Product (Re)forms
Part II
Improving Performance

Kristy Gardner
Computer Science Department
Amherst College

Rhonda Righter
IEOR Department
UC Berkeley

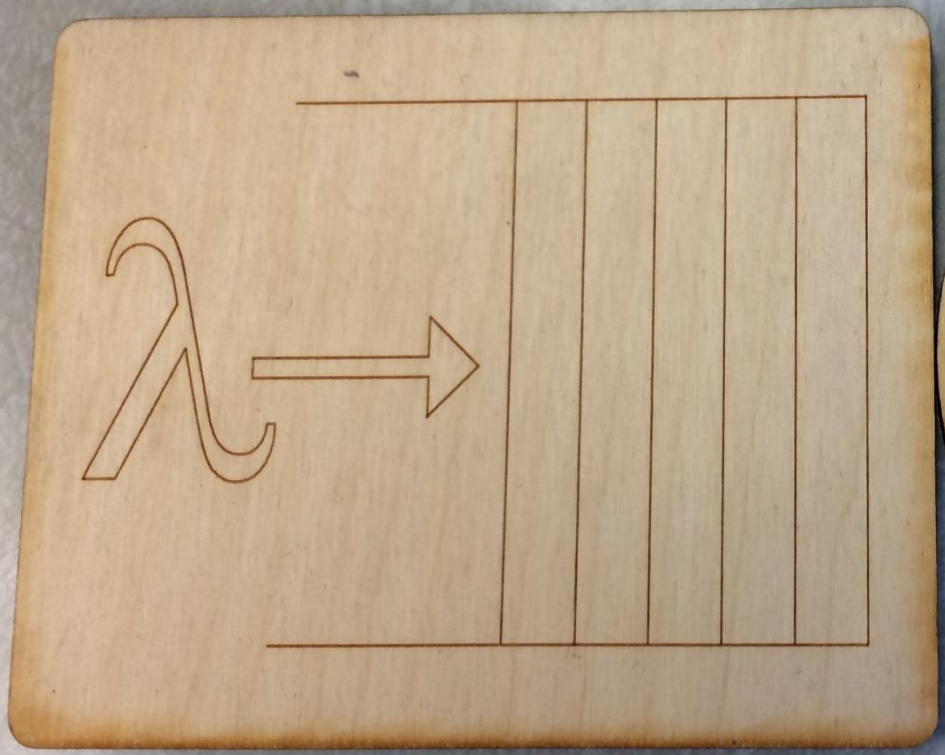
INFORMS APS, July 4, 2019


Lillian Dong


BERKELEY
IEOR
INDUSTRIAL ENGINEERING
& OPERATIONS RESEARCH


Cal

Tyler Maxey



Tyler Maxey

Outline – Improving Performance

1. Improving performance by collaborating (or not)
2. Improving performance by scheduling
3. Improving performance by increasing flexibility
4. Improving performance for whom?
5. Improving performance by how much?

Joint with Adan, Harchol-Balter, Hyytiä, Kleiner, Weiss
More details in talks by Gardner, Weiss

Improving Performance by Collaborating?

Does collaboration help?

Note: Even though we have product forms for steady-state distributions for both the collaborative and noncollaborative models, a direct comparison, even for steady-state means, is not possible except in simple cases.

We'll explore the question both for steady-state means and along sample paths.

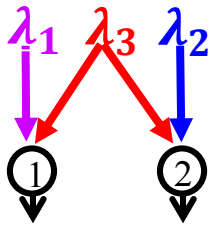
Improving Performance by Collaborating?

Does collaboration help? Sometimes!

Collaboration $\succ$ noncollaboration (C $\succ$ NC)

to minimize the number in system for:

1. **M/M/K:** One class, heterogeneous servers
   Collaboration $\Leftrightarrow$ M/M/1 with fast server
2. **Fully symmetric system**:
   Servers and job classes stochastically identical
   $\lambda_i = \lambda_j, \mu_k = \mu_l, \forall i, j, k, l$
   S_i symmetric: *d* randomly selected (power of d)
3. **Symmetric W model**: $\lambda_1 = \lambda_2 \quad \mu_1 = \mu_2$



Intuition: Collaboration keeps servers busy

Improving Performance by Collaborating?

Does collaboration help? Sometimes!

Collaboration $\succ$ noncollaboration (C $\succ$ NC)

to minimize the number in system for:

1. **M/M/K:** One class, heterogeneous servers
   Collaboration $\Leftrightarrow$ M/M/1 with fast server
2. **Fully symmetric system**:
   Servers and job classes stochastically identical
   $\lambda_i = \lambda_j, \mu_k = \mu_l, \forall i, j, k, l$
   S_i symmetric: *d* randomly selected (power of d)
3. **Symmetric W model**: $\lambda_1 = \lambda_2 \quad \mu_1 = \mu_2$

But sometimes **collaboration can be much worse…**

Asymmetric Case: N Model

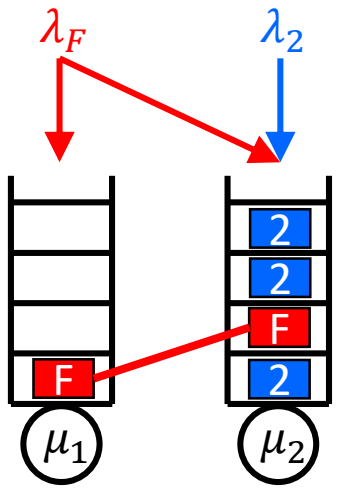
Thm: For the N model,

$N^C(t) - N^{NC}(t)$, $N_2^C(t) - N_2^{NC}(t)$, and $N_F^C(t) - N_F^{NC}(t)$

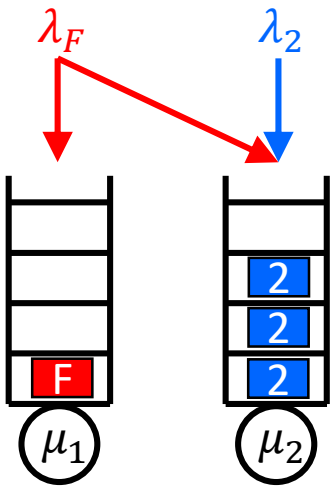
all can be arbitrarily large. (F for flexible)

Proof: By an example sample path

Collaborative (C)



Noncollaborative (NC)



State: $(2, F, 2, 2)$ $(2, F_1, 2, 2)$

Sample path:

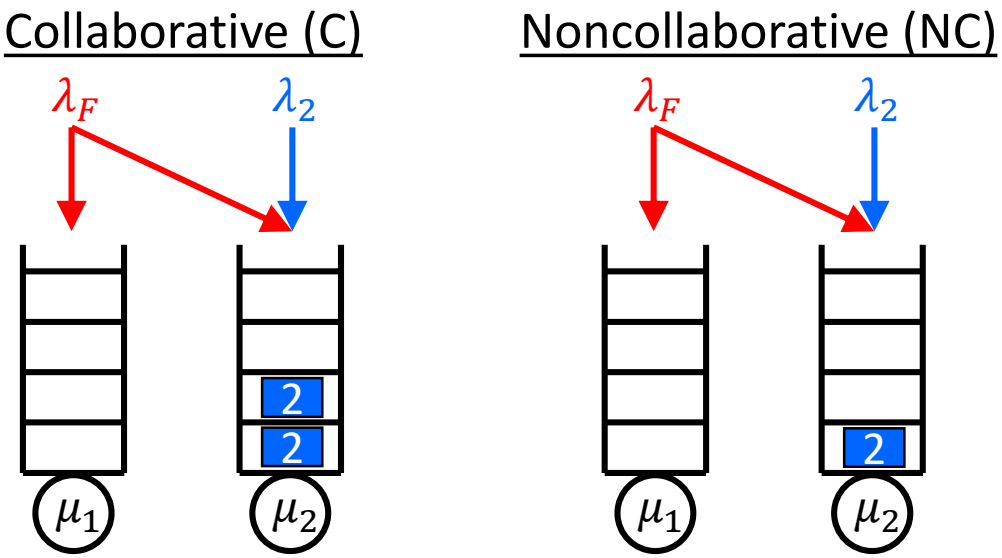
Asymmetric Case: N Model

Thm: For the N model,

$N^C(t) - N^{NC}(t)$, $N_2^C(t) - N_2^{NC}(t)$, and $N_F^C(t) - N_F^{NC}(t)$

all can be arbitrarily large.

Proof: By an example sample path



State: $(2,2)$ (2)

Sample path: Service on 2,2,1

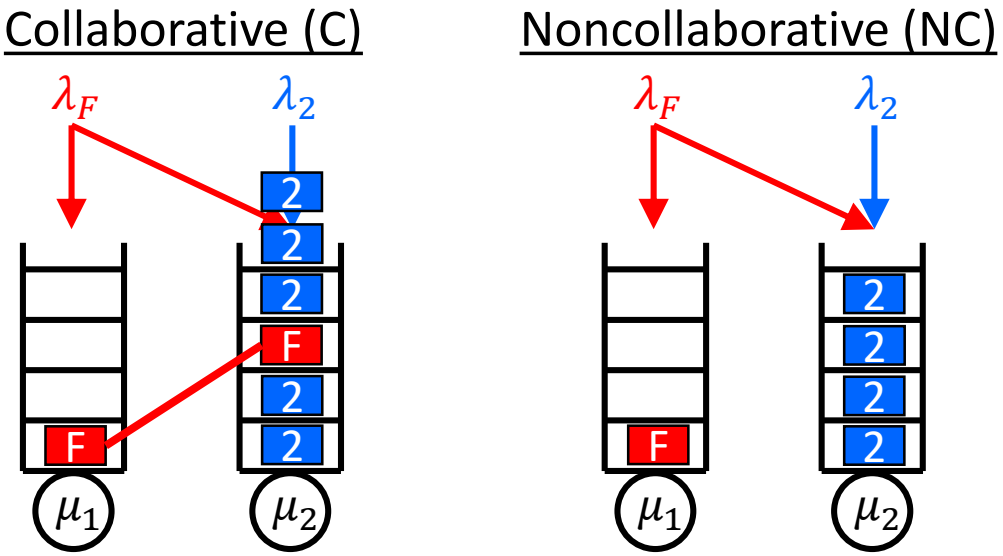
Asymmetric Case: N Model

Thm: For the N model,

$N^C(t) - N^{NC}(t)$, $N_2^C(t) - N_2^{NC}(t)$, and $N_F^C(t) - N_F^{NC}(t)$

all can be arbitrarily large.

