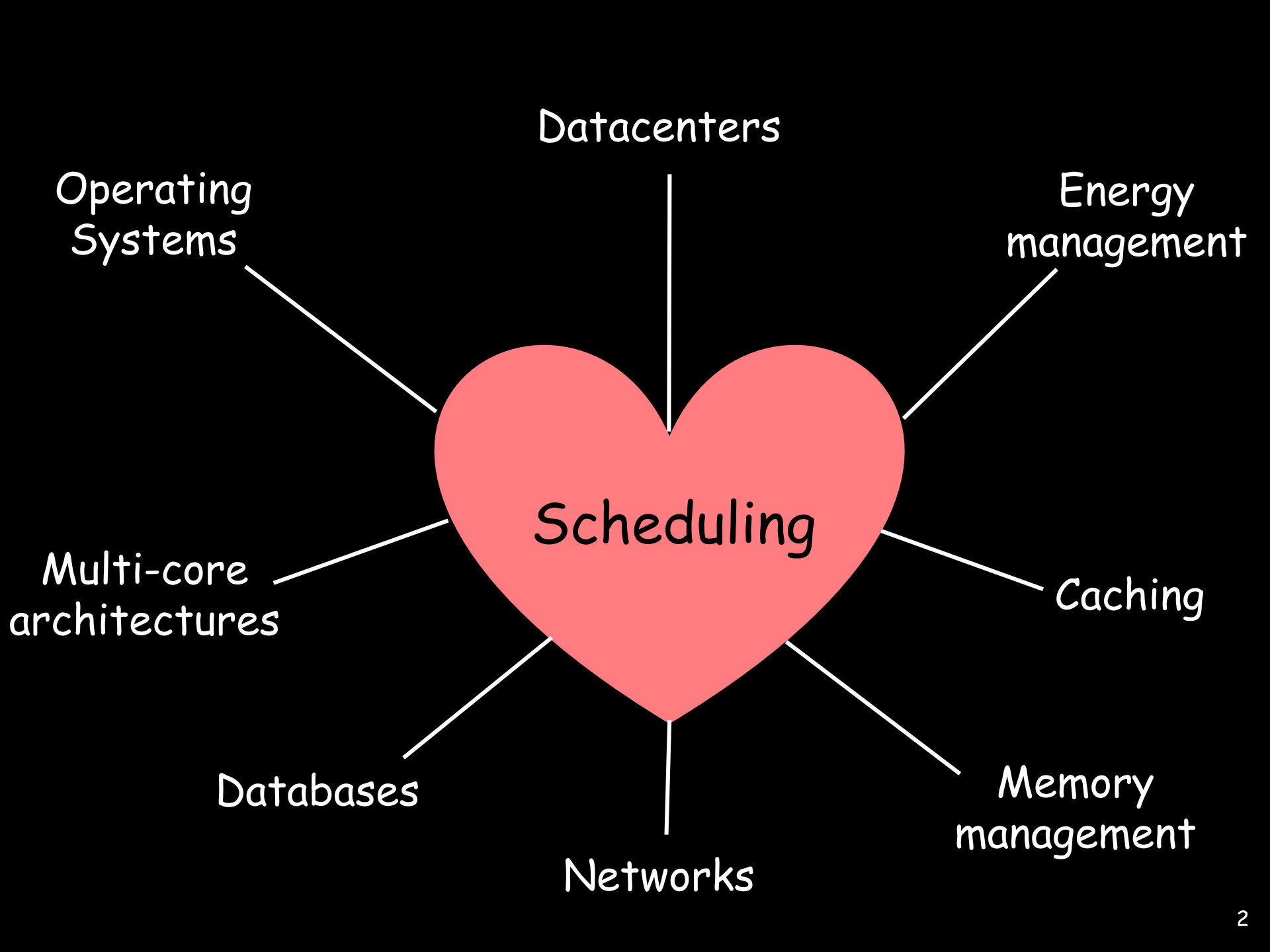
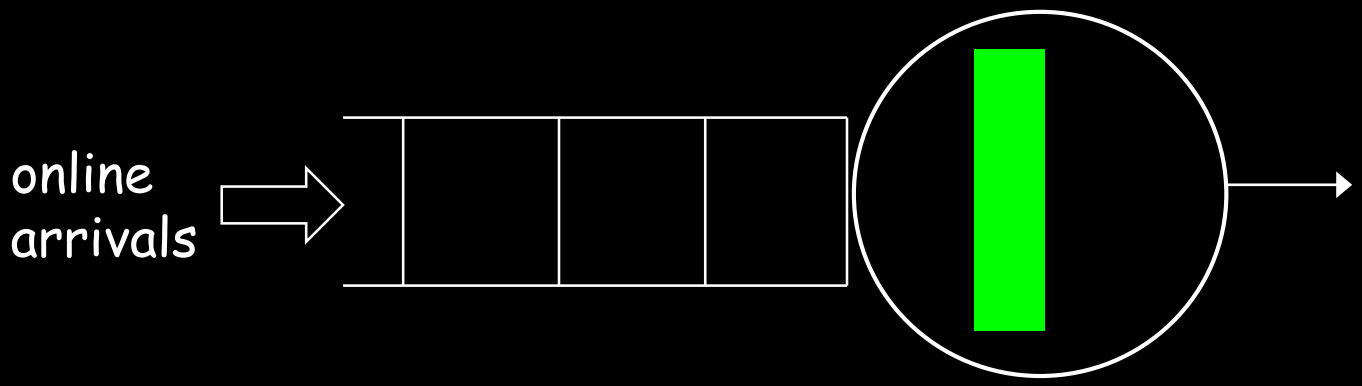
New Directions in Stochastic Scheduling

Mor Harchol-Balter
Computer Science Dept.
Carnegie Mellon University

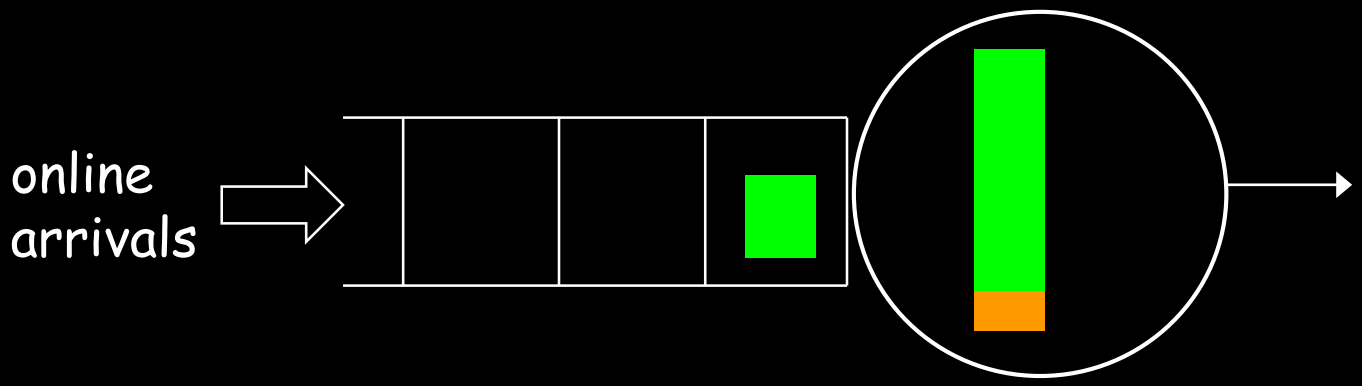
Based on: ``Open problems in queueing theory inspired by datacenter computing."
Queueing Systems , vol. 97, no. 1, 2021, pp. 3--37.


Datacenters
Operating Systems
Energy management
Scheduling
Multi-core architectures
Caching
Databases
Networks
Memory management

