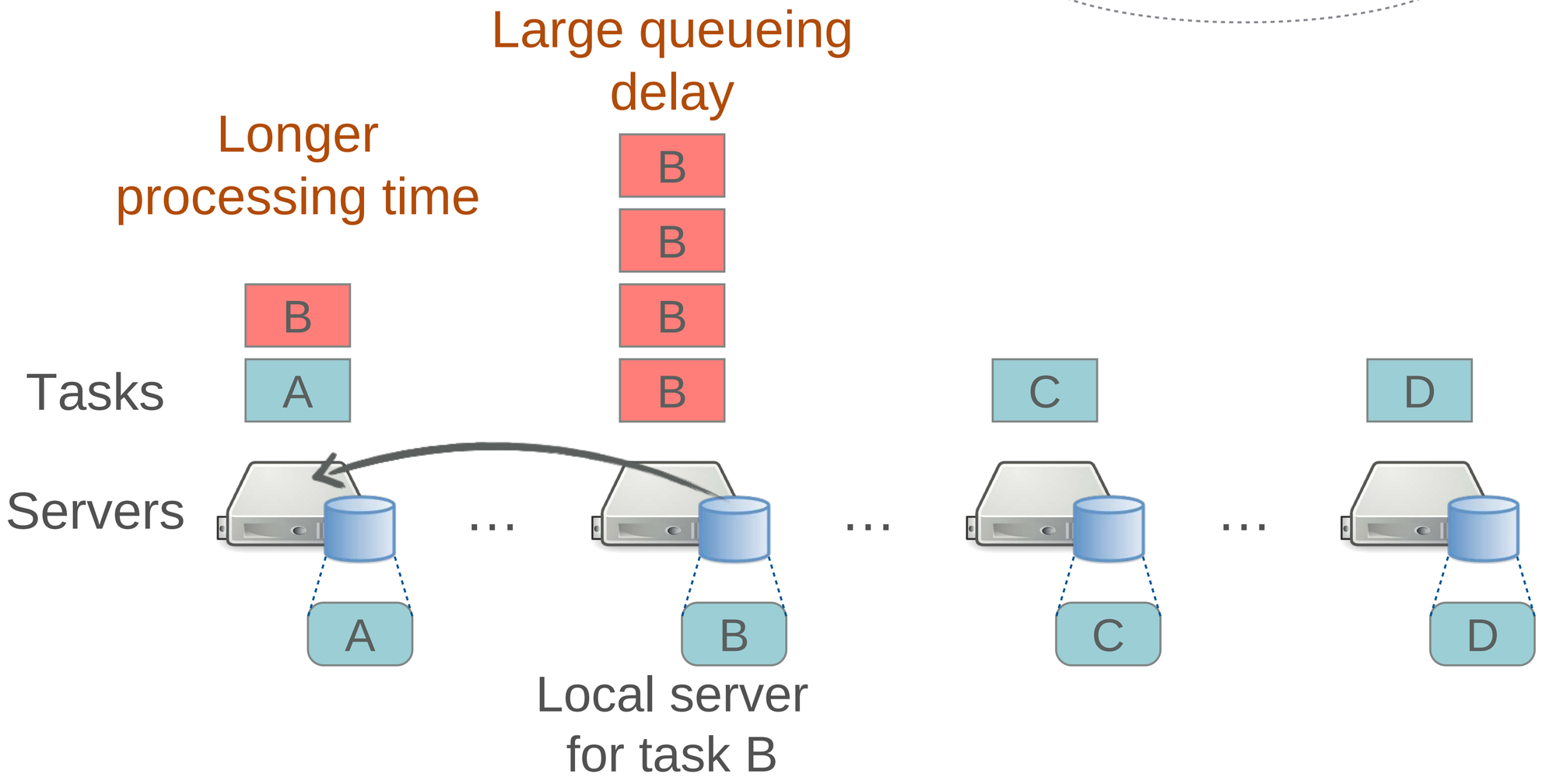
- If a task is routed to a server with the required data chunk, then it executes quickly; otherwise, it has to fetch the data from another server, thus leading to longer processing times



Load-Balancing and Communication

- Each task is associated with a data chunk





Modeling Data Locality

- For the rest of this part, we will not model the dynamics in the network
  - We abstract the network away and assume that

local processing rate > remote processing rate

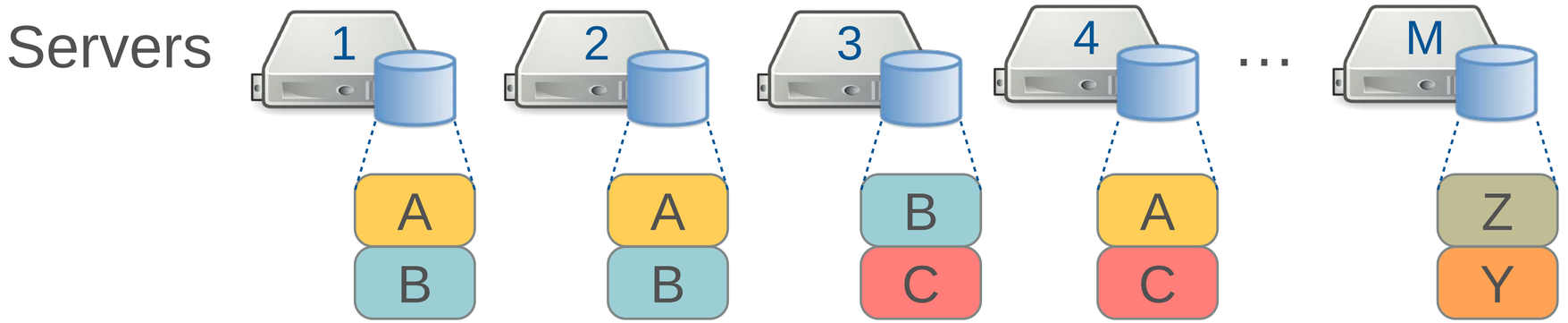
- We will focus on how to strike the right balance between load-balancing and communication to maximize throughput and minimize delay

Task Types

- Task type: servers that store its input data (local servers)

Type = (1, 2, 4)

Arrival rates $\lambda_{(1,2,3)}$ $\lambda_{(1,2,4)}$ $\lambda_{(1,2,5)}$ … $\lambda_{(i,\ j,\ k)}$ …

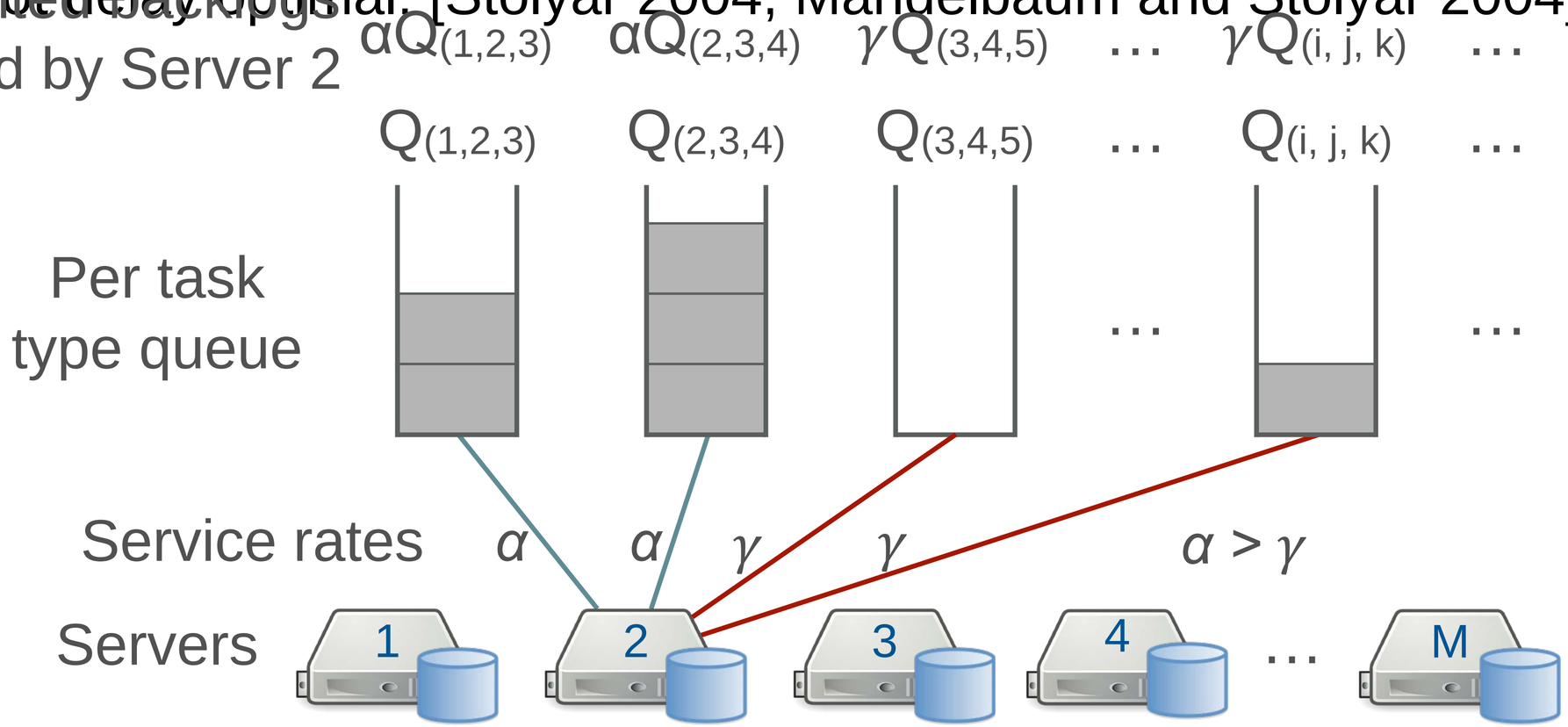


Max Weighted Backlog [Tassiulas and Ephremides 1993]

When server 2 becomes available:
serves the queue with the maximum weight

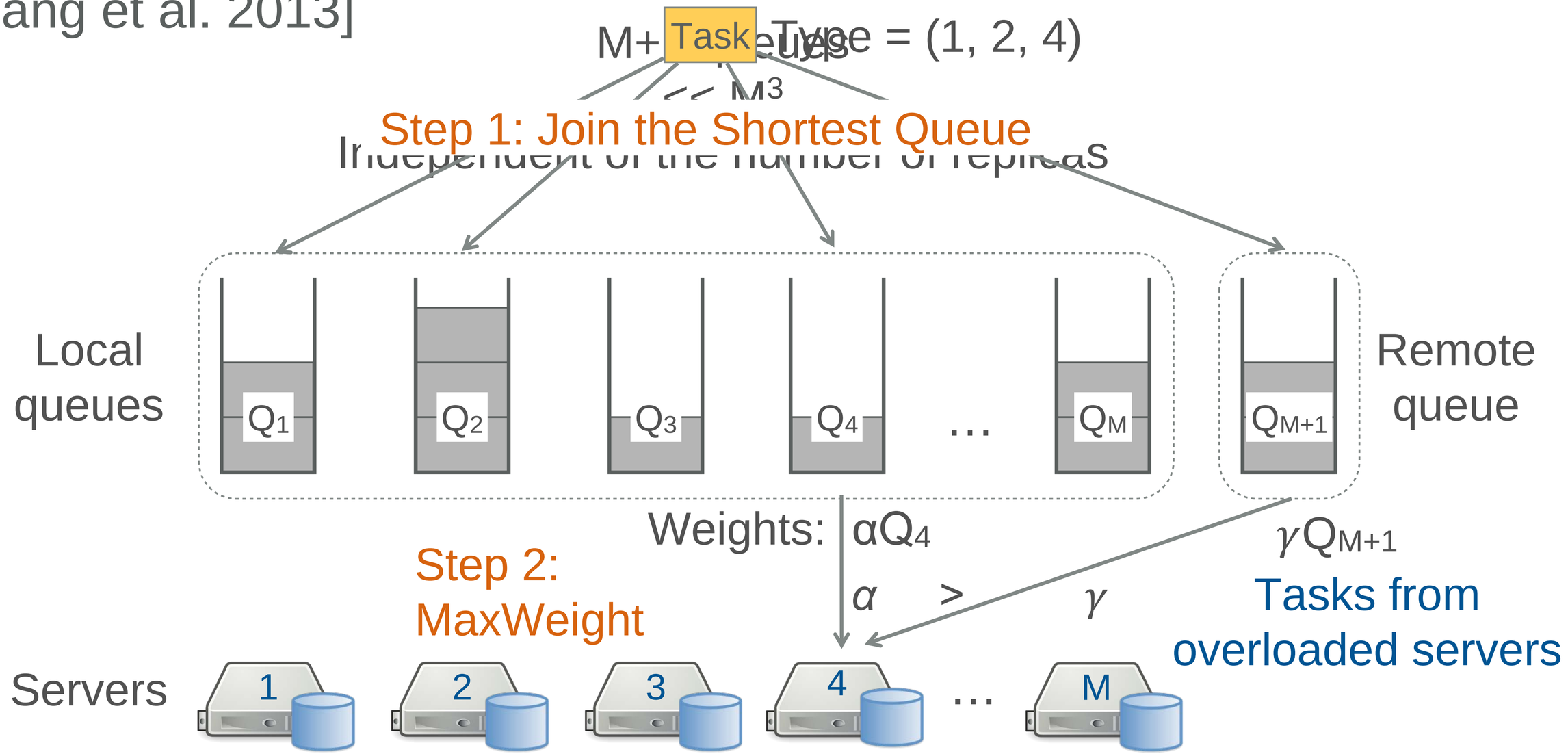
- Throughput optimal but …
  - Too many queues: $O(M^3)$ M ~ Tens of thousands
  - Not delay optimal: [Stolyar 2004, Mandelbaum and Stolyar 2004]

Weighted backlogs viewed by Server 2



Achieving Delay Optimality under Data Locality

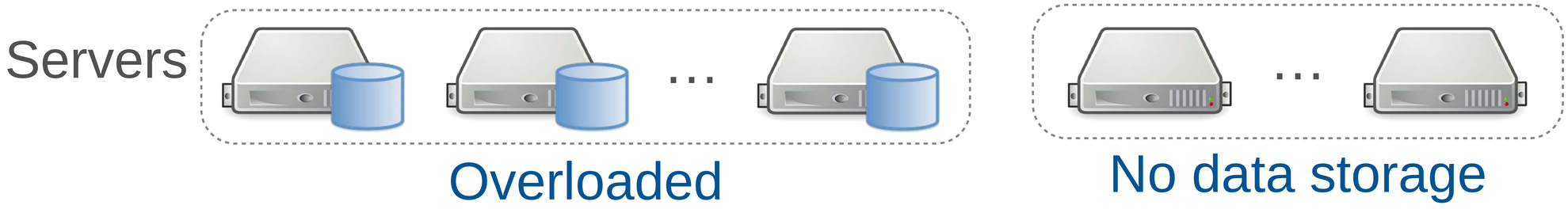
[Wang et al. 2013]



Performance

- Maximizes Throughput

- Delay optimality in the following heavy-traffic regime

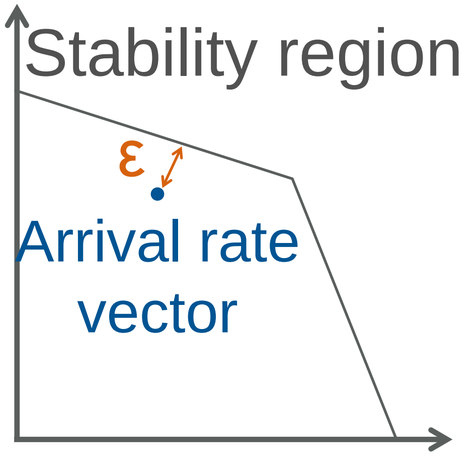


Peak hours of data centers

Delay Optimality

- Total backlog (based on Eryilmaz, S. 2012)

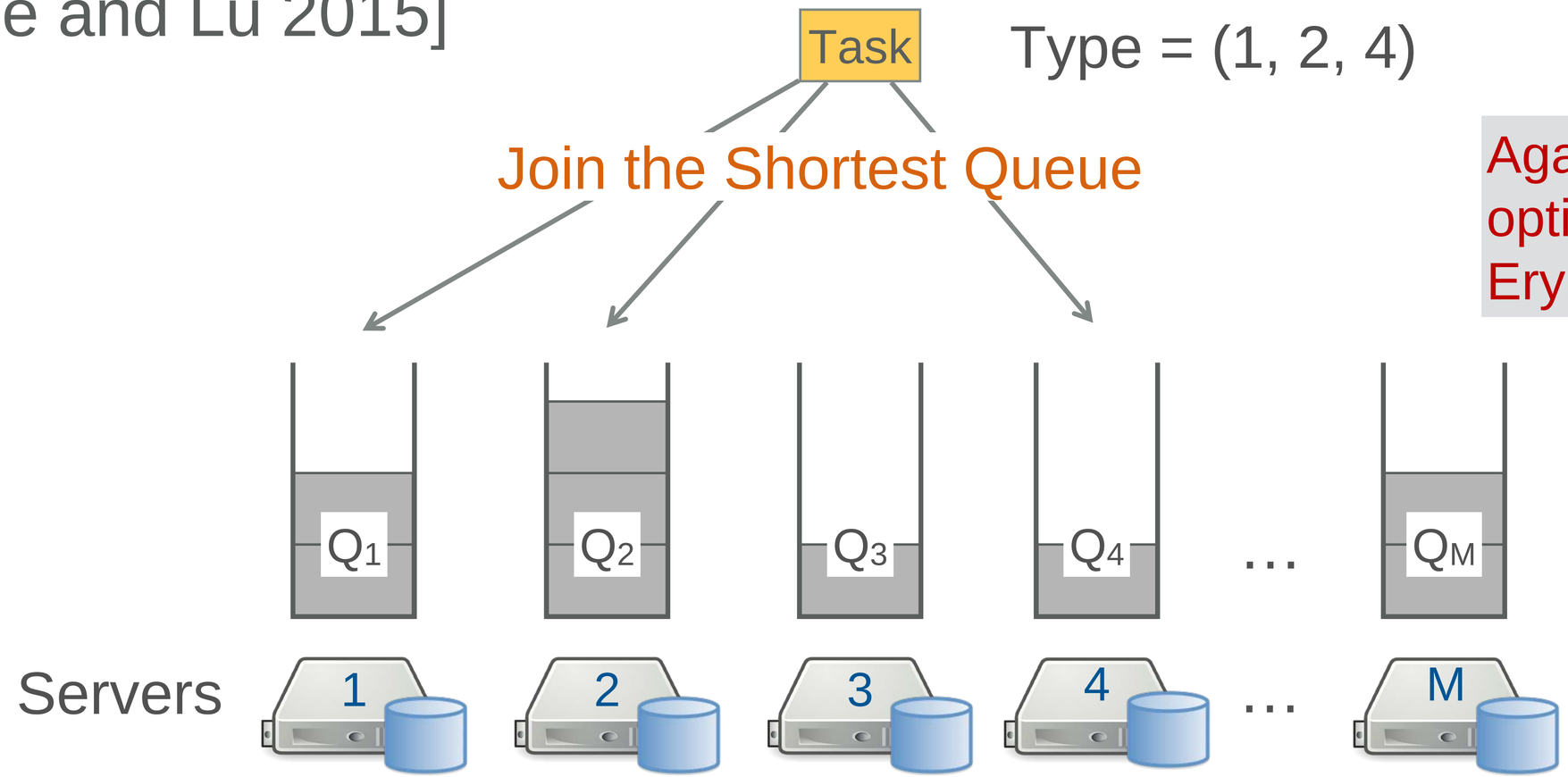
$$\mathbb{E}\left[\sum_{m=1}^{M+1} Q_m^{(\epsilon)}(t)\right] = \frac{\sigma^2 + \nu^2}{2\epsilon} + o\left(\frac{1}{\epsilon}\right)$$