Proof: By an example sample path



State: $(2,2,F,2,2,2)$ $(2,F_1,2,2,2)$

Sample path: Service on $2,2,1$, Arrival F, $2,2,2$

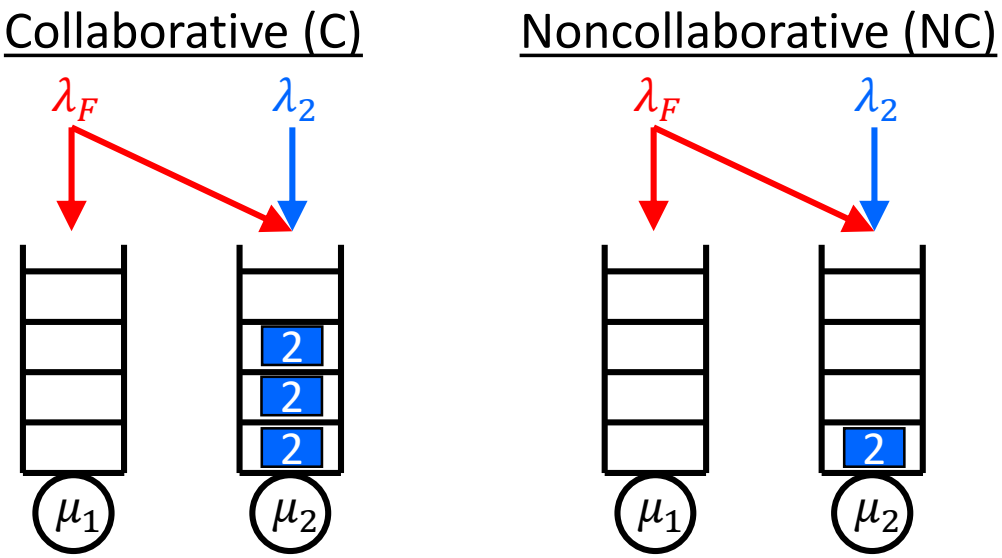
Asymmetric Case: N Model

Thm: For the N model,

$N^C(t) - N^{NC}(t)$, $N_2^C(t) - N_2^{NC}(t)$, and $N_F^C(t) - N_F^{NC}(t)$

all can be arbitrarily large.

Proof: By an example sample path



State: $(2,2,2)$ (2)

Sample path: Service on 2,2,1, Arrival F, 2,2,2, Service on 2,2,2,1

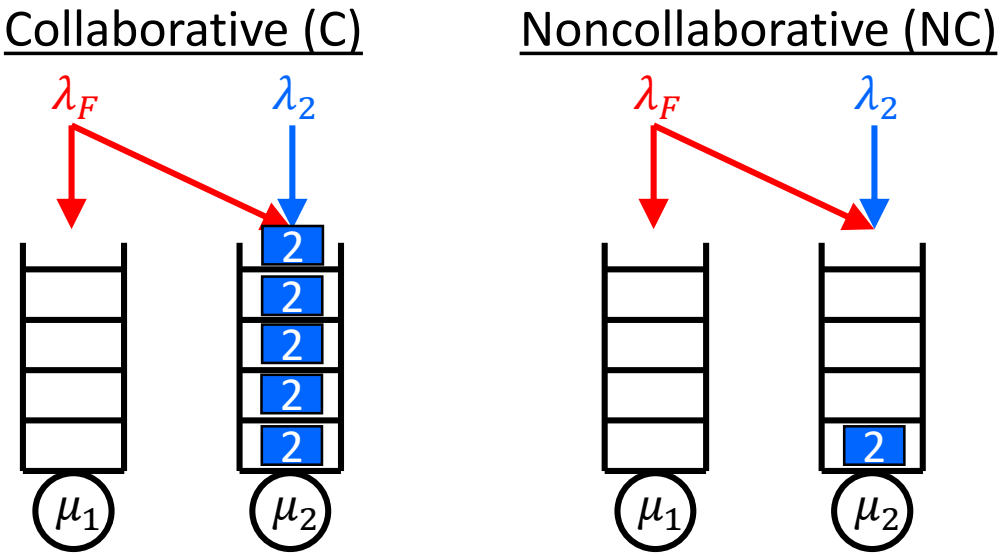
Asymmetric Case: N Model

Thm: For the N model,

$N^C(t) - N^{NC}(t)$, $N_2^C(t) - N_2^{NC}(t)$, and $N_F^C(t) - N_F^{NC}(t)$

all can be arbitrarily large.

Proof: By an example sample path



State: $(2,2,2,2,2)$ (2)

Sample path: Continuing – arbitrary build up of class 2 in the C system

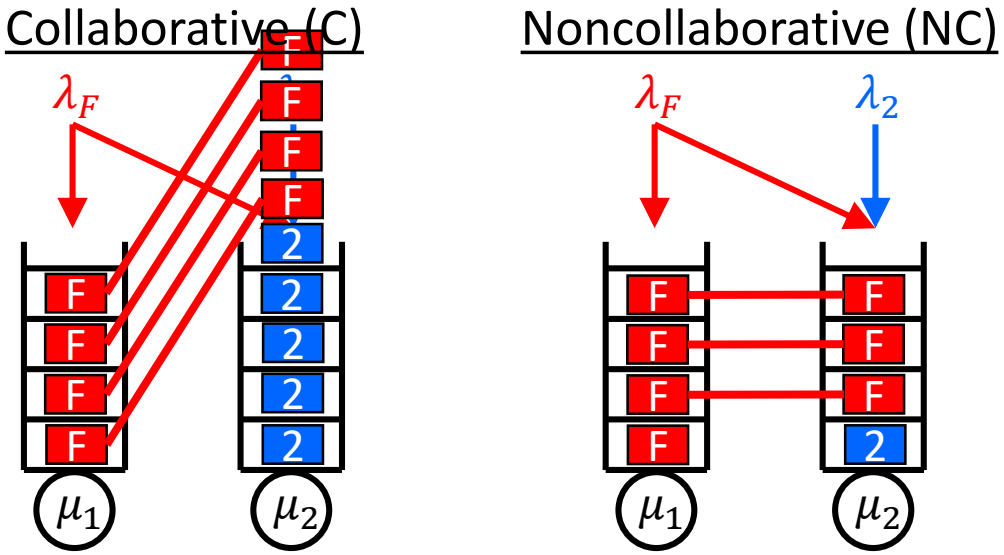
Asymmetric Case: N Model

Thm: For the N model,

$N^C(t) - N^{NC}(t)$, $N_2^C(t) - N_2^{NC}(t)$, and $N_F^C(t) - N_F^{NC}(t)$

all can be arbitrarily large.

Proof: By an example sample path



State: $(2,2,2,2,2,F,F,F,F)$ $(2,F_1,F,F,F)$

Sample path: Arrival F, F, F, F

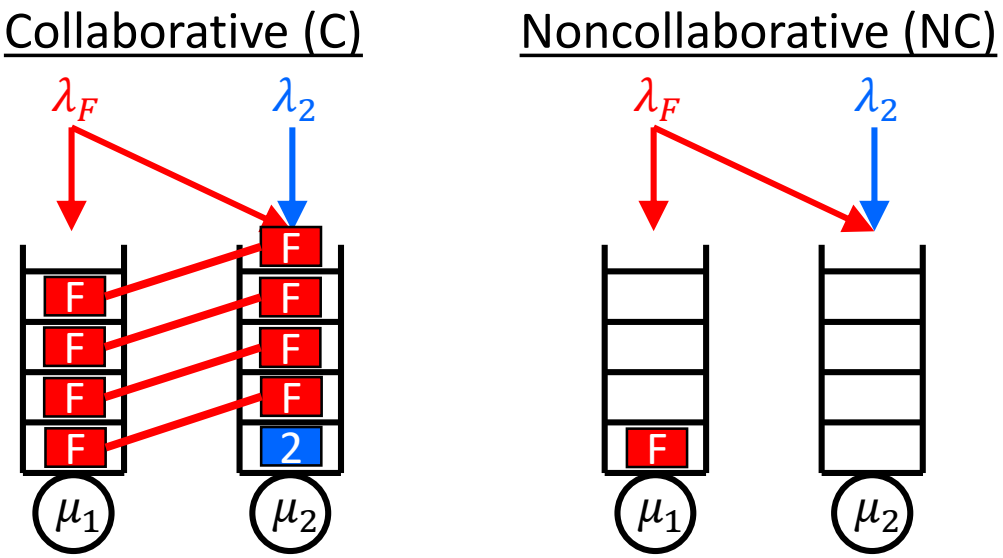
Asymmetric Case: N Model

Thm: For the N model,

$N^C(t) - N^{NC}(t)$, $N_2^C(t) - N_2^{NC}(t)$, and $N_F^C(t) - N_F^{NC}(t)$

all can be arbitrarily large.

