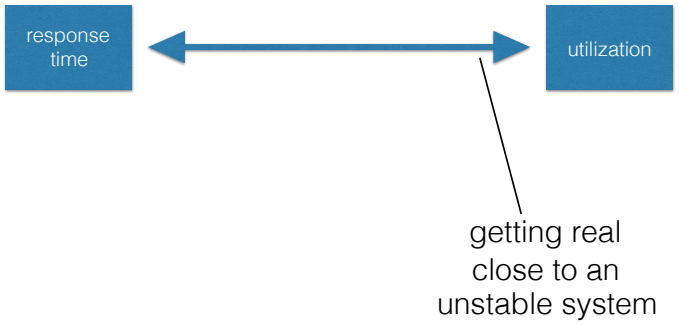
Operating systems

Scheduling 3: trends

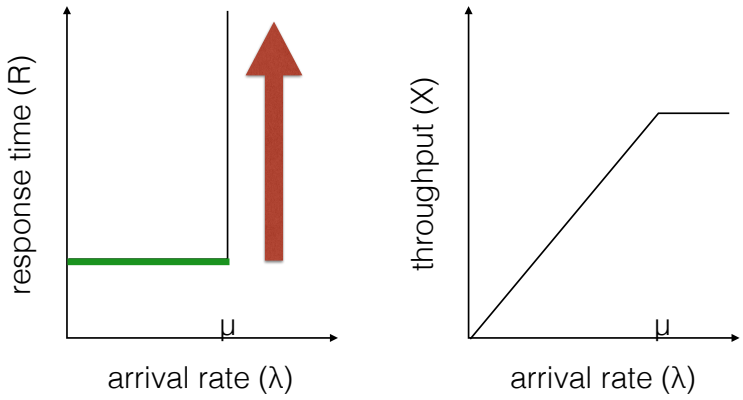
Today

- exponential distribution
- overhead management
- multi-processor scheduling
- real-time scheduling
- energy aware scheduling

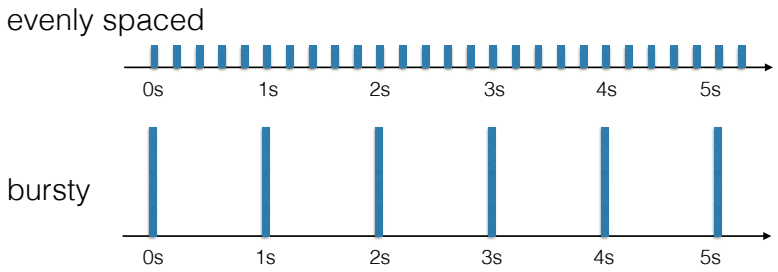
no free lunch



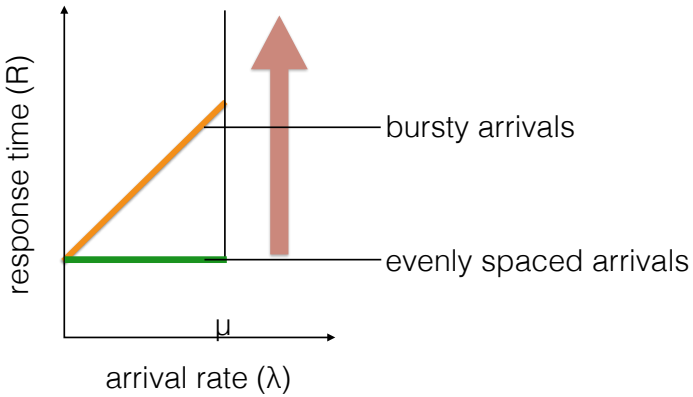
extreme scenario 1: evenly spaced arrivals



