Introduction to queueing theory

Andrea Marin

a.a. 2024/2025

Università Ca' Foscari Venezia
MSc in Computer Science

Software performance and scalability

Basic notions

Rules for all queues

The M/M/1 queue

The M/M/m queue

The M/M/∞ queue

The M/G/1 queue

The M/G/1/PS queue

The G/G/1 system

Size based scheduling: SRPT

Age based scheduling

Basic notions

Some credits. . .

Agner Krarup Erlang (1878-1929)

Danish mathematician founder of **queueing** theory

Some credits. . .

Leonard Kleinrock

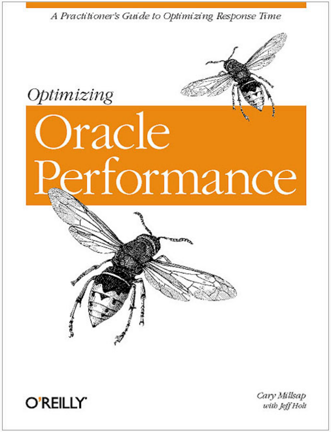
Basically, what I did for my PhD research in 1961-1962 was to establish a mathematical theory of packet networks . . .

We will use models. . .

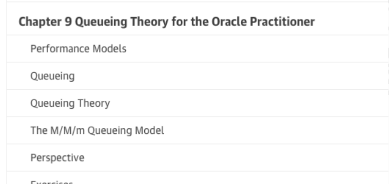
- All models are wrong, but some are useful [Box]
- Trying to avoid the **spherical cows**..

We will use Math. . .

- Mathematics is the most beautiful and most powerful creation of the human spirit [Banach]



Cary Millsap: Created the Optimal Flexible Architecture (OFA) configuration standard, which is implemented for every Oracle Database installation in the world. Founded and led the System Performance Group (15.8 million dollars annual revenue)



Chapter 9 Queueing Theory for the Oracle Practitioner

Performance Models

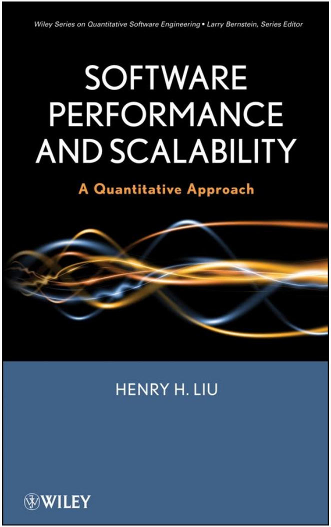
Queueing

Queueing Theory

The M/M/m Queueing Model

Perspective

Another (last) example



Wiley Series on Quantitative Software Engineering • Larry Bernstein, Series Editor

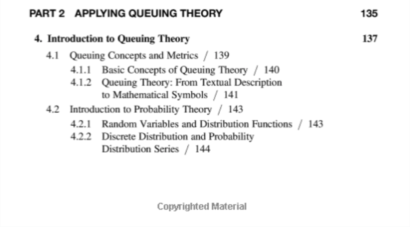
SOFTWARE PERFORMANCE AND SCALABILITY

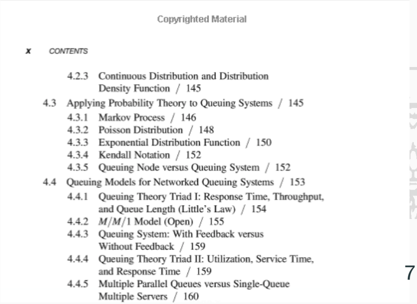
A Quantitative Approach

HENRY H. LIU

WILEY

Henry H. Liu: Lead Performance Engineer for iOS Systems at Apple





A theory for studying the performance

- *If the facts don't fit the theory, change the facts!* A. Einstein
- In computer systems the jobs can:
    - compete for resources
    - use the resource
    - synchronise
    - ...
- Queueing theory helps in predicting how long a job will spend in each of its phases
- Widely applied in practice

