SIMULATION AND MODELLING

Tony Field and **Giuliano Casale**

{ajf,gcasale}@doc.ic.ac.uk

Suggested Books

- Performance Modeling and Design of Computer Systems: Queueing Theory in Action
  Mor Harchol-Balter
  Cambridge University Press, 2013
- Probability, Markov Chains, Queues and Simulation
  William Stewart
  Princeton University Press, 2009
- Discrete-event System Simulation (5/e)
  J. Banks, J.S. Carson, B.L. Nelson and D.M. Nicol
  Prentice Hall International, 2010
- Simulation Modeling and Analysis
  A.M. Law and W.D. Kelton
  McGraw Hill, 2000

Syllabus

Part I (Tony)

1. Introduction
2. Operational laws
3. Poisson processes
4. Discrete-event simulation
5. Markov processes

Part II (Giuliano)

1. Markovian queues
2. Open queueing networks
3. Fork-join subsystems
4. Application examples
5. Parallel discrete-event simulation

Introduction

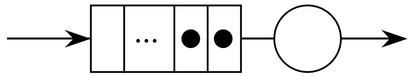
- This course is about using measurements and models to understand performance aspects of real-world systems
- We'll focus on computer systems but the principles are widely applicable
- Performance models capture the way jobs/customers/entities move around a system and compete for its resources
- It then becomes a tool for reasoning about the system's performance, e.g. in order to:
  - Understand the observable behaviour of an existing system
  - Guide changes, rewrites or upgrades to a system
  - Study new or imaginary systems

Applications

- There are many application areas, e.g.
  - Compute/web servers
  - Cloud computing/storage systems
  - Distributed systems
  - Mobile & sensor networks
  - Manufacturing
  - Transport & logistics
  - Healthcare provision
  - Military logistics & strategy
- We'll (try to!) balance theory and practice, so you'll understand how/why the techniques you'll be learning work

Motivation: Some Example Problems

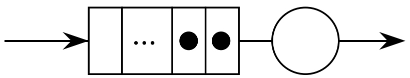
An in-memory TP system accepts and processes a stream of transactions, mediated through a (large) FIFO job queue:



- Transactions arrive "randomly" at some specified rate
- The service times are distributed exponentially, with some specified rate

Q: If both the arrival rate and service rate are doubled, what happens to the mean response time?

Consider an operating system scheduler where the job sizes (X) are highly variable

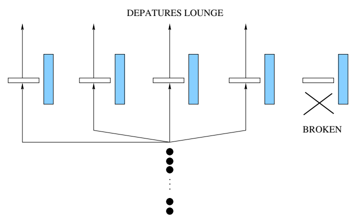


$$10 \quad \leq \quad C_X^2 = \frac{VAR(X)}{E(X)^2} \quad \leq \quad 50$$

Rank the following schedulers in order of mean processing time per job:

- Round Robin (preemptive)
- First-Come-First-Served
- Shortest Job First (non-preemptive)
- Shortest Remaining Processing Time (preemptive)

There are five security scanners between the check-in and departures area at Heathrow (T4); one of them is broken:

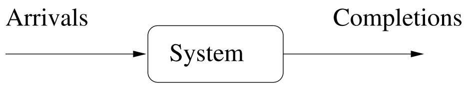


- Around 0.5 customers pass through the terminal each second and it takes just under 8 seconds on average to scan each passenger
- The average delay is about 30 minutes (1600 seconds)

Q: How long would it take on average if all 5 scanners were working?

Measurement Approaches and the Fundamental Laws

- Consider *any* “open” system with external arrivals and departures:



- Let’s assume we observe the system for time T whence
  - The number of arrivals is A
  - The number of completions is C
- From this we can immediately determine:
  - The *arrival rate* is $\lambda = A/T$
  - The *average inter-arrival time* is $\lambda^{-1} = T/A$
  - The *throughput* is $X = C/T$

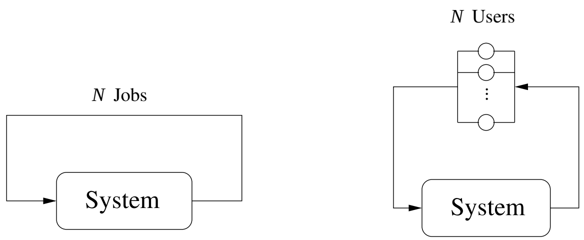
The Flow Balance Assumption

- It is typically the case that $\lambda = X$, i.e. either $A = C$ or $A - C$ is small in comparison to A and C
- In particular, systems that are in *equilibrium* (or *steady state*) must satisfy this property in the long term
- Note: If the flow balance assumption does not hold in the long term (as time $\to \infty$), then the system is fundamentally unstable
  - Motto: “What goes in must come out!”

