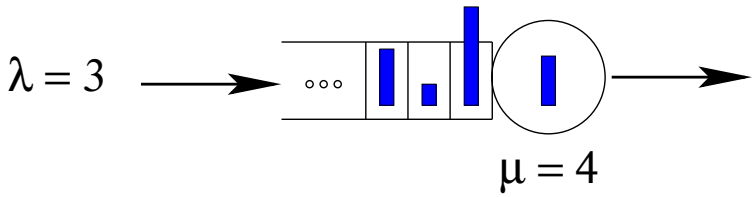
What is queueing theory?

Queueing theory is the theory behind what happens when you have lots of jobs arriving and scarce resources.

Examples of where queues occur:

_____	_____
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

Single-Server Queue Terminology



Avg. arrival rate, λ

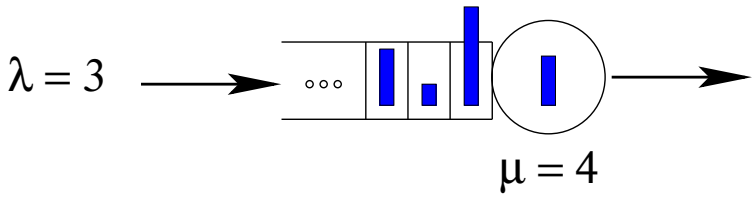
Job size, S

Avg. service rate, μ

Response time, T

Systems Speak	vs.	Queueing Speak

EXAMPLE 1: Scaling Capacity – Single-Server Queue



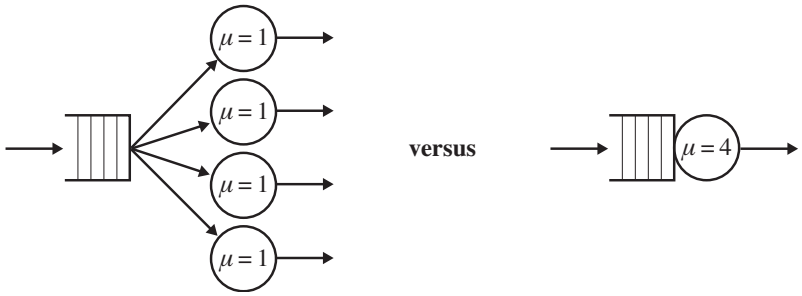
Question: Suppose $\lambda \longrightarrow 2\lambda$, but we want to keep $\mathbf{E}[T]$ the same?

(a) $\mu \longrightarrow 2\mu$?

(b) $\mu \longrightarrow < 2\mu$?

(c) $\mu \longrightarrow > 2\mu$?

EXAMPLE 2: Many slow machines vs. single fast one



Assume jobs are non-preemptible:

Question: Which is better for mean response time, $\mathbf{E}[T]$?

Question: Suppose we care about mean waiting time instead?

Assume jobs are preemptible:

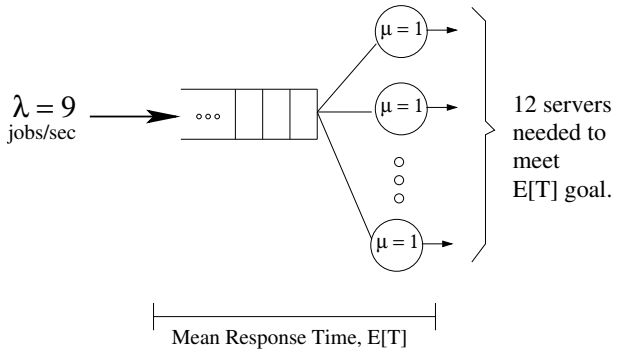
Question: Which is better for mean response time, $\mathbf{E}[T]$?

Question: What happens when price is also a factor?