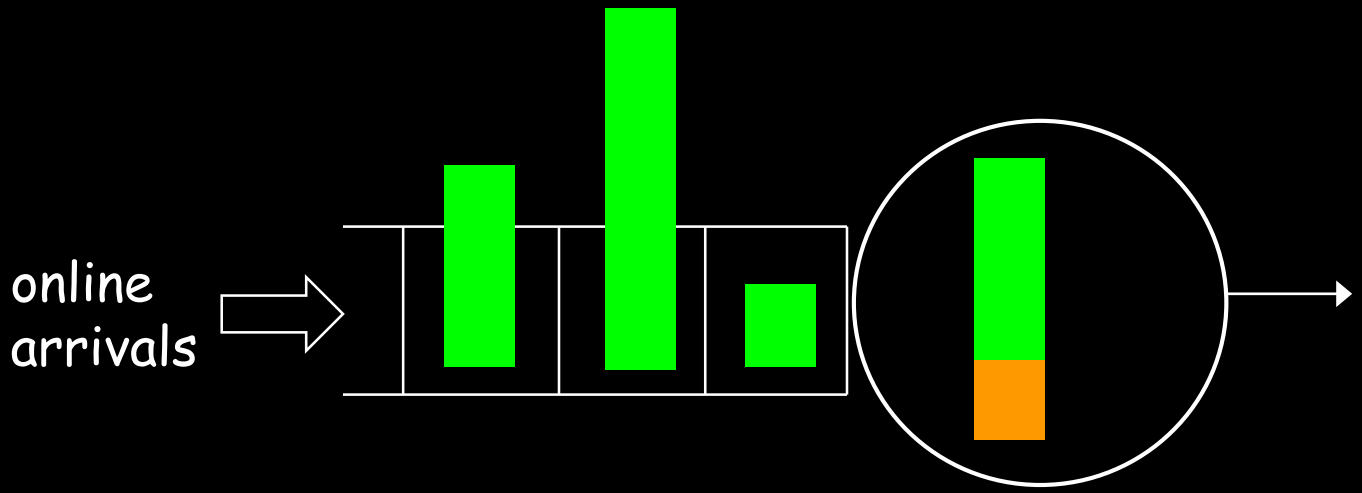
Basic Terminology



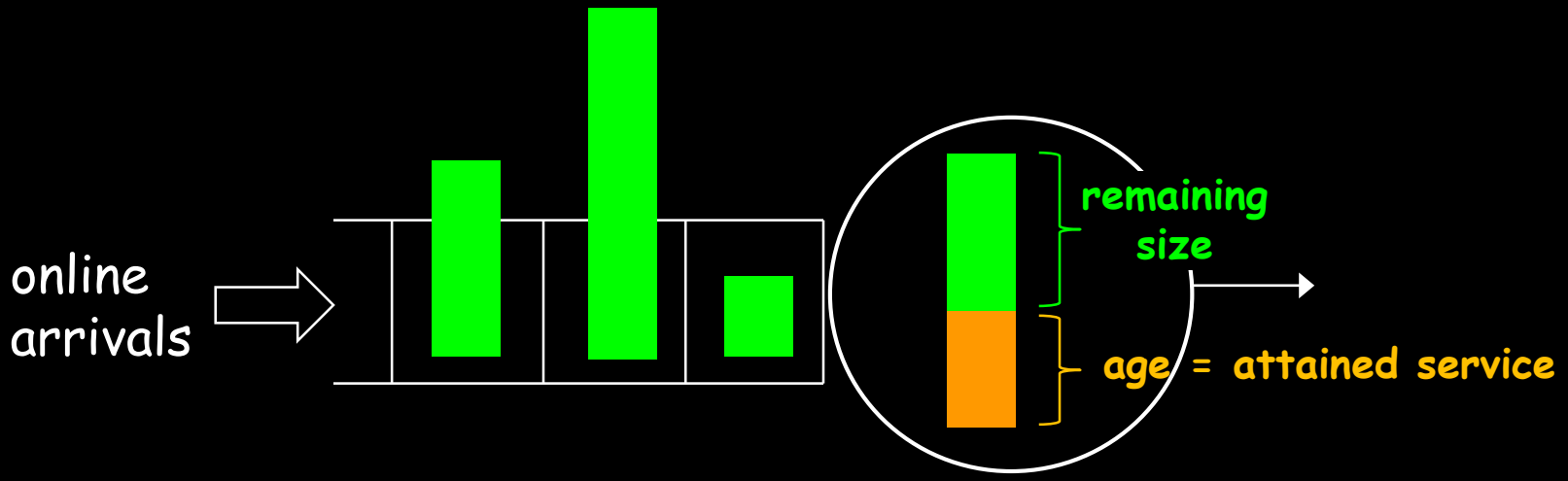
Basic Terminology



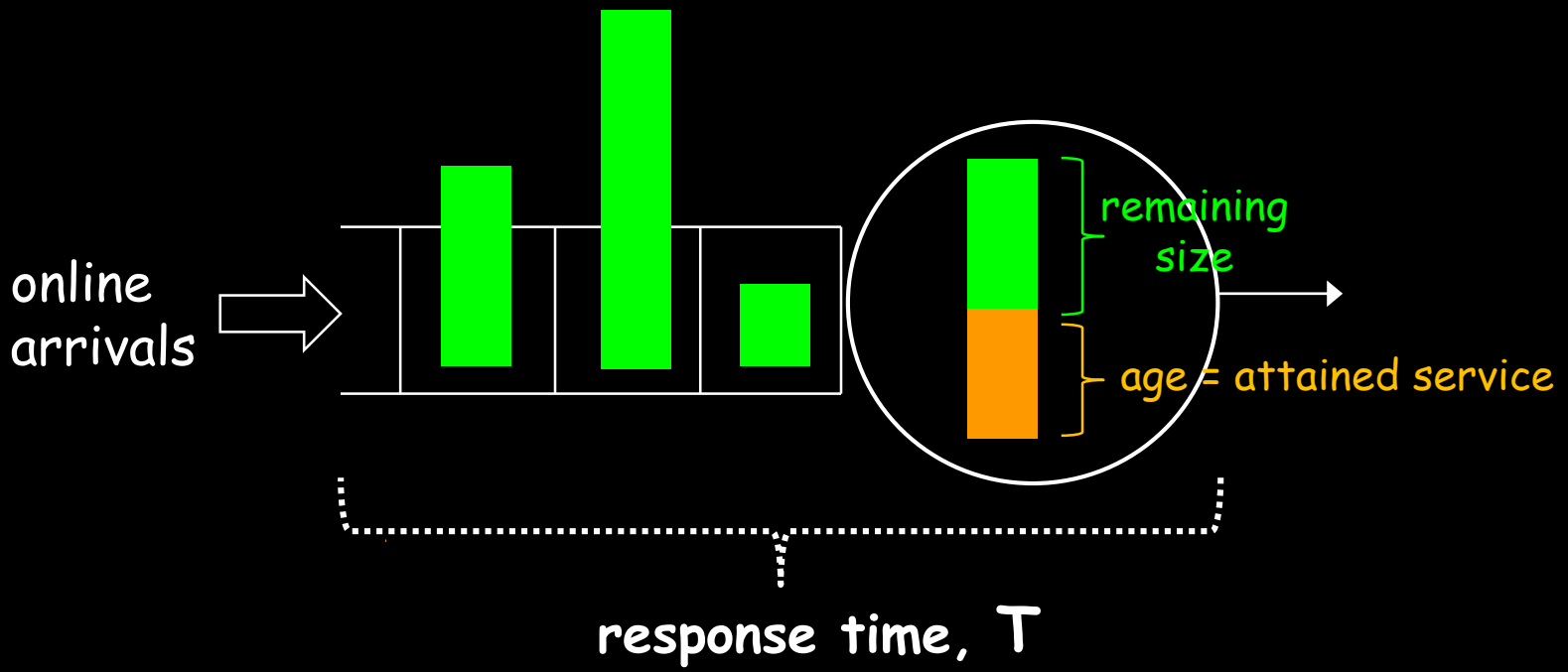
Basic Terminology



Basic Terminology

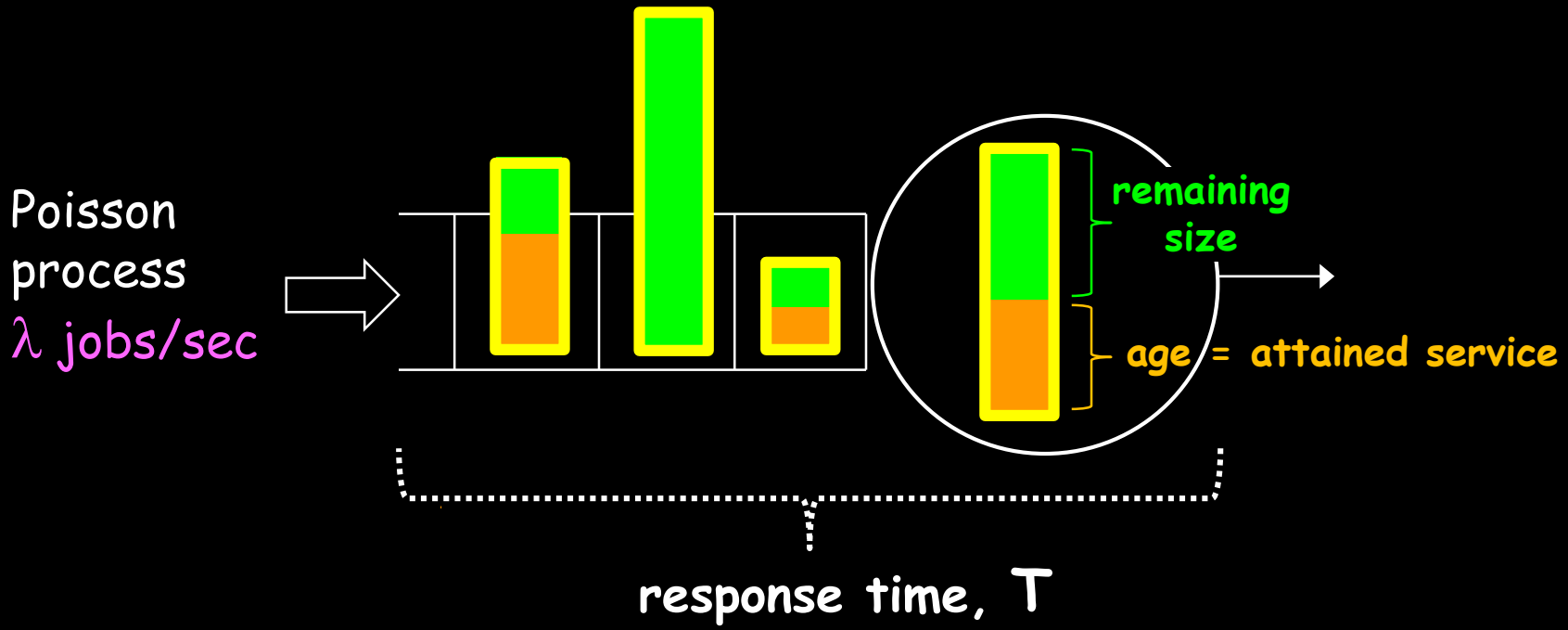


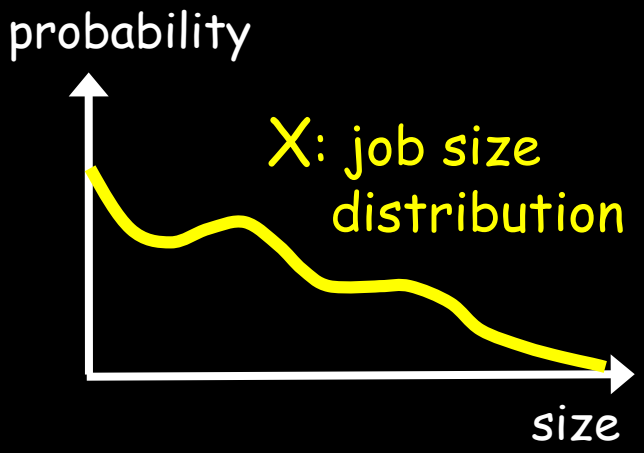
Basic Terminology



Scheduling Policy
(preempt-resume)

M/G/1 with Scheduling

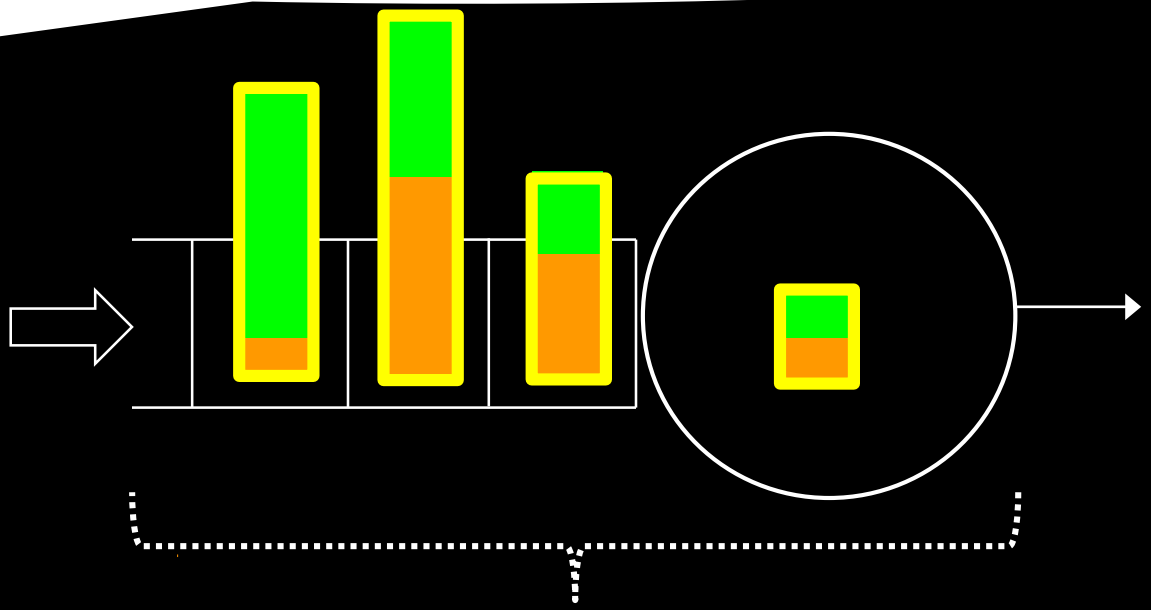




"Load" = fraction time server busy

$$\rho = \lambda \cdot E[X] < 1$$

Q: What scheduling policy minimizes E[T]?

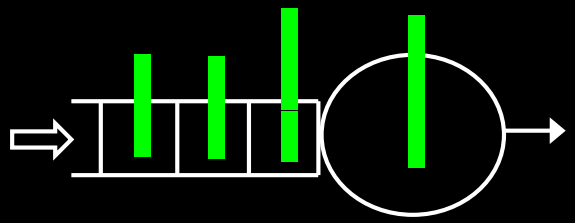


A: SRPT -Shortest Remaining Processing Time

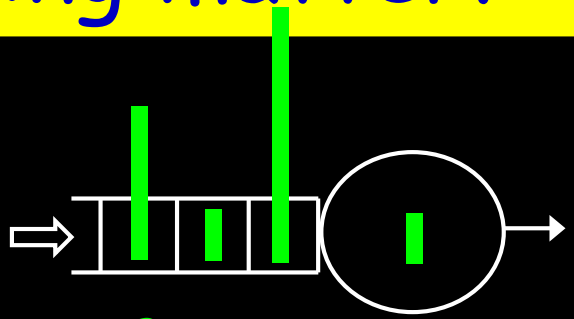
optimal in worst-case!

M/G/1 analysis – [Schrage 1966]

How much does scheduling matter?



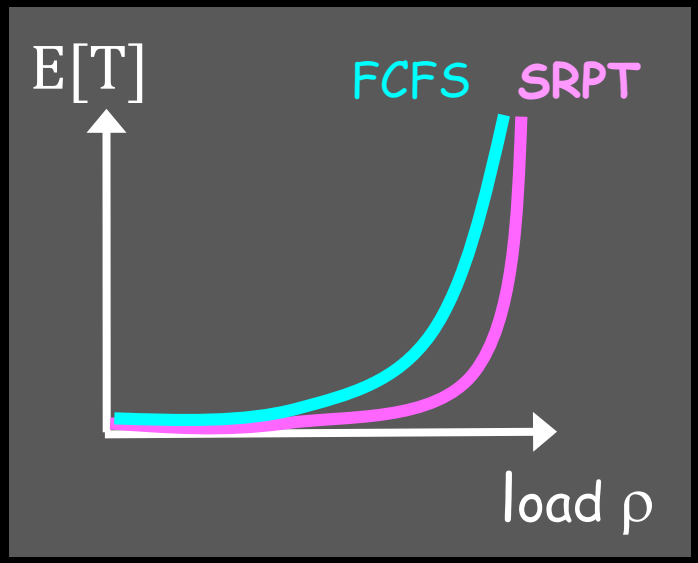
$C_X^2 = 1$



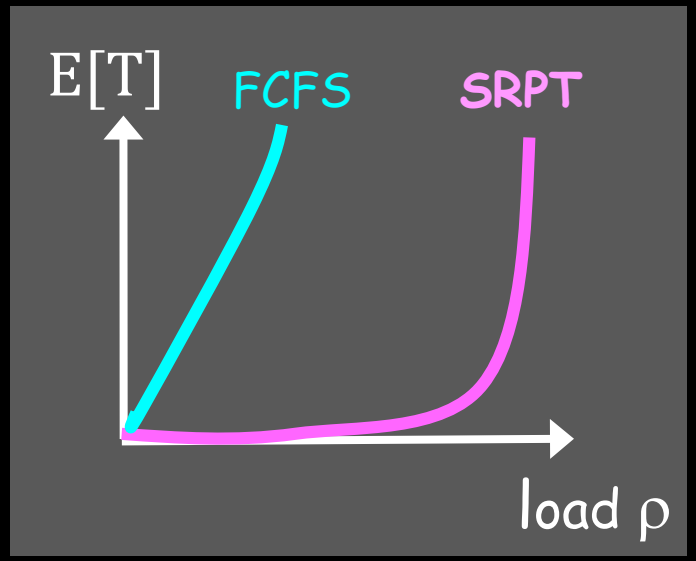
$C_X^2 = 100$

$$C_X^2 = \frac{Var(X)}{E[X]^2}$$

Low variability

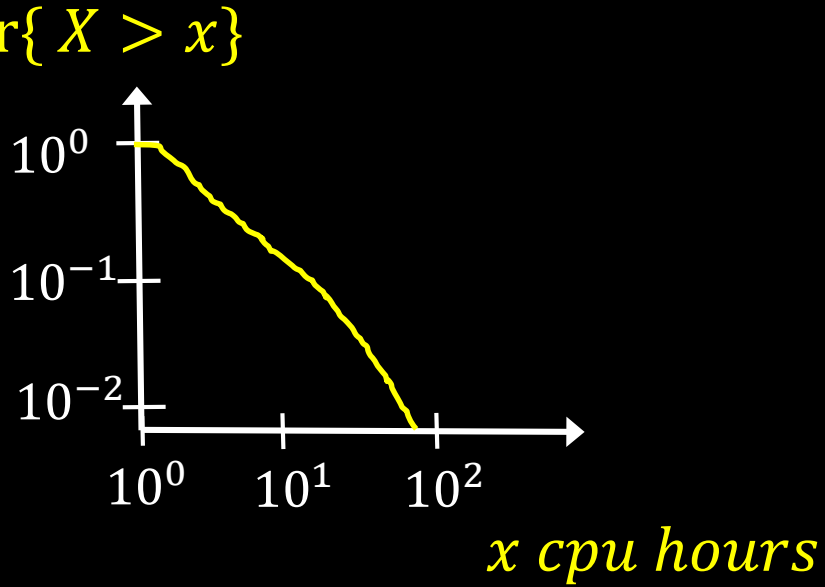


High variability



Empirical Job Size Distribution

UNIX jobs.
[Harchol-Balter, Downey – SIGMETRICS 1996]