- Coincides with the lower bound where all the servers are pooled together and running on full speed
- Asymptotically minimizes the total backlog
  - By Little's law, it asymptotically minimizes the average task delay

Delay Optimality for All Heavy-Traffic Regime

[Xie and Lu 2015]



Again heavy-traffic optimal in the sense of Eryilmaz, S. (2012)

Prioritized scheduling - Local tasks first

- No local tasks: serve the longest queue

Open Problem

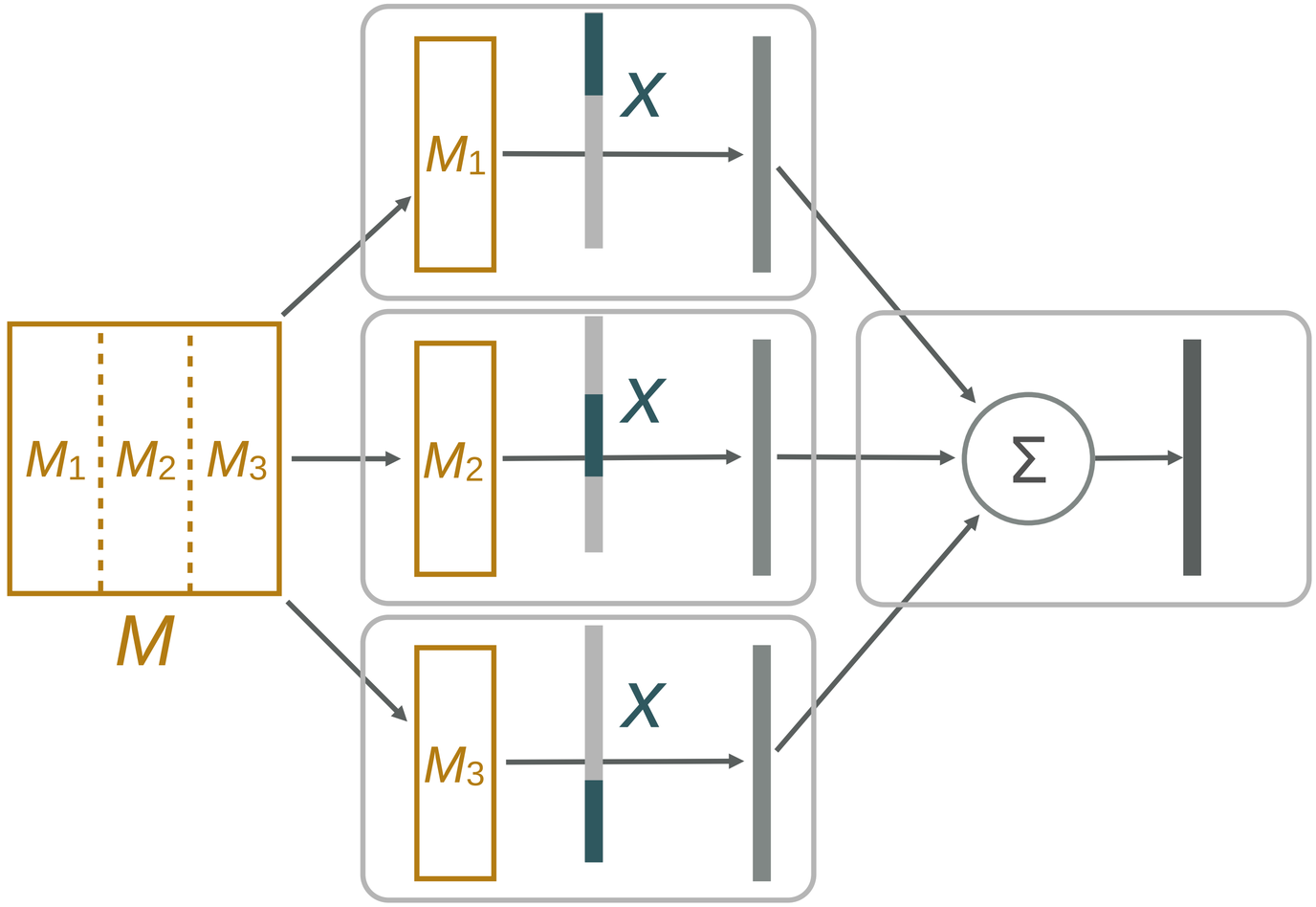
- In our discussion, we considered problems where each job consists of a single task
- Instead, suppose each job consists of multiple tasks, and a job is completed only when all the tasks in the job are completed
- Further each task can be allocated to a different server if needed
- Scheduling algorithms which minimize job delays are unknown
- But near-optimality can be achieved when there is no data locality

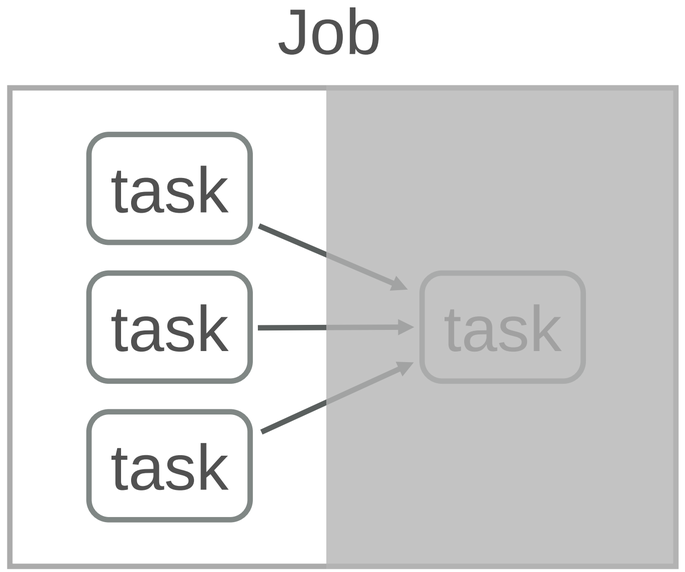
Outline of Part I

- Load-balancing in large data-storage systems
- Scheduling with Data Locality
- Job vs Task Scheduling (with no data locality)
- Minimizing data transfer delay in data center networks

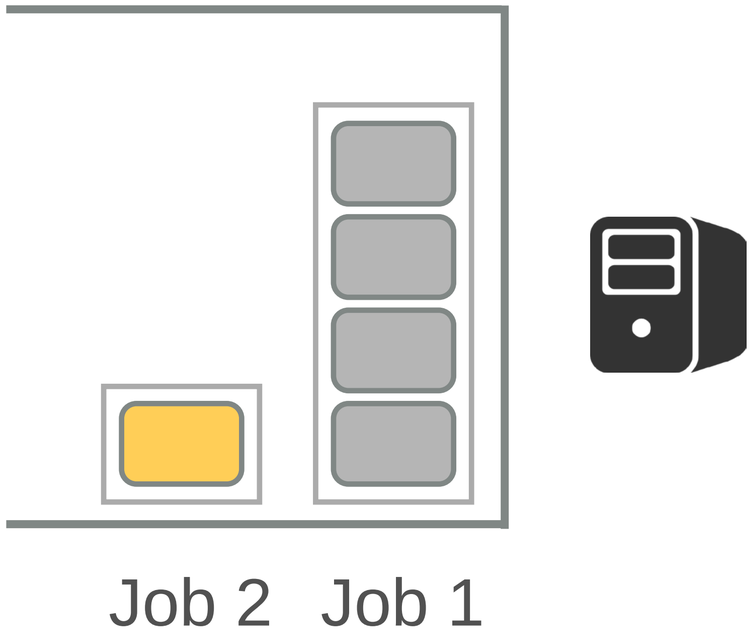
Jobs with Multiple Tasks

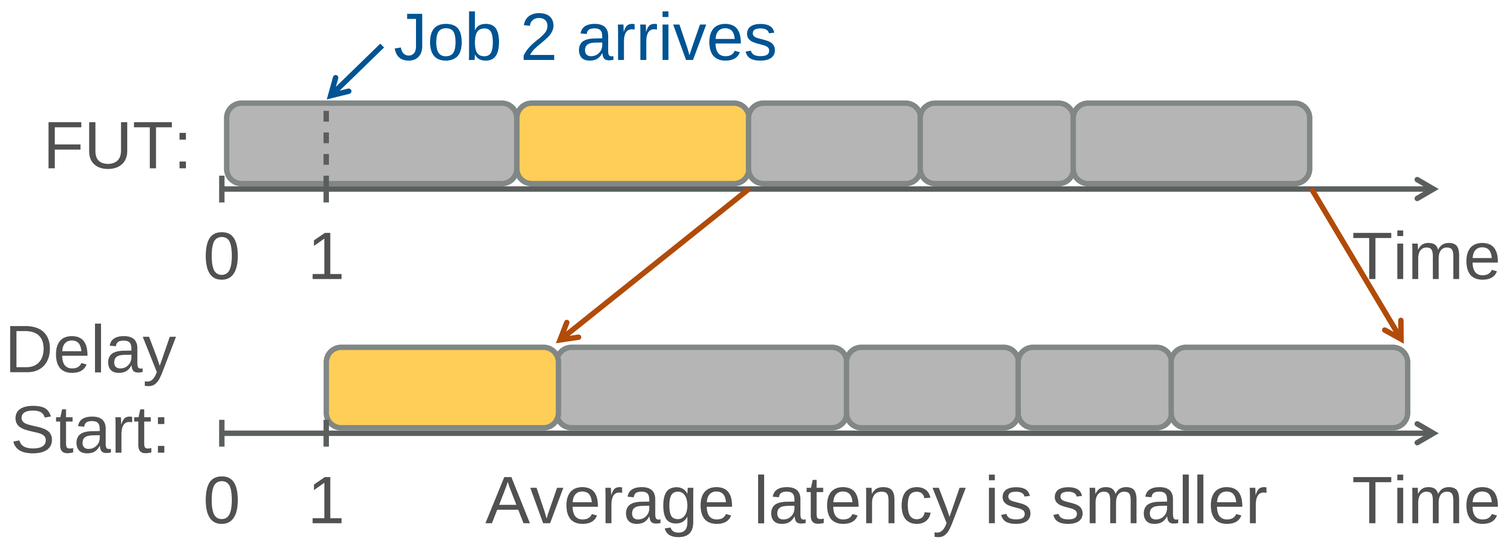
- Matrix-vector multiplication: *Mx*
  - Basic operation for PageRank, regression analysis, …





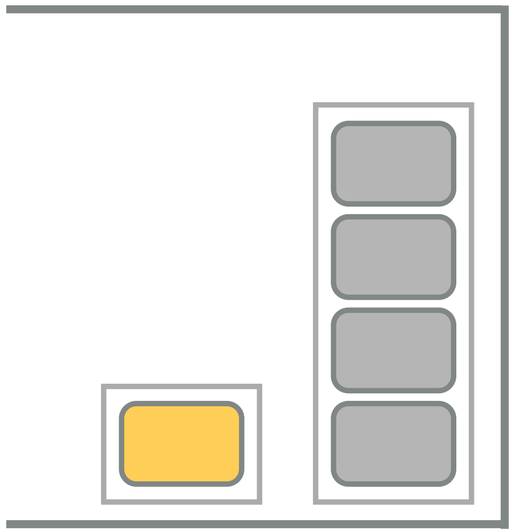
Single-Server System

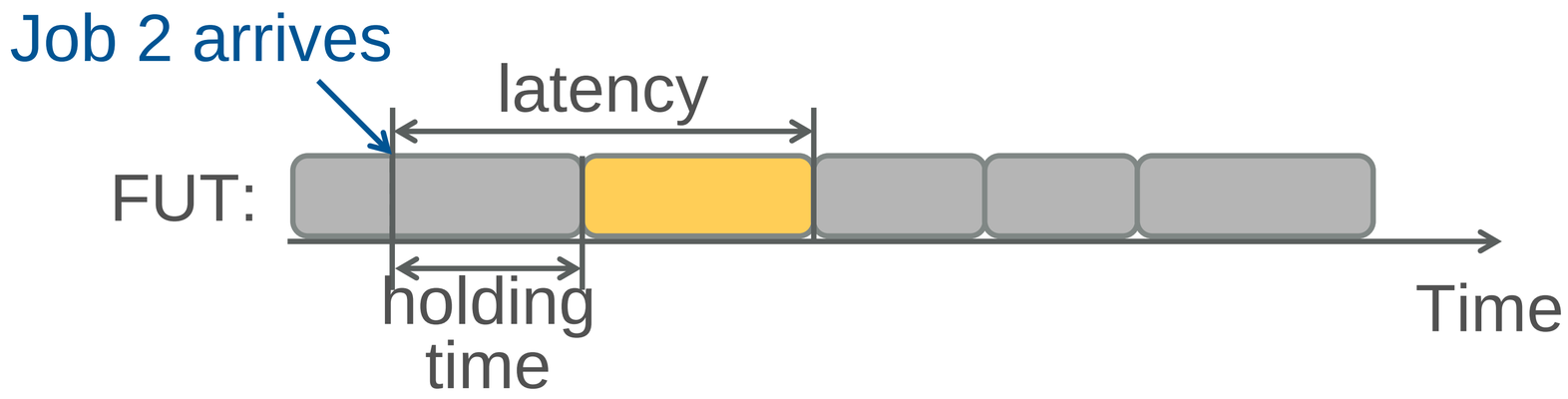




- Fewest-Unassigned-Tasks (FUT) first?
  - Analogy to Shortest-Remaining-Processing-Time (SRPT) policy
  - Not sample-path optimal for jobs with multiple tasks

Near Delay Optimality of FUT





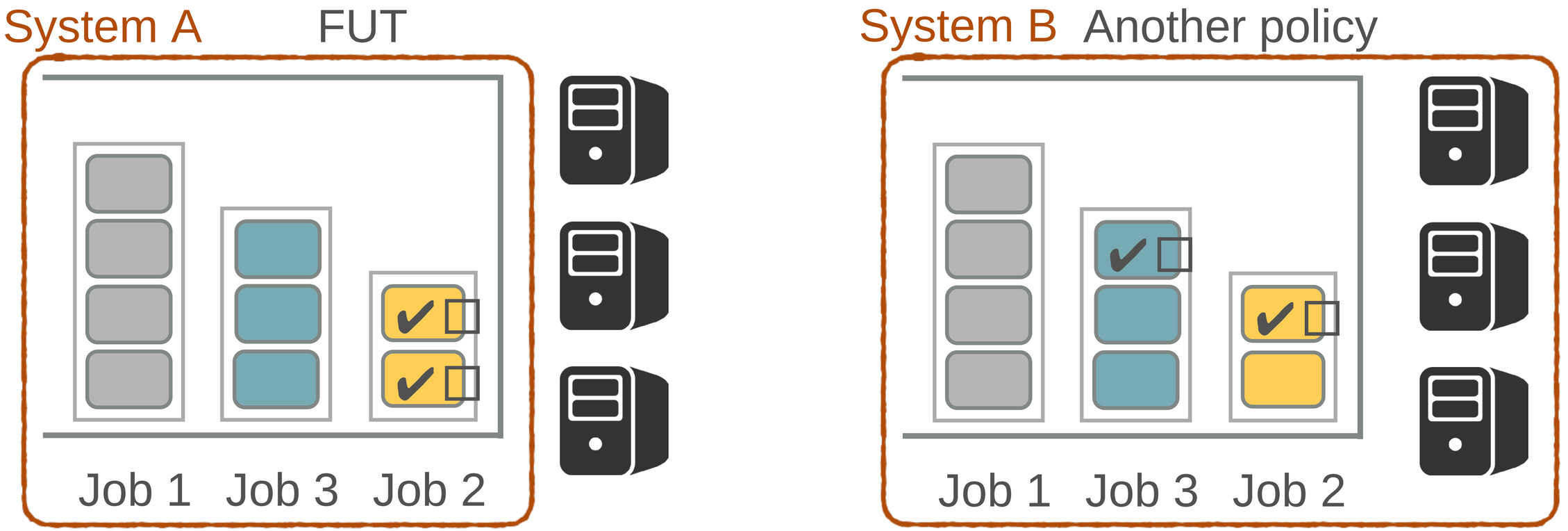
Near delay optimality:

Average holding time of FUT ≤ Average latency of any policy

[Sun, Koksal, Shroff 2017]

- Holding time of a job: amount of time a job spends in the queue
  - Single server system: latency − holding time = service time of one task
  - More generally: latency − holding time ≤ Constant

Intuition



- Average holding time of FUT = average latency of System A
  - Tasks leave System A from the job with the fewest remaining tasks
  - For any [0, t], task departures: System A ≥ System B

Open Problem

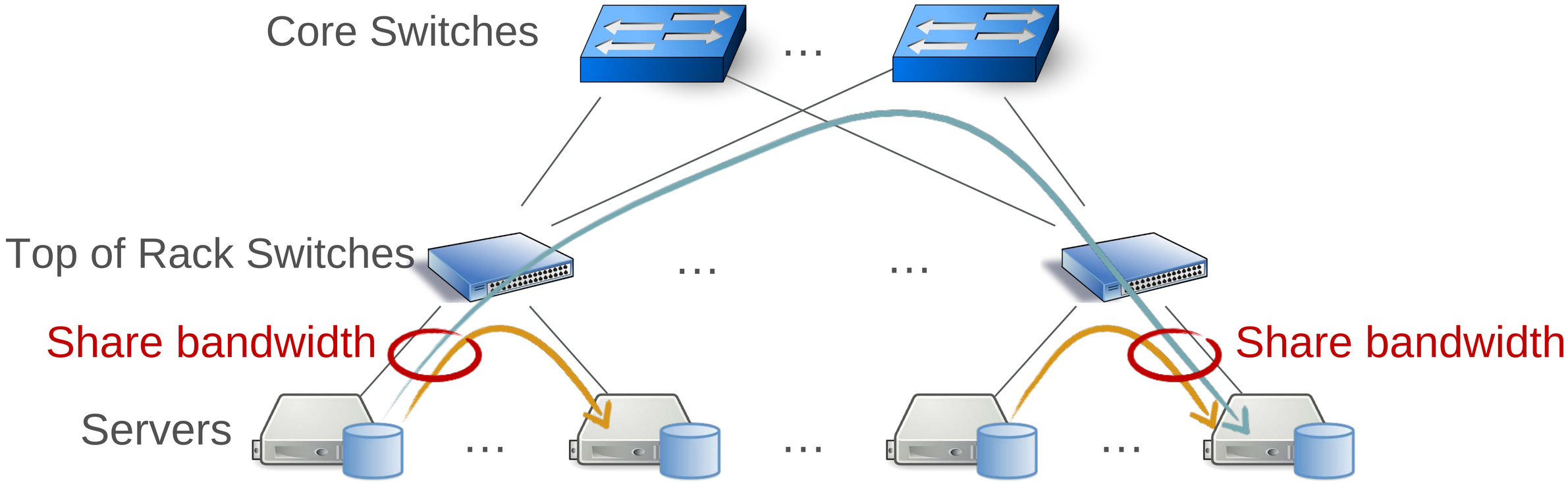
- Back to the matrix multiplication example
- The job consists of not only many tasks, but the tasks have to be executed in multiple stages, i.e., there are precedence constraints
- Delay-optimal or near-optimal scheduling policies for multi-stage, multi-task jobs are unknown
- Data locality and multiple stages???

Outline of Part I

- Load-balancing in large data-storage systems
- Scheduling with Data Locality
- Job vs Task Scheduling
- Minimizing data transfer delay in data center networks

Data Transfer Flows

- How to allocate bandwidth to flows to minimize file transfer delay?
- Links have bandwidth capacities, shared by flows



Resource Allocation

- x_r: rate allocated to a flow on route r
- n_r: # of flows on route r
- C_l: capacity of link l
- Constraints:

$$\sum_{r \ni l} n_r x_r \leq C_l \, , \forall l$$

- Goal: how should we choose $\{x_r\}$ to minimize data transfer delay?