Queueing notations: arrival process

- In a queueing system we serve *customers* or *jobs*
    - A HTTP request for Apache is a customer or a job
- If the customers arrive at time $t_1, t_2, \ldots, t_j$, then the times $T_j = t_j - t_{j-1}$ are called inter-arrival times
- We usually assume that T_j are independent random variables and identically distributed. These form the arrival process
- The most common arrival process is the Poisson arrival process
    - The inter-arrival times are i.i.d. exponential random variables

Queueing notation: service time distribution

- How long does a job need to be served?
- In general we do not know the size of job (the amount of time it will use the processor)
- We model the job service times as independent random variables. This is the service time distribution
  - When all the jobs are statistically identical we say that the system has a single class of jobs
  - If we can cluster the jobs into classes in which the elements are statistically identical then we say that the system is multi-class

Queueing notation: number of servers and system capacity

- We can have one or more servers that work on the jobs
- We consider the servers to be identical
  - Notice that the variability stays in the job size not in the server speed
- The system capacity is the maximum number of jobs that the system can contain
  - What happens when there is an arrival at a saturated system?

Queueing notation: population size

- The population size is the total number of potential jobs that can access the system
- The population size and the system capacity are different notions:
  - the system capacity is the maximum number of jobs that can stay in the system
  - the population size is the maximum number of jobs that are willing to enter in the system

Queueing notation: service discipline

- The service discipline is the order in which the jobs are served
- Most used disciplines:
  - First Come First Served (FCFS)
  - Last Come First Served (LCFS)
  - Last Come First Served with preemption (LCFSP)
  - Last Come First Served with preemption and resume (LCFSPR)
  - Processor Sharing (PS)
    - Approximation of the round robin discipline
  - Delay center or Infinite Servers (IS)
  - Shortest Remaining Processing Time (SRPT)

Work conserving disciplines

Definition (Work conserving disciplines)

A queueing discipline is work-conserving if:

- it never leaves idle a server that is allowed to work;
- it never wastes the work done on a job.

Kendall notation

- Fast way to describe a queueing system
- A/S/m/B/K/SD
  - A: inter-arrival distribution
  - S: service distribution
  - m: number of servers
  - B: system capacity
  - K: population size
  - SD: service discipline
- Abbreviations for A and S
  - M: Exponential
  - E_k: Erlang with k phases of service
  - H_k: Hyper-exponential with k phases of service
  - COX: Coxian
  - D: Deterministic
  - PH: Phase-type
  - G: General

Example

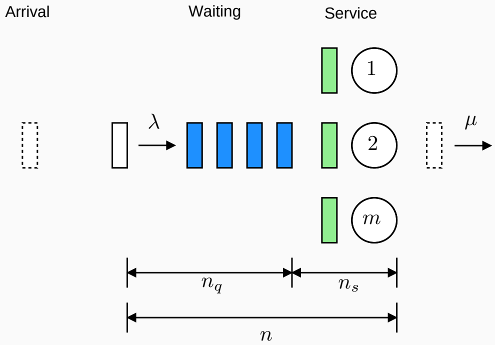
- What can we say about the queue $M/M/3/20/1500/FCFS$?
    - Poisson arrival process
    - Exponential service time distribution
    - 3 servers
    - Maximum 20 jobs in the system
    - population size: 1500
    - FCFS queueing discipline

Abbreviations

- If we do not specify the system capacity, we mean ∞
- If we do not specify the population size, we mean ∞
- If we do dot specify the scheduling discipline, we mean FCFS
- Examples:
  - $M/M/m$: Poisson arrivals, exponential service time, m servers
  - $G/G/1$: General inter-arrival distribution, general service time distribution, single server