$\underline{X = Job\ Size}$

$X \sim BoundedPareto(\alpha \approx 1)$

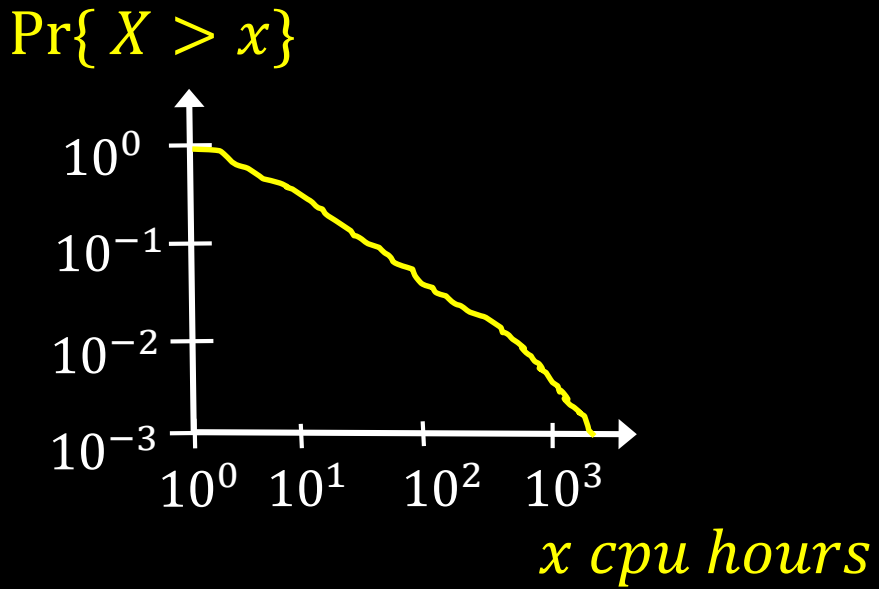
$C_X^2 = 50$

$Top\ 1\%\ of\ jobs = 50\%\ of\ load$

Upshot: Scheduling matters

Empirical Job Size Distribution

Borg Scheduler at Google [Tirmazi et al., EUROSYS 2020]



$\underline{X = Job\ Size}$

$X \sim BoundedPareto(\alpha = 0.69)$

$C_X^2 = 23{,}000$

$Top\ 1\%\ of\ jobs = 99\%\ of\ load$

Upshot: Scheduling REALLY matters!

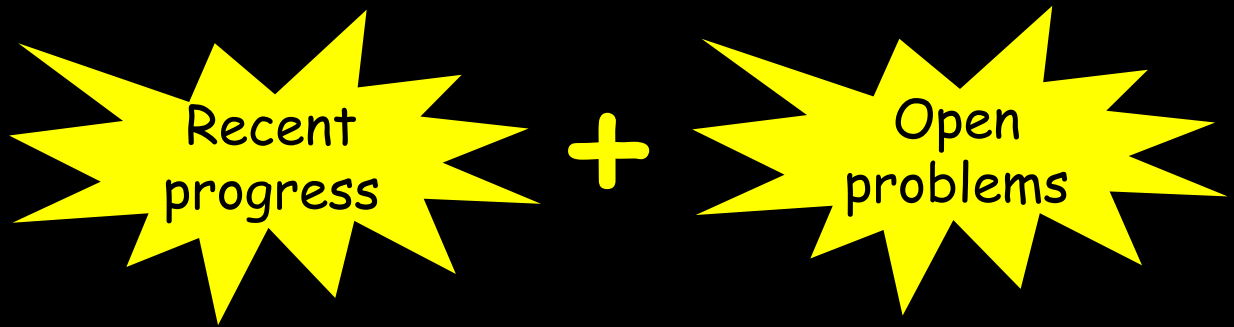
Outline: New Directions

I. Expanding Scheduling for M/G/1

II. Scheduling for multi-server M/G/k

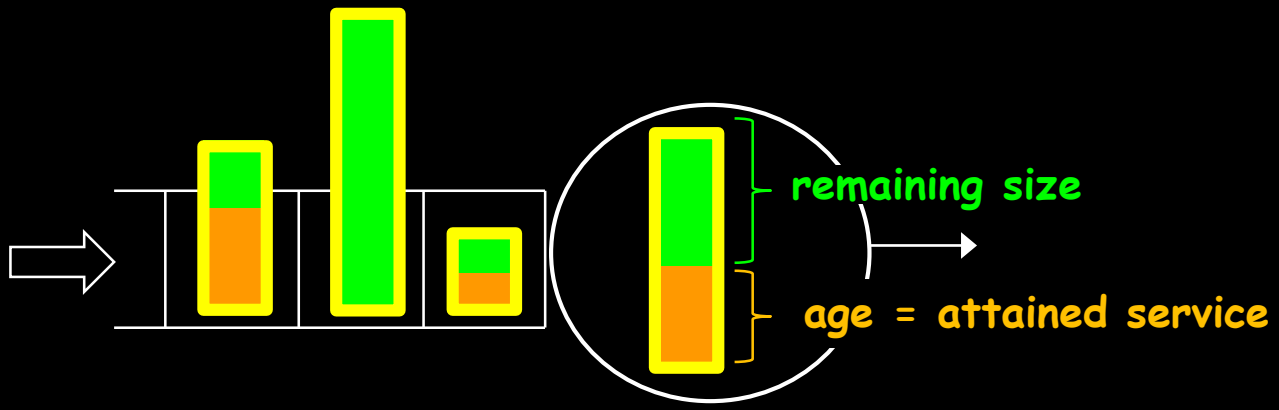
III. Today's multi-server jobs

IV. Scheduling malleable jobs with flexible parallelizability

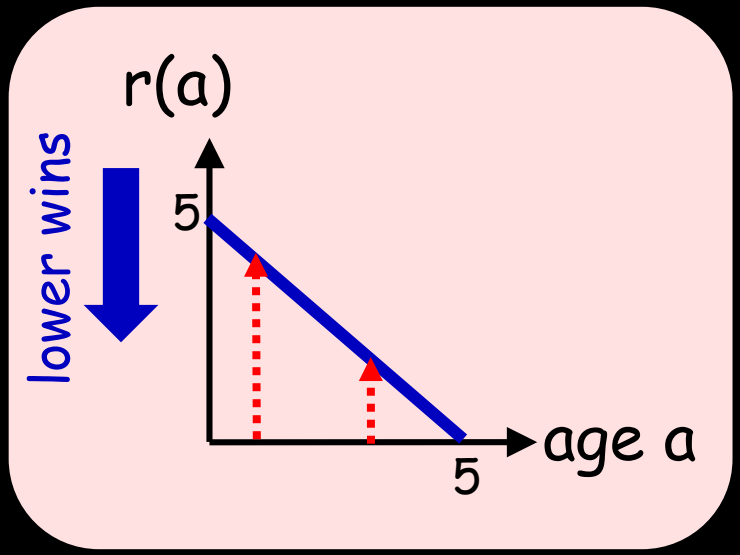


I. Expanding Scheduling for M/G/1

So far we've assumed known size …



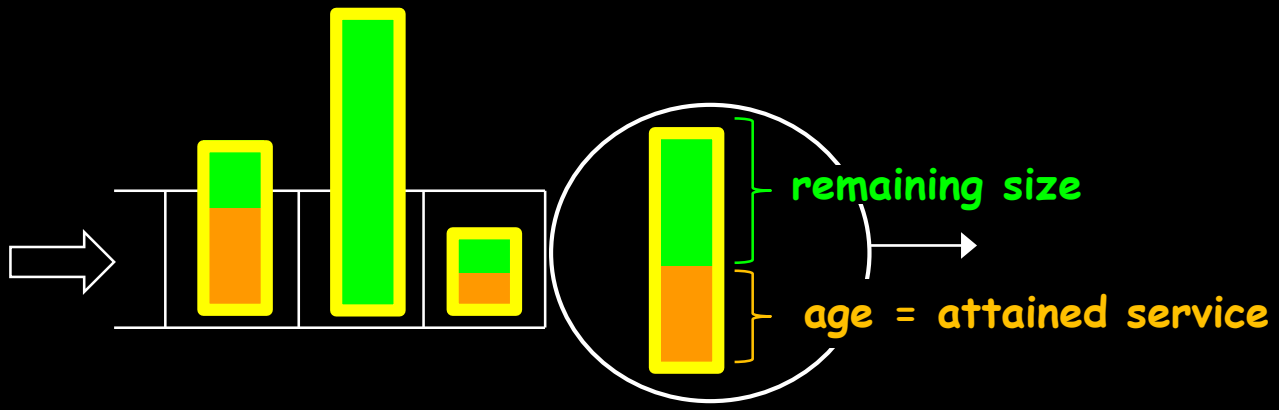
SRPT



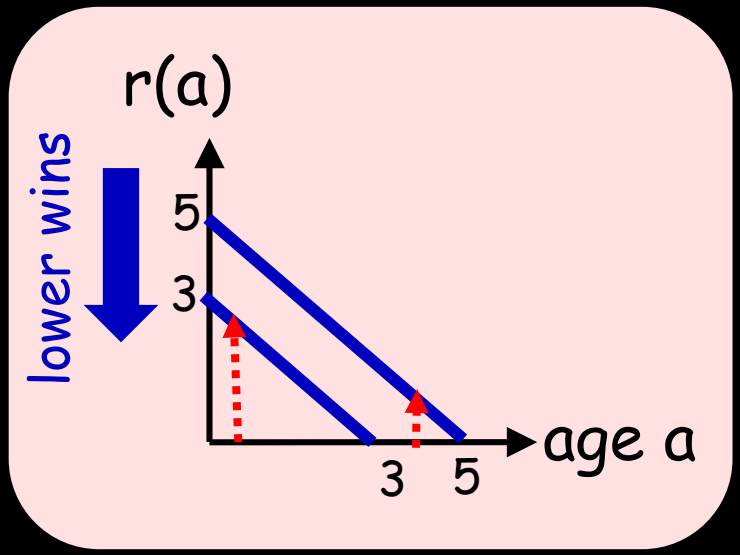
SRPT minimizes E[T].

I. Expanding Scheduling for M/G/1

So far we've assumed known size …



SRPT



SRPT minimizes E[T].

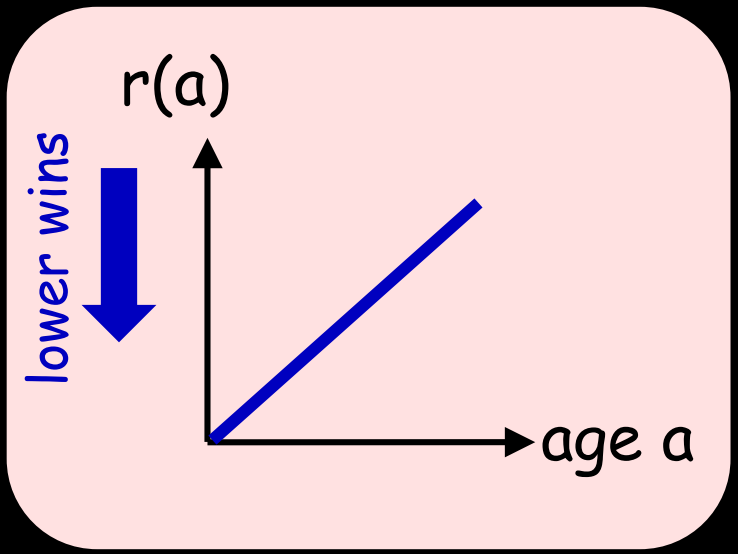
But what if size is NOT known?

IF job size distribution has Decreasing Failure Rate (DFR), then LAS minimizes E[T].

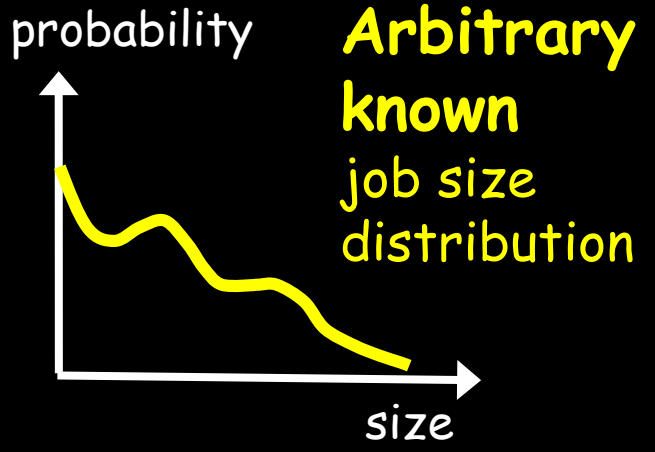
Least Attained Service

(see papers by Righter, Aalto, Ayesta)

LAS



But what if size is NOT known?