File Transfer Delay

- If a flow on route r arrives at time A_r and has file size F_r, then its delay D_r is determined by the equation:

$$\int_{A_r}^{A_r+D_r} x_r(t)\,dt = F_r \qquad (1)$$

- F_r: random variable
- The choice of $\{x_r\}$ affects the delay D_r through (1)

A Performance Criterion

- Suppose there are *N* servers in the system
- # of source-destination pairs is $O(N^2)$
- # of links in a data center network is $L \ll N^2$
  - Example: $N^2 \sim 10^8$, $L \sim 10^4$ in a tree structured data center network

- Goal: $\mathbb{E}$[# of flows] ~ $O(L)$ instead of $O(N^2)$

Proportional Fairness

- Choose $\{x_r\}$ to maximize

$$\sum_r n_r \log x_r$$

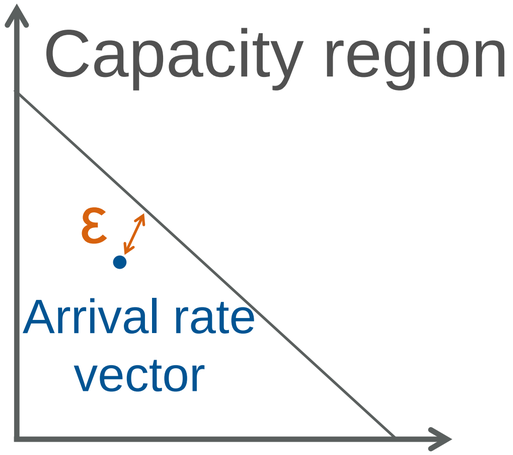
- $\log x_r$: utility of a flow on route r when allocated a rate of x_r

- Subject to resource constraints on each link

Results

- Asymptotically tight bounds on total backlog (Kang et al, 2009)

$$\mathbb{E}[\#flows] = \frac{L}{\epsilon} + o\left(\frac{1}{\epsilon}\right)$$



- L: number of saturated links in heavy traffic

- Insensitivity: the bounds hold for a general class of phase-type distributions for data sizes (Vlasiou et al, 2014)
  - This distribution class can approximate any data size distribution arbitrarily closely

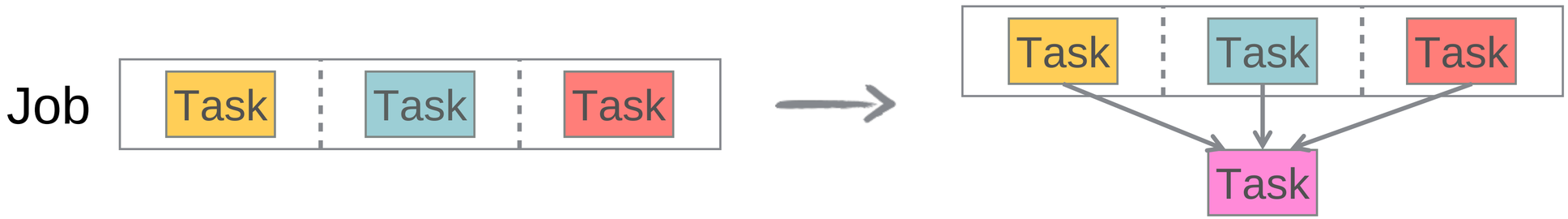
Open Problem

- The file transfers may occur to achieve some other load-balancing goal
  - Such as to fetch remote data at a server in the data locality problem

- Joint resource allocation for data-transfer and load balancing in the presence of data locality to achieve optimal or near-optimal job/task delays is an open problem

Open Problems

- Coding/Replication/Load Balancing Tradeoffs
- Analyzing/minimizing job delay
- Scheduling jobs with multiple stages
  - Dependence among tasks puts constraints on scheduling decisions



- Joint design of task scheduling and data transfer

Rest of the Tutorial

- Difficult to analyze most resource allocation schemes exactly
  - There are some exceptions

- Two asymptotic regimes
  - The number of servers is very large (mean-field limit)
  - The traffic load approaches the capacity of the system (heavy-traffic)

- The first regime is perhaps more realistic for data centers

- The second regime often provides insight into why certain policies behave better than others (even in light to moderate traffic)

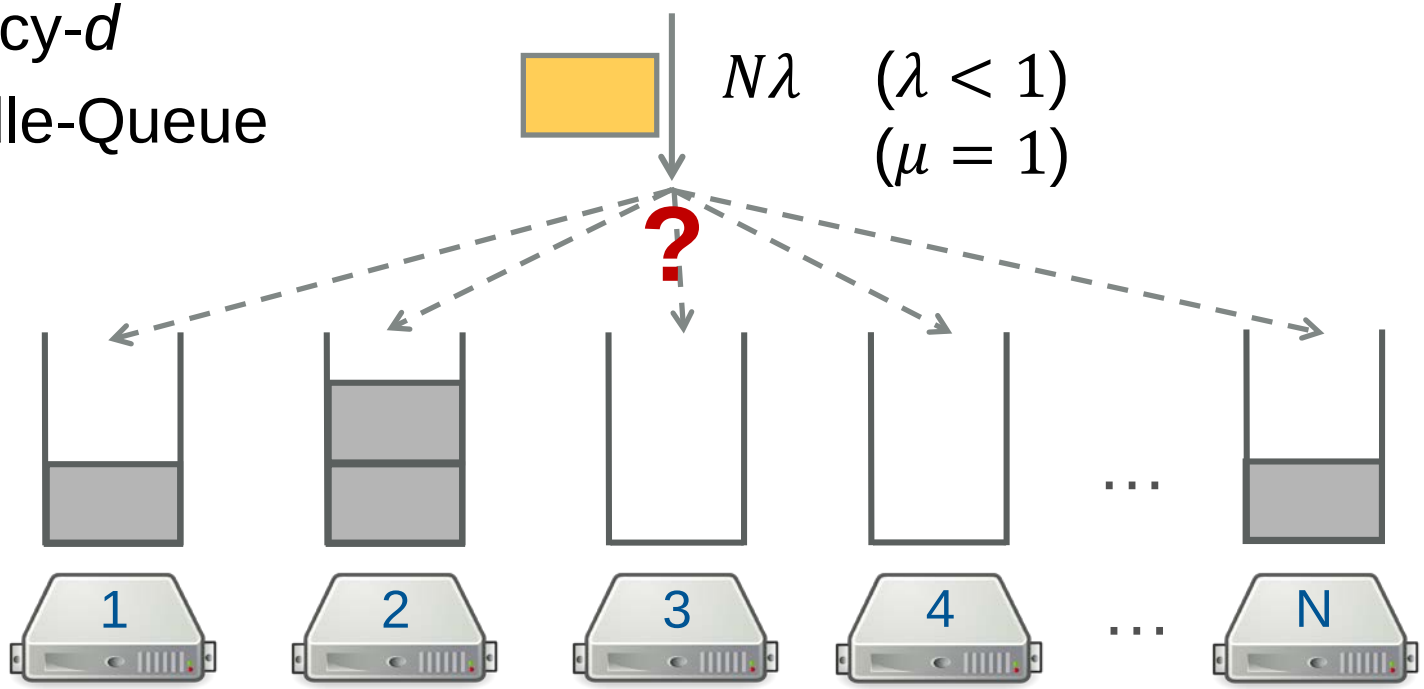
Tutorial on Mean-Field Analysis

R. Srikant and Weina Wang
University of Illinois at Urbana-Champaign

Load Balancing

- Join-the-Shortest-Queue
- Random Routing
- Power-of-*d*-Choices
- Batch-filling
- Redundancy-*d*
- Join-the-Idle-Queue

Arrival process is Poisson($N\lambda$)
Service times are exp(1)

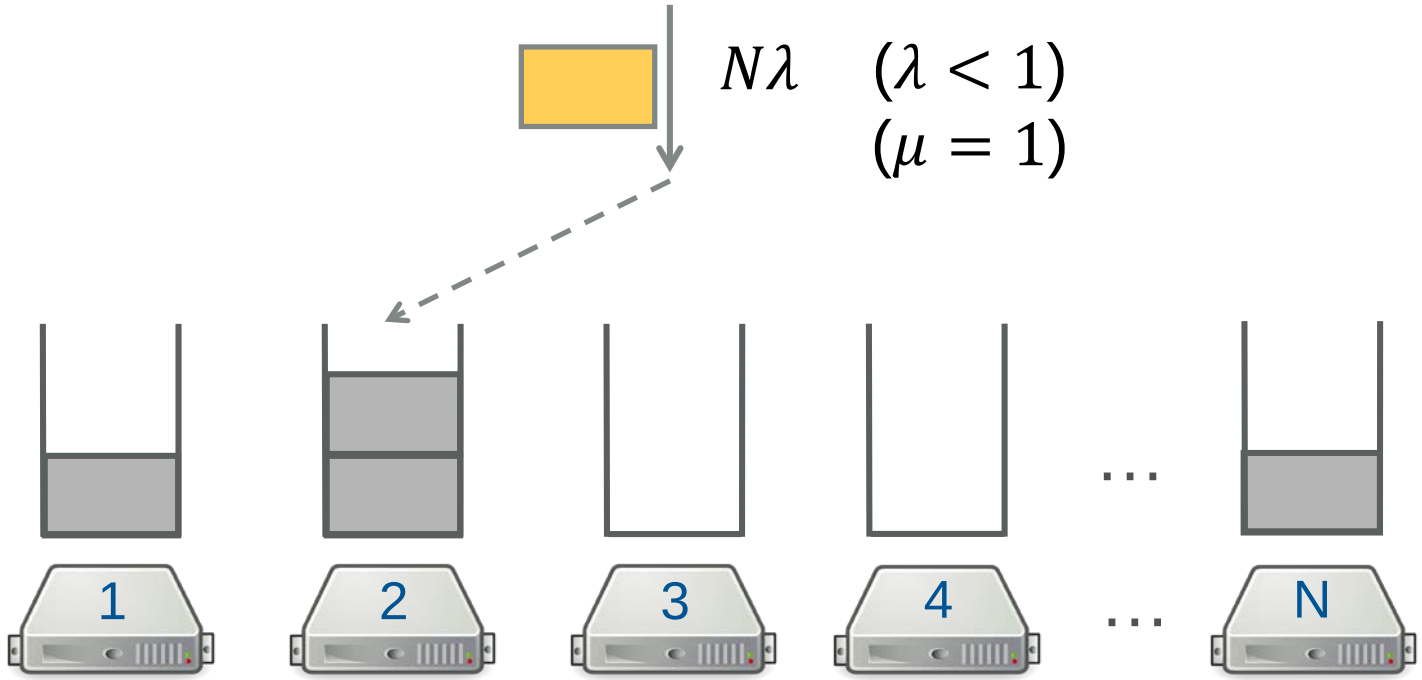


Random Routing

- Route an arrival to a queue uniformly at random
- N separate M/M/1 queues with load λ
- Queue length:

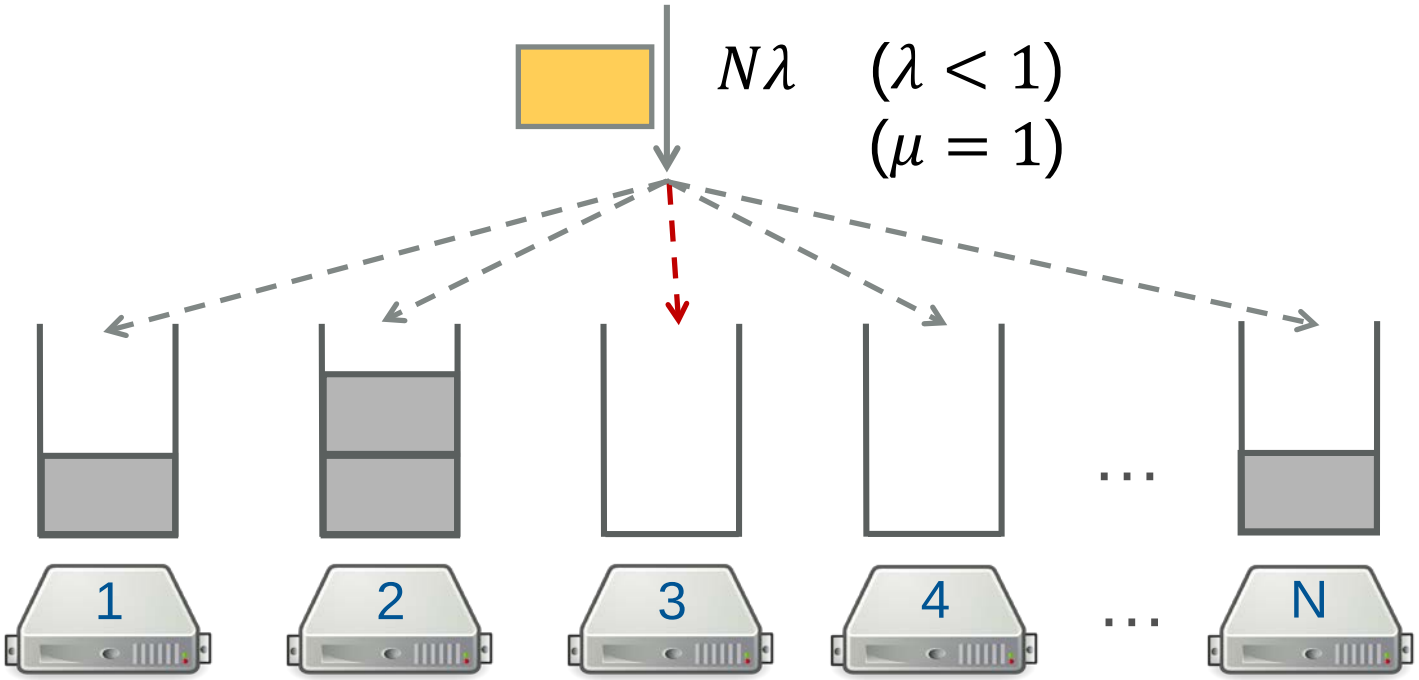
$$\Pr(Q \geq i) = \lambda^i$$

Can we do better?



Join-the-Shortest-Queue

- Each arrival joins the shortest queue among all the N queues
- Minimizes average delay
- Requires information about all queues: large overhead

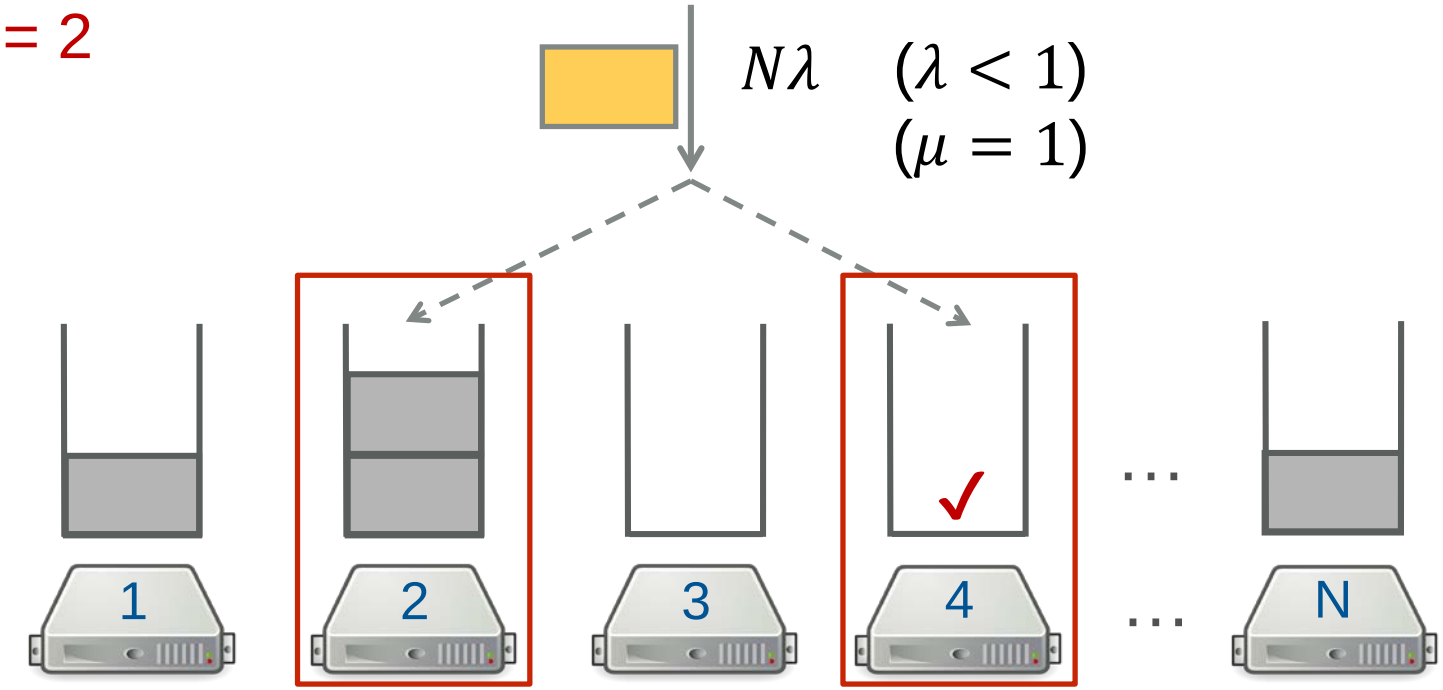


Power-of-*d*-Choices

- Each arrival picks *d* queues at random and joins the shortest one
- Queue length:

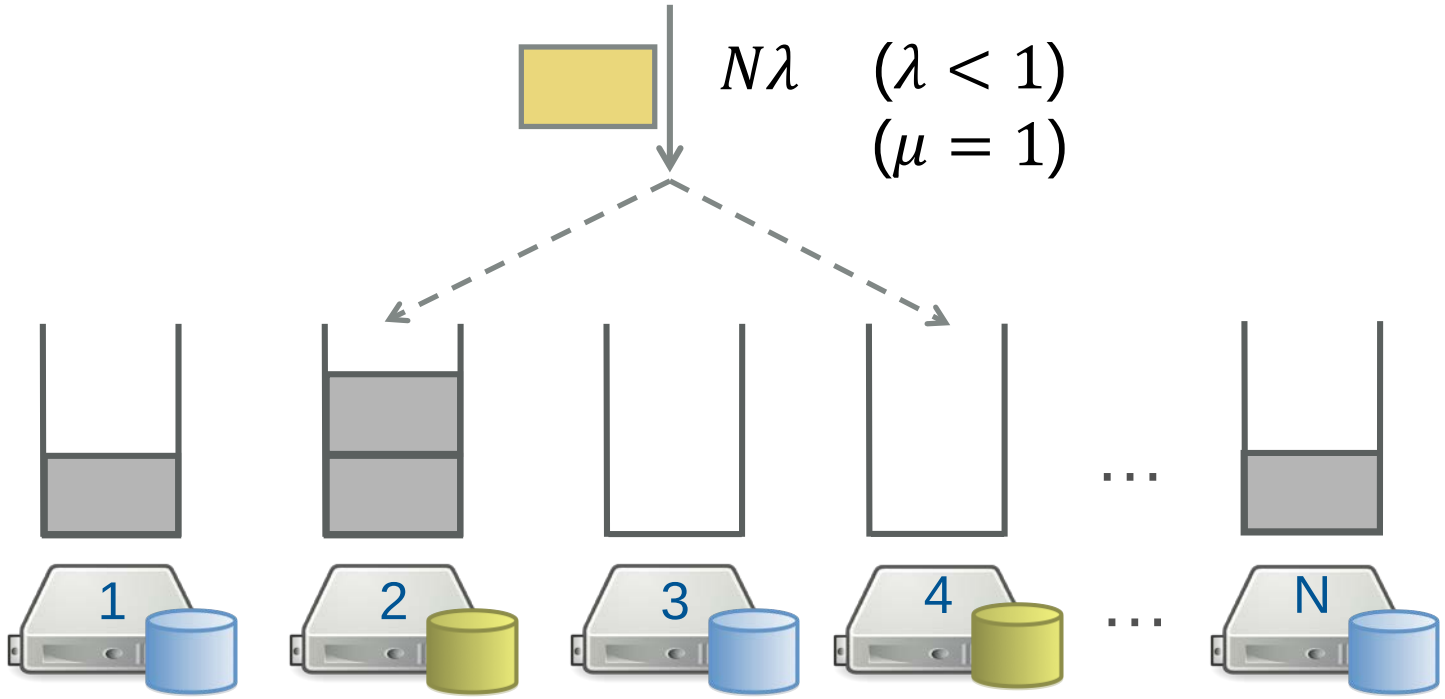
$$\lim_{n\to\infty} \Pr(Q \geq i) = \lambda^{\frac{d^i-1}{d-1}}$$ doubly exponential