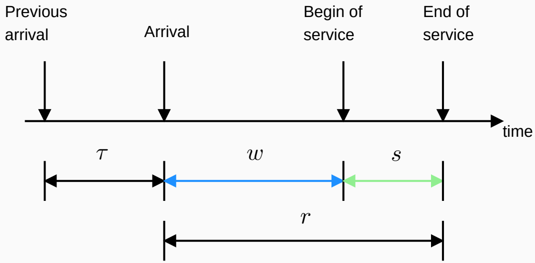
Rules for all queues

Key-variables in the state



- λ: arrival rate, μ: service rate
- n_q: number of waiting customers, n_s: number of customers in service, $n = n_s + n_q$: number of customers in the system (or *queue length*)

Key-variables in the time



- τ inter-arrival time per job, $\lambda = E[\tau]^{-1}$
- s service time per job, $\mu = E[s]^{-1}$
- w waiting time per job
- r response time per job: $r = w + s$

Random variables

- All the variables that we have introduced are random variables with the exception of λ and μ
- Since $n = n_q + n_s$ we have $E[n] = E[n_q] + E[n_s]$
- Since $r = w + s$ we have $E[r] = E[w] + E[s]$
- Our goal is to find a relation between the system's parameters (service discipline, λ, μ, $\dots$) and the distributions or the moments of the other random variables

Stability of a queueing system

- The system is unstable if the number of jobs grows continuously and becomes infinite
- Otherwise we say that the system is stable
- For infinite population and infinite buffer we must require that the maximum throughput of the system must be lower that the arrival rate:

$$\lambda < m\mu$$

Little's law

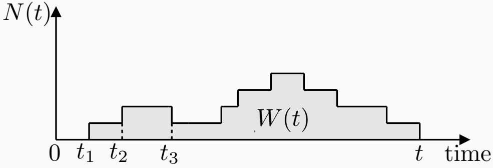
- $A(t)$ number of arrivals in $[0, t]$
- $C(t)$ number of departures in $[0, t]$
- $A(t) - C(t)$ is the number of customers in the system at t
- $W(t) = \int_0^t (A(\tau) - C(\tau)) d\tau$ is the cumulative work in $[0, t]$
- $\overline{N}(t) = W(t)/t$, $\overline{R}(t) = W(t)/C(t)$

Theorem (Little's law)

$$\overline{N}(t) = X(t)\overline{R}(t)$$

Sketch of the proof/1

- Consider a realization of process that begins and finishes with an empty system, call this interval $[0, t]$



- $N(\tau) = A(\tau) - C(\tau)$ with $0 \leq \tau \leq t$
- $W(t) = \int_0^t N(\tau) d\tau$
- The average number of jobs in the system is: $\overline{N}(t) = W(t)/t$

Sktetch o the proof/2

- In the cycle from empty to empty, the total time spent by all the customers that arrived at the system is $W(t)$
- Since we have $C(t)$ completions (and arrivals) the expected time in the system is $\overline{R}(t) = W(t)/C(t)$
- Hence:

$$\overline{N}(t) = \frac{W(t)}{t} == \frac{W(t)}{t}\frac{C(t)}{C(t)} = \overline{R}(t)X(t)$$

since $X(t) = C(t)/t$.

- Attention: the theorem holds even if $N(0) > 0$ or $N(t) > 0$ but the proof is more complicated!

Little's theorem

Assume the following limits exist:

- $\lambda = \lim_{t\to\infty} A(t)/t$ traffic intensity
- $\overline{N} = \lim_{t\to\infty} N(t)$ (the system is stable)
- $\overline{R} = \lim_{t\to\infty} \sum_{i=1}^{k} r_i$
    - k is the number of jobs served in $[0, t]$ and r_i the response time of the i-th service

Theorem

$$\overline{N} = \lambda \overline{R}$$

A notice on the traffic intensity and the throughput

Under the following assumptions

- No job loss
- Stability of the system (the population of the queue does not grow indefinitely)

We have:

$$\lambda = X$$

i.e., the traffic intensity is identical to the thoughtput

- We can rewrite Little's theorem as:

$$\overline{N} = X\overline{R}$$

Utilisation in single server queues

Theorem (Utilisation of the single server queue)

In a single server queue with constant arrival rate and service rate, infinite capacity and work-conserving queueing discipline, the utilisation of the queue is:

$$U = \frac{\lambda}{\mu}$$

where λ is the intensity of the arrival process and μ^{-1} is the expected job size or equivalently the expected service time.