Proof: By an example sample path



State: $(2, F, F, F, F)$ (F_1)

Sample path: Arrival F, F, F, F, Service on 2,2,2,2

Improving Performance through Collaboration?

Collaboration is better in symmetric systems

Collaboration can be worse than noncollaboration:

We've seen that

$N^C(t) - N^{NC}(t)$

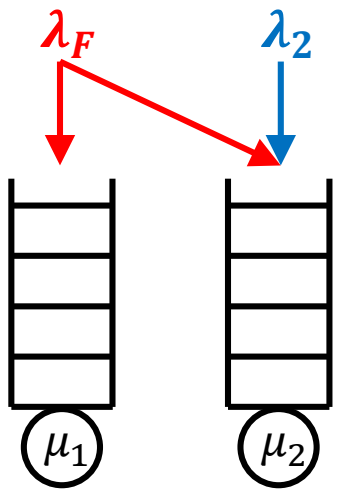
can be arbitrarily large on a sample path

More flexible jobs can block less flexible jobs in the C (collaborative) system

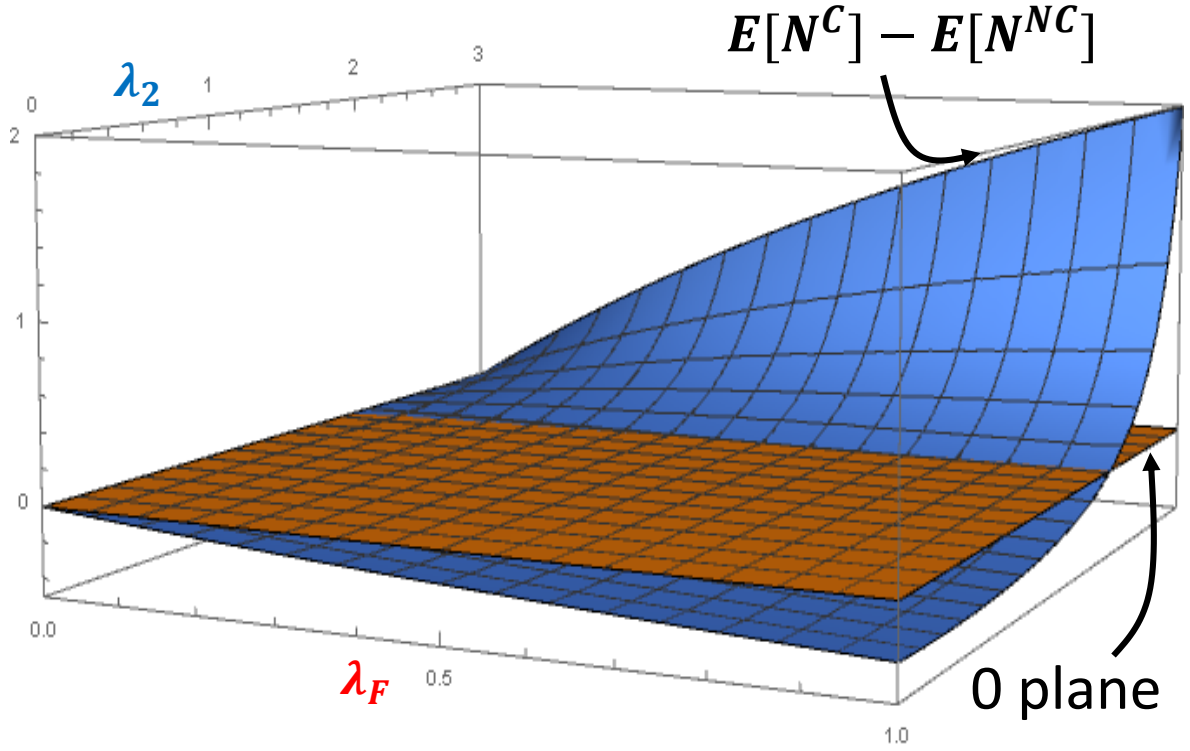
Are these sample paths sufficiently rare so that the mean is bounded?

Improving Performance by Collaborating?

No! **Collaboration can be much worse**, even on average, for asymmetric systems.



$\mu_1 = 1, \mu_2 = 3$



Improving Performance through Collaboration?

We've seen that the **cost of collaboration**, in terms of # of jobs, can be arbitrarily large:

$N^C(t) - N^{NC}(t)$ can be arbitrarily large

$E[N^C] - E[N^{NC}]$ can be arbitrarily large

Can the **benefit of collaboration** also be arbitrarily large?

No! The benefit of collaboration is bounded by M = # of servers

$N^{NC}(t) - N^C(t) \leq M$ on coupled sample paths

Collaboration Benefit Upper Bound

Thm: We can couple the C and NC systems so that

$\{N^{NC}(t)\} \leq \{N^{C}(t)\} + M$ with probability 1 and

$\{N_i^{NC}(t)\} \leq \{N_i^{C}(t)\} + |S_i|$ w.p. 1,

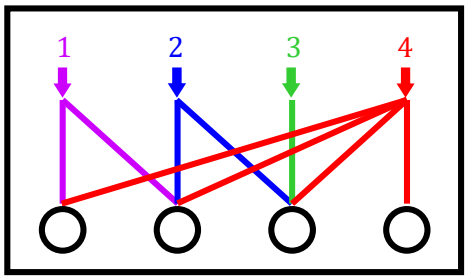
where $|S_i|$ = # servers compatible with class *i*

Proof Idea: Recall from Part I that

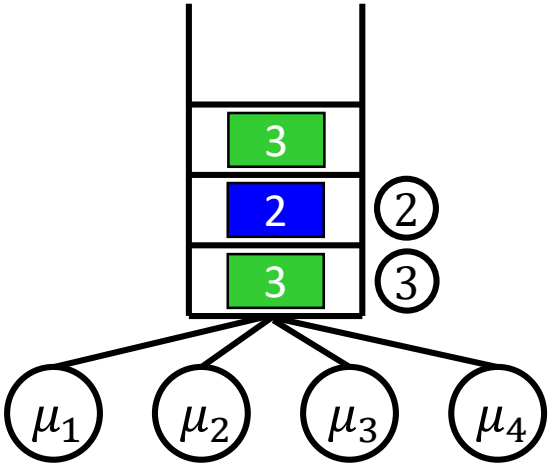
$\{N_Q^{NC}(t) \mid \text{keep all servers busy}\} =_{st} \{N^{C}(t)\}$

Collaborative vs. Noncollaborative

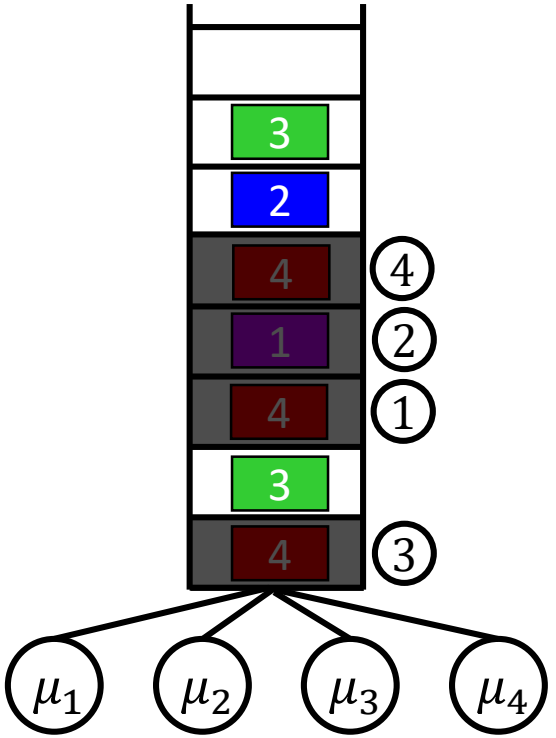
Collaborative (C)



$\vec{c}_n = (3,1,2,3)$



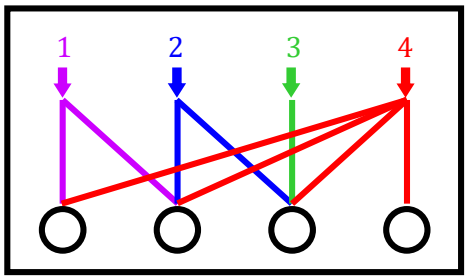
Noncollaborative (NC)



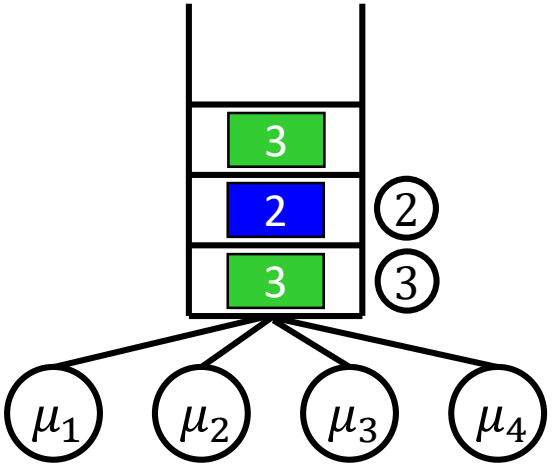
Conditioned on all servers busy in NC, the noncollaborative **queue** has the same stationary distribution as the collaborative **system**