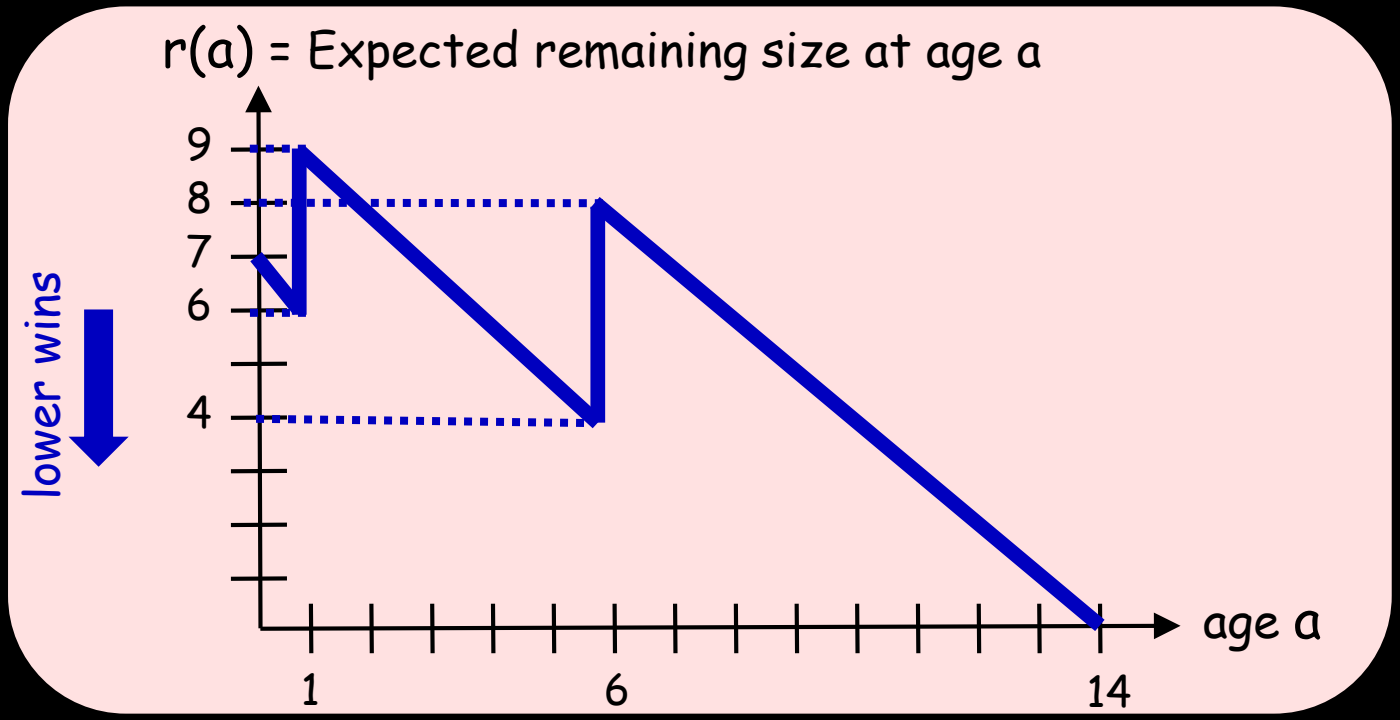
Arbitrary known job size distribution

SERPT -- Shortest Expected Remaining Processing Time?

SERPT

$$X = \begin{cases} 1 & w.p. \ \frac{1}{3} \\ 6 & w.p. \ \frac{1}{3} \\ 14 & w.p. \ \frac{1}{3} \end{cases}$$

$$r(a) = E[X - a \mid X > a]$$



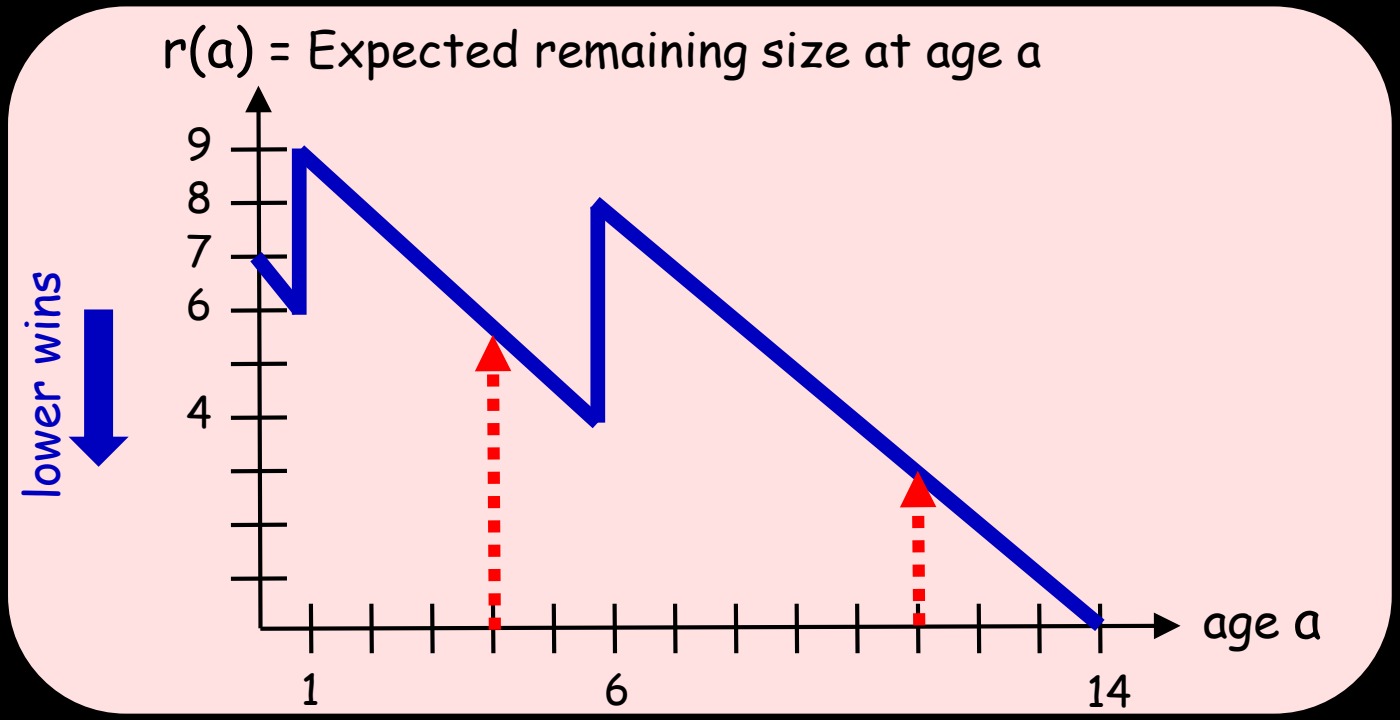
rank NOT monotonic

SERPT

Always run job with lowest rank

$$X = \begin{cases} 1 & w.p. \ \frac{1}{3} \\ 6 & w.p. \ \frac{1}{3} \\ 14 & w.p. \ \frac{1}{3} \end{cases}$$

$$r(a) = E[X - a \mid X > a]$$



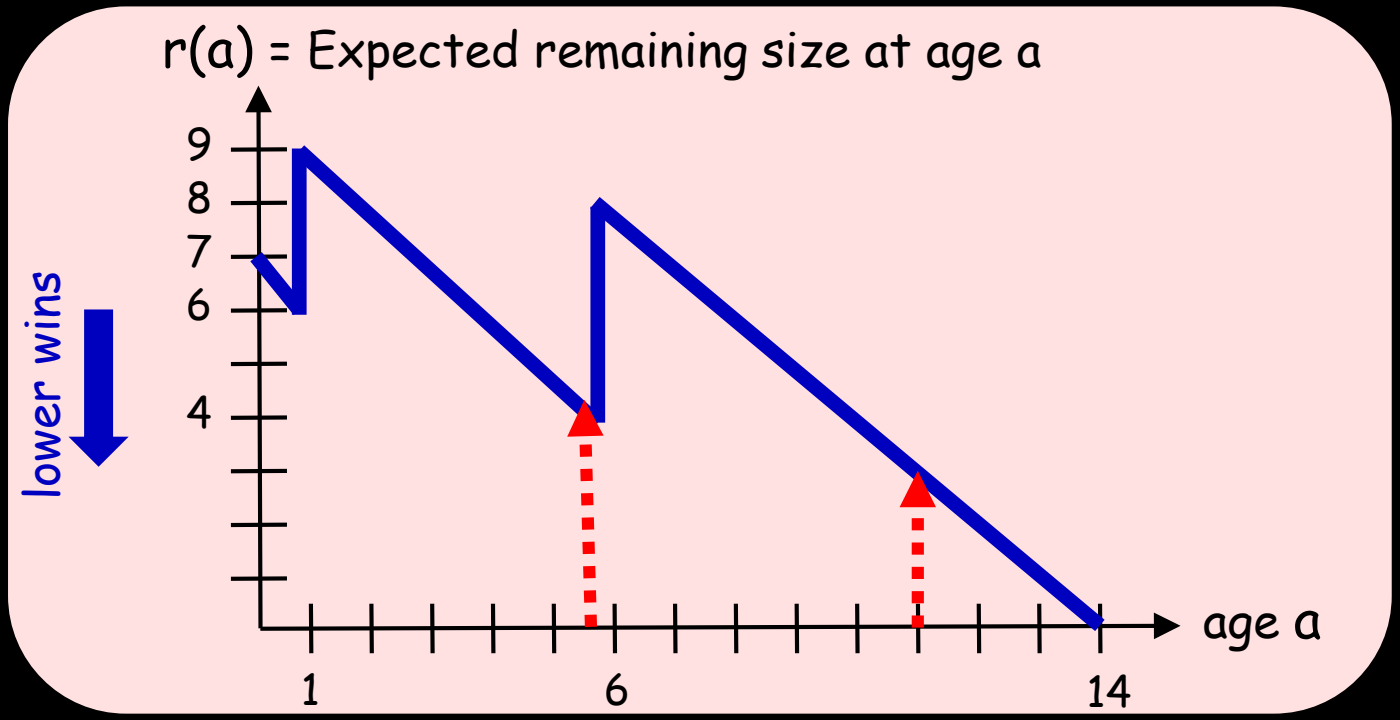
rank NOT monotonic

SERPT

Always run job with lowest rank

$$X = \begin{cases} 1 & w.p. \ \frac{1}{3} \\ 6 & w.p. \ \frac{1}{3} \\ 14 & w.p. \ \frac{1}{3} \end{cases}$$

$$r(a) = E[X - a \mid X > a]$$



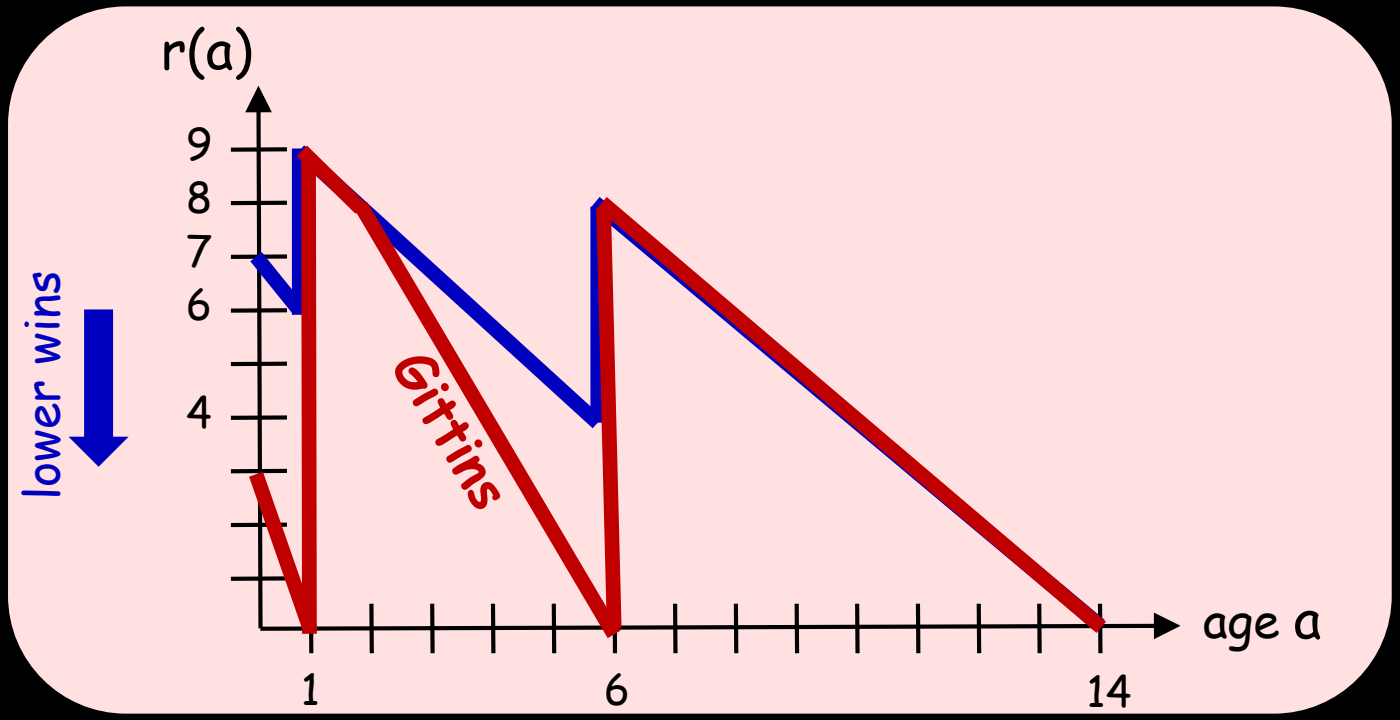
rank NOT monotonic

Gittins Index

Always run job with lowest rank

$$X = \begin{cases} 1 & w.p. \ \frac{1}{3} \\ 6 & w.p. \ \frac{1}{3} \\ 14 & w.p. \ \frac{1}{3} \end{cases}$$

$$r(a) = \inf_{\Delta} \frac{E[\min\{X - a\,,\ \Delta\} \mid X > a]}{\Pr\{X \leq a + \Delta \mid X > a\}}$$



rank NOT monotonic

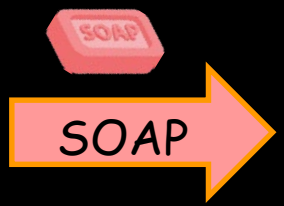
State-of-art in Scheduling

Pre-2018 couldn't analyze response time for policies like SERPT, GITTINS, …

… couldn't handle non-monotonic rank functions

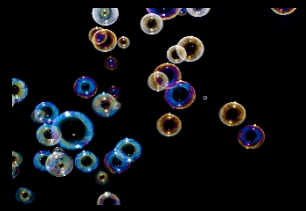
Recent progress

Given: **any** rank function



[Scully et al. Sigmetrics '18]

Closed-form response time (mean & transform)



"SOAP bubbles" paper

Open problems

- What if don't have accurate age?
- What if can only estimate job size distribution?