Example: d = 2



Equivalent model: Simple Load-Balancing for Data Locality

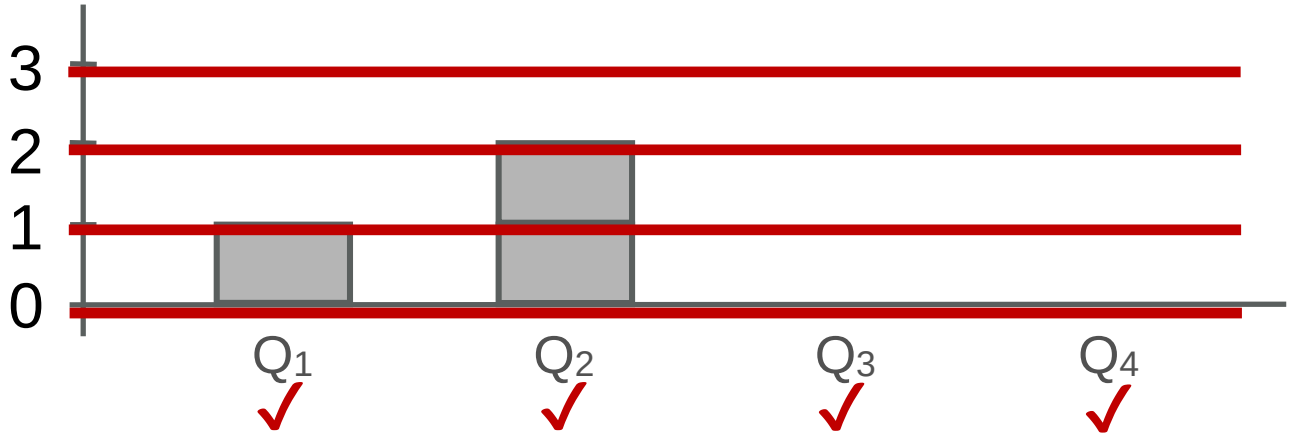
- Each data chunk is in *2* servers: $\binom{N}{2}$ places where a data chunk can be
- Arrival checks the servers that have its data and joins the shortest one
- Equivalent to the Power-of-*2*-Choices



State Representation

- Queue length vector: $Q = (Q_1, Q_2, Q_3, Q_4) = (1, 2, 0, 0)$
- Equivalent representation:

$s_i(t)$: fraction of queues with at least i tasks at time t



$s_3 = 0$

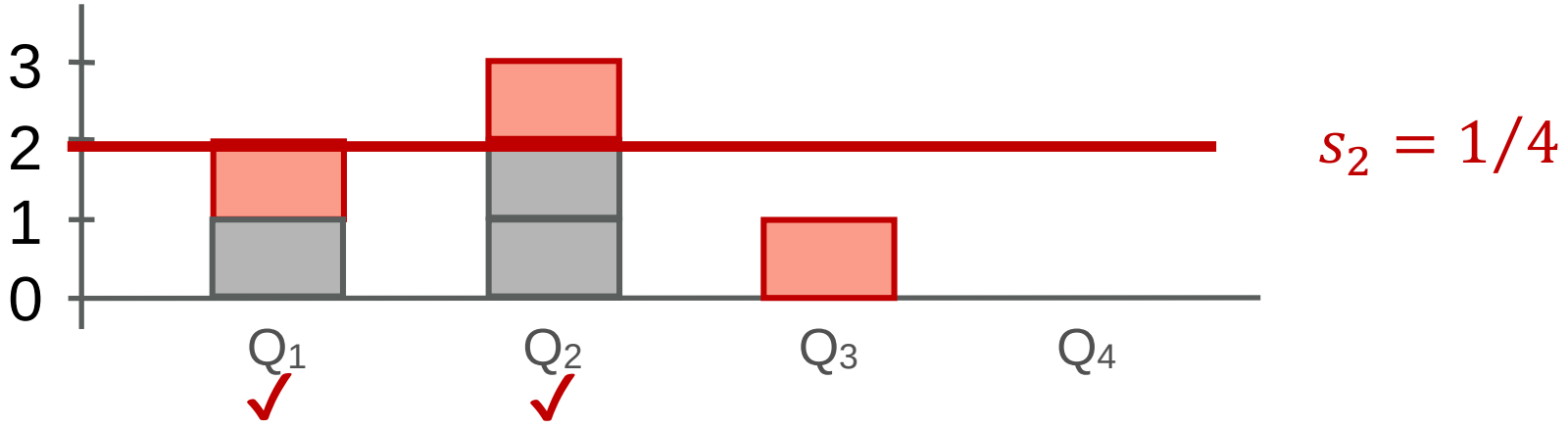
$s_2 = 1/4$

$s_1 = 2/4 = 1/2$

$s_0 = 4/4 = 1$

Dynamics

- $s_i(t)$: fraction of queues with at least i tasks at time t
- When does s_i change?
  - s_i does not change when a task arrives to queue with $< i - 1$ tasks or $\geq i$ tasks
  - $s_i \rightarrow s_i + 1/N$ when a task arrives to a queue with $i - 1$ tasks
  - s_i does not change when a task departures from queue with $< i$ tasks or $\geq i + 1$ tasks
  - $s_i \rightarrow s_i - 1/N$ when a tasks departs from a queue with i tasks



$s_2 = 1/4$

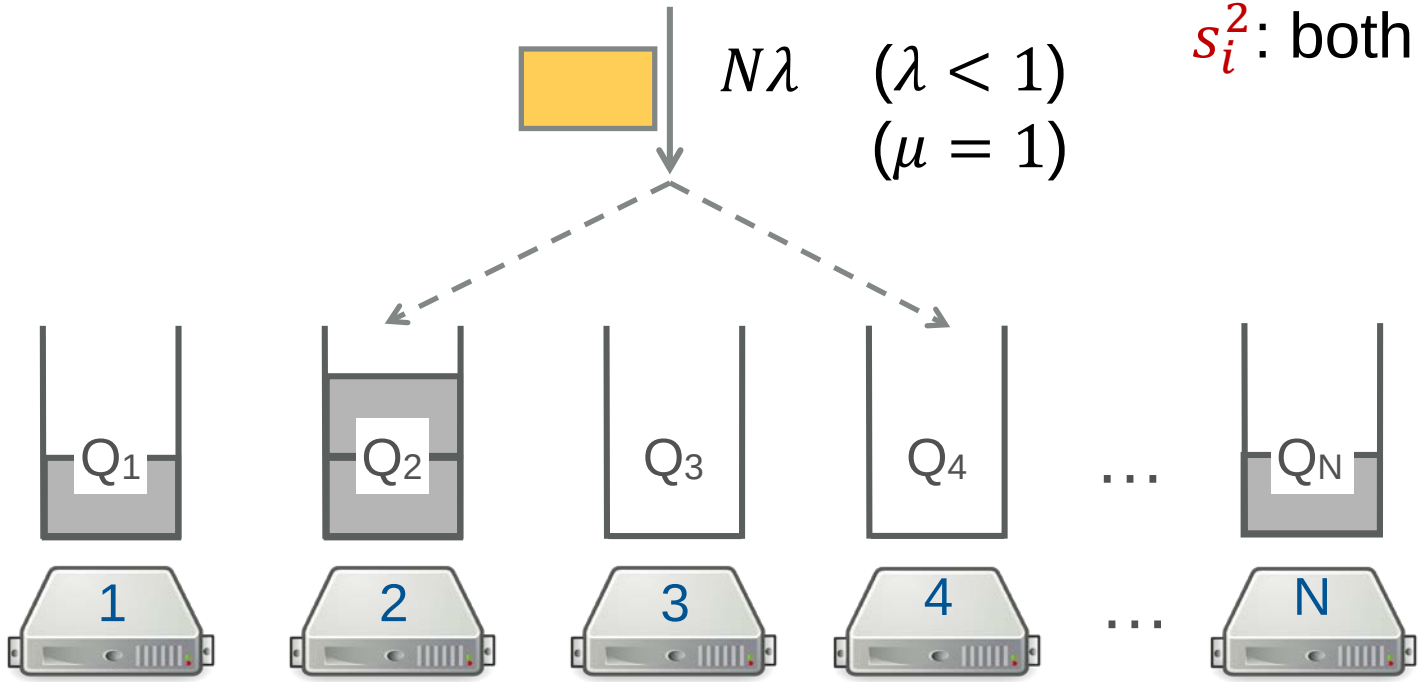
Power-of-2-Choices: Arrivals

- $s_i(t+\delta) = s_i + 1/N$ when a task arrives to a queue with $i-1$ tasks
- Happens when an arrival chooses two queues where the shorter one has $i-1$ tasks

Probability $\approx N\lambda\delta\left(s_{i-1}^2 - s_i^2\right)$

s_{i-1}^2: both queues $\geq i-1$

s_i^2: both queues $\geq i$



Power-of-2-Choices: Departures

- $s_i(t+\delta) = s_i - 1/N$ when a tasks departs from a queue with i tasks
- # queue with i tasks = $N(s_i - s_{i+1})$
- Probability $\approx N(s_i - s_{i+1})\delta$

Mean-Field Analysis

- Drift:

$$\mathbb{E}\left[\frac{s_i(t+\delta)-s_i(t)}{\delta}\middle| s\right] = \frac{1}{\delta}\left(\frac{1}{N}\overbrace{N\lambda\delta\left(s_{i-1}^2 - s_i^2\right)}^{\text{arrival}} - \frac{1}{N}\overbrace{N(s_i - s_{i+1})\delta}^{\text{departure}}\right)$$

$$= \lambda\left(s_{i-1}^2 - s_i^2\right) - (s_i - s_{i+1})$$

- Mean field approximation:

as $n \to \infty$ (not proved here)

$$\frac{ds_i}{dt} = \lambda\left(s_{i-1}^2 - s_i^2\right) - (s_i - s_{i+1})$$

- Use the fixed point of this set of differential equations as an approximation for the original system

Solving for the Fixed Point

$$\frac{ds_i}{dt} = \lambda\left(s_{i-1}^2 - s_i^2\right) - (s_i - s_{i+1})$$

- Fixed point: $\lambda\left(s_{i-1}^2 - s_i^2\right) = (s_i - s_{i+1})$
- Sufficient condition

$$\lambda s_i^2 = s_{i+1}, i = 1, 2, \ldots$$

- Solution: $s_i = \lambda^{2^i - 1}, i = 1, 2, \ldots$
- Queue length probability decreases double exponentially
- Uniqueness (not proved here)

Steps Involved in the MFA

- Change the state description in terms of fractions of queues with at least a certain queue length
- Compute

$$\lim_{\delta \to 0} E\left(\frac{s_i(t+\delta) - s_i(t)}{\delta} | s_i(t) = s_i\right)$$

- Use the above as an approximation to $\dot{s}_i$
- Study the differential equations

Power-of-*d*-Choices

- Arrival: $s_i(t+\delta) = s_i + 1/N$ when a task arrives to a queue with $i-1$ tasks

$$\text{Probability} \approx N\lambda\delta\left(s_{i-1}^d - s_i^d\right)$$

- Departure: same as power-of-2-choices

- Drift:

$$\mathbb{E}\left[\frac{s_i(t+\delta) - s_i(t)}{\delta}\middle| s\right] = \frac{1}{\delta}\left(\frac{1}{N}\overbrace{N\lambda\delta\left(s_{i-1}^d - s_i^d\right)}^{\text{arrival}} - \frac{1}{N}\overbrace{N(s_i - s_{i+1})\delta}^{\text{departure}}\right)$$

$$= \lambda\left(s_{i-1}^d - s_i^d\right) - (s_i - s_{i+1})$$

$$\downarrow \quad \text{as } n \to \infty \text{ (need to prove)}$$

- Mean field analysis

$$\frac{ds_i}{dt} = \lambda\left(s_{i-1}^d - s_i^d\right) - (s_i - s_{i+1})$$

Power-of-*d*-Choices: Fixed Point

$$\frac{ds_i}{dt} = \lambda\left(s_{i-1}^d - s_i^d\right) - (s_i - s_{i+1})$$

- Fixed point: $\lambda\left(s_{i-1}^d - s_i^d\right) - (s_i - s_{i+1}) = 0$
- Sufficient condition

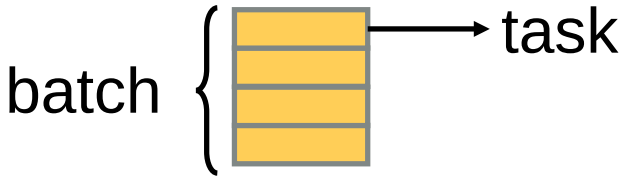
$$\lambda s_i^d = s_{i+1}, i = 1, 2, \ldots$$

- Solution:

$$s_i = \lambda^{\frac{d^i - 1}{d-1}}, i = 1, 2, \ldots$$

Batch-Filling

- Batch arrivals: each batch has m tasks
- Poisson batch arrivals with rate $N\lambda/m$
  - Total task arrival rate is still $N\lambda$
- Each batch arrival checks md queues
- d: probe ratio



Batch-Filling

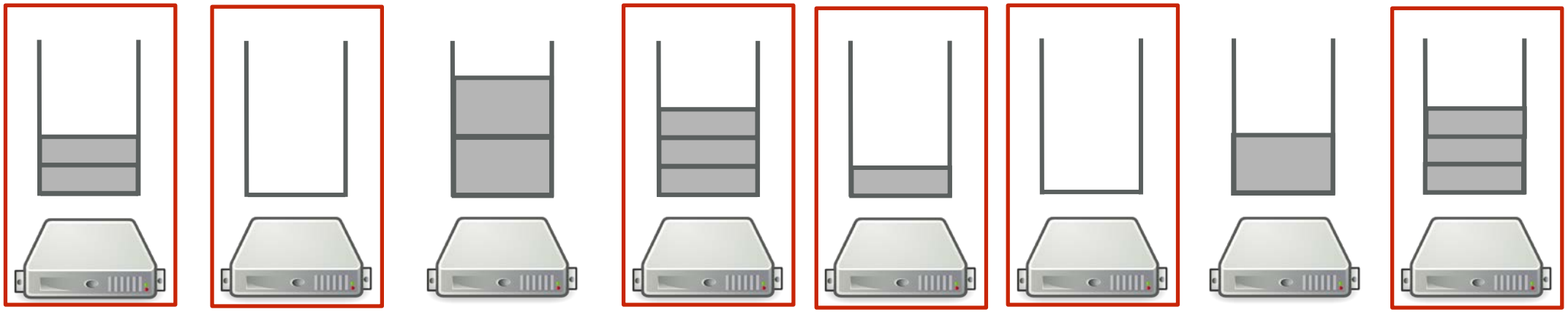
Waterfilling:

- Fill the smallest queue until it equals the second smallest queue
- Then fill these until they become equal to the third smallest queue, and so on…

Example: m=4, d = 1.5

$N\lambda \quad (\lambda < 1)$

$(\mu = 1)$



Batch-Filling

- Queue length:

$$\lim_{n\to\infty} \Pr(Q = i) = \begin{cases} 1-\lambda & i = 0, \\ (1-\lambda)\lambda d(1+\lambda d)^{i-1} & 1 \le i \le \bar{Q}-1, \\ 1-(1-\lambda)d(1+\lambda d)^{\bar{Q}-1} & i = \bar{Q}, \\ 0 & otherwise, \end{cases}$$

where

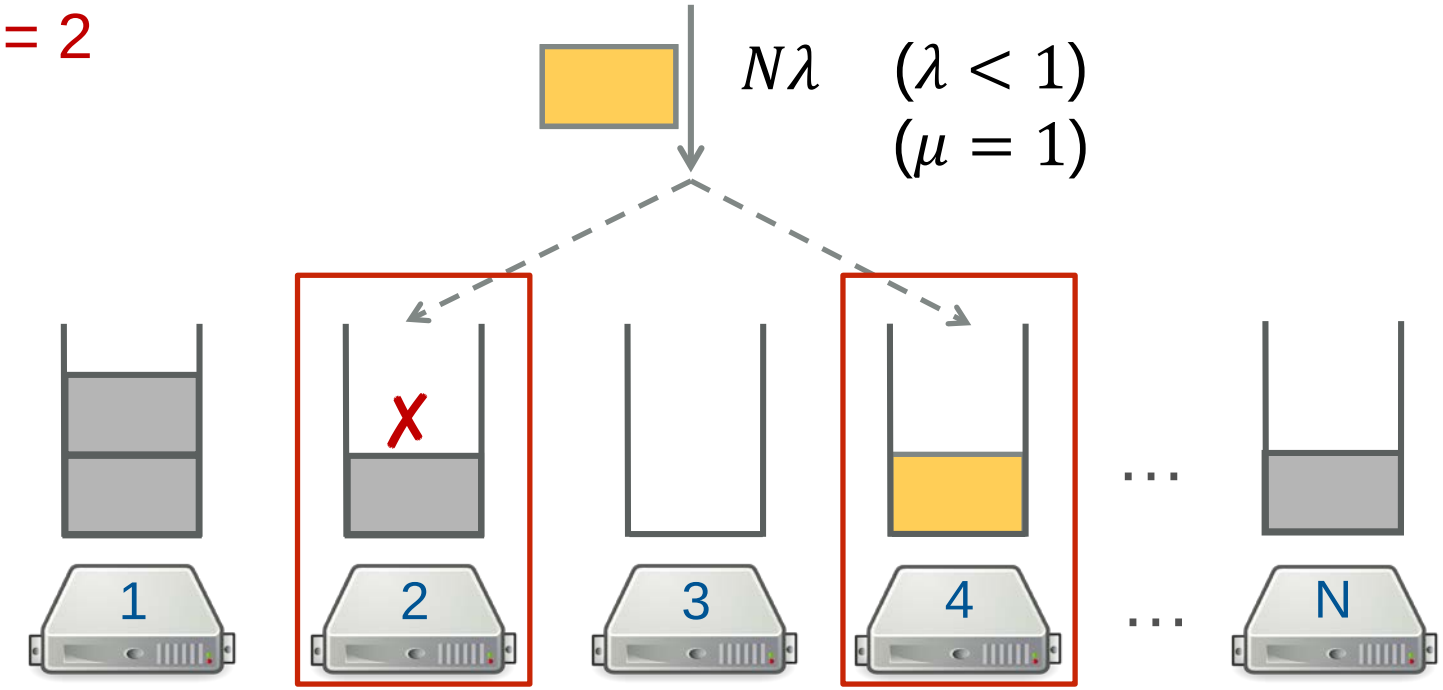
$$\bar{Q} = \left\lceil \frac{\log\left(\frac{1}{1-\lambda}\right)}{\log(1+\lambda d)} \right\rceil$$

- Finite queue length

Redundancy-*d*

- Each arrival makes copies at *d* servers chosen at random
- The other copies are killed once one copy is completed
- Harder to analyze