Collaborative vs. Noncollaborative

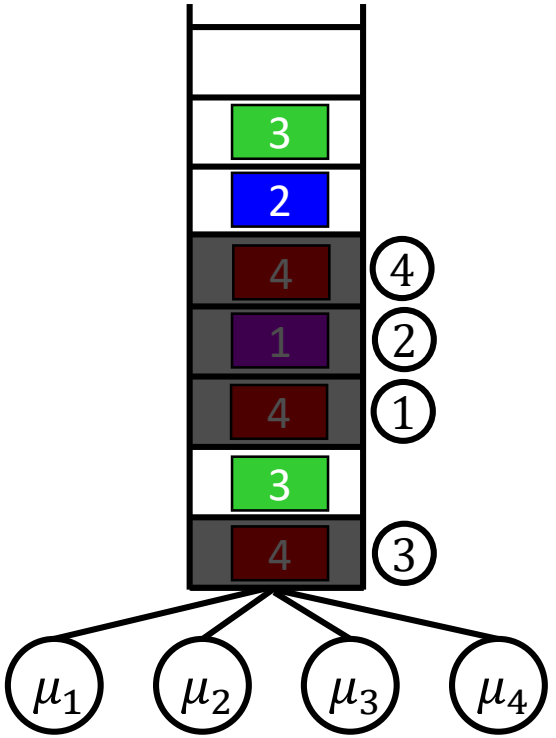
Collaborative (C)



$\vec{c}_n = (3,1,2,3)$



Noncollaborative (NC)



If we keep all servers busy in NC, the noncollaborative **queue** has the **same sample paths** w.p. 1 as the collaborative **system**

Collaboration Benefit Upper Bound

Thm: We can couple the C and NC systems so that

$\{N^{NC}(t)\} \leq \{N^{C}(t)\} + M$ with probability 1 and

$\{N_i^{NC}(t)\} \leq \{N_i^{C}(t)\} + |S_i|$ w.p. 1,

where $|S_i|$ = # servers compatible with class *i*

Proof Idea: Recall from Part I that

$$\{N_Q^{NC}(t) \mid \text{keep all servers busy}\} =_{st} \{N^{C}(t)\} \qquad *$$

$$\{N^{NC}(t)\} \leq_{st} \{N_Q^{NC}(t) \mid \text{all servers busy}\} + M \quad \text{(easy)}$$

$$=_{st} \{N^{C}(t)\} + M \quad \text{(from *)}$$

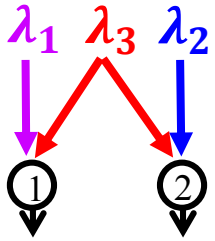
Tighter Bound for the W Model

Thm: For the W model, we can couple the NC and C systems so that, w.p.1,

$$\{N^{NC}(t)\} \leq \{N^{C}(t)\} + 1$$

$$\{N_i^{NC}(t)\} \leq \{N_i^{C}(t)\} + 1, i = 1,2$$

$$\{N_i^{NC}(t) + N_3^{NC}\} \leq \{N_i^{C}(t) + N_3^{C}\} + 1, i = 1,2$$



➔ NC has **at most** one more job than C
(call it the tagged job)

And if it does, then among all non-tagged jobs:
NC has **fewer** jobs of each class than C (nonstrictly)

The theorem also holds for W models embedded in larger nested systems.

W Model Bound – Proof Idea

- In **C and NC**: Class i jobs FIFO, $i = 1,2$ (they leave in arrival order)
- In **C** (only!): Class 3 jobs FIFO
- In **C**: Class i jobs + class 3 jobs FIFO (Class i jobs can't "pass" class 3 jobs)

➔

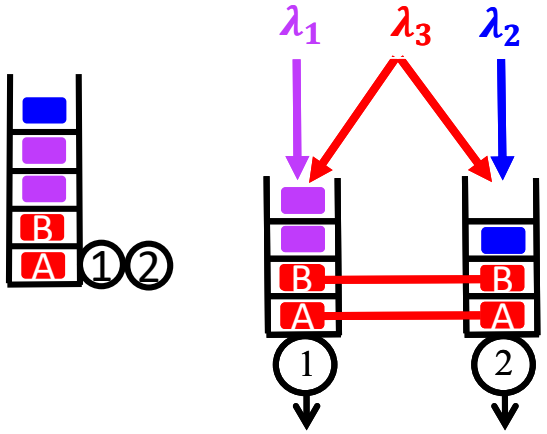
If NC has an extra tagged job T, and otherwise fewer of each class than C

- T must be in service in NC,
- The "next" job (A) on the other server must be in service in both C and NC

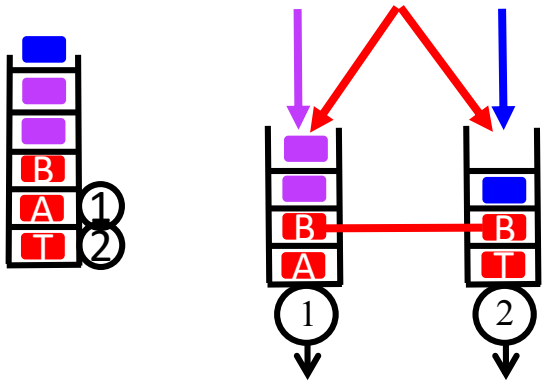
➔

A forward induction proof

Collaborative (C)



NonCollaborative (NC)



Outline – Improving Performance

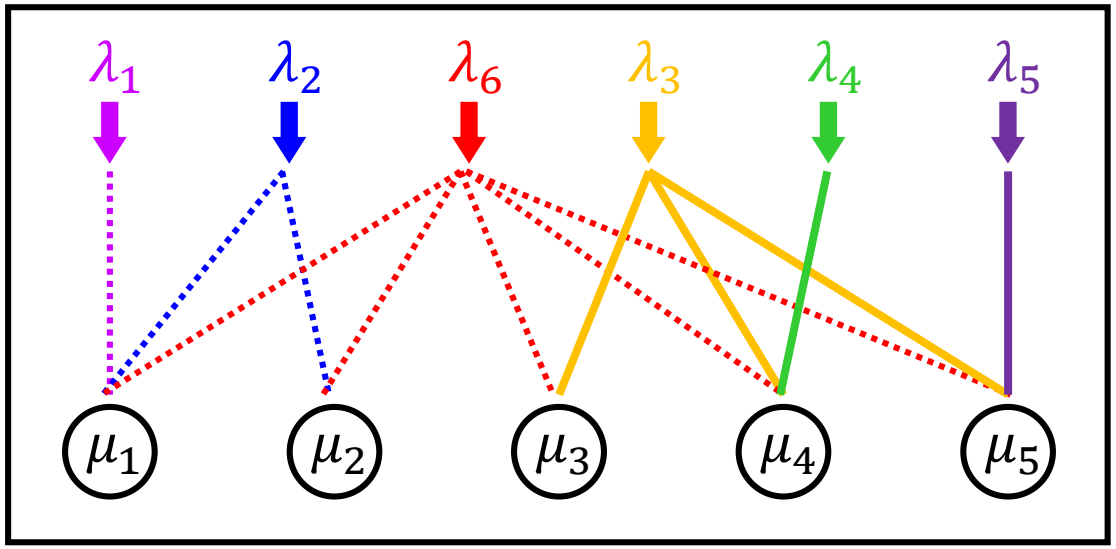
1. Improving performance by collaborating (or not)
2. **Improving performance by scheduling**
3. Improving performance by increasing flexibility
4. Improving performance for whom?
5. Improving performance by how much?

Scheduling in Nested Systems

Recall for nested systems:

For every pair of job classes i and j, either:

- $S_i \subset S_j$
- $S_j \subset S_i$
- $S_i \cap S_j = \emptyset$



$R(i)$ $= \{j : S_j \subseteq S_j\}$ e.g., $R(\mathbf{3}) = \{\mathbf{3}, \mathbf{4}, \mathbf{5}\}$

= the set of classes that **R**equire at least one of the servers in S_i

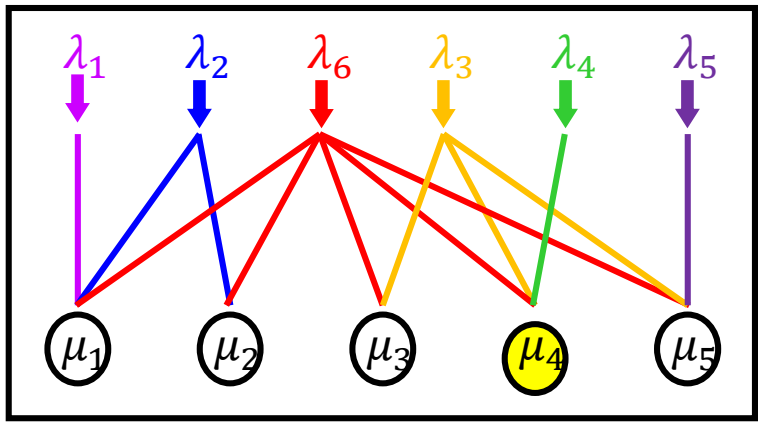
= class i plus all of the classes within its subsystem

$K_i(t) = \sum_{j \in R(i)} N_j(t)$ = the total number of jobs in class i's subsystem

E.g., $K_3(t) = \boldsymbol{N_3} + \boldsymbol{N_4} + \boldsymbol{N_5}$

Scheduling in Nested Systems

Thm: For nested (collaborative or noncollaborative) systems, preemptive non-idling **LFF (least-flexible-first)** minimizes $\{\vec{K}(t)\}$ w.p. 1 for coupled sample paths, where $\vec{K}(t) = (K_1(t), K_2(t), \ldots, K_J(t))$ and $K_i(t) =$ the total number of jobs in class i's subsystem



LFF: $4 \succ 3 \succ 6$ on server 4, etc.