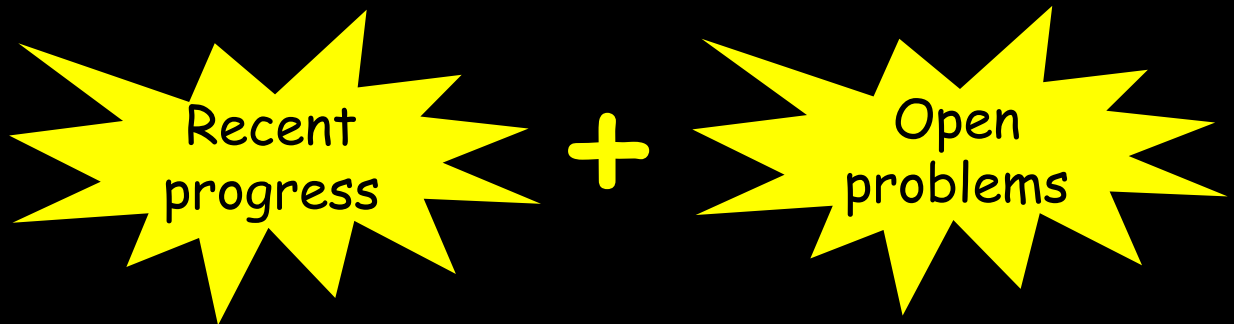
Outline: New Directions

I. Expanding Scheduling for M/G/1

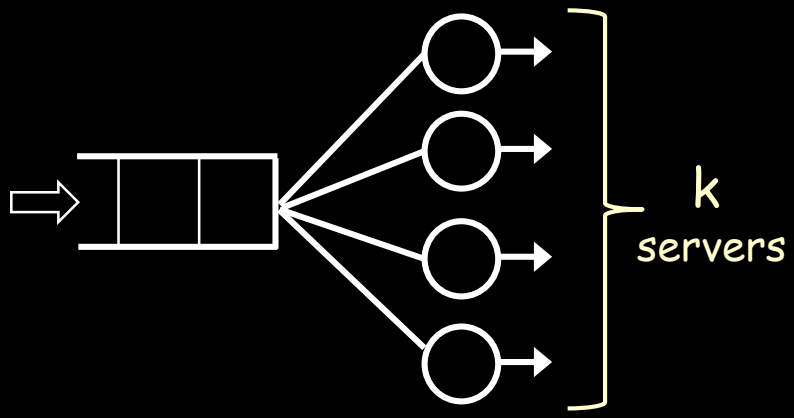
II. Scheduling for multi-server M/G/k

III. Today's multi-server jobs

IV. Scheduling malleable jobs with flexible parallelizability

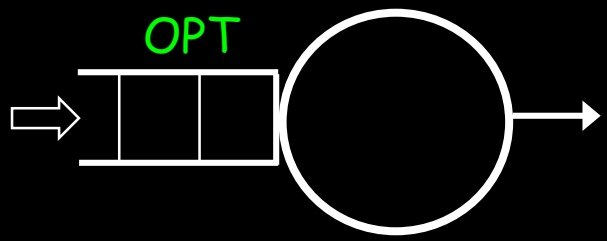


II. Multi-server system



Q: How to schedule to minimize E[T]?
(sizes known)

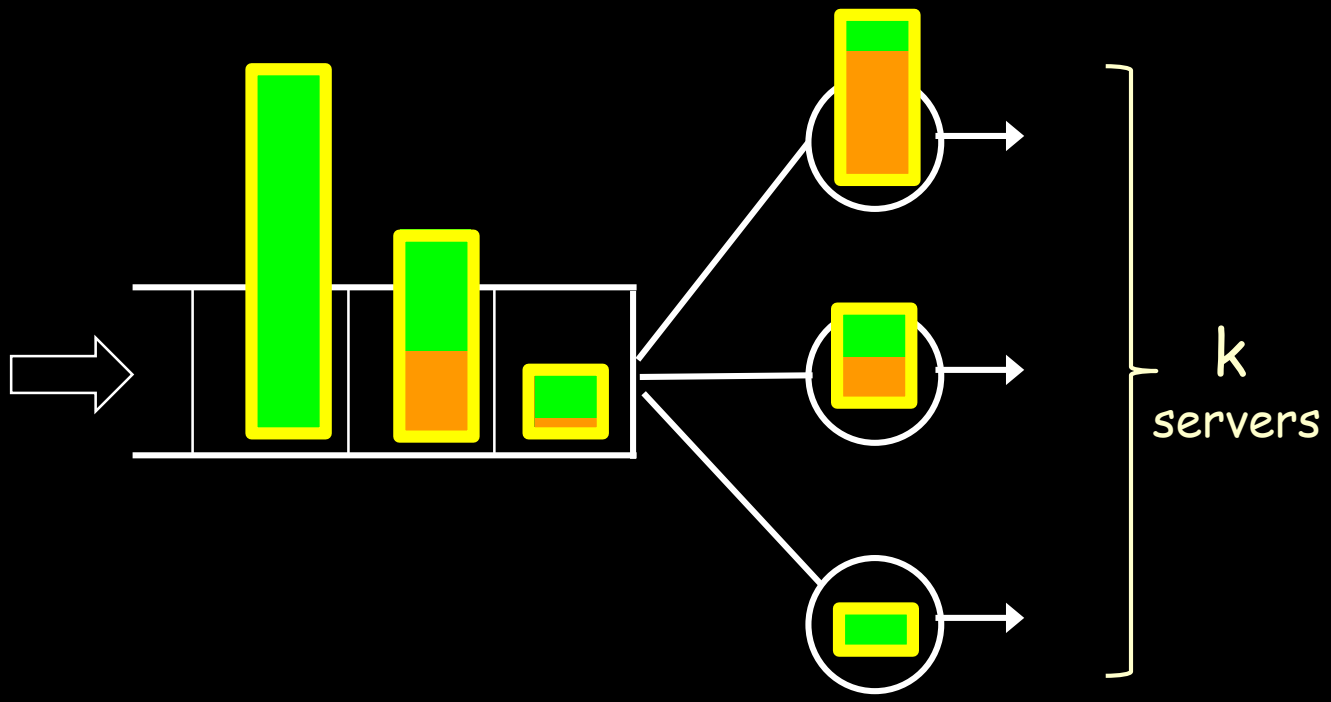
Lower Bound



Want to match single powerful OPT server.

OPT is SRPT!

SRPT-k



At all times run k jobs with shortest remaining times.

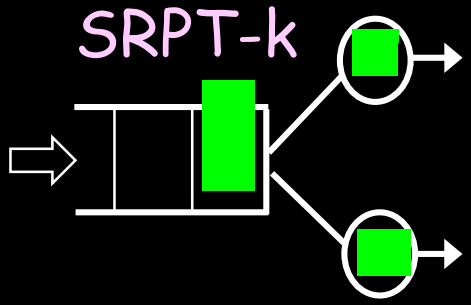
SRPT-k is FAR from OPT in worst-case

Theorem: [Leonardi, Raz 1997]

arrivals!

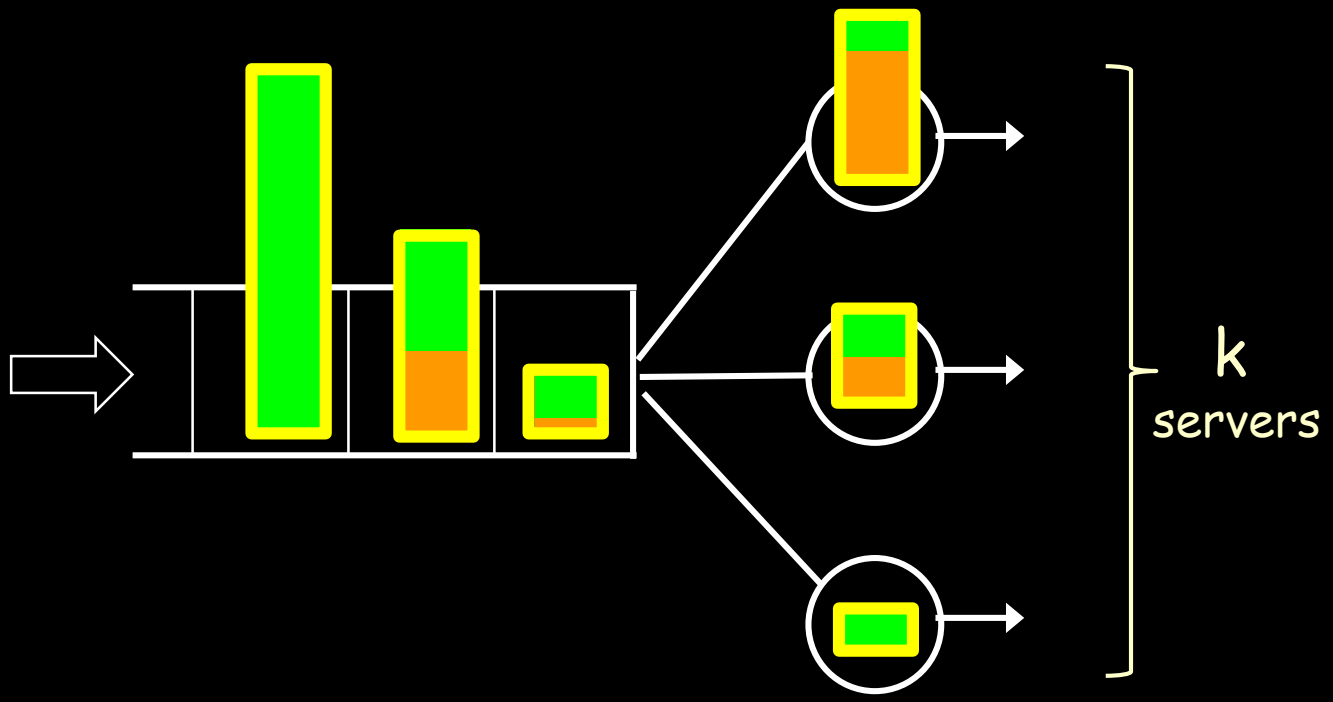
$$\text{Competitive Ratio SRPT-}k\ (k \geq 2) = O\left(\log\left(\min\left(\frac{n}{k}, \frac{\text{Max job size}}{\text{Min job size}}\right)\right)\right)$$

ratio can be high!



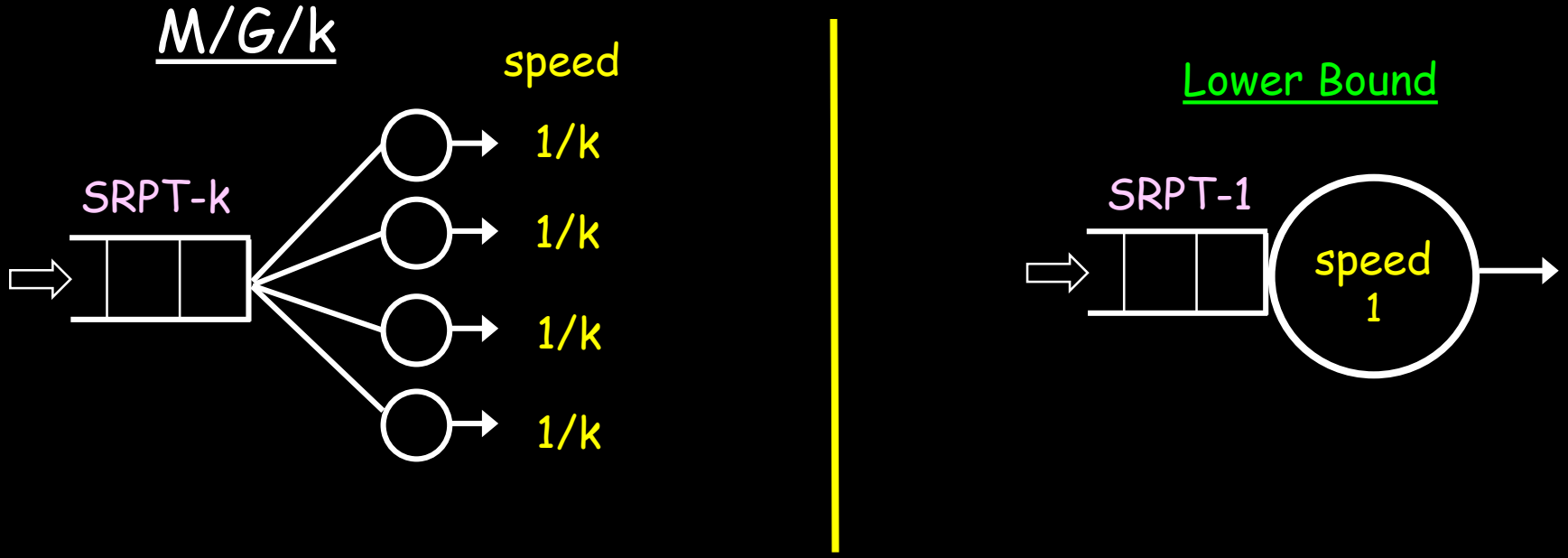
... and no other policy does better

But what about SRPT-k for M/G/k?