Throughput and Utilisation in single server queues

Theorem

In a single server queue with constant arrival rate and service rate and work-conserving queueing discipline, we have the following relation between the throughput and the utilisation:

$$X = \mu \cdot U$$

PASTA property

- Poisson Arrivals see Time Averages

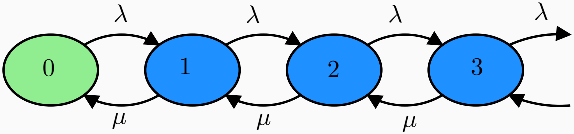
Theorem (PASTA)

In a queueing system with a Poisson arrival process, the distribution seen by a job immediately before its arrival is the same as the random observer's one.

The M/M/1 queue

The M/M/1 queueing system

- Exponential independent inter-arrival times (Poisson arrivals)
- Exponential independent service times
- Single server, infinite buffer, FCFS discipline



Properties of the CTMC underlying the M/M/1 queue

- Irreducible state space
- Infinite state space
- We do not know if the chain is ergodic
  - Pick an arbitrary reference state (usually 0)
  - Solve the GBEs for all the other states
  - Try to compute $\pi(0)$ by normalising the probability distribution
    - If you obtain 0 then the chain is not ergodic
    - If you obtain a value between $0 < \pi(0) < 1$ then the chain is ergodic

The GBEs of the M/M/1 queue

$$\begin{cases} \pi(0)\lambda = \pi(1)\mu \\ \pi(n)(\lambda+\mu) = \pi(n-1)\lambda + \pi(n+1)\mu \quad n > 0 \end{cases}$$

- By fixing $\pi(0)$ we can derive $\pi(1) = \pi(0)\lambda/\mu$
- From the equation associated with $\pi(1)$ we derive $\pi(2) = \pi(0)(\lambda/\mu)^2$
- In general we have:

$$\pi(n) = \pi(0)\left(\frac{\lambda}{\mu}\right)^n \quad n \geq 0$$

The load factor

- The load factor ρ of the queue is the ratio λ/μ
- It represents the ratio between the arrival rate and the service rate
- We already observed that the queue is stable if $\rho < 1$, i.e., $\lambda < \mu$
- We can rewrite the steady-state probabilities (if they exist) as:

$$\pi(n) = \pi(0)\rho^n$$

Computation of $\pi(0)$

- We must impose that $\sum_{n=0}^{\infty} \pi(n) = 1$, i.e.:

$$\sum_{n=0}^{\infty} \pi(0)\rho^n = \pi(0) \sum_{n=0}^{\infty} \rho^n = \pi(0) \frac{1}{1-\rho}$$

where the geometric series converge if and only if $\rho < 1$ (same condition of the stability!)

- Last step:

$$\pi(0)\frac{1}{1-\rho} = 1 \quad \Longrightarrow \quad \pi(0) = (1-\rho)$$

Conclusion on the steady-state

- The CTMC is ergodic when the queue is stable, i.e., $\rho < 1$
- In stability the steady-state distribution is:

$$\pi(n) = (1 - \rho)\rho^{n}$$

- The steady-state probability of observing n jobs in the queue has a geometric distribution with ratio ρ