Intuition: Save more flexible jobs to run when servers might otherwise be idle

Product form no longer holds, though least flexible jobs experience an M/M/1 queue

LFF Optimality Proof for the W model

Thm: For the W model, LFF minimizes, w.p. 1,

$\{(N_1(t), N_2(t), N_1(t) + N_2(t) + N_3(t))\}$

Proof:

Don't idle – simple argument ✔

For policy $\pi \neq$ LFF,

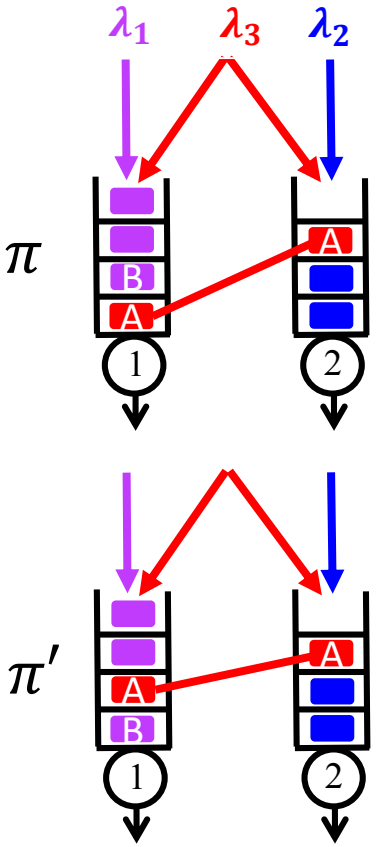
we will construct construct a better policy, π'

At time 0, π serves a class 3 job (**A**) on server 1

when there is a class 1 job (**B**) available

Let π' serve the class 1 job (**B**) on server 1

and agree with π on server 2 at time 0



LFF Optimality Proof for the W model

Thm: For the W model, LFF minimizes, w.p. 1,

$\{(N_1(t), N_2(t), N_1(t) + N_2(t) + N_3(t))\}$

Proof:

At time 0, π serves a class 3 job (**A**) on server 1

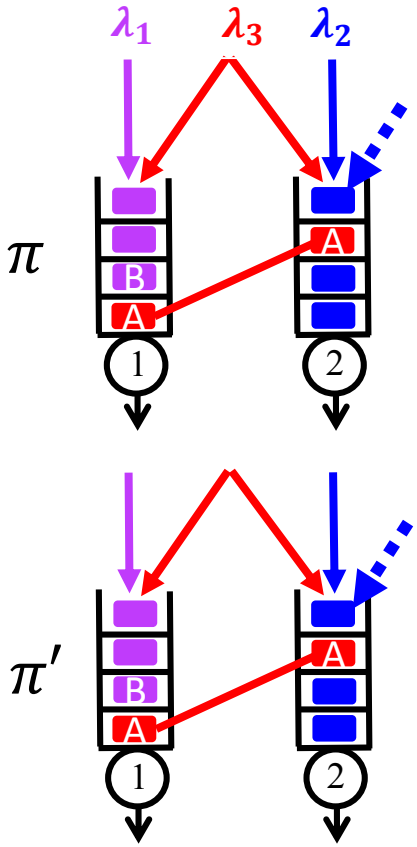
Let π' serve the class 1 job (**B**) on server 1

Case 1. The next event is

an arrival or server 2 completion:

Let $\pi' = \pi$ thereafter, so

$N'_i(t) = N_i(t)$ for all i,t



LFF Optimality Proof for the W model

Thm: For the W model, LFF minimizes, w.p. 1,

$\{(N_1(t), N_2(t), N_1(t) + N_2(t) + N_3(t))\}$

Proof:

Case 2. Next event a server 1 completion:

Let π' serve **A** whenever π serves **B**;

otherwise let $\pi' = \pi$

(OK because of **nested structure**, may **idle**)

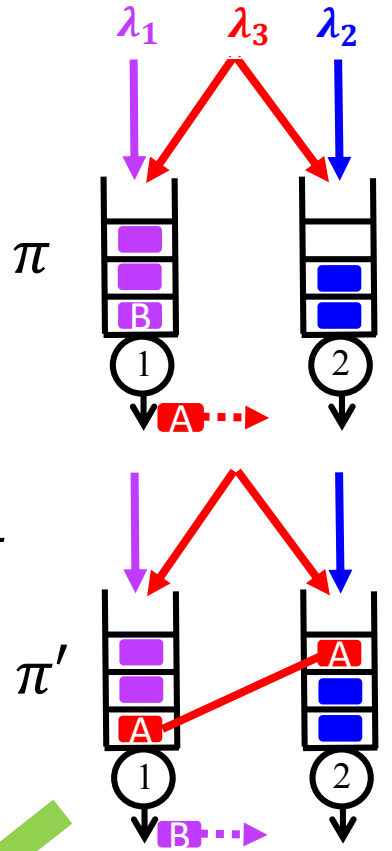
All job completions at same times,

but **A**, **B** interchanged: **A** leaves earlier under π

and **B** leaves earlier under π'

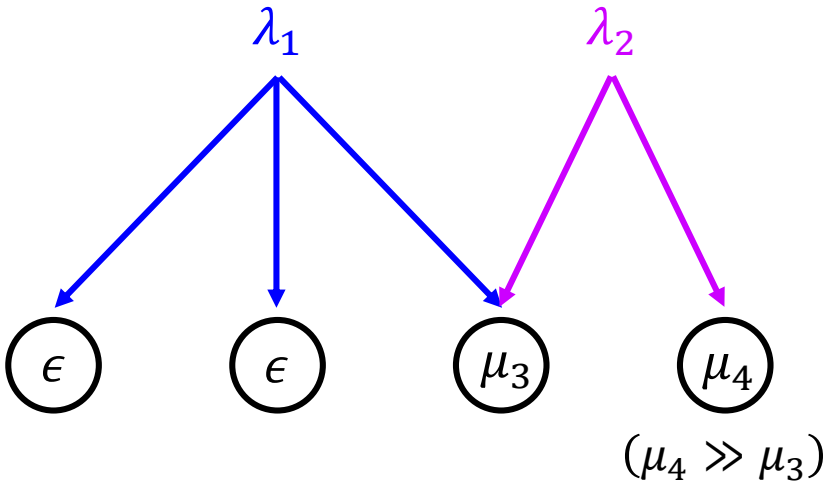
$N_1'(t) \le N_1(t), N_2'(t) \le N_2(t),$

$N_1'(t) + N_2'(t) + N_3'(t) \le N_1(t) + N_2'(t) + N_3(t)$



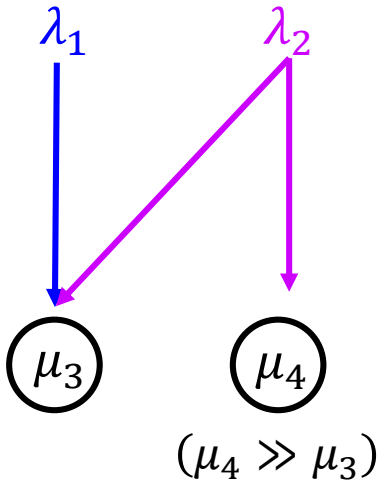
Non-nested Systems: LFF is not optimal

Example Non-nested System



$\varepsilon \to 0$

An N (nested) system



LFF prioritizes class 2 on server 3

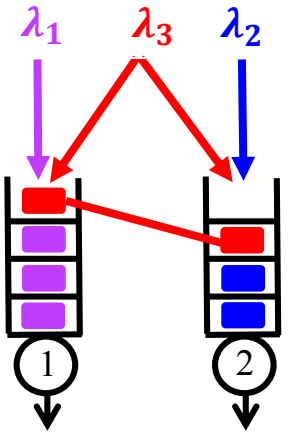
LFF = optimal and LFF prioritizes class 1 on server 3

Is Collaboration a Good Thing?

For FCFS – Sometimes!

For LFF in nested systems – Always!

For example, in the W model,
class 3 jobs are only served on server *i*
when no class *i* jobs are present, *i* = 1,2
– when using both servers can only help.



Outline – Improving Performance

1. Improving performance by collaborating (or not)
2. Improving performance by scheduling
3. **Improving performance by increasing flexibility**
4. Improving performance for whom?
5. Improving performance by how much?

Improving Performance with Flexibility?

Under **FCFS**, more flexibility **helps in symmetric systems**, when flexibility added symmetrically

1. **Fully symmetric system**

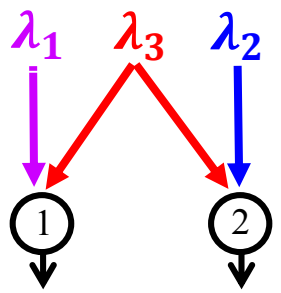
   Servers and job classes stochastically identical

   $\lambda_i = \lambda_j, \mu_k = \mu_l, \forall i, j, k, l$

   S_i symmetric: *d* randomly selected (power of d)