Q: What does $A - C$ represent at any point in time?

Q: When might we have $\lambda > X$ and (notionally) a stable system?

- The system may also be “closed”, in which case a number of jobs, N, circulate around the system, e.g.



Batch system — Interactive system

- The key difference is that N is now *fixed* (the “multiprogramming level”, or “user population”)

Resources

- If the system includes a resource (e.g. a server, lock, network port...) whose total busy time is B then:
- The *utilisation* of the resource is $U = B/T$
- The average *service time* of each job at the resource is $S = B/C$
- Note: we often talk about the service *rate*, which is $\mu = 1/S = C/B$
- The *Fundamental Laws* (or *Operational Laws*) define important relationships between these various measures – REMEMBER THEM!

The Utilisation Law

- Since $U = B/T = C/T \times B/C$ we have:

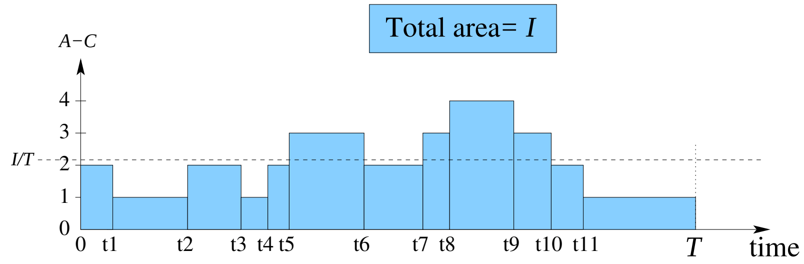
$$\boxed{U = X \times S}$$

- Often we work with service *rates* rather than *times*, in which case we have $U = X/\mu$
- Importantly, note that $U \leq 1$ so we require $\mu = 1/S \geq X$

Q: What if $\mu = 1/S = X = \lambda$? Can we ever have a utilisation that is exactly 1?

Little's Law

- Suppose we plot $(A - C)$ over the observation period $(0, T)$:



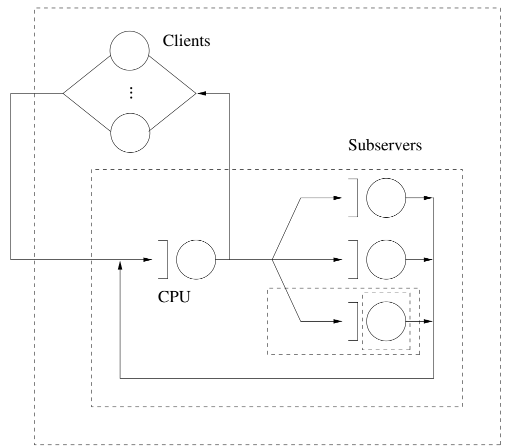
- $A - C$ represents the population of the system at a given time
- Let the accumulated area be I (in "request-seconds")

Little's Law continued

- The average number of jobs in the system is $N = I/T$
- The average time each job spends in the system (the average *response time*) is $R = I/C$
- Since $I/T = C/T \times I/C$ we have:

$$\boxed{N = X \times R}$$

- Little's Law can be applied to *any* system (and subsystem) in "equilibrium", e.g.



- For example: $N_Q = XR_Q$, $U = XS(!)$, where R_Q and N_Q are the mean queueing time and mean number of jobs queued waiting to be served

The Response Time Law

- The special case of a closed interactive system comprising N users in "think/compute" mode is a special case of Little's Law
- Let Z be the "think time", i.e. the time between completion of one request and the submission of the next
- N is the total population of users and X the request submission rate

- The average total time for each cycle is $R + Z$, so from

  Little's Law, $N = X(R + Z)$, or

$$\boxed{R = N/X - Z}$$

The Forced Flow Law

- Suppose the system in question comprises a number of resources that a job can "visit" during its time in the system
- Let C_k be the number of job completions at resource k
- The average number of visits each job makes to resource k is then $V_k = C_k/C$

- Rearranging: $C_k = V_k C$ so, dividing both sides by T,

  $C_k/T = V_k\ C/T$, i.e.

$$\boxed{X_k = V_k \times X}$$

where X_k is the throughput of resource k

The Service Demand/Bottleneck Laws

- If, on average, the number of times a job visits resource k is V_k and the average service time at the resource is S_k then the average service demand of each job at resource k is $D_k = V_k\ S_k$