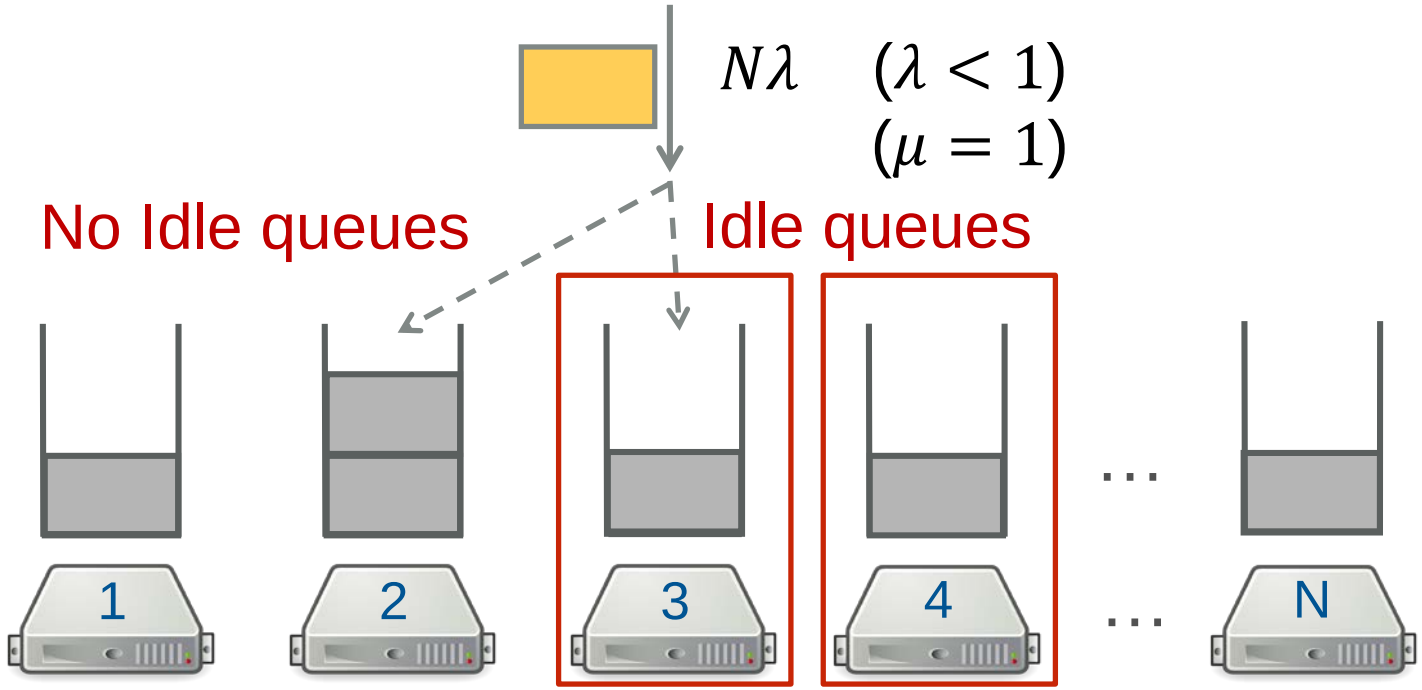
Example: d = 2



Join-the-Idle-Queue

- An arrival is routed to one of the idle servers at random if there is any; if not, the arrival is routed to a server at random
- Queue length:

$$\lim_{n\to\infty} \Pr(Q \geq i) = 0, \qquad i \geq 2$$



Incomplete List of References

- PoD: Vvedenskaya, Dobrushin, Karpelevich (1996), Mitzenmacher (1996)
- Batch: Ying, S., Kang (2015, 2017)
- JIQ: Lu et al (2011), Stolyar (2015,…)
- Redundancy: Gardner et al (2015,…)

- Many other scaling regimes: Borst SIGMETRICS 2017 Talk

Heavy-Traffic Analysis for Discrete-Time Systems

R. Srikant and Weina Wang
University of Illinois at Urbana-Champaign

Outline

- Kingman Bound for a Single Queue
- Join-the-Shortest-Queue (JSQ) Routing Policy (Eryilmaz, S., 2012)
- JSQ-MaxWeight for Scheduling with Data Locality (Wang et al, 2013)

Kingman Bound for a Single Queue

Single Queue

- $q(k+1) = q(k) + a(k) - s(k) + u(k)$
  - $a(k)$: # arrivals
  - $s(k)$: # potential departures
  - $u(k)$: unused service

- Want to bound $\mathbb{E}[q(k)]$ in steady state
- Set the drift of Lyapunov function $V(q) = q^2$ to zero:

$$\mathbb{E}[q^2(k+1)] - \mathbb{E}[q^2(k)] = 0$$

- Drift equation:

$$\begin{aligned} & q^2(k+1) - q^2(k) \\ = & \ 2q(k)\big(a(k) - s(k)\big) + \big(a(k) - s(k)\big)^2 \\ & +2u(k)\big(q(k) + a(k) - s(k) + u(k)\big) - u^2(k) \end{aligned}$$

Drift Equation

$$q^2(k+1) - q^2(k)$$
$$= 2q(k)\big(a(k) - s(k)\big) + \big(a(k) - s(k)\big)^2$$
$$+2u(k)\big(q(k) + a(k) - s(k) + u(k)\big) - u^2(k)$$

- $\mathbb{E}\big[q(k)\big(a(k) - s(k)\big)\big] = (\lambda - \mu)\mathbb{E}[q(k)]$
- $\mathbb{E}\left[\big(a(k) - s(k)\big)^2\right] \to \sigma^2$ as $\lambda \to \mu$

$$q(k+1) = q(k) + a(k) - s(k) + u(k)$$

In each time slot *k*,
$a(k)$: # arrivals
$s(k)$: # potential departures
$u(k)$: unused service

Drift Equation (cont'd)

$$q(k+1) = q(k) + a(k) - s(k) + u(k)$$

In each time slot k,
$a(k)$: # arrivals
$s(k)$: # potential departures
$u(k)$: unused service

$$q^2(k+1) - q^2(k)$$
$$= 2q(k)\big(a(k) - s(k)\big) + \big(a(k) - s(k)\big)^2$$
$$+ 2u(k)\big(q(k) + a(k) - s(k) + u(k)\big) - u^2(k)$$

- $q(k+1)u(k) = 0 \quad \forall k$
- $\mathbb{E}[u^2(k)] \le s_{\max}(\mu - \lambda)$ since

$u^2(k) \le s_{\max}\, u(k)$, and

$$\mathbb{E}[q(k+1)] = \mathbb{E}[q(k)] \Rightarrow \mathbb{E}[a(k) - s(k) + u(k)] = 0 \Rightarrow \mathbb{E}[u(k)] = \mu - \lambda$$

Kingman Bound

$$q(k+1) = q(k) + a(k) - s(k) + u(k)$$

In each time slot *k*,
$a(k)$: # arrivals
$s(k)$: # potential departures
$u(k)$: unused service

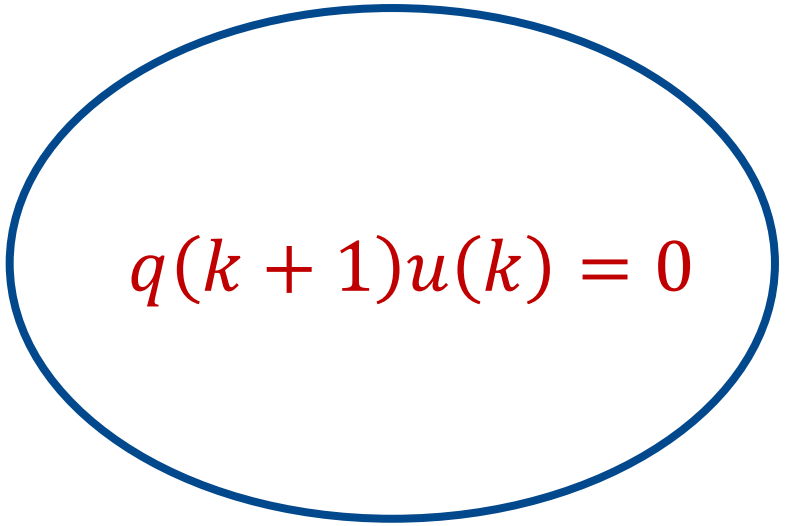
$$q^2(k+1) - q^2(k)$$
$$= 2q(k)\big(a(k) - s(k)\big) + \big(a(k) - s(k)\big)^2$$
$$+2u(k)\big(q(k) + a(k) - s(k) + u(k)\big) - u^2(k)$$

- $\mathbb{E}[q^2(k+1)] - \mathbb{E}[q^2(k)] = 0$ in steady state yields

$$\mathbb{E}[q] = \frac{\mathbb{E}[(a-s)^2]}{2(\mu-\lambda)} + o\left(\frac{1}{\mu-\lambda}\right)$$

This term is small compared to the first term when $\lambda \to \mu$

Key Fact about Unused Service

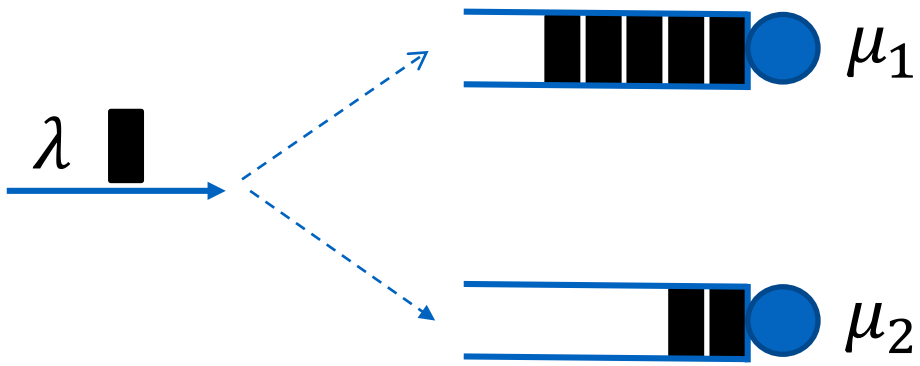


Join-the-Shortest-Queue (JSQ) Routing Policy

$$\big(q_1(k+1)+q_2(k+1)\big)u_1(k)\approx 0$$

JSQ

- Discrete-time model
- Route packet arrivals in each time slot to the shorter of the two queues, breaking ties at random
- Well known that JSQ is heavy-traffic optimal; will derive this result using the Kingman-type drift argument



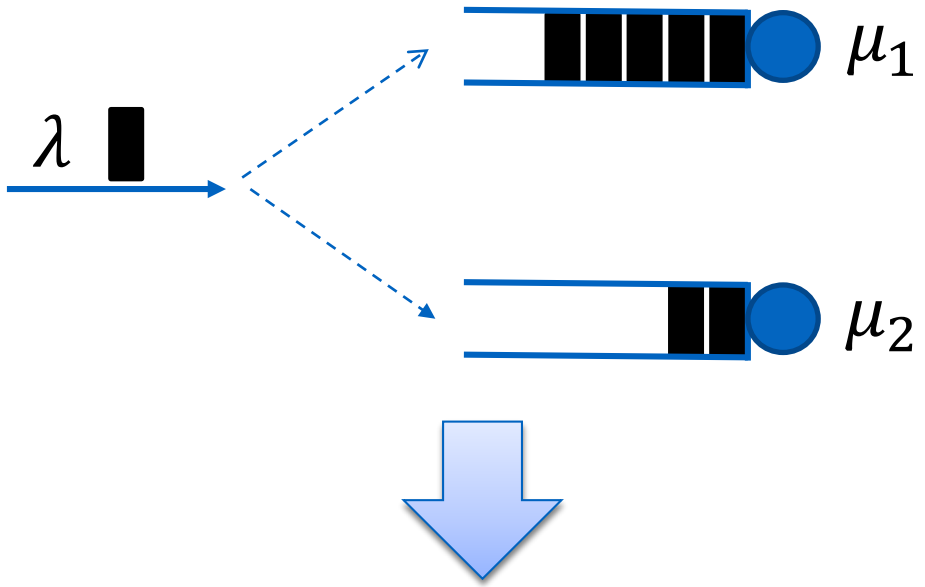
Universal Lower Bound: Resource Pooling

- For any routing policy, $q_1 + q_2$ is lower bounded by the queue length in the system where the service resource is pooled together

- Heavy-traffic parameter $\epsilon = \mu_1 + \mu_2 - \lambda$

- By the Kingman bound for the single queue system,

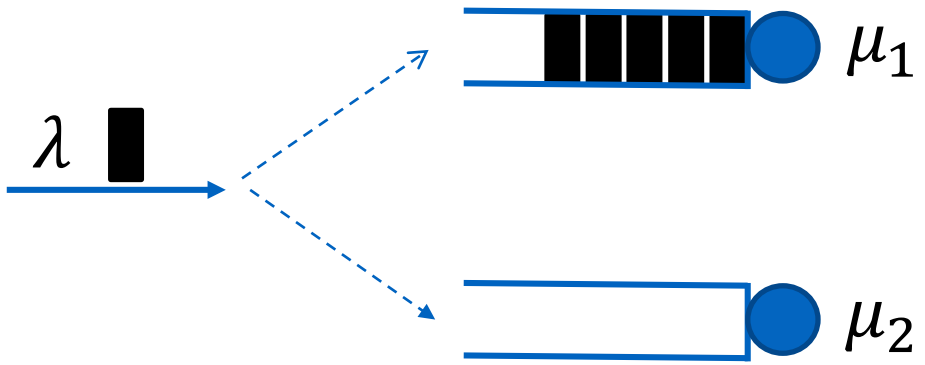
$$\mathbb{E}[q_1 + q_2] \geq \frac{\mathbb{E}[(a-s)^2]}{2\epsilon} + o\left(\frac{1}{\epsilon}\right)$$



Lower bounding system

JSQ: What can go wrong?

- Because of the inherent randomness in the service times, one queue can become empty when the other is not
- This means that one server is idling (i.e., can have unused service) when it can be doing work
- However, in heavy traffic, this should happen rarely under the join-the-shortest queue policy



State-Space Collapse

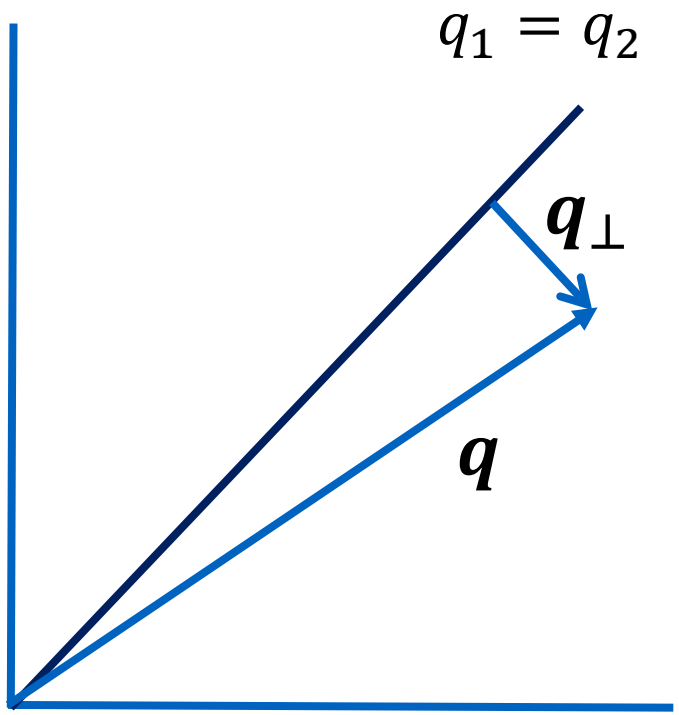
- For the servers to avoid unnecessarily idling we need to show $q_1 \approx q_2$ under JSQ

- What we actually prove is that

$$\mathbb{E}[\|q_\perp\|^2] \leq M,$$

where M does not depend on the heavy-traffic parameter $\epsilon = \mu_1 + \mu_2 - \lambda$

  - This is called state-space collapse because this means $\boldsymbol{q}_\perp$ is small compared to $q_1 + q_2$ which is $O\left(\frac{1}{\epsilon}\right)$ in expectation (why?)



Upper Bound for JSQ

- Set the drift of $V(q) = (\sum_l q_l)^2$ equal to zero:

$$\mathbb{E}\left[V\left(q(k+1)\right)\right] - \mathbb{E}\left[V\left(q(k)\right)\right] = 0$$

- Why this choice of $V(q)$?
  - From the state-space collapse result, we expect the queues to behave like a single queue as in the lower bound; all queues are roughly equal, so they would all hit zero "simultaneously"
  - So we expect $\sum_l q_l$ to behave like a single-server queue

Drift Equation

- The terms in the drift equation look very similar to the lower bound, except for the red term below:

$$\begin{aligned}
&\big(q_1(k+1)+q_2(k+1)\big)^2-\big(q_1(k)+q_2(k)\big)^2\\
=\;&2\big(q_1(k)+q_2(k)\big)\big(a_1(k)+a_2(k)-s_1(k)-s_2(k)\big)\\
&+\big(a_1(k)+a_2(k)-s_1(k)-s_2(k)\big)^2\\
&+2\big(u_1(k)+u_2(k)\big)\big(q_1(k+1)+q_2(k+1)\big)\\
&-\big(u_1(k)+u_2(k)\big)^2
\end{aligned}$$

- Lower bounding system:

$$\begin{aligned}
&q^2(k+1)-q^2(k)\\
=\;&2q(k)\big(a(k)-s(k)\big)+\big(a(k)-s(k)\big)^2\\
&+2u(k)q(k+1)-u^2(k)
\end{aligned}$$

Using State-Space Collapse

- The terms in the drift equation look very similar to the lower bound, except for terms of the form:

$$\big(q_1(k+1)+q_2(k+1)\big)\big(u_1(k)+u_2(k)\big)$$

- Note that $q_1(k+1)u_1(k)=0$, but $q_2(k+1)u_1(k)\neq 0$

- But, from state-space collapse, $q_1(k+1)\approx q_2(k+1)$, and thus,

$$q_2(k+1)u_1(k)\approx 0$$

Upper Bound for JSQ

- $\mathbb{E}\left[V\left(q(k+1)\right)\right] - \mathbb{E}\left[V\left(q(k)\right)\right] = 0$ in steady state yields

$$
\begin{aligned}
&\mathbb{E}\left[\left(q_1(k+1)+q_2(k+1)\right)^2 - \left(q_1(k)+q_2(k)\right)^2\right] && 0\\
&= \mathbb{E}\left[2\left(q_1(k)+q_2(k)\right)\left(a_1(k)+a_2(k)-s_1(k)-s_2(k)\right)\right] && -2\epsilon\mathbb{E}[q_1+q_2]\\
&+\mathbb{E}\left[\left(a_1(k)+a_2(k)-s_1(k)-s_2(k)\right)^2\right] && \mathbb{E}[(a-s)^2]\\
&+\mathbb{E}\left[2\left(u_1(k)+u_2(k)\right)\left(q_1(k+1)+q_2(k+1)\right)\right] && o(1)\\
&-\mathbb{E}\left[\left(u_1(k)+u_2(k)\right)^2\right] && o(1)
\end{aligned}
$$

- Thus,

$$\mathbb{E}[q_1+q_2] = \frac{\mathbb{E}[(a-s)^2\,]}{2\epsilon} + o\left(\frac{1}{\epsilon}\right)$$

Heavy-Traffic Delay Optimality

- Under JSQ,

$$\mathbb{E}[q_1 + q_2] = \frac{\mathbb{E}[(a-s)^2]}{2\epsilon} + o\left(\frac{1}{\epsilon}\right)$$

- This coincides with the lower bound for any policy

$$\mathbb{E}[q_1 + q_2] \geq \frac{\mathbb{E}[(a-s)^2]}{2\epsilon} + o\left(\frac{1}{\epsilon}\right)$$

- JSQ asymptotically minimizes the backlog in heavy traffic
- By Little's law, JSQ asymptotically minimizes the average delay in heavy traffic

Key Steps

- Lower Bound
  - Resource Pooling
- Establish State-Space Collapse
  - Using insight about why the algorithm might achieve the lower bound
  - We haven’t yet discussed how to show state-space collapse
- Obtain an Upper Bound
  - Again using the insight from the lower bound and the state-space collapse to choose an appropriate function whose drift is equal to zero

State-Space Collapse

(Hajek, 1982)