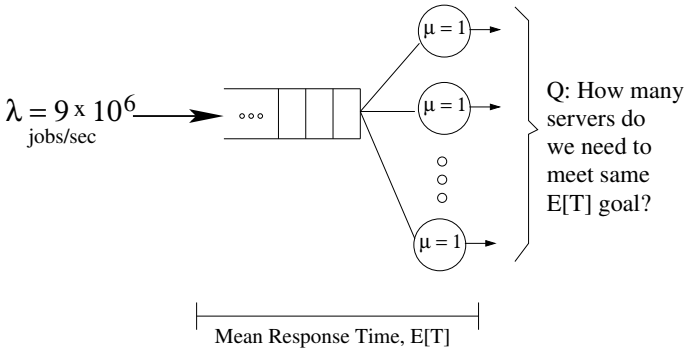
For real-world application, see: [Gandhi, Harchol-Balter, Das, Lefurgy, "Optimal power allocation in server farms" *ACM SIGMETRICS 2009*]

EXAMPLE 3: Capacity Provisioning

Your Lab:



At Meta:



Question: How many servers are needed to get same $\mathbf{E}[T]$?

(a) 12×10^6?

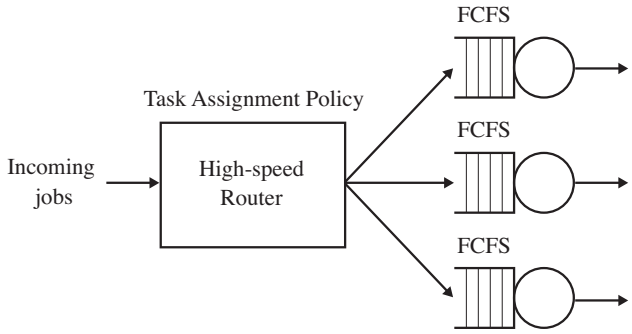
(b) $< 12 \times 10^6$?

(c) $> 12 \times 10^6$?

Story: Dynamic Power Management for Meta

Anshul Gandhi, Mor Harchol-Balter, Ram Raghunathan, and Mike Kozuch. "AutoScale: Dynamic, Robust Capacity Management for Multi-Tier Data Centers." *ACM Transactions on Computer Systems*, 2012.

EXAMPLE 4: Task Assignment



Model:

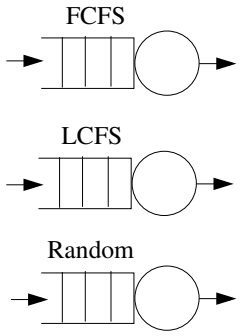
1. Hosts are identical
2. Job sizes highly variable
3. Jobs not preemptible

Question: How should we balance jobs between hosts to minimize $\mathbf{E}[T]$?

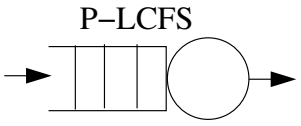
See: [Harchol-Balter, Scheller-Wolf, Young "Surprising Results on Task Assignment in Server Farms with High-Variability Workloads" *ACM SIGMETRICS 2009*]

EXAMPLE 5: Scheduling Policies

Question: Which of these non-preemptive scheduling policies is best for minimizing $\mathbf{E}[T]$?



Question: How about if we make LCFS preemptive?



So MANY more queueing problems!

Some of my current obsessions:

1. **Optimal core allocation among jobs with different speedup functions** ML training jobs are highly parallelizable, but different jobs have different speedup gain as a function of the number of GPUs that they are run on. Given a stream of jobs, and a limited number of GPUs, how should we allocate the GPUs across the different jobs to minimize overall mean response time?

2. **Scheduling jobs with different holding costs and sizes**. A job's holding cost is the price it costs us every day that the job is not done. Given a stream of jobs with different holding costs and sizes, how should we schedule jobs so as to minimize our total time-average holding cost? Now what if the holding cost changes over time?

3. **Pricing and queueing**. Some people are very willing to pay to avoid waiting in line. Others are less willing. How should we design priority levels and price these to maximize revenue?