- Multiplying the RHS by X_k/X_k, we get

  $D_k = V_k/X_k \times X_k S_k = U_k/X$; thus:

$$\boxed{D_k = U_k/X \quad \text{or} \quad U_k = X D_k}$$

- Recall: X is the overall *system* throughput

Bottlenecks and Throughput Bounds

- Since $U_k = D_k X$ and $U_k \leq 1$, we have $X \leq 1/D_k$ for all k; thus $X \leq \frac{1}{D_{max}}$ where $D_{max} = \max_k D_k$
- Under heavy load $U_{max} \approx 1$ and $X \approx 1/D_{max}$
- $1/D_{max}$ is the upper asymptotic bound on throughput under heavy load
- The resource with the highest demand (D_{max}) is called the bottleneck resource
- In an open system the arrival rate and throughput are the same ($\lambda = X$), which means that we require $\lambda \leq \frac{1}{D_{max}}$ for the system to be stable

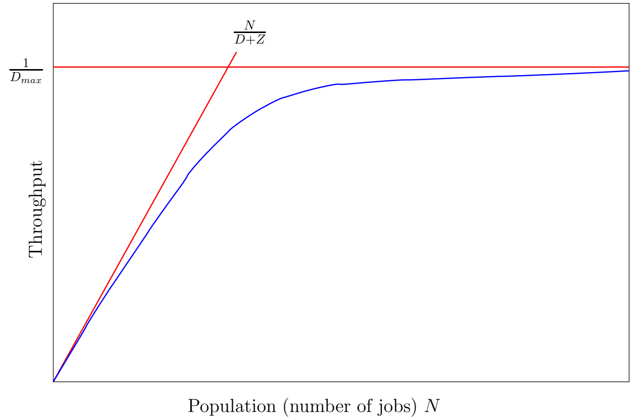
- Under light load no job ever has to queue and the time each job spends at resource k is just D_k and, minimally, $R = \sum_k D_k$
- Thus, for a closed system, from the response time law, $X = N/(R+Z) \leq N/(D+Z)$ where $D = \sum_k D_k$.
- $N/(D+Z)$ is the upper asymptotic bound on throughput under light load

- In general, we have

$$\boxed{X \leq \min(\frac{1}{D_{max}}, \frac{N}{D+Z})}$$

- For an open system $Z = 0$, but recall that N isn't fixed: the bound still applies, but it's not as tight as for a closed system

- A typical throughput plot looks like this:



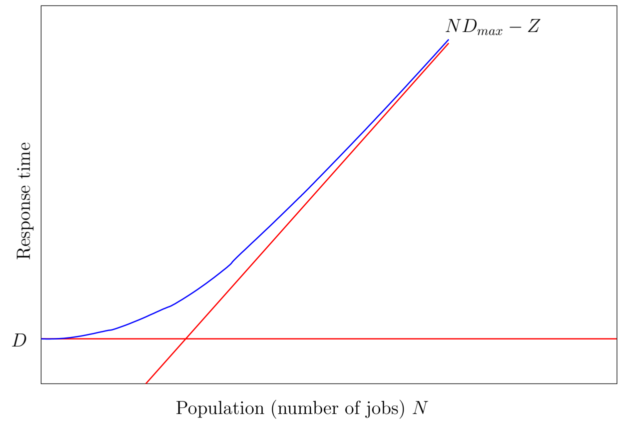
Response time Bounds

- Similar to throughput bounds, we can also derive response time bounds...
- Under high load, since $X \leq 1/D_{max}$, we have $R = N/X - Z \geq ND_{max} - Z$
- Under low load every job experiences the average service demand at each node *without queueing*, i.e. total demand D
- The response time can never be lower than this, so $R \geq D$

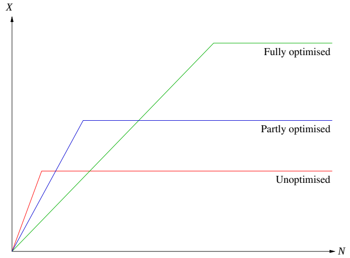
- In general, therefore:

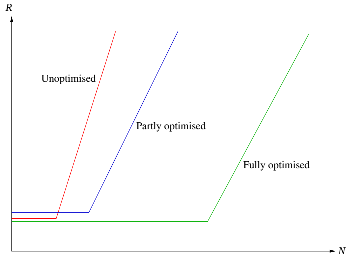
$$\boxed{R \geq \max(D, ND_{max} - Z)}$$

- A typical response time plot looks like this:



- Performance optimisation involves identifying and fixing bottlenecks:





- Operational laws define how measures are related; they enable us to "fill in gaps"
- Bounds plots tell us how well we can do in the limit, given the service demands for each resource
- However, neither enable us to *predict* performance measures directly
- **Example**: given D_i, U_i, S_i, X_i... compute R_i or N_i
- To compute such *derived* measures we need to build a model of a system, making *assumptions* about, for example:
  - The distribution of inter-arrival times, job service times etc.
  - The way jobs are routed among a collection of resources
  - The way resources are claimed and released by jobs
  - The job scheduling strategy (queueing discipline)
- The rest of this course is about how to build and analyse such models...