   Response time decreasing in *d*

2. **Symmetric W model**



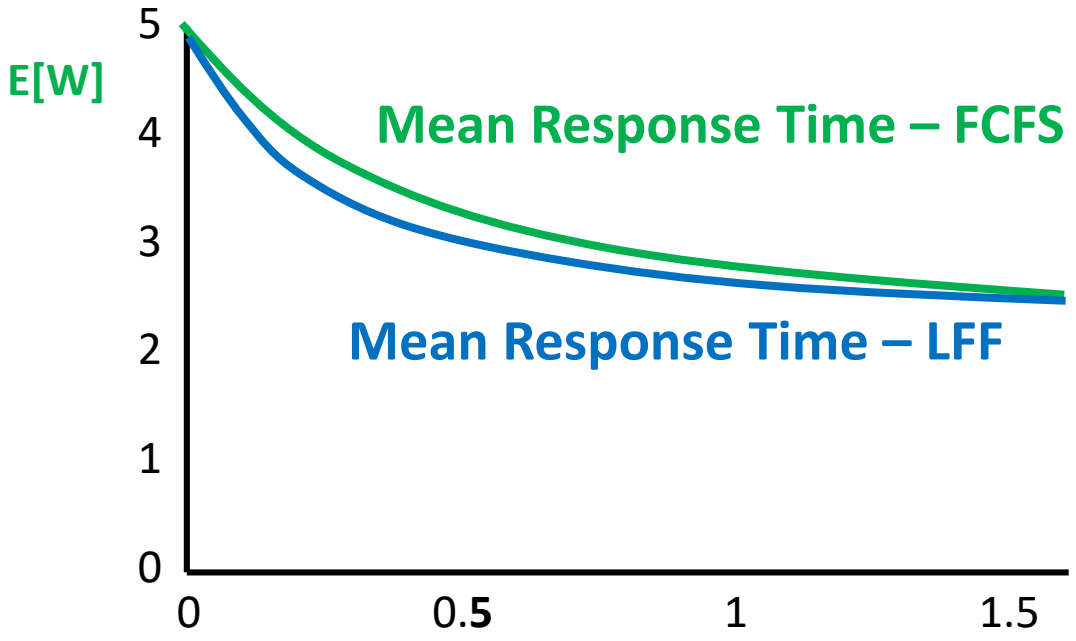
$\mu_1 = \mu_2, \lambda_3 = \Delta$

$\lambda_1 = \lambda_2 = \lambda - \frac{\Delta}{2}$

Response time decreasing in Δ

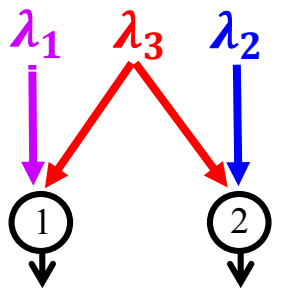
W Model: Adding Flexibility Symmetrically

Under **FCFS** (and LFF) more flexibility **helps in symmetric systems**



Δ = shift from classes 1 and 2 to class 3

Collaborative W



$$\mu_1 = \mu_2 = 1$$

$$\lambda_1 = \frac{1.6-\Delta}{2}$$

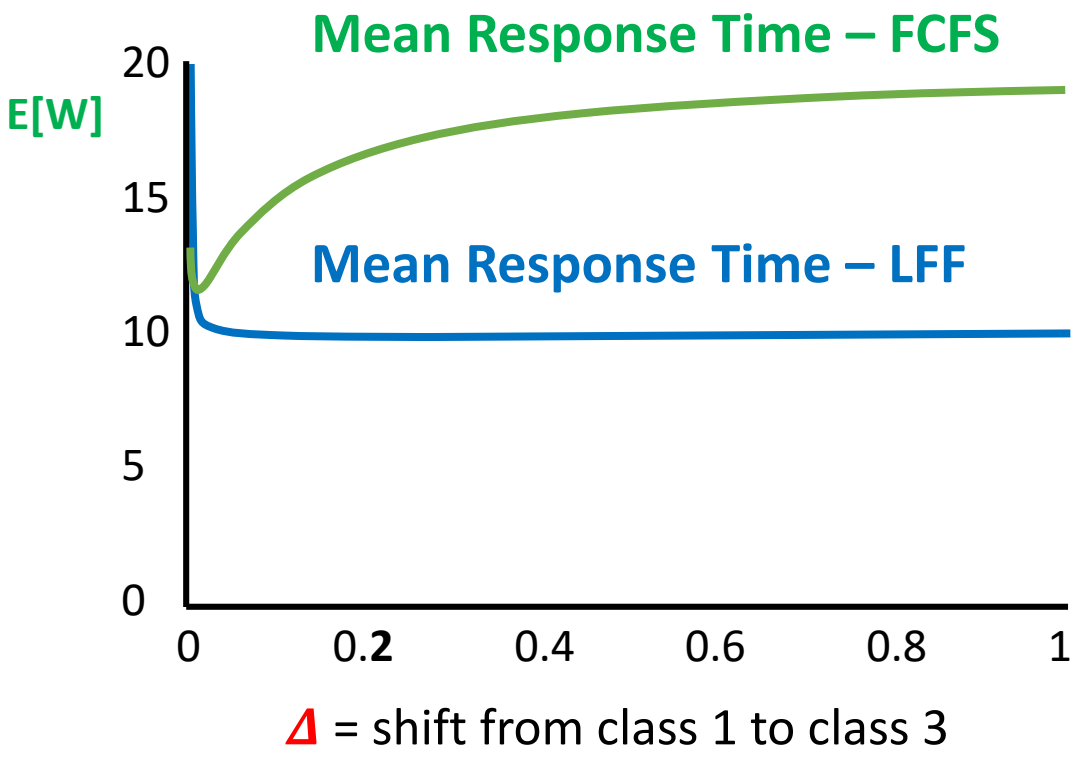
$$\lambda_2 = \frac{1.6-\Delta}{2}$$

$$\lambda_3 = \Delta$$

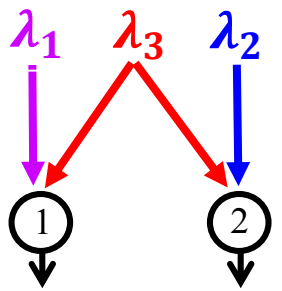
Improving Performance with Flexibility?

Under **FCFS**, more flexibility **can hurt** (asymmetric)

Under **LFF** and nested, more flexibility **always helps**



Collaborative W



$\mu_1 = 1$

$\mu_2 = 100$

$\lambda_1 = 1 - \Delta$

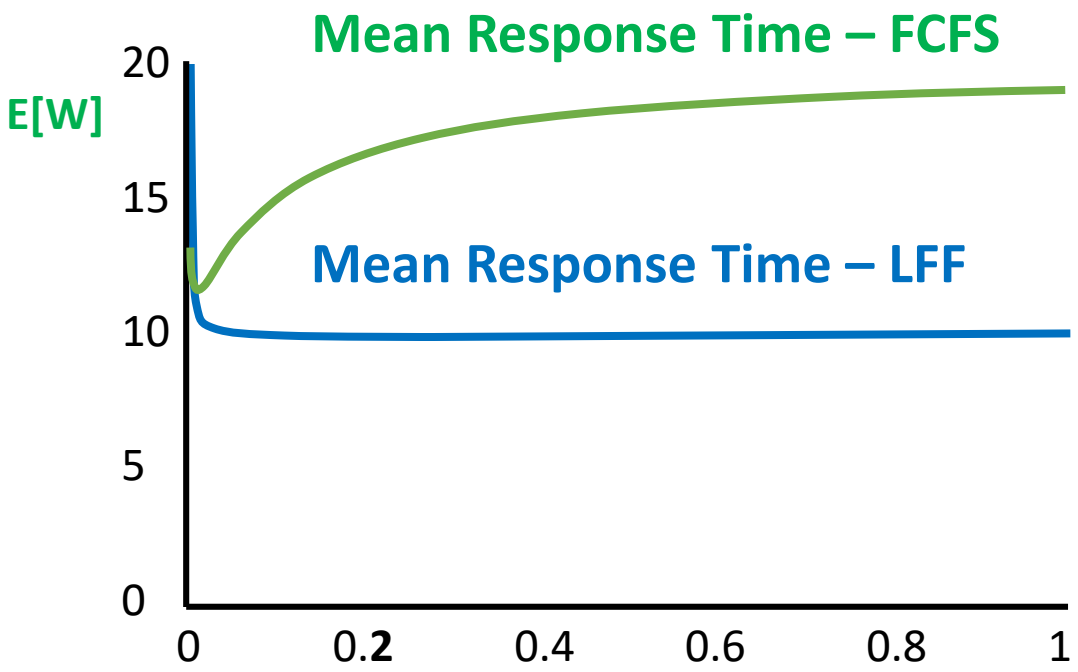
$\lambda_2 = 99.9$

$\lambda_3 = \Delta$

Improving Performance with Flexibility?

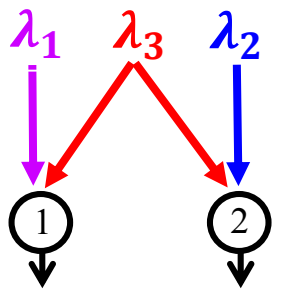
LFF: A little bit (of flexibility) goes a long way

FCFS: A little bit more goes the wrong way!



Δ = shift from class 1 to class 3

Collaborative W



$\mu_1 = 1$

$\mu_2 = 100$

$\lambda_1 = 1 - \Delta$

$\lambda_2 = 99.9$

$\lambda_3 = \Delta$

Outline – Improving Performance

1. Improving performance by collaborating (or not)
2. Improving performance by scheduling
3. Improving performance by increasing flexibility
4. **Improving performance for whom?**
5. Improving performance by how much?

Improving Performance for Whom?