Expected number of customers in the system

- We have that $\overline{N} = E[n] = \sum_{n=0}^{\infty} n\pi(n) = \sum_{n=0}^{\infty} n\pi(0)\rho^n$:

$$\overline{N} = \sum_{i=0}^{\infty} i\pi(i) = (1-\rho)\sum_{i=1}^{\infty} i\rho^i = (1-\rho)\rho\sum_{i=0}^{\infty}(i+1)\rho^i$$

$$= (1-\rho)\rho\sum_{i=0}^{\infty}\frac{\partial\rho^{i+1}}{\partial\rho} = (1-\rho)\rho\frac{\partial}{\partial\rho}\sum_{i=0}^{\infty}\rho^{i+1} = (1-\rho)\rho\frac{\partial}{\partial\rho}\frac{\rho}{1-\rho}$$

$$= (1-\rho)\rho\frac{1-\rho+\rho}{(1-\rho)^2} = \frac{\rho}{1-\rho}\,.$$

Utilisation and expected number of customers in servers

- The steady-state probability of finding the server busy is $U = 1 - \pi(0) = \rho$
- The expected number of customer is the service room is:

$$E[n_s] = \sum_{i=1}^{\infty} 1 \cdot \pi(i) = 1 - \pi(0) = \rho$$

- Recall that $E[n] = E[n_q] + E[n_s]$ and hence:

$$E[n_q] = \frac{\rho}{1-\rho} - \rho = \frac{\rho^2}{1-\rho}$$

Expected response time

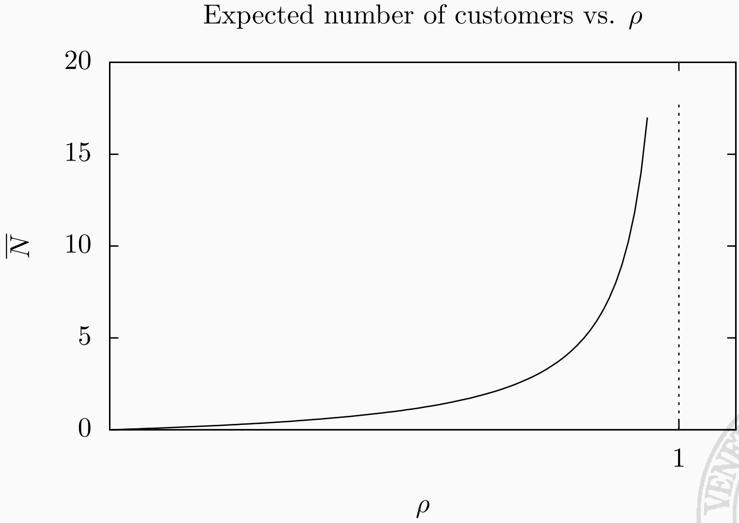
- In stability, the throughput $X = \lambda$
- We can compute the expected response time by Little's theorem:

$$\overline{R} = E[r] = \frac{\overline{N}}{X} = \frac{1}{\mu - \lambda}\,.$$

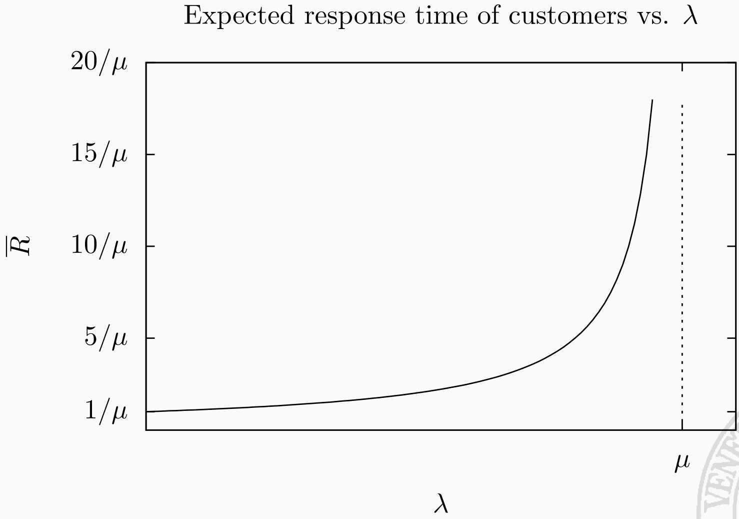
- Since we know that $E[s] = 1/\mu$:

$$E[w] = E[r] - E[s] = \frac{1}{\mu - \lambda} - \frac{1}{\mu} = \frac{\lambda}{\mu(\mu - \lambda)}$$