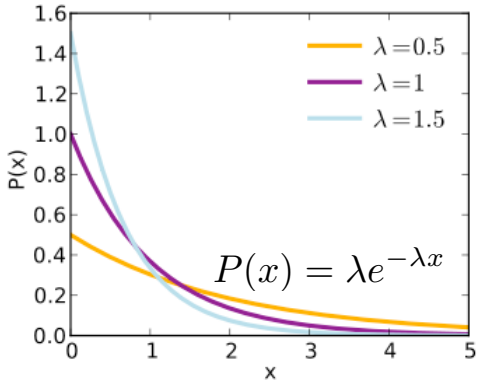
example: λ = 5 requests/s



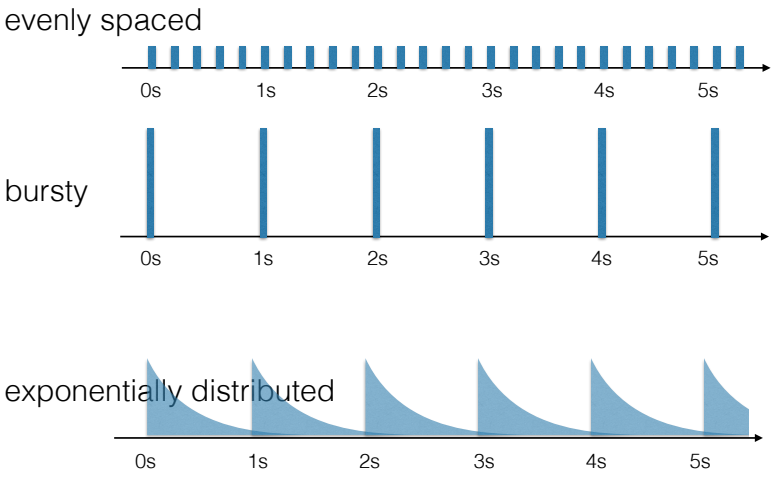
extreme scenario 2: bursty arrivals



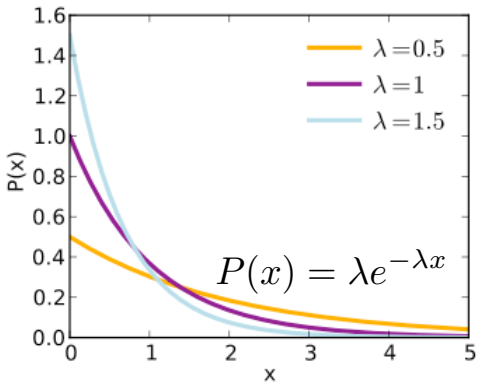
Exponential distribution



example: λ = 5 requests/s



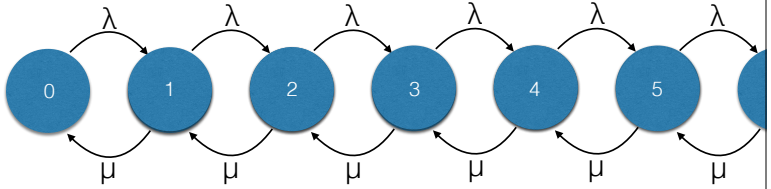
Exponential distribution



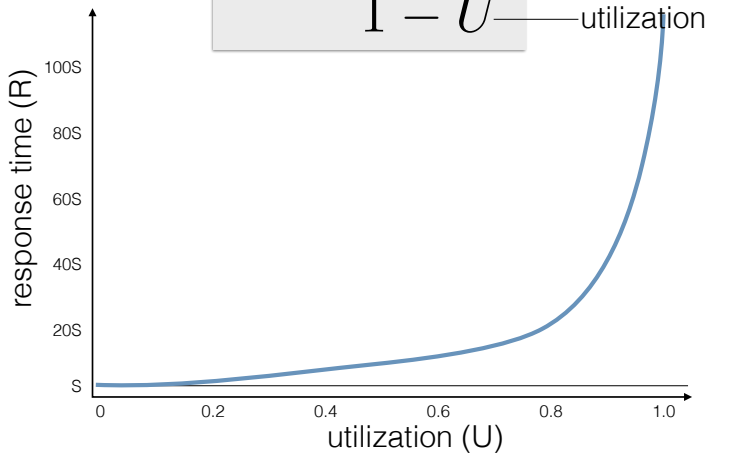
- memoryless
- light-tailed

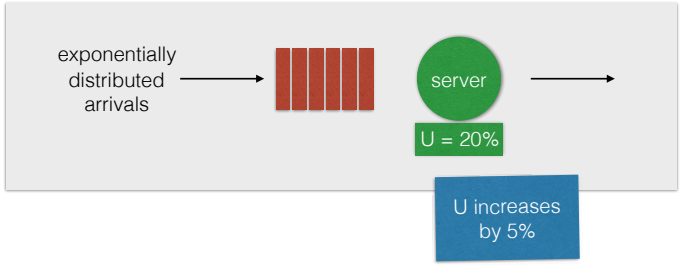
arrival time:
memory-full or
memoryless
probability distribution?