Collaboration, scheduling, flexibility are all means for (possibly) improving **overall** performance

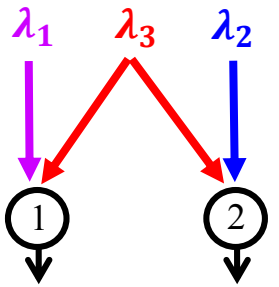
But even when overall performance improves, some classes of jobs may be hurt

1. FCFS ➔ LFF hurts most flexible jobs
   FCFS ➔ LFF helps least flexible jobs

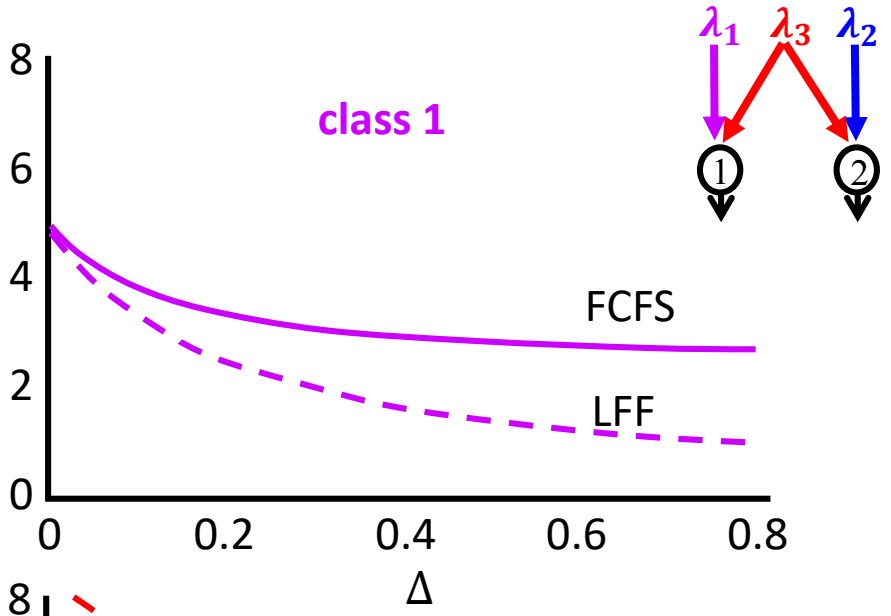
W model (collaborative) example:

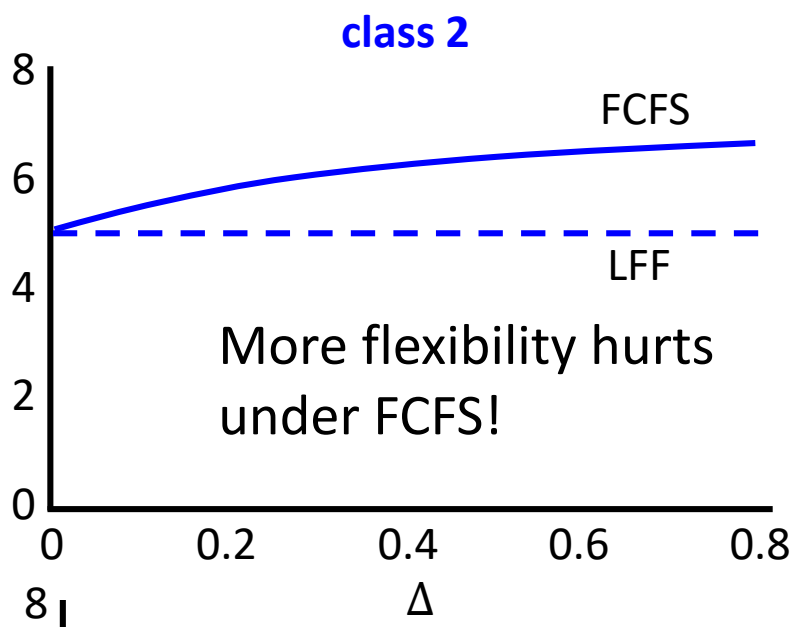
$$\mu_1 = \mu_2 = 1, \lambda = 1.6, \lambda_2 = 0.8$$
$$\lambda_1 = 0.8 - \Delta, \lambda_3 = \Delta$$

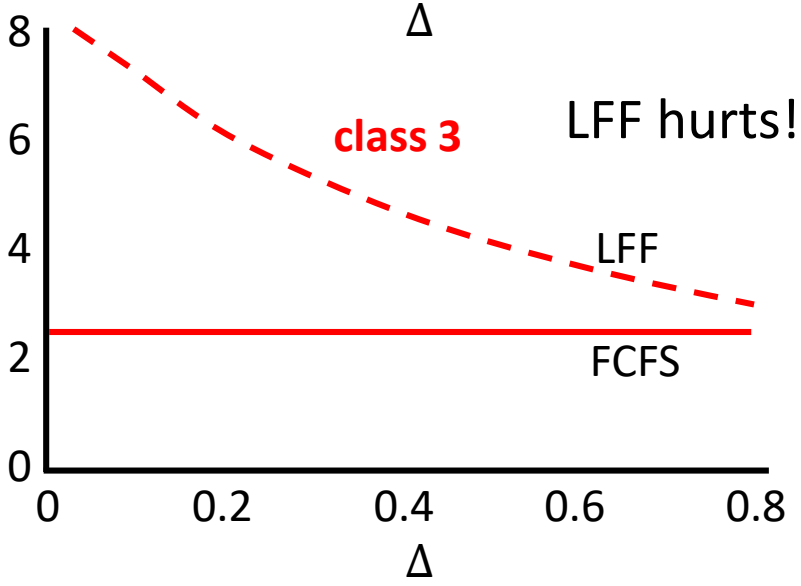
Δ = shift from class 1 to class 3

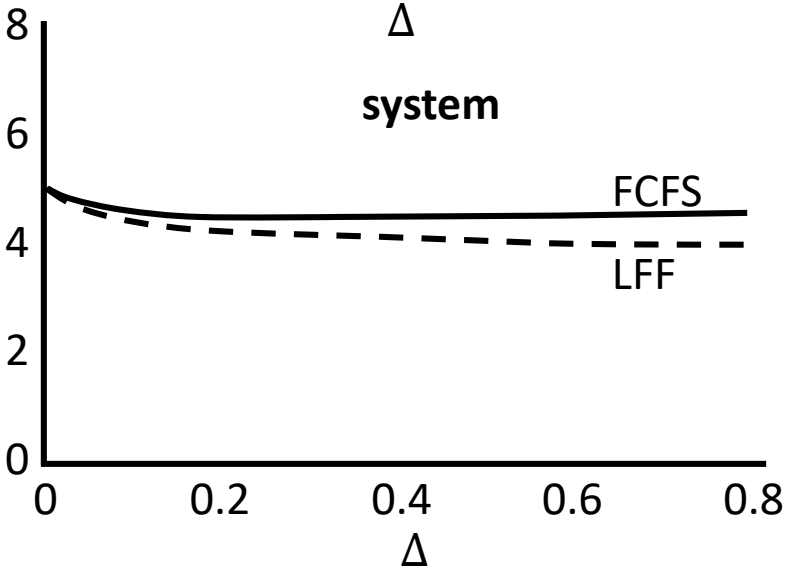


W Model Response Times – Shift 1 to 3









Improving Performance for Whom?

1. Going from FCFS ➔ LFF hurts some job classes
2. Under FCFS
   - even when *more flexibility* helps overall, it may hurt some job classes
   - *collaboration* may hurt some job classes
3. Under LFF, in nested systems
   - *more flexibility* helps all classes (win-win!)
   - *collaboration* helps all classes

   How can we improve performance, relative to FCFS, without hurting any job class?
4. Going from FCFS to the PF (primaries first) policy ensures more flexibility helps all classes

Primaries First Policy

Objective: Improve from status quo

Starting with an initial FCFS system,
call it System *P* (primary)

Create a new system through scheduling and increased flexibility that provides improved performance for all job classes.

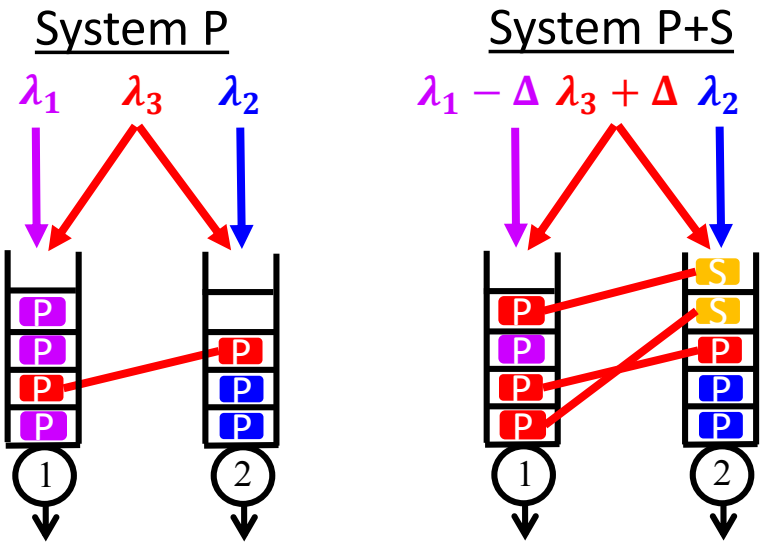
Primaries First Policy

System *P* (primary): Initial FCFS system with $S_i \subset S_j$.

System *P*+*S* (primary plus secondary):

Shift arrival rates from λ_i and λ_j

to $\lambda_i - \Delta$ and $\lambda_j + \Delta$ (still stable!)