Expected number of customers vs. ρ



Expected response time vs. λ



Example

- Consider an $M/M/1$ queue with arrival rate $\lambda = 2.5$ jobs/second
- Compute the service rate to have an expected response time lower than 4 seconds

$$\overline{R} = \frac{1}{\mu - \lambda}$$

- So we have:

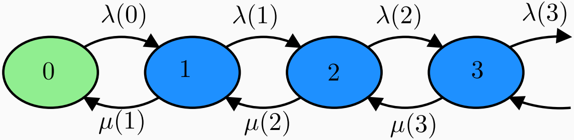
$$\mu = \frac{1 + \overline{R}\lambda}{\overline{R}} = \frac{1 + 4 \cdot 2.5}{4} = 2.75 \text{ jobs/second}$$

Exercise

- Study the $M/M/1/c$ queueing system with loss (jobs arriving at the saturated queue are lost)
- Give the stability condition
- Derive the steady-state probabilities
- Derive the throughput in stability
- Derive the expected number of jobs in the queue
- What can we say about the expected response time?

The M/M/m queue

A generalisation of the M/M/1 queue: Birth and death processes



If the chain is ergodic:

$$\pi(n) = \pi(0)\frac{\prod_{i=0}^{n-1}\lambda(i)}{\prod_{i=1}^{n}\mu(i)}$$

The M/M/m queue

- We have m independent servers, Poisson arrival, exponential service time
- We use the generalised formula with $\lambda(n) = \lambda$ and:

$$\mu(n) = \begin{cases} n\mu & \text{if } n \leq m \\ m\mu & \text{otherwise} \end{cases}$$

- Steady-state distribution:

$$\pi(n) = \begin{cases} \pi(0)\frac{1}{n!}\left(\frac{\lambda}{\mu}\right)^n & \text{if } n \leq m \\ \pi(0)\frac{1}{m!}\left(\frac{\lambda}{\mu}\right)^n \frac{1}{m^{n-m}} & \text{if } n > m \end{cases}$$

Expression for $\pi(0)$

- The queue is stable if and only if $\lambda < m\mu$
- This is also the condition for the ergodicity

$$\pi(0) = \left(\sum_{j=0}^{m-1} \left(\frac{\lambda}{\mu}\right)^j \frac{1}{j!} + \left(\frac{\lambda}{\mu}\right)^m \frac{1}{m!} \frac{1}{1 - \frac{\lambda}{m\mu}} \right)^{-1}$$

Probability of finding all the servers busy

- Let $\rho = \lambda/(m\mu)$
- Let $C(m, \lambda/\mu)$ be the probability of finding all the servers busy
- Then, we have the Erlang-C formula:

$$C\left(m, \frac{\lambda}{\mu}\right) = \frac{1}{1 + (1-\rho)\left(\frac{m!}{(m\rho)^m}\right)\sum_{k=0}^{m-1}\frac{(m\rho)^k}{k!}}$$

- Notice that if $m = 1$ we have the M/M/1 queue and:

$$C\left(1, \frac{\lambda}{\mu}\right) = \frac{1}{1 + (1-\rho)\frac{1}{\rho}} = \rho$$