X is Markov chain, V(x) is some function (satisfying a certain condition) defined over the state-space of the Markov chain. If

$$E(V(X_{k+1}) - V(x)|X_k = x) \leq -\delta,$$

for $V(x) \geq B$, then

$$\lim_{k\to\infty} E\left(e^{\theta V(X_k)}\right) \leq M_\delta$$

A Useful Property of JSQ

- Define $W(\boldsymbol{q}) = \|\boldsymbol{q}_{\perp}\|$
- In the 2-d case, $W(\boldsymbol{q}) = \frac{1}{\sqrt{2}}|q_1 - q_2|$

- Drift: $\mathbb{E}\left[W\left(\boldsymbol{q}(k+1)\right) - W\left(\boldsymbol{q}(k)\right) \mid \boldsymbol{q}(k) = \boldsymbol{q}\right]$

- JSQ:
  - If $q_1 > q_2$, then q_1 decreases and q_2 increases, independent of ε
  - Similarly when $q_2 > q_1$
  - Conclusion: Drift is independent of ε

Moments & State-Space Collapse

- $E\left[W\left(\boldsymbol{q}(k+1)\right) - W\left(\boldsymbol{q}(k)\right)\middle|\boldsymbol{q}(k) = \boldsymbol{q}\right] \leq -\delta$, independent of the heavy-traffic parameter ϵ

- Following Hajek (1982), this implies the following steady-state estimate:

  $$E\left(\left\|\boldsymbol{q}_{\perp}\right\|\right) \leq M,$$

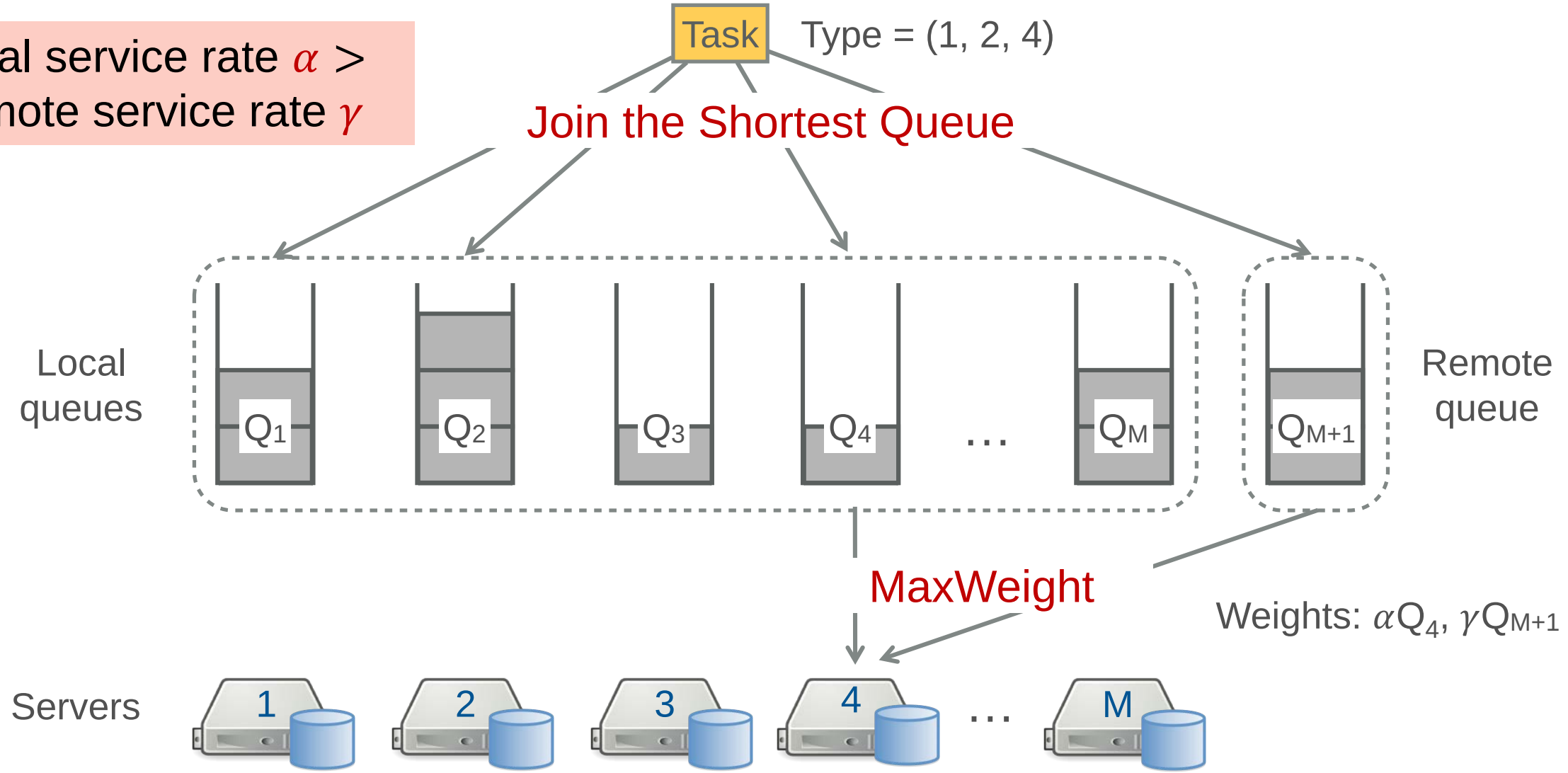
  independent of ϵ

- State-space collapse: Recall that $E\left(\sum_l q_l\right)$ is $\Omega\left(\frac{1}{\epsilon}\right)$; thus $\boldsymbol{q}_{\perp}$ is small compared to $\boldsymbol{q}$ in heavy-traffic

JSQ-MaxWeight for Scheduling with Data Locality

JSQ-MaxWeight

local service rate α > remote service rate γ



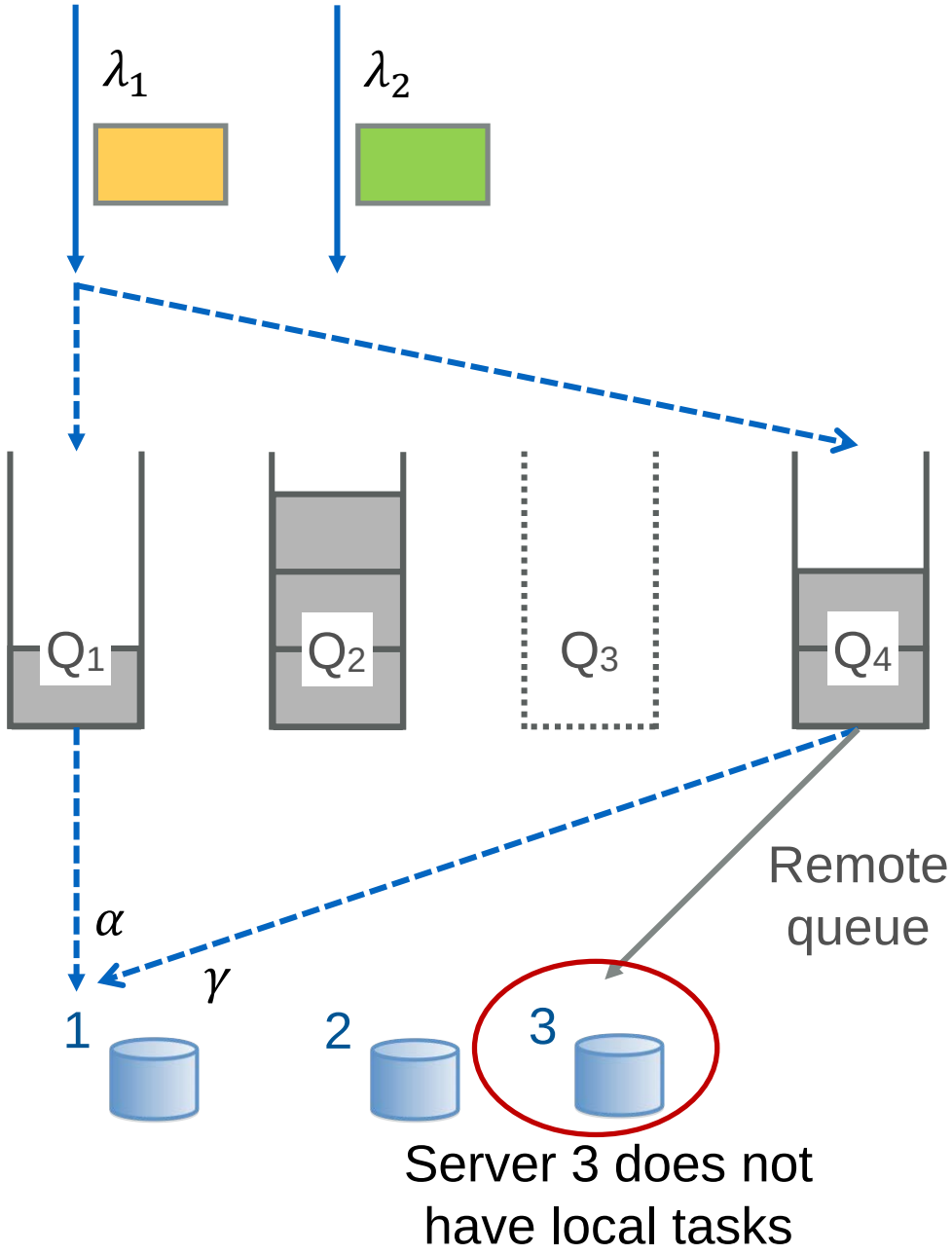
Heavy-Traffic Regime

- Some servers do not store data, so they do not have local tasks
  - Backup servers for peak hours
- Servers with data are overloaded

Example

- Server 3 does not have local tasks
- Task types: Type 1 is local to server 1, and Type 2 is local to server 2
- Servers with local tasks are overloaded: $\lambda_1 > \alpha, \lambda_2 > \alpha$
- Heavy-traffic parameter:

$$\epsilon = 2\alpha + \gamma - (\lambda_1 + \lambda_2)$$



Key Steps

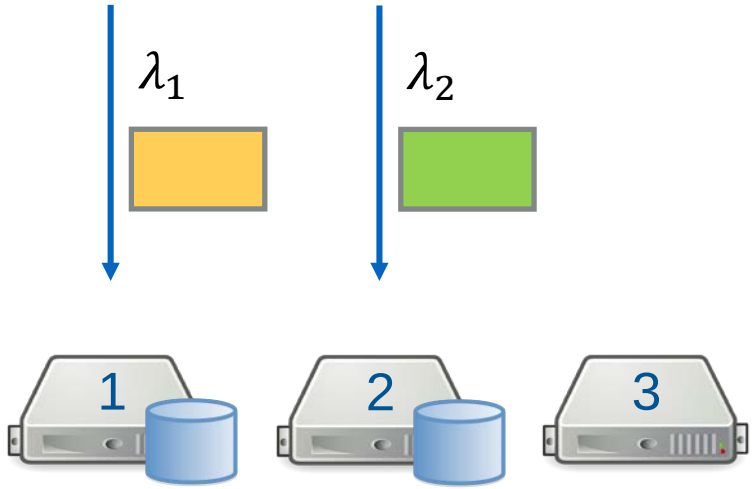
- Lower Bound
  - Resource Pooling

- Establish State-Space Collapse
  - Using insight about why the algorithm might achieve the lower bound

- Obtain an Upper Bound
  - Again using the insight from the lower bound and the state-space collapse to choose an appropriate function whose drift is equal to zero

Universal Lower Bound

- For any scheduling policy, the backlog is lower bounded by the queue length in the system where
  - Servers are running at maximum speeds
  - The service resource is pooled together



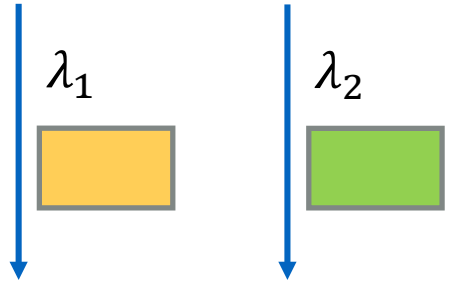
- By the Kingman bound for single queue,

$$\mathbb{E}[\text{backlog}] \geq \frac{\mathbb{E}[(a-s)^2]}{2\epsilon} + o\left(\frac{1}{\epsilon}\right)$$

Lower bounding system

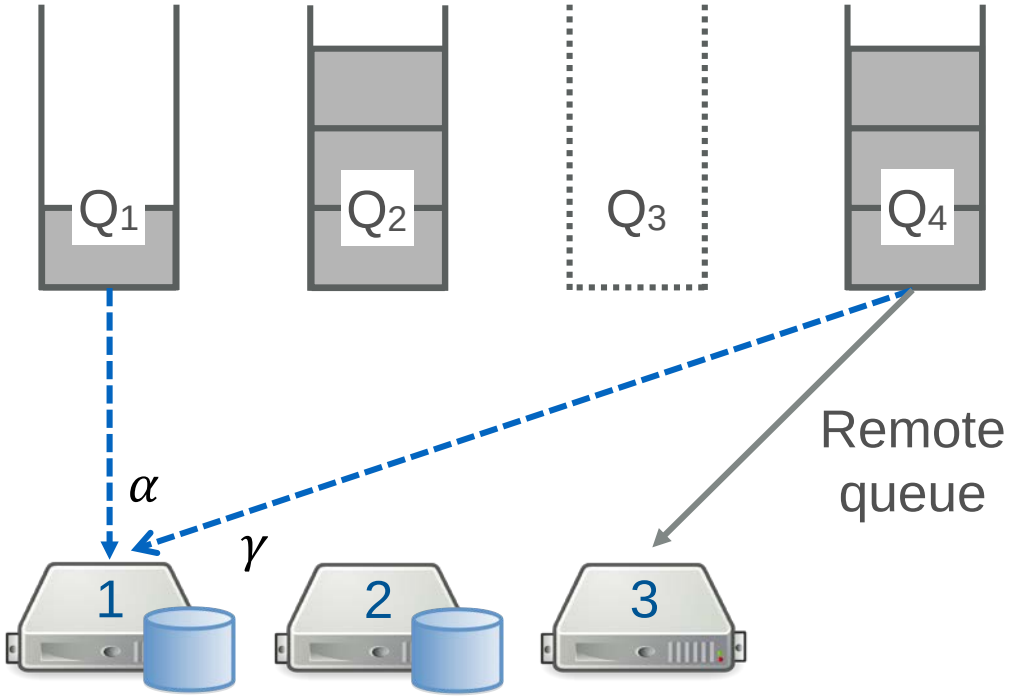
JSQ-MaxWeight: What can go wrong?

- Similar to JSQ, a server can be idling when there is still unfinished work in the system



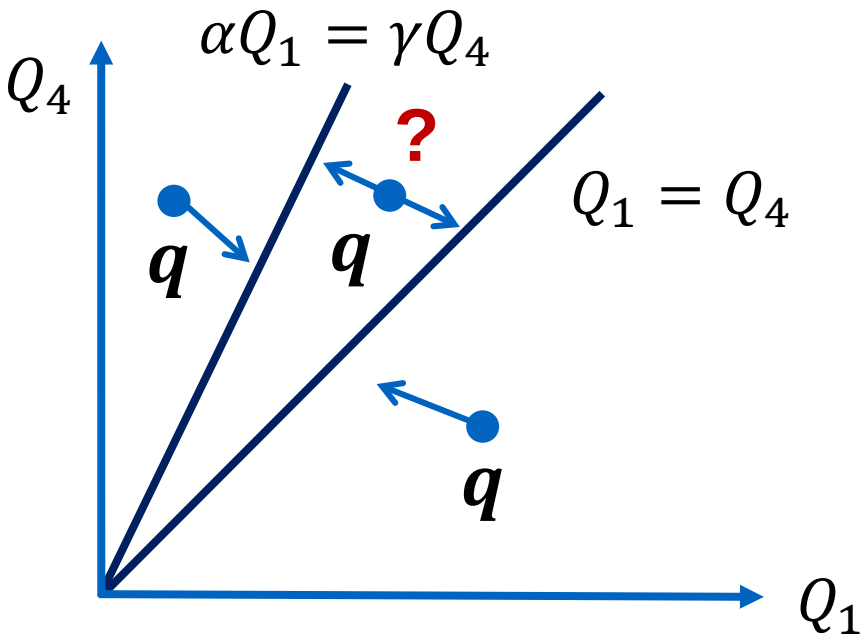
- A server can be working on a remote task when it has local tasks
- Then it is not running at full speed

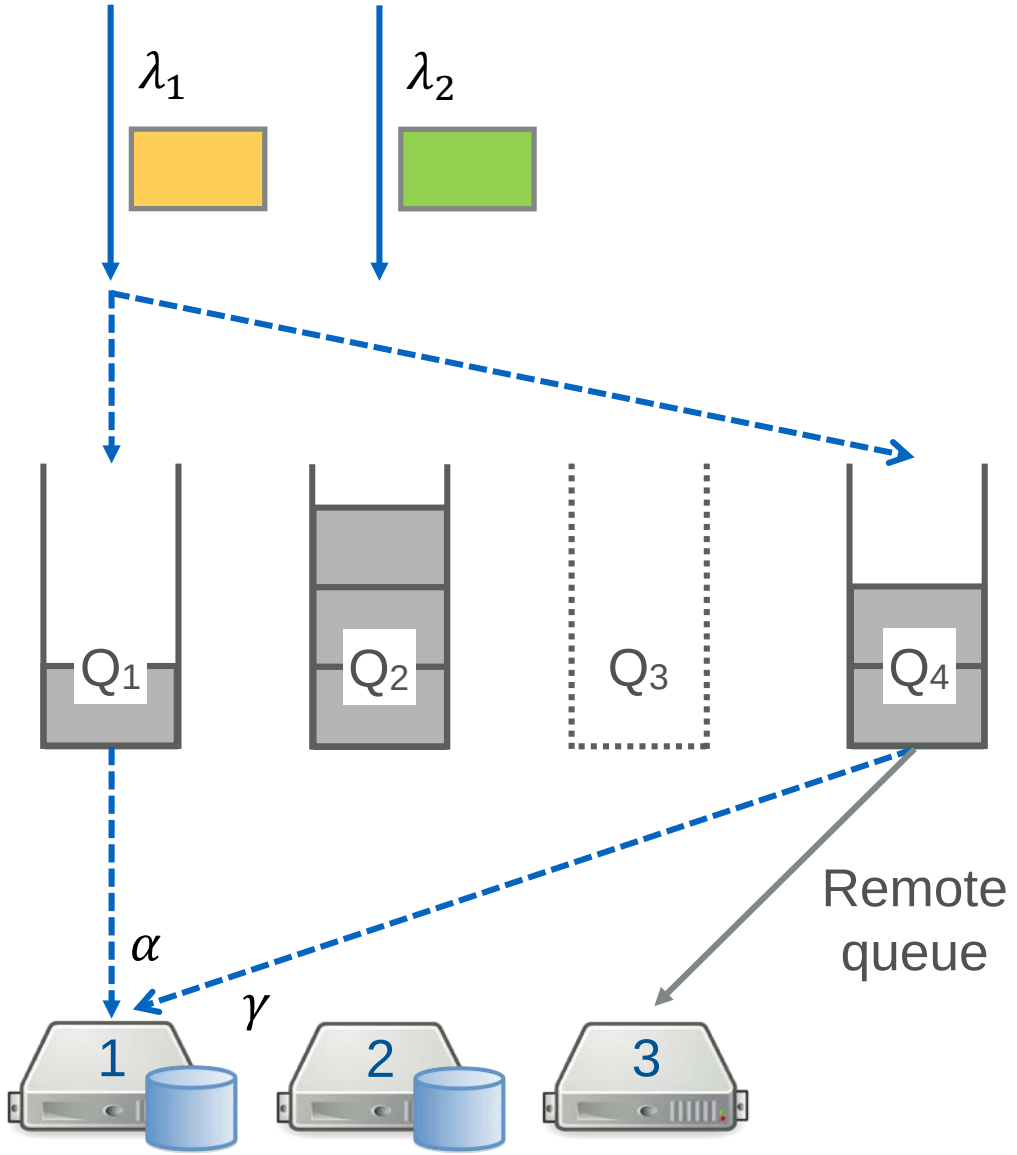
- Again, in heavy traffic, these should happen rarely under the JSQ-MaxWeight



State-Space Collapse

- JSQ tries to balance Q_1 and Q_4
- MaxWeight tries to balance αQ_1 and γQ_4
- Where does the state space collapses to?