if the arrival process is a memoryless distribution



response time — $R = \frac{S}{1-U}$ — service time; utilization

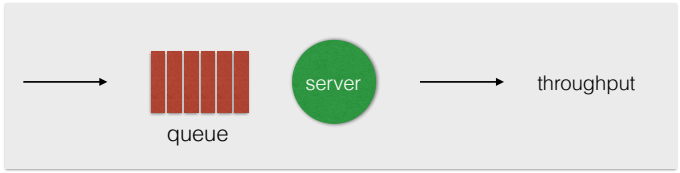




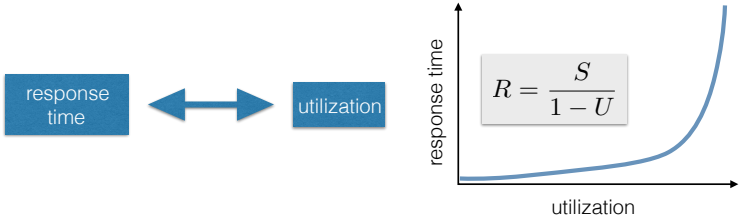
1. by how much does the response time increase?
2. how does that increase compare to the case when utilization goes from 90% to 95%?

$$R = \frac{S}{1-U}$$

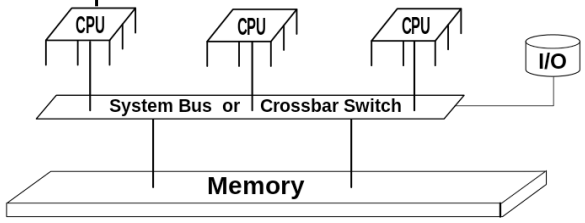
in summary…



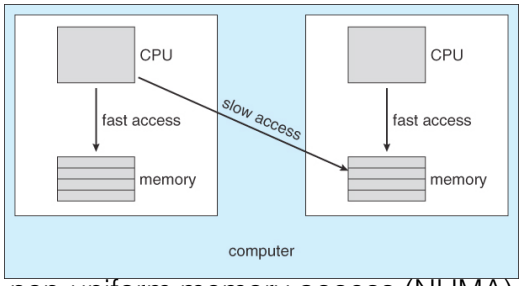
Little's law $N = XR$



Multiprocessor architecture



symmetric multiprocessing system



non-uniform memory access (NUMA)

Multi-processor scheduling

- symmetric multiprocessing (SMP): each processor is self-scheduling
- processor affinity: preference for a process to prefer to continue running on its current processor
- load balancing: the practice of keeping workloads evenly distributed

processor affinity ⟷ load balancing

How to manage overload?

reject service

routers
dropping packets

degrade service

streaming services
degrading resolution

design the resource manager with constant overhead

software-level demands
for responsiveness

↓

energy-aware scheduling

↑

hardware support
for power optimization

clock speed ⟷ energy usage

- processor design
- processor usage
- I/O device power

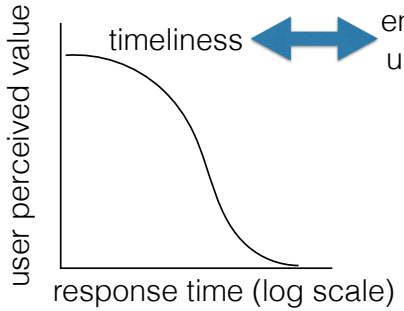
software-level demands
for responsiveness

↓

energy-aware scheduling

↑

hardware support
for power optimization



- UI: optimize response time according to human perception
- background tasks: balance tradeoff according to resource availability

software-level demands
for responsiveness

↓

energy-aware scheduling

↑

hardware support
for power optimization

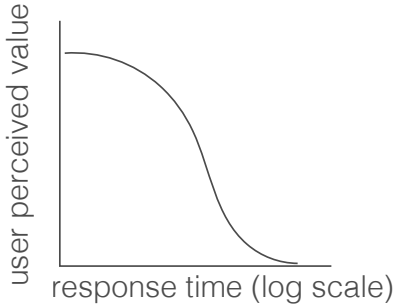
Real-time constraints

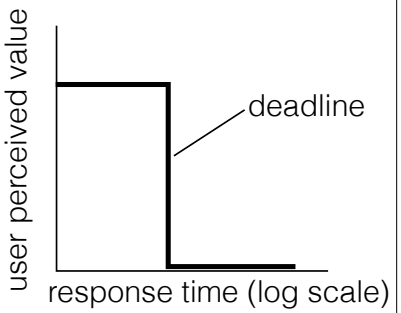
airplane's flight path

anti-lock brakes

movie

Real-time constraints





Approaches to real-time scheduling

Obvious

- give tasks with deadlines higher priority than other tasks
- problem: priority inversion
- how can you guarantee constraints are met?

Better

- over-provisioning
- earliest deadline first
- priority donation: when a higher priority task is waiting for a lock held by a lower priority task, donate its priority