Scheduling for M/G/k is non-existent

Recent Progress [Grosof et al., IFIP Performance 2018]



2 results:

1) First Bound: $E[T]^{SRPT-k} \leq E[T]^{SRPT-1} + \frac{2}{\rho} kE[X] \ln\left(\frac{1}{1-\rho}\right)$

2) Optimality: $\lim_{\rho \to 1} \frac{E[T]^{SRPT-k}}{E[T]^{SRPT-1}} = 1$

Empiricaly SRPT-k is great at all loads.

Scheduling in M/G/k: Wrap-up

Recent progress

[Grosof et al. in Performance '18]

First Bounds on: SRPT-k, PSJF-k, LAS-k, RS-k

[Scully et al. in Sigmetrics '21]

First Bounds on: Gittins-k

Open problems

1. Better upper bounds for lighter loads
2. Better lower bound than single fast server
3. Usual questions for Gittins-k on learning job size distribution, etc.

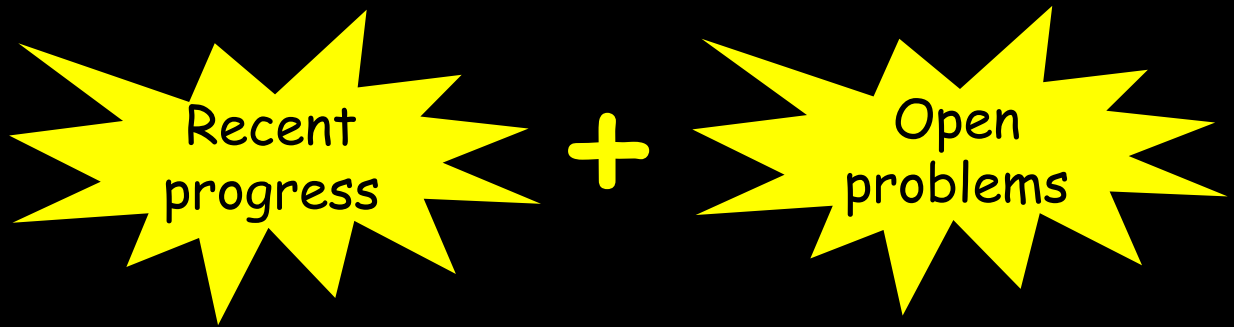
Outline: New Directions

I. Expanding Scheduling for M/G/1

II. Scheduling for multi-server M/G/k

III. Today's multi-server jobs

IV. Scheduling malleable jobs with flexible parallelizability



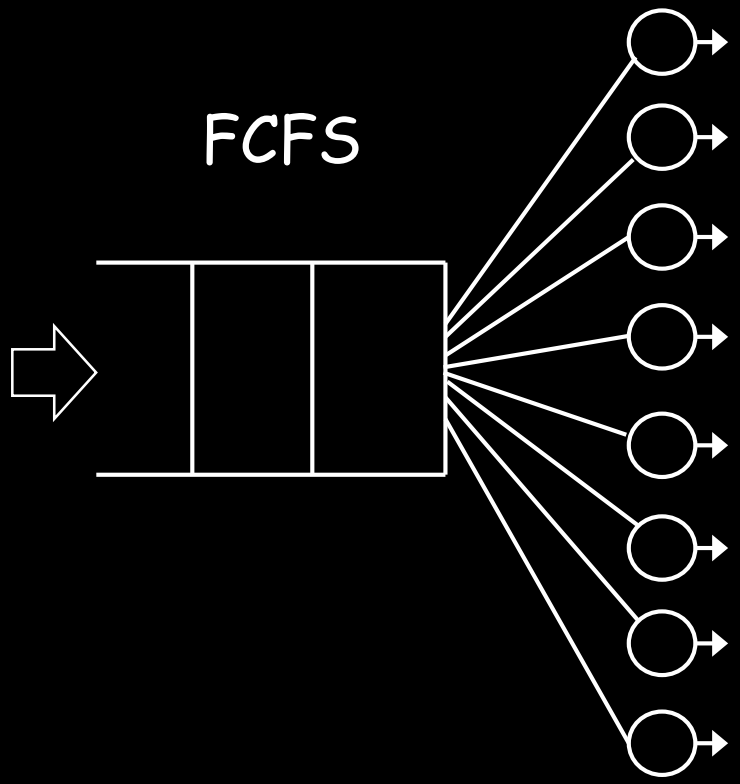
III. Today's Multi-server Jobs

job = (# servers, time)

Numbers from Google Borg Trace 2020. Numbers are normalized.

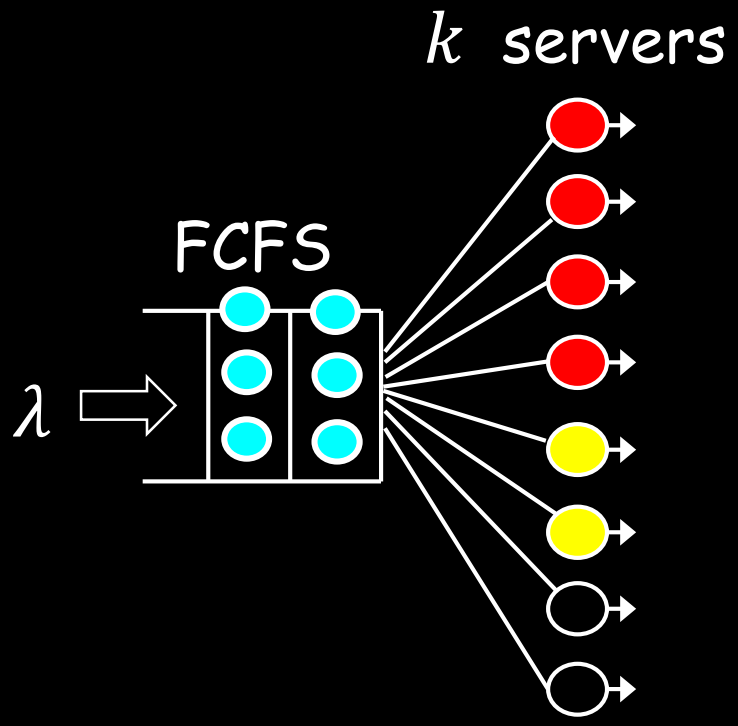
Today's Multi-server Jobs

job = (# servers, time)



CloudSim, iFogSim, EPSim, GridSim, Borg Scheduler, etc.

Multi-server job model



Each arrival:

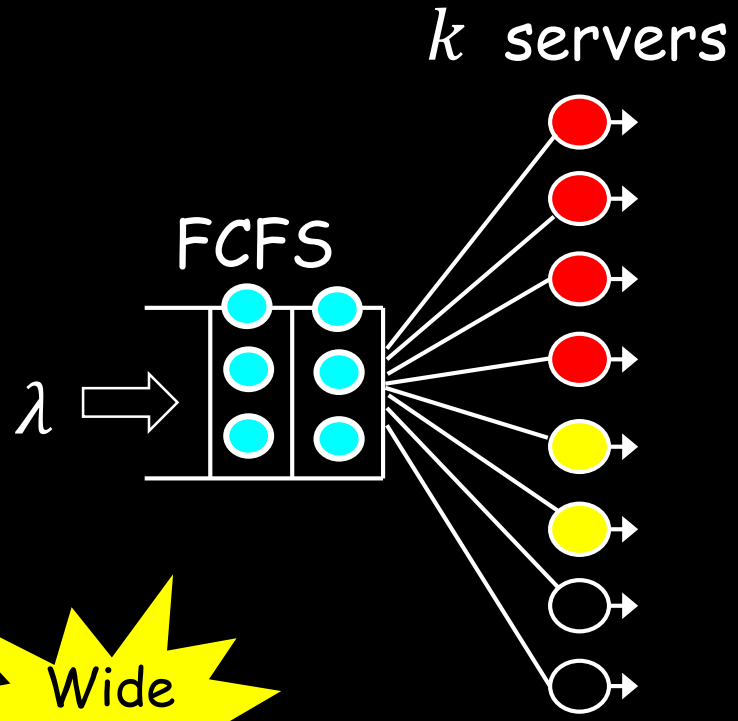
with prob. p_i

- request n_i servers
- for S_i time

$$\rho \equiv \frac{\lambda \cdot \sum_i p_i \, n_i \, E[S_i]}{k}$$

CloudSim, iFogSim, EPSim, GridSim, Borg Scheduler, etc.

Multi-server job model



Each arrival:

with prob. p_i

- request n_i servers
- for S_i time

$$\rho \equiv \frac{\lambda \cdot \sum_i p_i \, n_i \, E[S_i]}{k}$$

Wide OPEN!

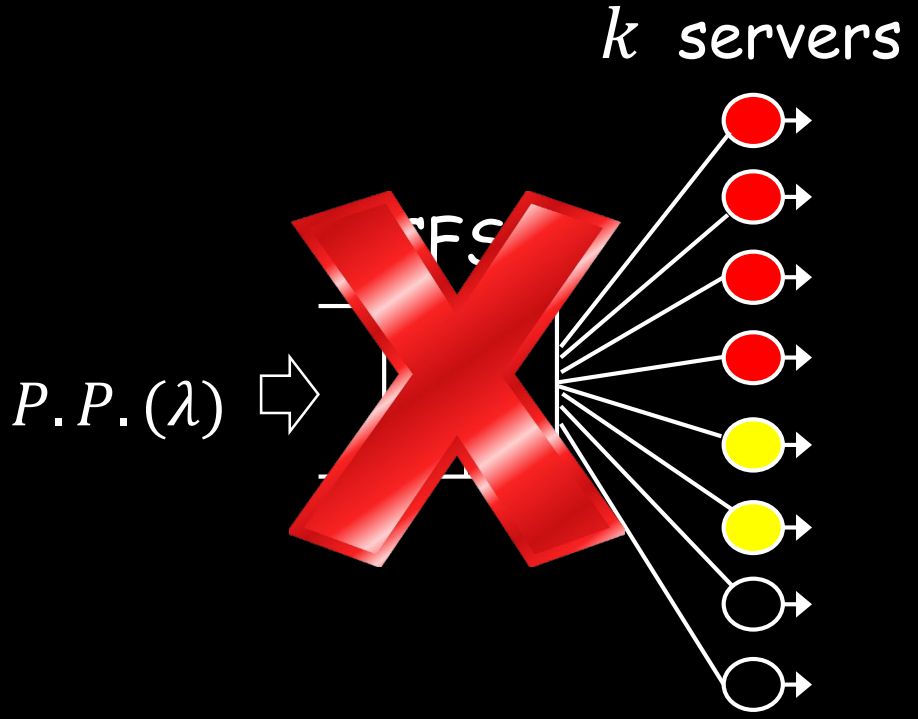
Q: What is E[T] ?

Open for k>2 servers
[Brill, Green 84], [Filippopoulos,Karatza 07]

Q: What is the stability region ?

Only known if $S_i \sim Exp(\mu)\ \forall i$
[Rumyantsev, Morozov 2017]

Dropping model is much easier!



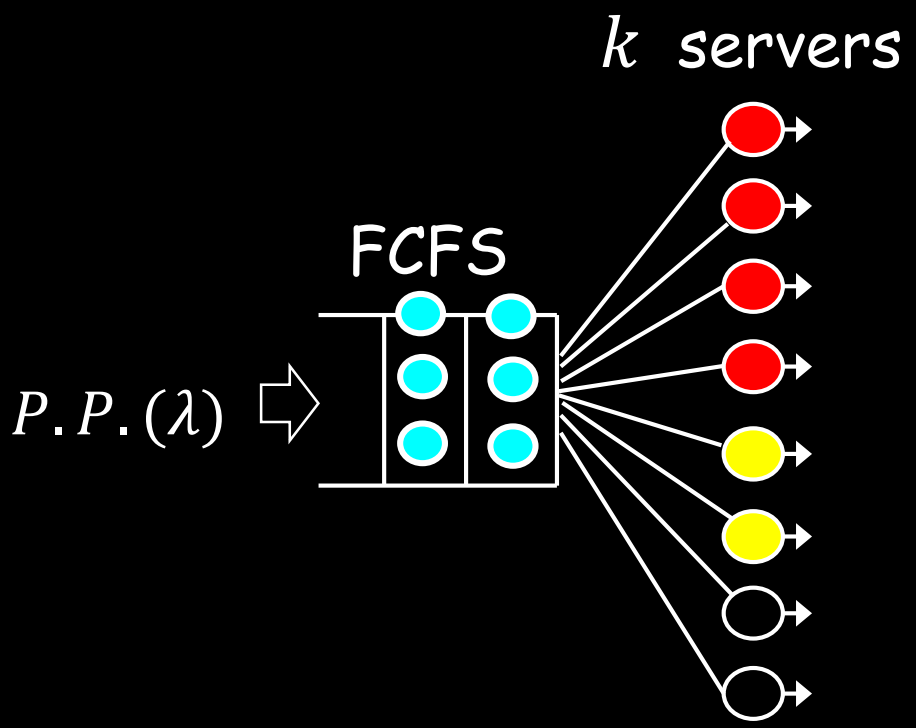
Much Easier!