- Original copies are primary
- New copies are secondary
- Primaries have preemptive priority over secondaries
- FCFS within primaries
- FCFS within secondaries

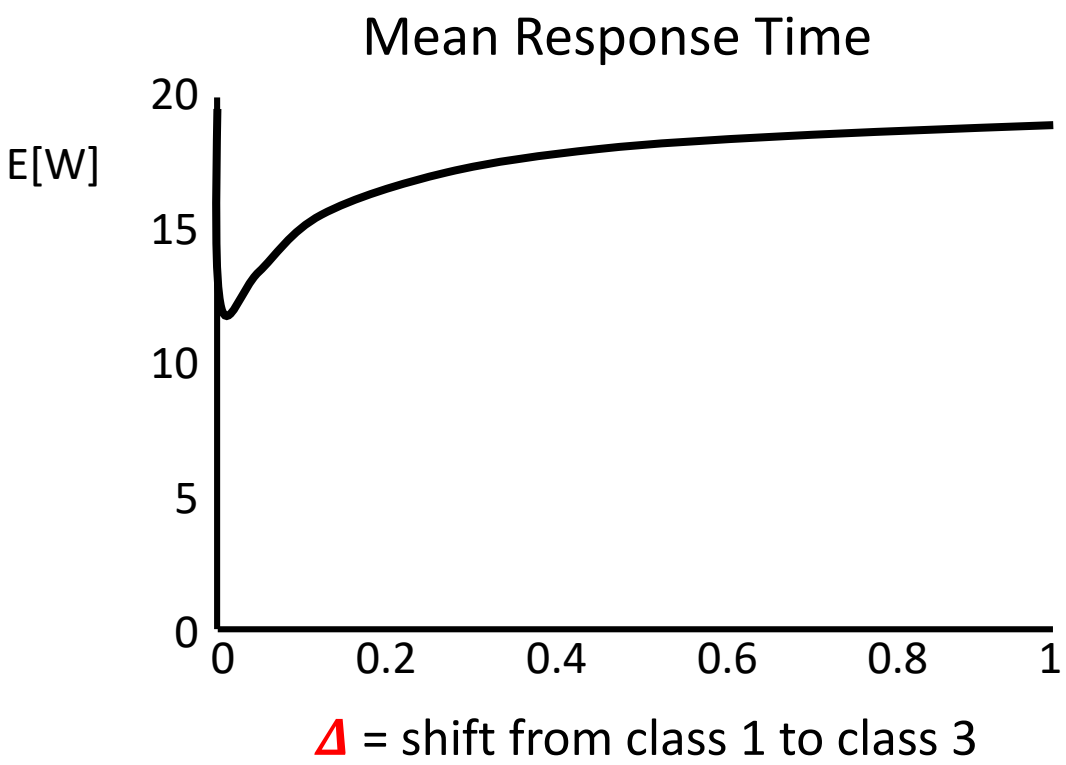
Under PF, all job classes benefit from increasing flexibility

Outline – Improving Performance

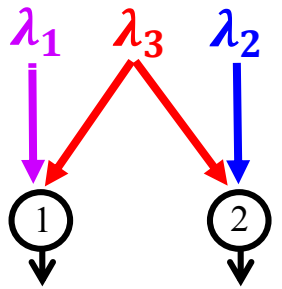
1. Improving performance by collaborating (or not)
2. Improving performance by scheduling
3. Improving performance by increasing flexibility
4. Improving performance for whom?
5. **Improving performance**
   (through increasing flexibility, collaborative)
   by how much?

FCFS and Convexity

Under **FCFS**, more flexibility **can hurt** overall response time, response time may be neither convex nor concave as a function of flexibility



Collaborative W



$\mu_1 = 1$

$\mu_2 = 100$

$\lambda_1 = 1 - \Delta$

$\lambda_2 = 99.9$

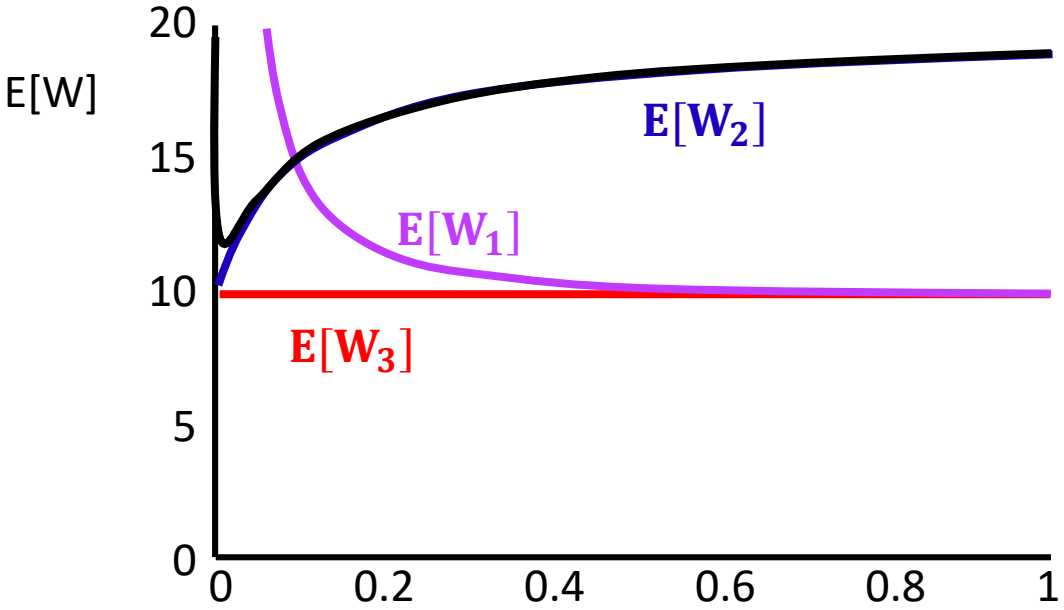
$\lambda_3 = \Delta$

FCFS and Per Class Convexity

Thm: As Δ, the arrival rate shifted from class 1 to class 3, increases

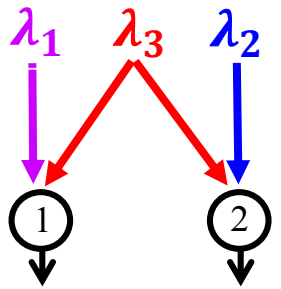
- $\mathrm{E}[W]$ may be nonmonotonic, nonconvex, nonconcave
- $\mathrm{E}[W_1]$ is decreasing and convex
- $\mathrm{E}[W_2]$ is increasing and concave
- $\mathrm{E}[W_3]$ is constant

Mean Response Time



Δ = shift from class 1 to class 3

Collaborative W



$\mu_1 = 1$

$\mu_2 = 100$

$\lambda_1 = 1 - \Delta$

$\lambda_2 = 99.9$

$\lambda_3 = \Delta$

FCFS and Per Class Convexity

Thm: As Δ, the arrival rate shifted from class 1 to class 3, increases

- $\mathrm{E}[W]$ may be nonmonotonic, nonconvex, nonconcave
- $\mathrm{E}[W_1]$ is decreasing and convex
- $\mathrm{E}[W_2]$ is increasing and concave
- $\mathrm{E}[W_3]$ is constant

The proof for the class-based response times follows from Part I.

Corollary:

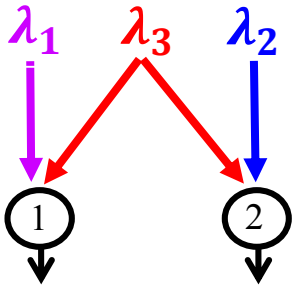
$$W_i =_{st} W\left(\lambda\left(R(S_i)\right), \mu(S_i)\right) + \sum_{j:S_i \subset S_j} W^Q\left(\lambda_j, \mu(S_j) - \lambda\left(R(S_j)\right) + \lambda_j\right)$$

The theorem can be generalized to arbitrary nested collaborative systems.

LRF and Convexity (Collaborative, Nested)

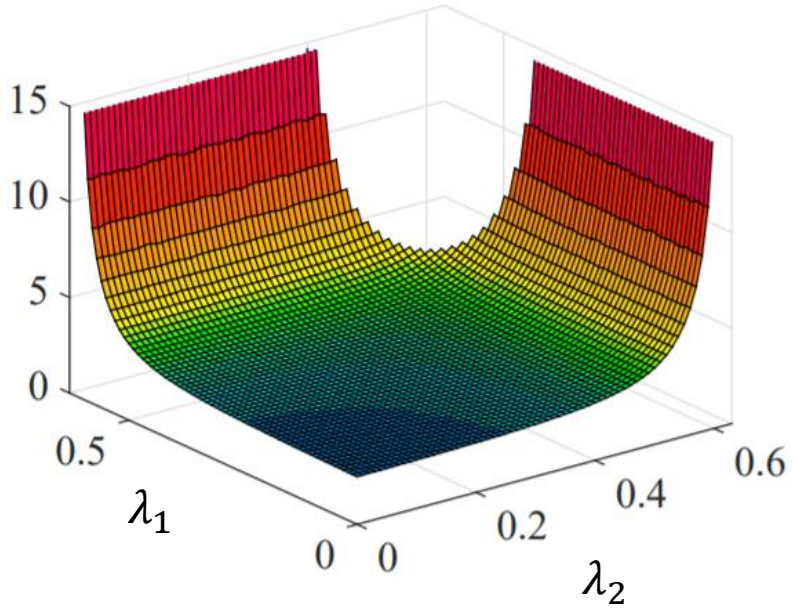
Thm: Overall mean response time is decreasing and convex in λ_3, holding $\lambda_1 + \lambda_1 + \lambda_3$ fixed.
The benefit of shifting from λ_1 to λ_3 is increasing in the amount shifted from λ_2 to λ_3.

Collaborative W



$\mu_1 = \mu_2 = 1$
$\lambda_1 + \lambda_1 + \lambda_3 = 1.8$

Mean Overall Response Time



Conclusions – Part I

1. Product forms for both collaborative and noncollaborative models can be shown easily by considering detailed states and order independence
2. Aggregating states yields partial results for performance measures
3. Fully flexible class experiences an M/M/1 system in the collaborative model
4. Complete, simple results hold for nested collaborative systems
5. Results for the collaborative model hold for the noncollaborative model given all servers busy

Conclusions – Part II

1. Collaboration and flexibility may not help
   - They help in symmetric FCFS systems
   - They help in nested LFF systems
2. Improving overall performance may not be fair
   - Under Primaries First (PF) more flexibility helps all classes
   - Under PF collaboration helps all classes
3. Under LFF, in nested systems, flexibility always helps
   - Diminishing marginal returns
   - Increasing cross-derivative effects