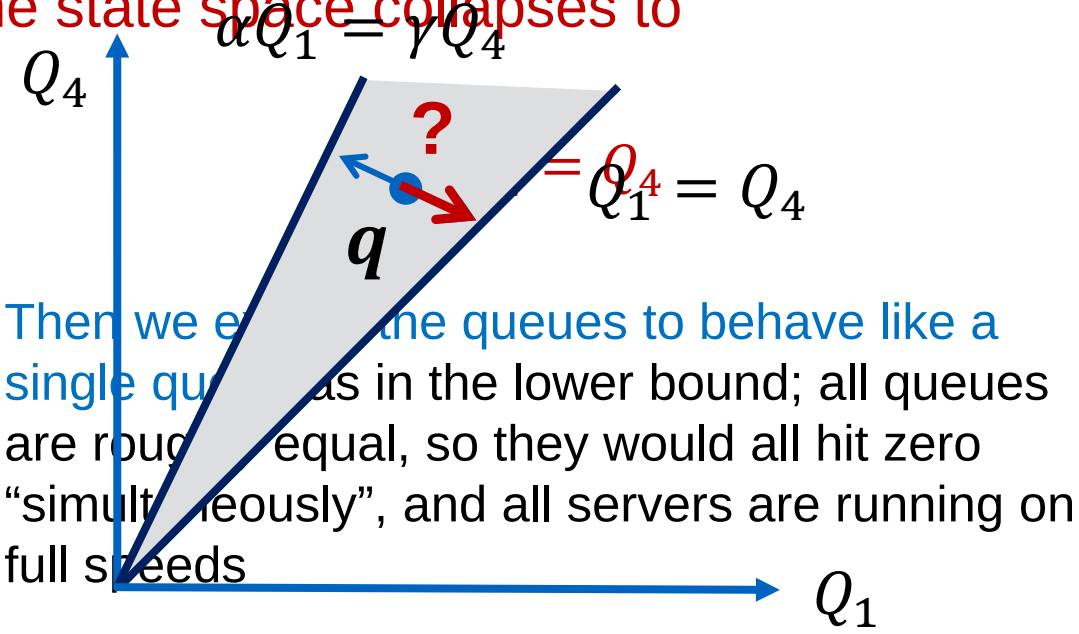




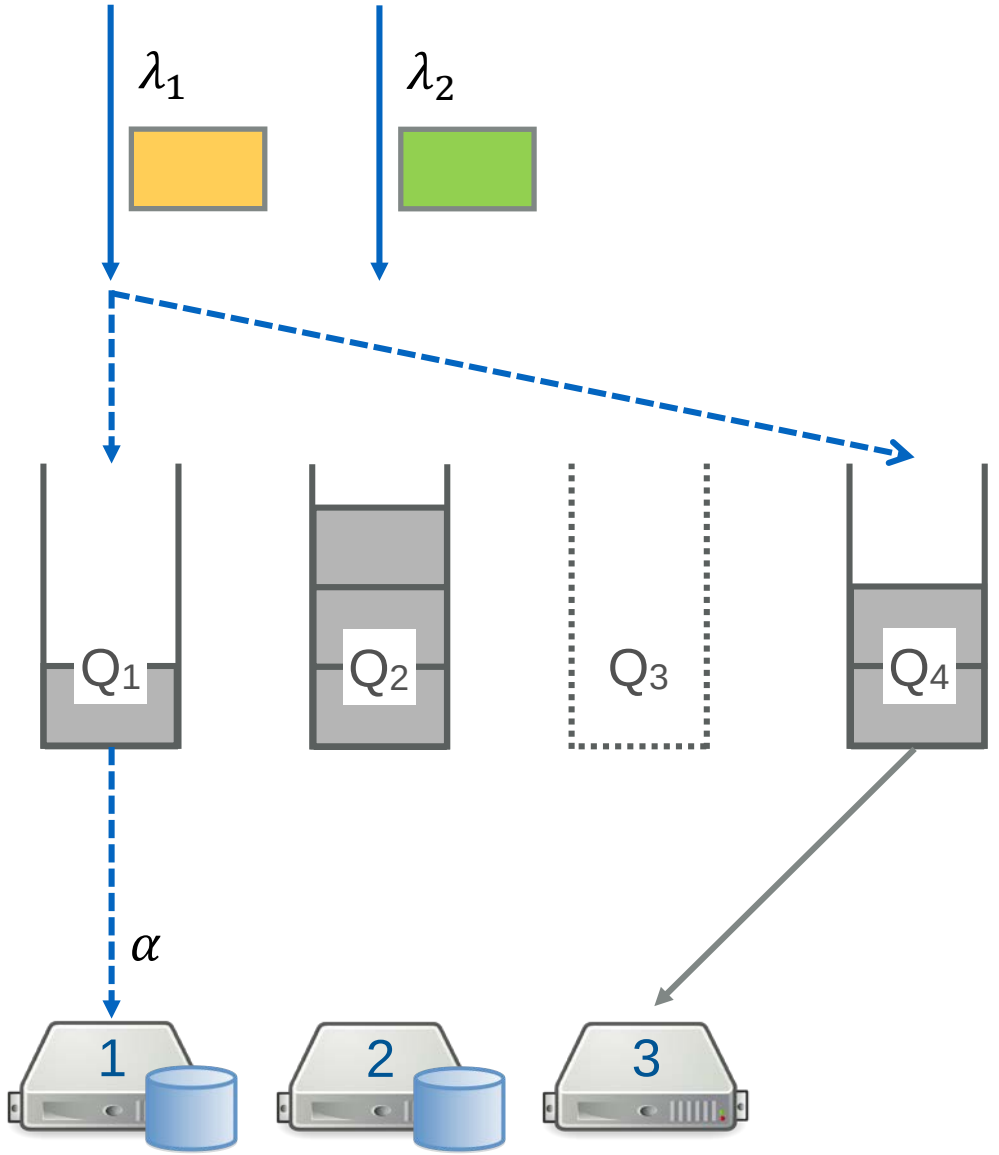
State-Space Collapse

- When $\alpha Q_1 \geq \gamma Q_4$, server 1 works on Q_1
- Then the system behaves like a JSQ system
- So the queue vector further goes towards $Q_1 = Q_4$

- The state space collapses to



  - Then we e[illegible]he queues to behave like a single qu[illegible]as in the lower bound; all queues are rou[illegible]equal, so they would all hit zero "simult[illegible]eously", and all servers are running on full s[illegible]eeds



Upper Bound

- Want to prove the system indeed behaves like a JSQ system
- That is to prove Server 1 serves Q_1 most of the time and Server 2 serves Q_2 most of the time
- Queue dynamics:

$$Q_m(k+1) = Q_m(k) + A_m(k) - S_m(k) + U_m(k), \qquad m = 1, 2, 4$$

- We prove that $\big(S_1(k), S_2(k), S_4(k)\big) \approx \big(S_1'(k), S_2'(k), S_4'(k)\big)$, where $S_m'(k)$'s are "ideal" service: $S_1'(k)$ and $S_2'(k)$ have rate α, $S_4'(k)$ has rate γ

Upper Bound

- Then obtaining the upper bound is similar to JSQ
- Setting the drift of $V(q) = (\sum_l q_l)^2$ equal to zero yields:

$$\mathbb{E}[\text{backlog}] \leq \frac{\mathbb{E}[(a-s)^2]}{2\epsilon} + o\left(\frac{1}{\epsilon}\right)$$

- Coincides the universal lower bound for any scheduling policy

Heavy-Traffic Analysis for Continuous-Time Systems

R. Srikant and Weina Wang
University of Illinois at Urbana-Champaign

Outline

- Single Queue in Continuous Time
- Join-the-Shortest-Queue in Continuous Time
- Proportionally Fair Bandwidth Sharing (Wang et al, in progress)

Single Queue in Continuous Time

Single Queue in Continuous Time

- Packet arrivals: Poisson process with rate λ
- Service time: exponential with mean $1/\mu$

- $n(t)$: # packets

  $n \to n+1$ with rate λ

  $n \to n-1$ with rate $\mu \mathbf{1}_{\{n>0\}}$ $\longrightarrow$ $\mu - \mu U$

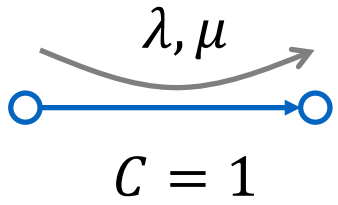
Load: $\rho = \lambda/\mu$

- Unused service: $U = 1 - \mathbf{1}_{\{n>0\}}$
- Note that

$$n \cdot U = 0$$

Equivalent Model: One Link, One Flow Type

- Flow arrivals: Poisson process with rate λ
- Each flow corresponds to the transfer of a file whose size is exponentially distributed with mean $1/\mu$



- $n(t)$: # of flows
- x: bandwidth allocated to each flow, $x = \begin{cases} \frac{1}{n}, & \text{if } n > 0 \\ 0, & \text{if } n = 0 \end{cases}$

Load: $\rho = \lambda/\mu$

$n \to n+1$ with rate λ

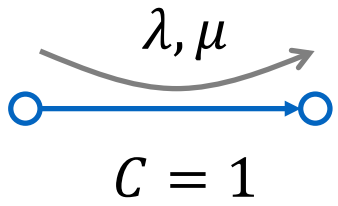
$n \to n-1$ with rate $nx\mu$ $\longrightarrow \mu - \mu U$

- Unused bandwidth: $U = 1 - nx = 1 - \mathbf{1}_{\{n>0\}}$
- Note that

$$n \cdot U = 0$$

Single Queue

- Want to bound $\mathbb{E}[n(t)]$ in steady state



- Set the drift of Lyapunov function $V(n) = n^2$ to zero:

$$\mathbb{E}\left[\left(n(t+\Delta t)\right)^2\right] = \mathbb{E}\left[\left(n(t)\right)^2\right]$$
$$\Leftrightarrow \mathbb{E}[\Delta\, n^2] = 0$$

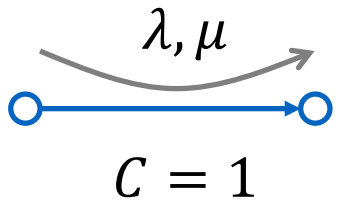
Load: $\rho = \lambda/\mu$

- Drift equation:

$$\begin{aligned}\Delta\, n^2 &= \lambda\left((n+1)^2 - n^2\right) + (\mu - \mu U)\left((n-1)^2 - n^2\right)\\ &= 2(\lambda - \mu + U)n + \lambda + \mu - \mu U\\ &= 2(\rho - 1)\cdot n\mu + 2U \cdot n\mu + \lambda + \mu - \mu U\end{aligned}$$

Bound on Backlog

$$\Delta n^2 = 2(\rho - 1) \cdot n\mu + 2U \cdot n\mu + \lambda + \mu - \mu U$$



Load: $\rho = \lambda/\mu$

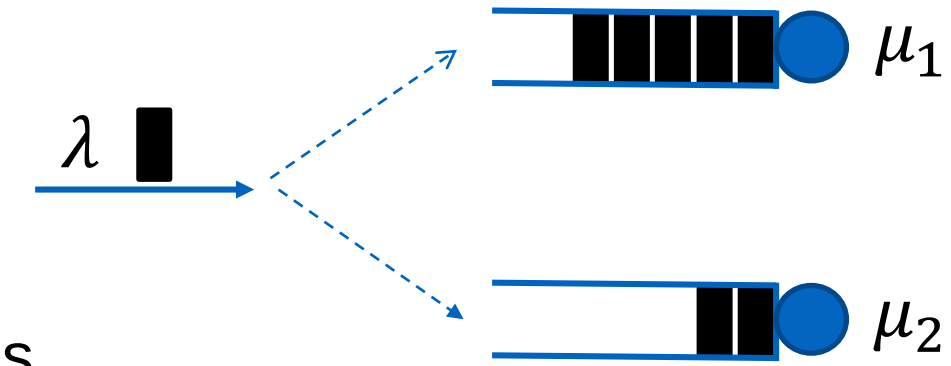
- $2U \cdot n\mu = 0$
- $\mathbb{E}[\mu - \mu U] = \lambda$: departure rate = arrival rate
- $\mathbb{E}[\Delta n^2] = 0$ in steady state: $0 = 2(\rho - 1) \cdot \mathbb{E}[n]\,\mu + 2\lambda$, which yields

$$\mathbb{E}[n] = \frac{\rho}{1 - \rho}$$

Join-the-Shortest-Queue in Continuous Time

JSQ

- Continuous-time model



- Route arrivals to the shorter of the two queues, breaking ties at random

$(n_1, n_2) \to (n_1 + 1, n_2)$ with rate $\lambda \mathbf{1}_{\{n_1 < n_2\}} + \frac{\lambda}{2} \mathbf{1}_{\{n_1 = n_2\}}$

$(n_1, n_2) \to (n_1, n_2 + 1)$ with rate $\lambda \mathbf{1}_{\{n_1 > n_2\}} + \frac{\lambda}{2} \mathbf{1}_{\{n_1 = n_2\}}$

$(n_1, n_2) \to (n_1 - 1, n_2)$ with rate $\mu_1 \mathbf{1}_{\{n_1 > 0\}} \longrightarrow \mu_1 - \mu_1 U_1$

$(n_1, n_2) \to (n_1, n_2 - 1)$ with rate $\mu_2 \mathbf{1}_{\{n_2 > 0\}} \longrightarrow \mu_2 - \mu_2 U_2$

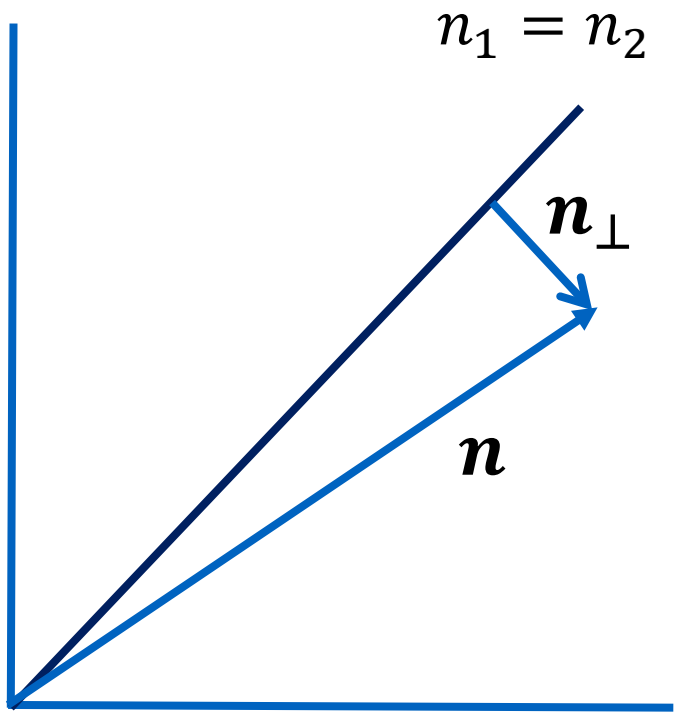
Unused service

$U_1 = 1 - \mathbf{1}_{\{n_1 > 0\}}$

$U_2 = 1 - \mathbf{1}_{\{n_2 > 0\}}$

State-Space Collapse

- Heavy traffic parameter $\epsilon = \mu_1 + \mu_2 - \lambda$
- Similar to the discrete-time model, we prove that $n_1 \approx n_2$ in heavy traffic



Upper Bound for JSQ

- Set the drift of $V(n) = (n_1 + n_2)^2$ equal to zero:

$$\mathbb{E}[\Delta(n_1 + n_2)^2] = 0$$

- Drift equation

$$\begin{aligned}
\Delta\,(n_1 + n_2)^2 &= \lambda\big((n_1 + n_2 + 1)^2 - (n_1 + n_2)^2\big) \\
&\quad +(\mu_1 - \mu_1 U_1 + \mu_2 - \mu_2 U_2)\big((n_1 + n_2 - 1)^2 - (n_1 + n_2)^2\big) \\
&= 2(n_1 + n_2)(\lambda - \mu_1 - \mu_2 + \mu_1 U_1 + \mu_2 U_2) \\
&\quad +(\lambda + \mu_1 - \mu_1 U_1 + \mu_2 - \mu_2 U_2) \\
&= 2(n_1 + n_2)(\lambda - \mu_1 - \mu_2) + 2(n_1 + n_2)(\mu_1 U_1 + \mu_2 U_2) \\
&\quad +(\lambda + \mu_1 - \mu_1 U_1 + \mu_2 - \mu_2 U_2)
\end{aligned}$$

Upper Bound for JSQ

$$\Delta\,(n_1+n_2)^2 = 2(n_1+n_2)(\lambda-\mu_1-\mu_2) + 2(n_1+n_2)(\mu_1U_1+\mu_2U_2) + (\lambda+\mu_1-\mu_1U_1+\mu_2-\mu_2U_2)$$

- Still focus on the term $(n_1+n_2)(\mu_1U_1+\mu_2U_2)$
- Note that $n_1U_1=0$, but $n_2U_1\neq 0$
- But, from state-space collapse, $n_1\approx n_2$, and thus, $n_2U_1\approx 0$

- Then $\mathbb{E}[\Delta(n_1+n_2)^2]=0$ yields

$$\mathbb{E}[n_1+n_2]=\frac{\lambda}{\epsilon}+o\left(\frac{1}{\epsilon}\right)$$

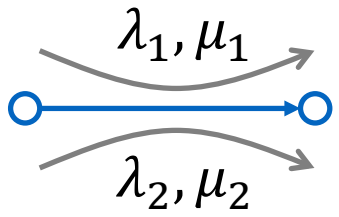
Proportionally Fair Bandwidth Sharing

One Link, Two Flow Types

- x_r: bandwidth allocated to each flow of type r
- Proportionally fair bandwidth allocation:

$$\max_{\{x_1,x_2\}} \quad n_1 \log x_1 + n_2 \log x_2$$

$$\text{subject to} \quad n_1 x_1 + n_2 x_2 \le 1$$



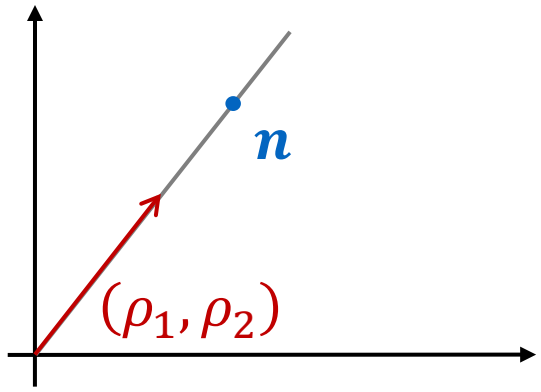
- Solution:

$$n_1 x_1 = n_1/p\,, \qquad n_2 x_2 = n_2/p\,,$$

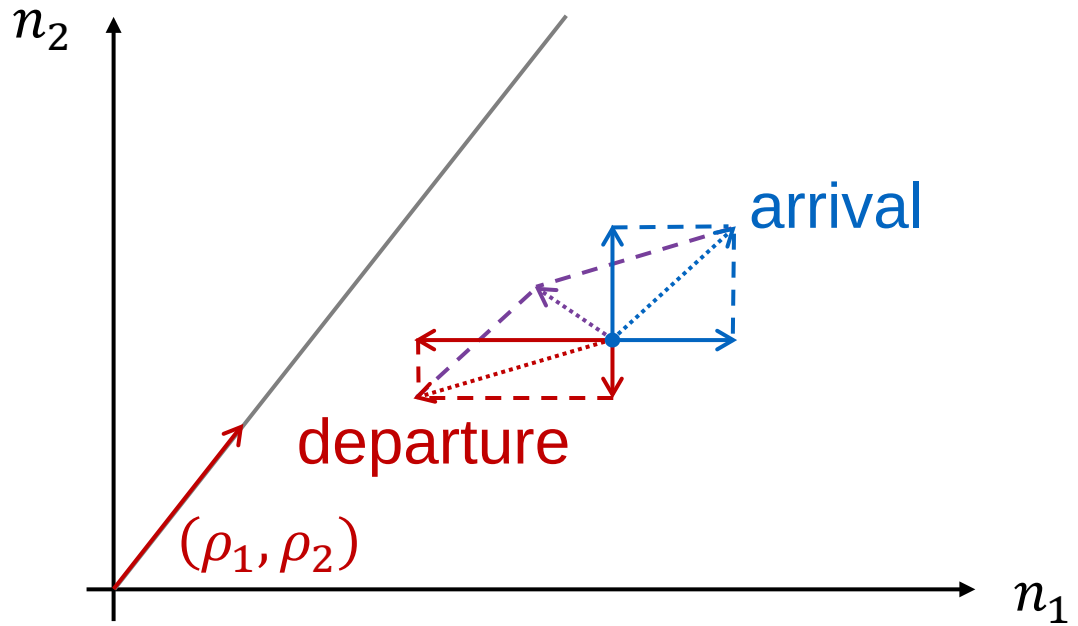
$p \ge 0$: Lagrange multiplier of the constraint