Product-form solution

[Arthurs, Kaufman '79]
[Whitt '85]
[vanDijk '89]
[Tikhonenko '05]

Dropping model is common in streaming for communication networks.

Multi-server job model: Wrap-up



Q: What is E[T] ?

Q: What is the stability region ?

Q: How should we think about "job size"?
-- for prioritization
-- for isolating smalls, e.g. SITA

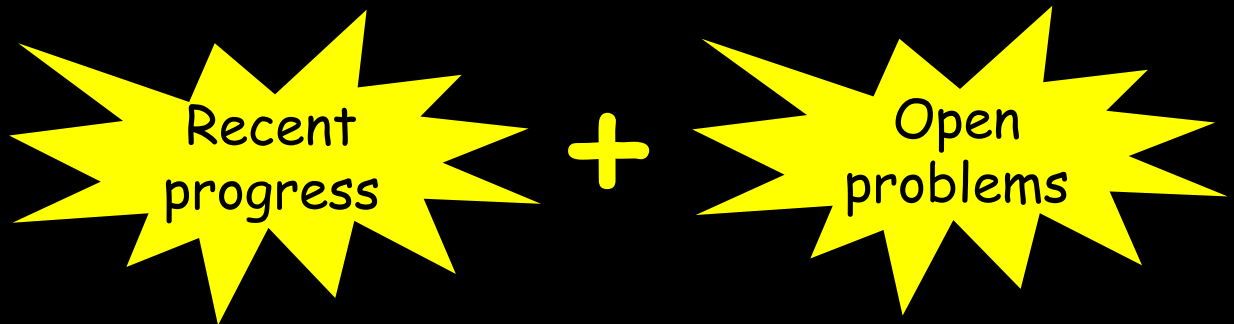
Outline: New Directions

I. Expanding Scheduling for M/G/1

II. Scheduling for multi-server M/G/k

III. Today's multi-server jobs

IV. Scheduling malleable jobs with flexible parallelizability



IV. Scheduling Malleable Jobs

Situations with limited number of "servers" …

e.g., multi-core machines

→ Need careful allocation of cores to jobs

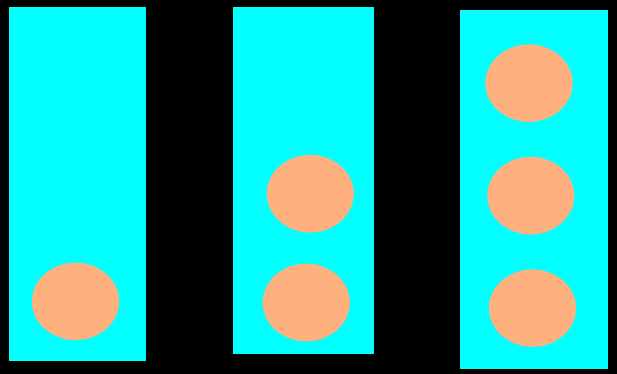
Many jobs are malleable!

- database queries
- machine learning jobs
- scientific computing jobs

can be parallelized across different numbers of cores

Malleable Jobs

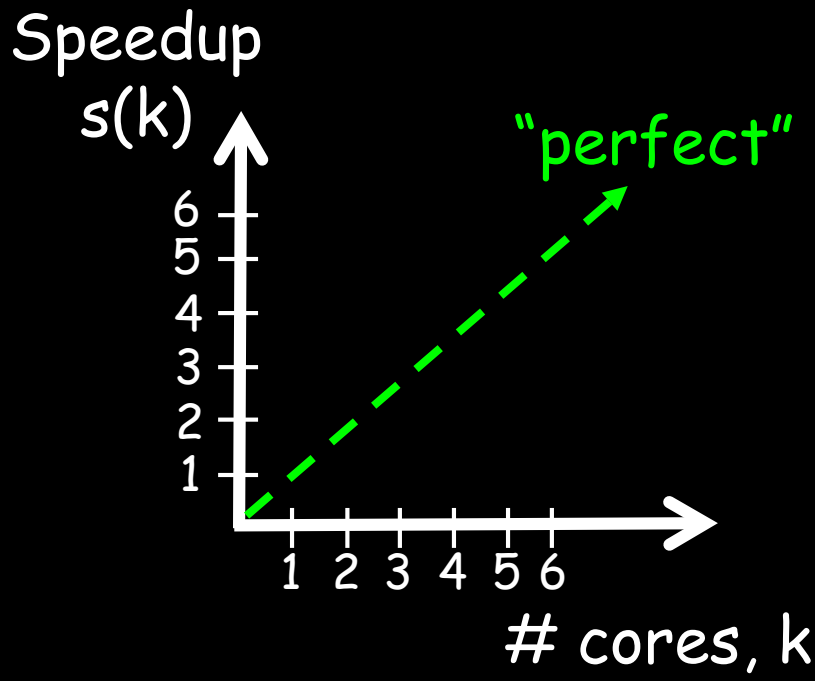
inherent job size x



runtime x

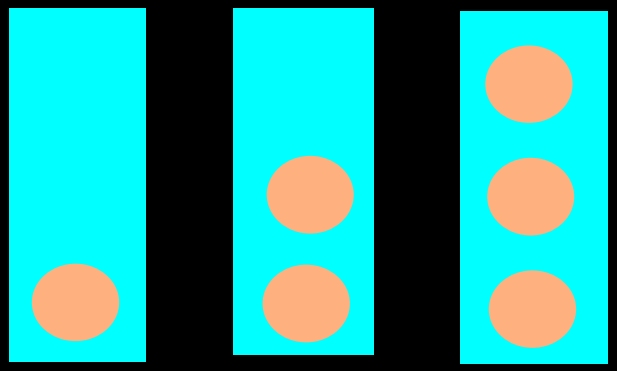
runtime $\frac{x}{2}$

runtime $\frac{x}{3}$



Malleable Jobs

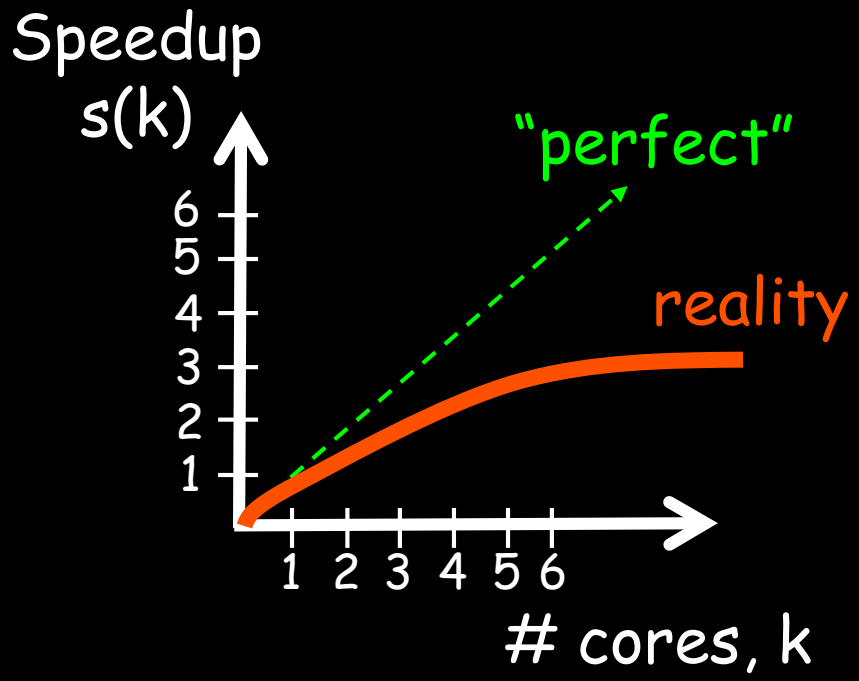
inherent job size x



runtime x

runtime $\frac{x}{s(2)}$

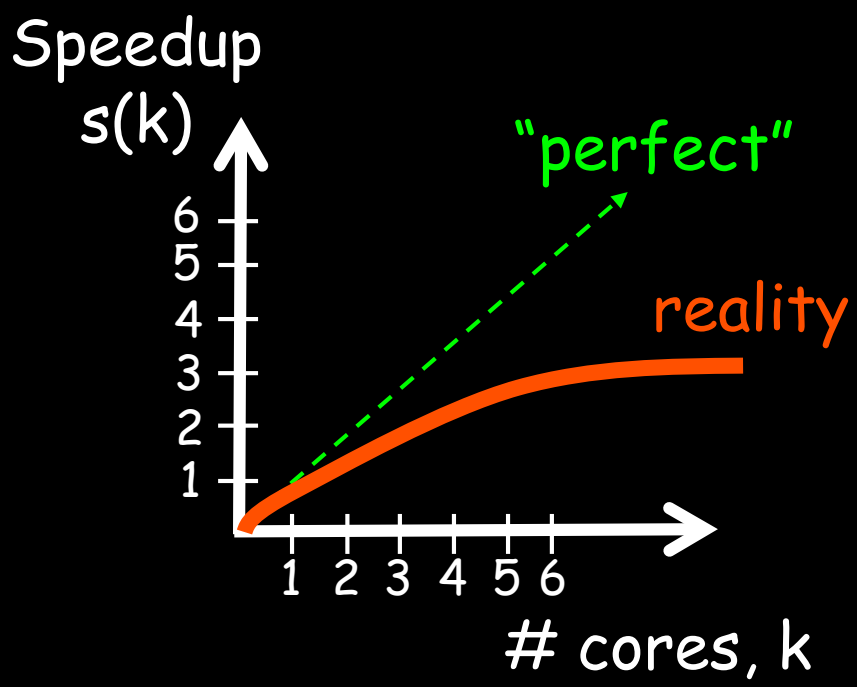
runtime $\frac{x}{s(3)}$



Key Question

Setting:
Jobs arrive over time.
Number of cores is limited.

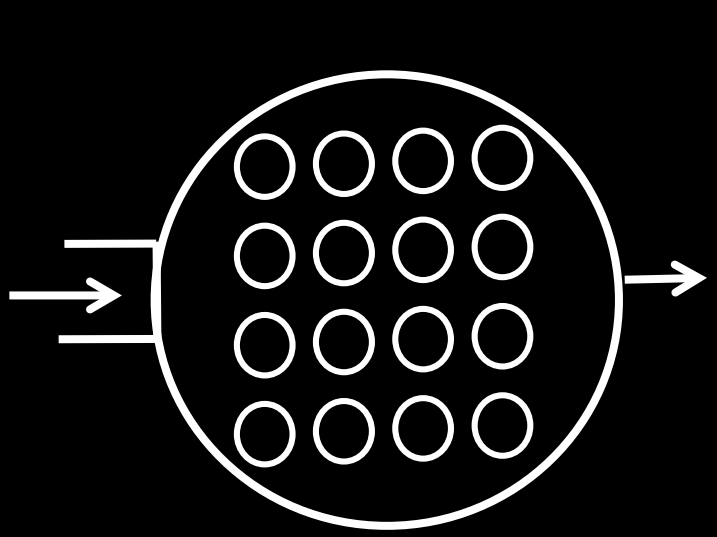
Question:
At every moment,
how many cores should we allocate to each job to minimize E[Response Time]?

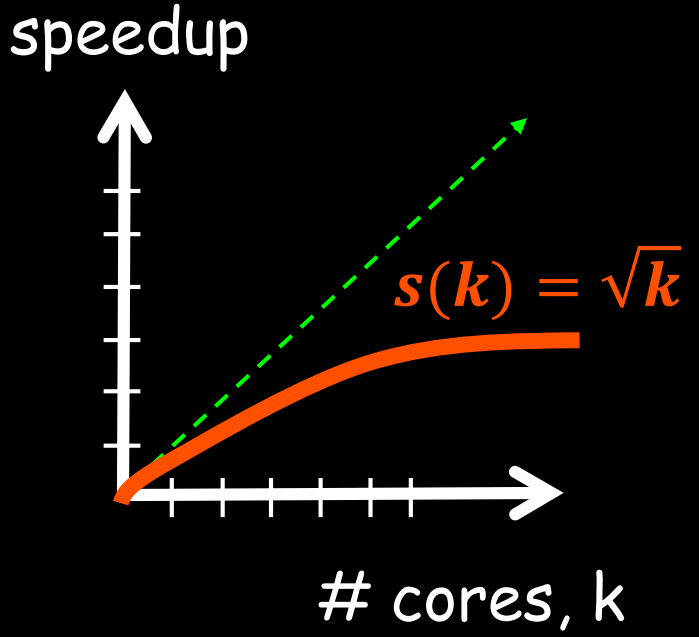


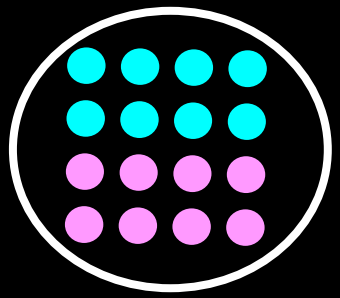
Intuition Building

$x_1 = 20$

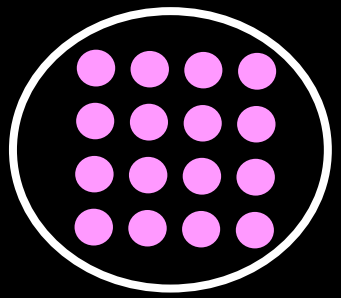
$x_2 = 10$



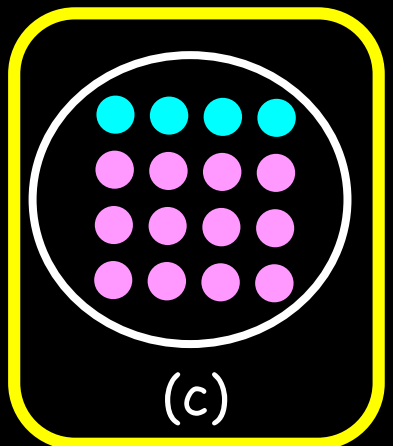




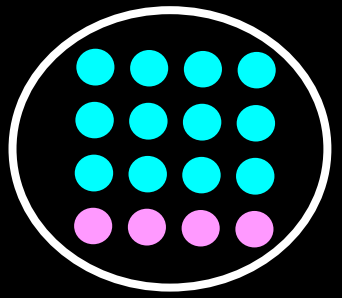
(a)



(b)

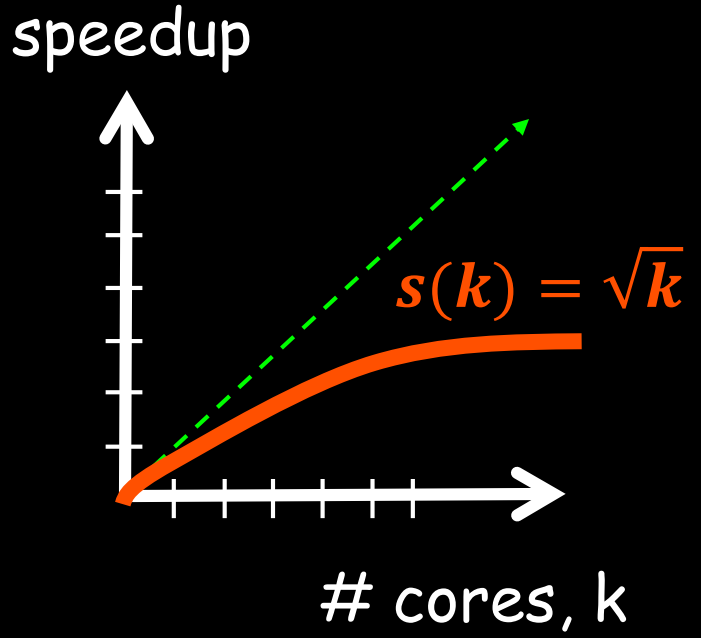


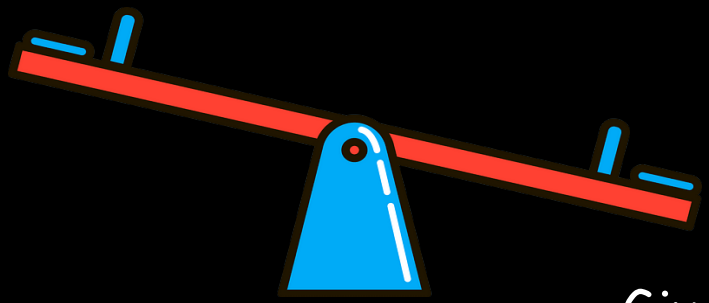
(c)



(d)

Intuition Building





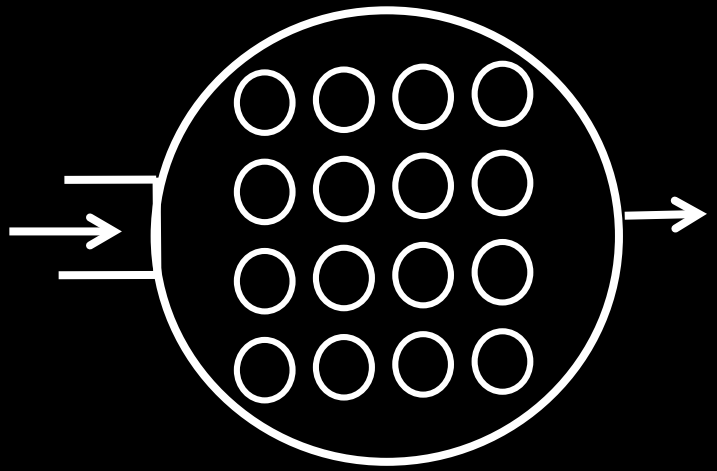
Give all cores to shortest job

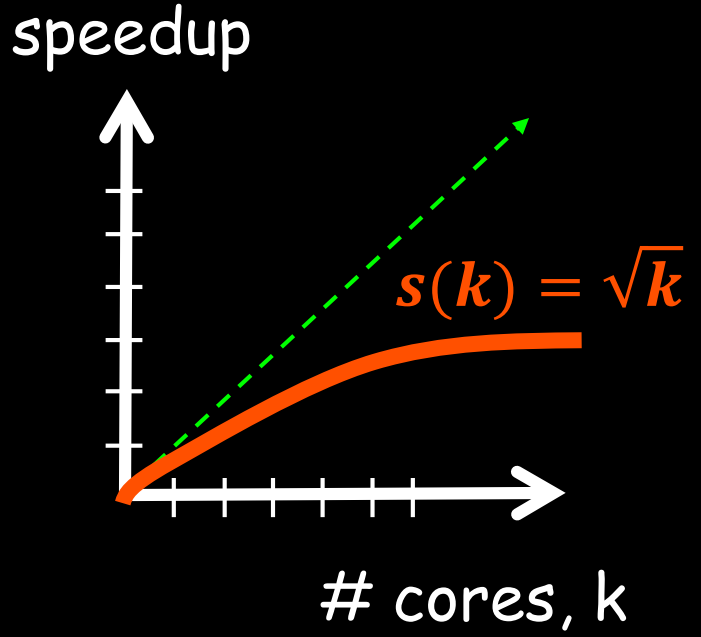
Give core to job that benefits most

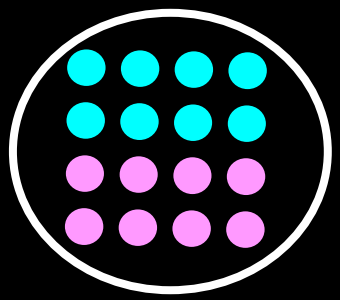
Intuition Building

$x_1 = 10$

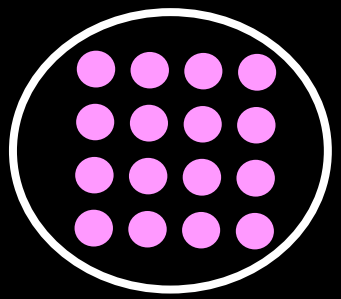
$x_2 = 10$



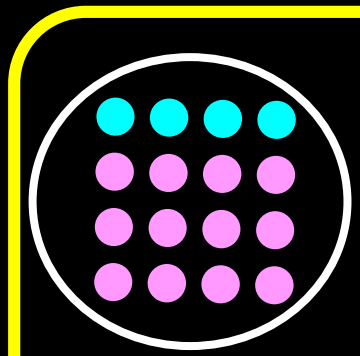




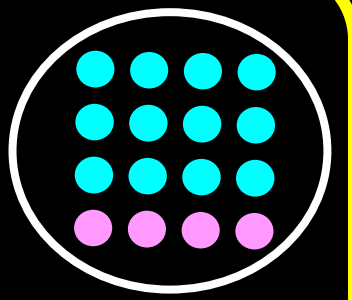
(a)



(b)

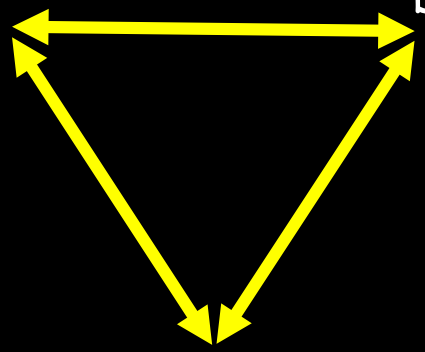


(c)



(d)

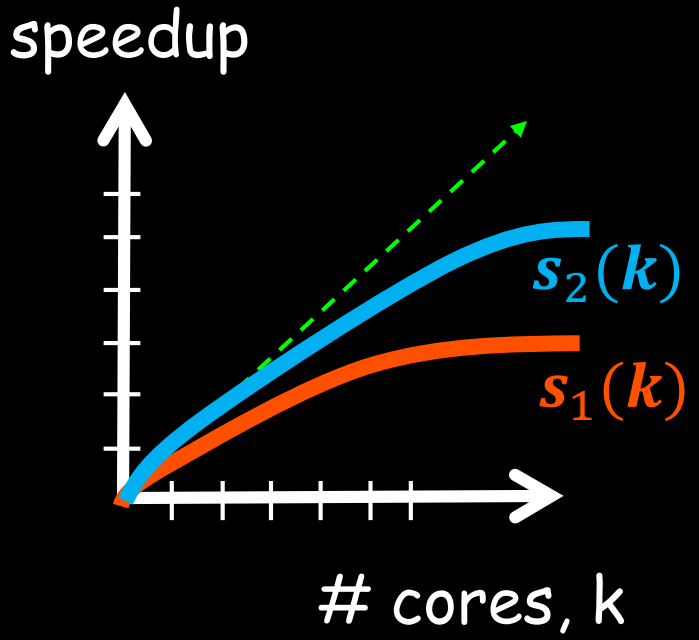
Intuition Building



Give all cores to shortest job

Give core to job that benefits most

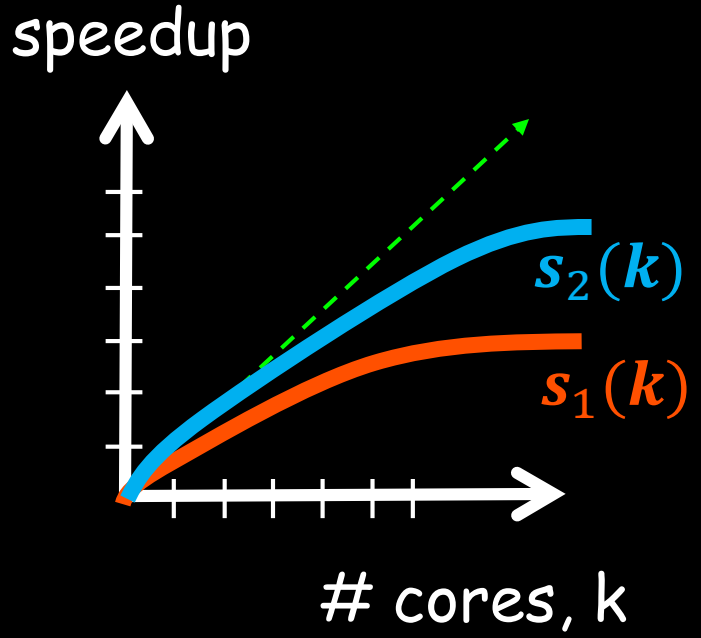
Save the blues … they're more flexible



Scheduling Malleable Jobs: Wrapup

Recent progress

See papers by Berg et al.
-- IFIP Performance '21
-- IFIP Performance '20
-- SPAA '20
-- Sigmetrics '18



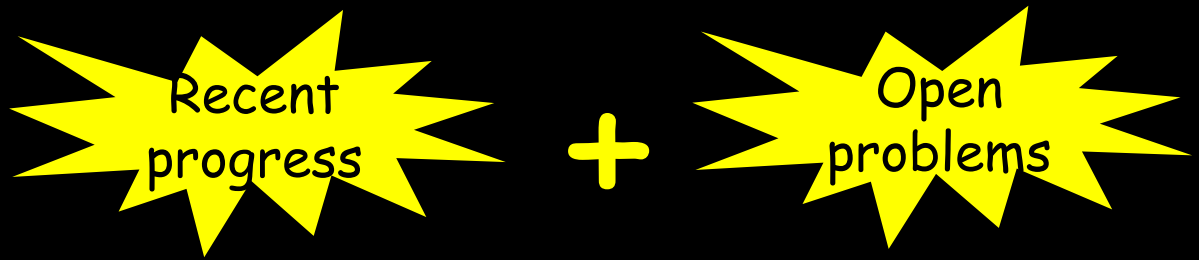
Open problems

	known sizes	unknown sizes
single speedup	✓✓?	✓??
multiple speedups	✓??	???

Open problems in every category

Conclusion

- ✓ I. Expanding Scheduling for M/G/1
- ✓ II. Scheduling for multi-server M/G/k
- ✓ III. Today's multi-server jobs
- ✓ IV. Scheduling malleable jobs with flexible parallelizability



Based on: ``Open problems in queueing theory inspired by datacenter computing." *Queueing Systems* , vol. 97, no. 1, 2021, pp. 3--37.