- Stability:

$$\begin{aligned} & n_1 x_1 \approx \rho_1, && n_2 x_2 \approx \rho_2 \\ \Rightarrow\ & n_1 \approx \rho_1 p, && n_2 \approx \rho_2 p \\ \Rightarrow\ & (n_1, n_2) \approx p(\rho_1, \rho_2) \end{aligned}$$



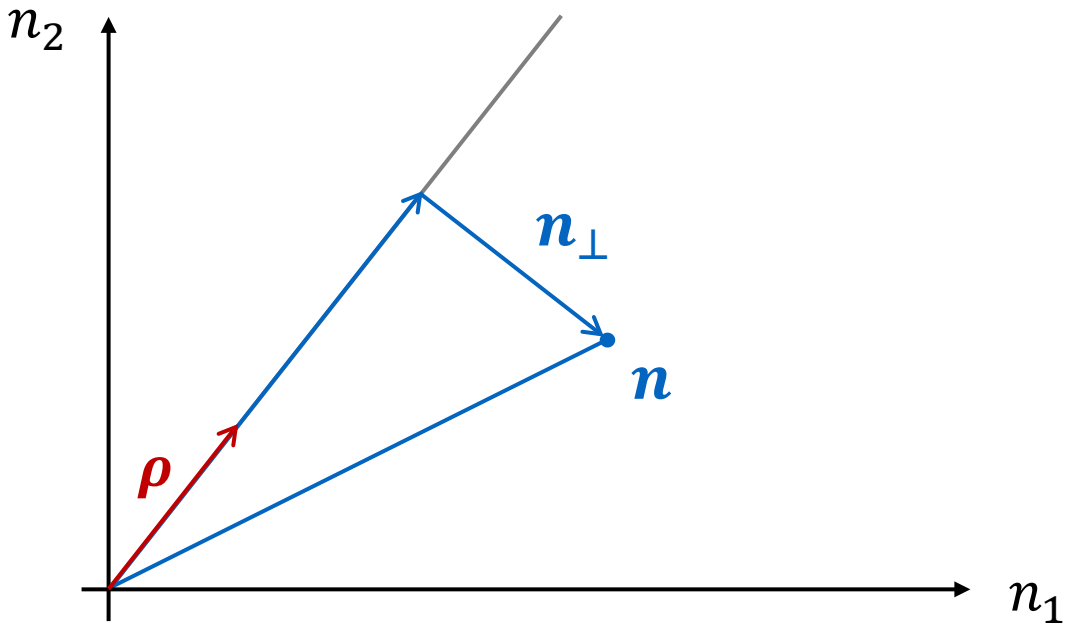
State-Space Collapse



Service rates are proportional to # of flows

When the flow count vector (n_1, n_2) is far from the direction (ρ_1, ρ_2), the departure rates pull it back

State-Space Collapse



- Heavy-traffic parameter: $\epsilon = 1 - \rho_1 - \rho_2$
- State space collapse:

$$\mathbb{E}[\|\boldsymbol{n}_\perp\|] = o\left(\frac{1}{\sqrt{\epsilon}}\right), \qquad \frac{\mathbb{E}[\|\boldsymbol{n}_\perp\|]}{\mathbb{E}[\|\boldsymbol{n}\|]} = o\left(\sqrt{\epsilon}\right)$$

Dynamics

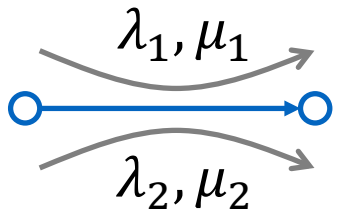
- State transition rates

  $(n_1, n_2) \rightarrow (n_1 + 1, n_2)$ with rate λ_1

  $(n_1, n_2) \rightarrow (n_1, n_2 + 1)$ with rate λ_2

  $(n_1, n_2) \rightarrow (n_1 - 1, n_2)$ with rate $n_1 x_1 \mu_1$

  $(n_1, n_2) \rightarrow (n_1, n_2 - 1)$ with rate $n_2 x_2 \mu_2$



Loads: $\rho_1 = \lambda_1 / \mu_1$

$\rho_2 = \lambda_2 / \mu_2$

- Unused bandwidth

$$U = 1 - (n_1 x_1 + n_2 x_2)$$

- Note that

$U > 0$ only when $n_1 = 0, n_2 = 0$

Bound on Backlog

- Set the drift of $V(n) = (c_1 n_1 + c_2 n_2)^2$ equal to zero for some c_1 and c_2

- Drift equation

$$\begin{aligned}\Delta\,(c_1 n_1 + c_2 n_2)^2 = {} & \lambda_1\big((c_1(n_1+1) + c_2 n_2)^2 - (c_1 n_1 + c_2 n_2)^2\big) \\ & + \lambda_2\Big(\big(c_1 n_1 + c_2(n_2+1)\big)^2 - (c_1 n_1 + c_2 n_2)^2\Big) \\ & + n_1 x_1 \mu_1\big((c_1(n_1-1) + c_2 n_2)^2 - (c_1 n_1 + c_2 n_2)^2\big) \\ & + n_2 x_2 \mu_2\Big(\big(c_1 n_1 + c_2(n_2-1)\big)^2 - (c_1 n_1 + c_2 n_2)^2\Big) \\ = {} & 2(c_1 n_1 + c_2 n_2)(c_1\lambda_1 + c_2\lambda_2 - c_1 n_1 x_1 \mu_1 - c_2 n_2 x_2 \mu_2) \\ & + c_1^2\lambda_1 + c_2^2\lambda_2 + c_1^2 n_1 x_1 \mu_1 + c_2^2 n_2 x_2 \mu_2\end{aligned}$$

Drift Analysis

$$\Delta\,(c_1 n_1 + c_2 n_2)^2 = 2(c_1 n_1 + c_2 n_2)(c_1\lambda_1 + c_2\lambda_2 - c_1 n_1 x_1 \mu_1 - c_2 n_2 x_2 \mu_2) + c_1^2\lambda_1 + c_2^2\lambda_2 + c_1^2 n_1 x_1 \mu_1 + c_2^2 n_2 x_2 \mu_2$$

- We know that $\epsilon = 1 - \rho_1 - \rho_2$ and $U = 1 - n_1 x_1 - n_2 x_2$
- We should choose $c_1 = \frac{1}{\mu_1}, c_2 = \frac{1}{\mu_2}$ to get

$$c_1\lambda_1 + c_2\lambda_2 - c_1 n_1 x_1 \mu_1 - c_2 n_2 x_2 \mu_2 = \rho_1 + \rho_2 - n_1 x_1 - n_2 x_2 = -\epsilon + U$$

- Note that $(c_1 n_1 + c_2 n_2)U = 0$

Drift Analysis

$$\Delta\,(c_1 n_1 + c_2 n_2)^2 = 2(c_1 n_1 + c_2 n_2)(c_1\lambda_1 + c_2\lambda_2 - c_1 n_1 x_1 \mu_1 - c_2 n_2 x_2 \mu_2) + c_1^2\lambda_1 + c_2^2\lambda_2 + c_1^2 n_1 x_1 \mu_1 + c_2^2 n_2 x_2 \mu_2$$

- We have chosen $c_1 = \frac{1}{\mu_1}, c_2 = \frac{1}{\mu_2}$

- State space collapse: $\frac{n_1}{n_2} \approx \frac{\rho_1}{\rho_2}$
- Then

$$c_1 n_1 + c_2 n_2 \approx \left(\frac{\rho_1}{\rho_2 \mu_1} + \frac{1}{\mu_2}\right) n_2 \approx \frac{\lambda_1/\mu_1^2 + \lambda_2/\mu_2^2}{\rho_1 + \rho_2}(n_1 + n_2)$$

Drift Analysis

$$\Delta\,(c_1 n_1 + c_2 n_2)^2 = 2(c_1 n_1 + c_2 n_2)(c_1\lambda_1 + c_2\lambda_2 - c_1 n_1 x_1 \mu_1 - c_2 n_2 x_2 \mu_2)$$
$$+c_1^2\lambda_1 + c_2^2\lambda_2 + c_1^2 n_1 x_1 \mu_1 + c_2^2 n_2 x_2 \mu_2$$

- $\mathbb{E}[n_r x_r \mu_r] = \lambda_r$, $r = 1,2$: departure rate = arrival rate

- Then

$$\mathbb{E}\left[c_1^2\lambda_1 + c_2^2\lambda_2 + c_1^2 n_1 x_1 \mu_1 + c_2^2 n_2 x_2 \mu_2\right] = 2\left(\lambda_1/\mu_1^2 + \lambda_2/\mu_2^2\right)$$

Bound on Backlog

$$\Delta\,(c_1 n_1 + c_2 n_2)^2 = 2(c_1 n_1 + c_2 n_2)(c_1\lambda_1 + c_2\lambda_2 - c_1 n_1 x_1 \mu_1 - c_2 n_2 x_2 \mu_2) + c_1^2\lambda_1 + c_2^2\lambda_2 + c_1^2 n_1 x_1 \mu_1 + c_2^2 n_2 x_2 \mu_2$$

- Setting $\mathbb{E}[\Delta(c_1 n_1 + c_2 n_2)^2] = 0$ yields

$$\mathbb{E}[n_1 + n_2] = \frac{1}{\epsilon} + o\left(\frac{1}{\epsilon}\right)$$

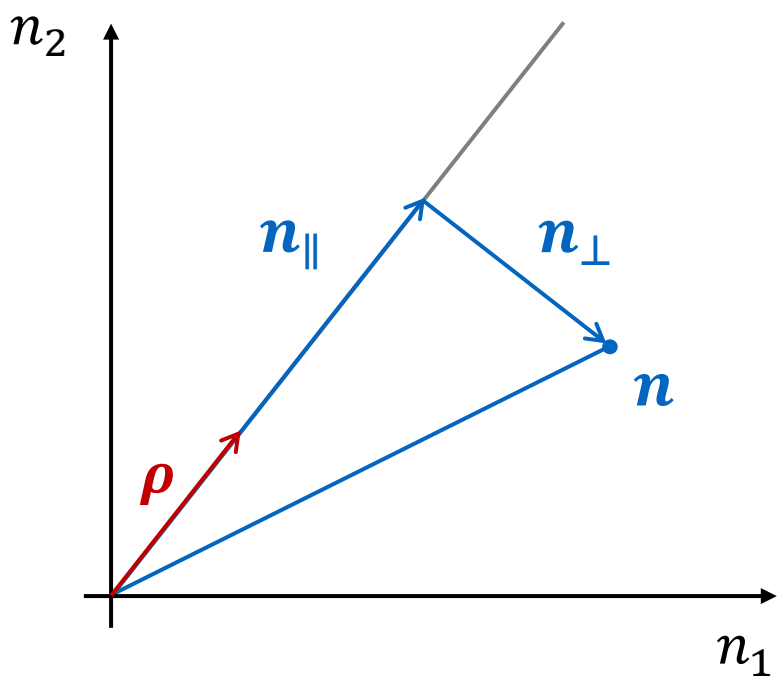
Weighted Inner Product

- $\boldsymbol{n}_{\parallel}$: projection of $\boldsymbol{n}$ onto the (half) subspace spanned by $\boldsymbol{\rho}$

- The Lyapunov function $V(n) = (n_1/\mu_1 + n_2/\mu_2)^2$ is the same as $V(n) = \left\|\boldsymbol{n}_{\parallel}\right\|^2 = \langle \boldsymbol{\rho}, \boldsymbol{n} \rangle^2$ when we use the following weighted inner product

$$\langle \boldsymbol{\rho}, \boldsymbol{n} \rangle = \begin{pmatrix} \rho_1 & \rho_2 \end{pmatrix} \begin{pmatrix} \frac{1}{\lambda_1} & 0 \\ 0 & \frac{1}{\lambda_2} \end{pmatrix} \begin{pmatrix} n_1 \\ n_2 \end{pmatrix}$$

$$= n_1/\mu_1 + n_2/\mu_2$$



Extending to General Networks

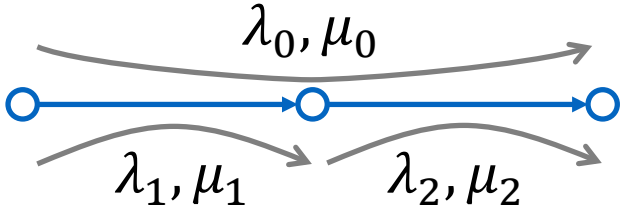
Two Links, Three Flow Types

- Proportionally fair bandwidth allocation:

$$\max_{\{x_0,x_1,x_2\}} \quad n_0 \log x_0 + n_1 \log x_1 + n_2 \log x_2$$

$$\text{subject to} \quad n_0 x_0 + n_1 x_1 \le C_1$$

$$n_0 x_0 + n_2 x_2 \le C_2$$



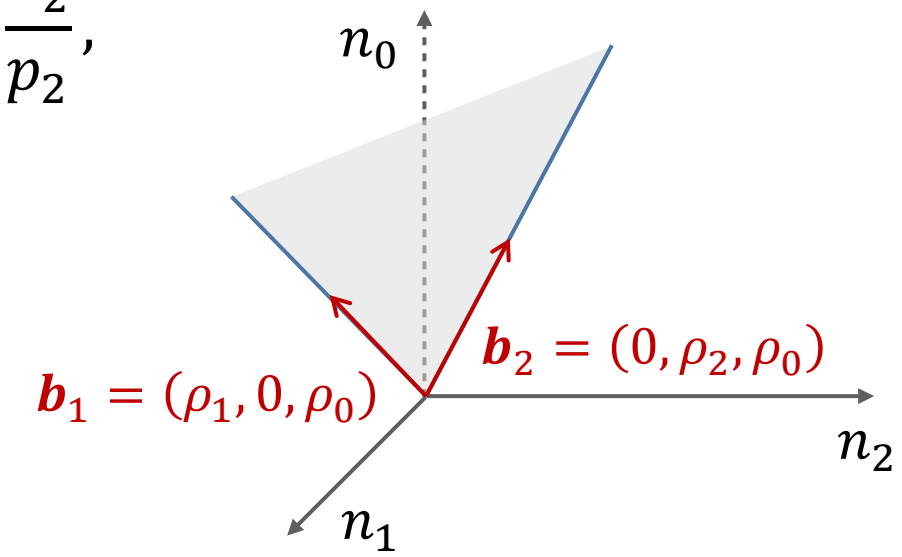
- Solution:

$$n_0 x_0 = \frac{n_0}{p_1 + p_2}, \qquad n_1 x_1 = \frac{n_1}{p_1}, \qquad n_2 x_2 = \frac{n_2}{p_2},$$

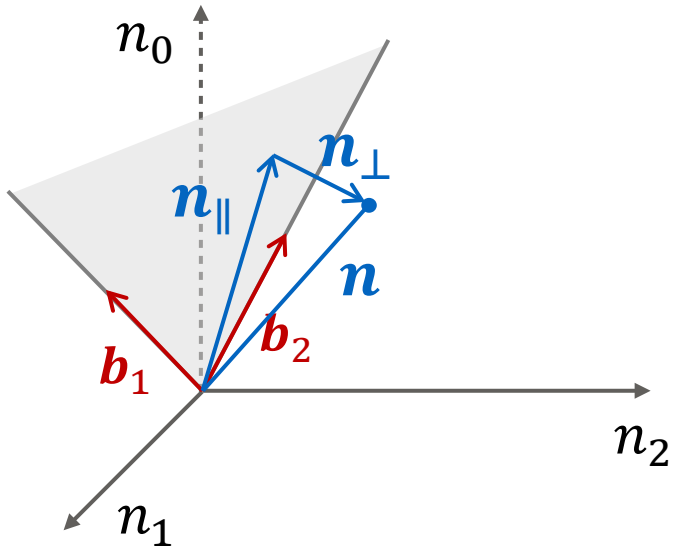
p_1, p_2: Lagrange multipliers of the constraints

- Stable: $n_0 x_0 \approx \rho_0,\ n_1 x_1 \approx \rho_1,\ n_2 x_2 \approx \rho_2$

$$\Rightarrow \quad (n_1, n_2, n_0) \approx p_1(\rho_1, 0, \rho_0) + p_2(0, \rho_2, \rho_0)$$



State-Space Collapse



- $\boldsymbol{n}_\parallel$: projection onto the cone spanned by $\{\boldsymbol{b}_1, \boldsymbol{b}_2\}$

- Heavy-traffic regime: $\rho_0 + \rho_1 = (1-\epsilon)C_1,\ \rho_0 + \rho_2 = (1-\epsilon)C_2$
- State space collapse:

$$\mathbb{E}[\|\boldsymbol{n}_\perp\|] = o\left(\frac{1}{\sqrt{\epsilon}}\right), \qquad \frac{\mathbb{E}[\|\boldsymbol{n}_\perp\|]}{\mathbb{E}[\|\boldsymbol{n}\|]} = o\left(\sqrt{\epsilon}\right)$$

Bound on Backlog

- Still consider the Lyapunov function $V(n) = \left\|\boldsymbol{n}_{\|}\right\|^2$, with the weighted inner product

$$\langle \boldsymbol{a}, \boldsymbol{n} \rangle = \begin{pmatrix} a_1 & a_2 & a_0 \end{pmatrix} \begin{pmatrix} \frac{1}{\lambda_1} & 0 & 0 \\ 0 & \frac{1}{\lambda_2} & 0 \\ 0 & 0 & \frac{1}{\lambda_0} \end{pmatrix} \begin{pmatrix} n_1 \\ n_2 \\ n_0 \end{pmatrix}$$

- Setting $\mathbb{E}[\Delta V(n)] = 0$ yields

$$\mathbb{E}[n_0 + n_1 + n_2] = \frac{2}{\epsilon} + o\left(\frac{1}{\epsilon}\right)$$

Bound on Backlog in a General Network

- L: # links in the network
- The state space collapses to an L-dimensional cone
- Setting $\mathbb{E}\left[\Delta\left\|\boldsymbol{n}_{\|}\right\|^2\right] = 0$ yields

$$\mathbb{E}\left[\sum_r n_r\right] = \frac{L}{\epsilon} + o\left(\frac{1}{\epsilon}\right)$$

Insensitivity

- The backlog bound

$$\mathbb{E}[\text{backlog}] = \frac{L}{\epsilon} + o\left(\frac{1}{\epsilon}\right)$$

  holds for a general class of phase-type distributions for file sizes
  - This distribution class can approximate any file size distribution arbitrarily closely
  - Need a more involved weighted inner product

- The delay performance of the proportionally fair bandwidth sharing policy is insensitive to the file size distribution