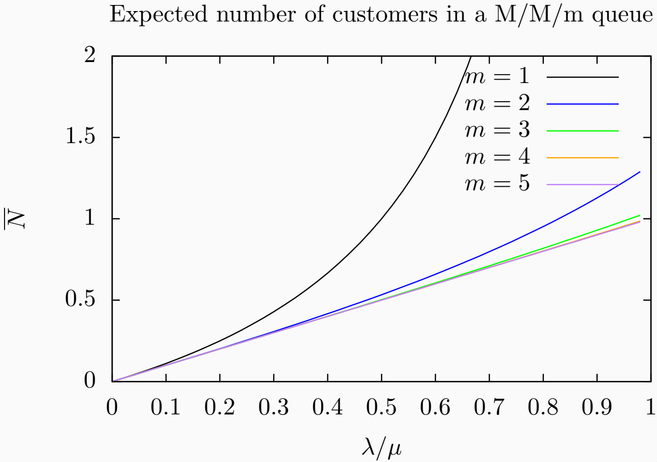
Expected number of customers and expected response time

- Assuming stability we have $\lambda = X$, where X is the throughput
- We can derive the steady-state performance measures, where $\overline{R}$ is derived by using Little's theorem

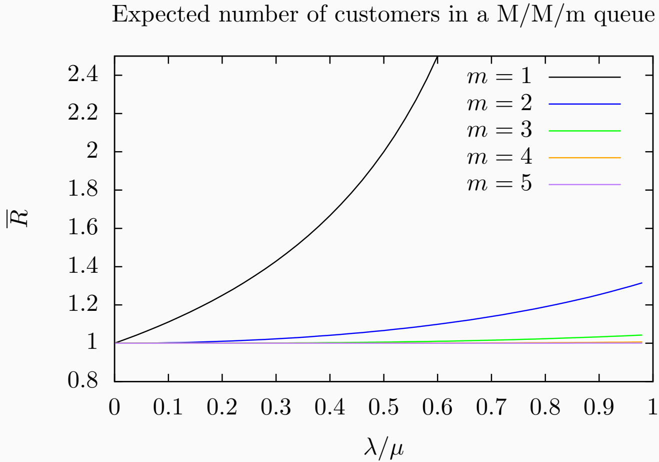
$$\overline{N} = E[n] = \frac{\rho}{1-\rho} C\left(m, \frac{\lambda}{\mu}\right) + \frac{\lambda}{\mu}$$

$$\overline{R} = E[r] = \frac{C\left(m, \frac{\lambda}{\mu}\right)}{m\mu - \lambda} + \frac{1}{\mu}$$

Plot of the expected number of customers



Plot of the expected response time



Case study: The Erlang-B formula

- m available connections
- λ calls per second is the arrival rate
- $1/\mu$ is the expected duration per call
- Calls that arrive at the system when all the m lines are busy are lost
    - The income is lost
- How many available connections do we need to have a loss-rate below a certain threshold?
    - Problem used in the call centers

The model

- To solve this type of problems we use a $M/M/m/m$ queue
- Arrivals at a saturated queue are lost
- Differences between the Erlang-B and the Erlang-C formulas:
  - The Erlang-C computes the probability that a customer that arrives at a $M/M/m$ queue has to wait to obtain the service
  - The Erlang-B computes the probability that a customer that arrives at a $M/M/m/m$ queue is not rejected

The Erlang-B formula

- The model is unconditionally stable because it has finite capacity
- The probability of rejection is:

$$E_B\left(m, \frac{\lambda}{\mu}\right) = \frac{(\lambda/\mu)^m}{m!}\left(\sum_{i=0}^{m}\frac{(\lambda/\mu)^i}{i!}\right)^{-1}$$

- The presence of the factorials and the sum makes the formula difficult to compute
- Not known a closed form expression for m

Erlang-B formula computation

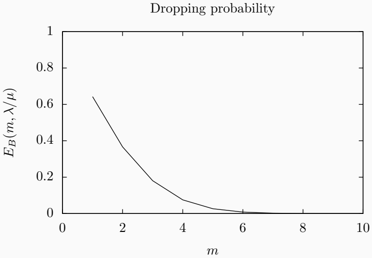
- The following recursive scheme is more numerically stable than the direct computation

$$\begin{cases} E_B\left(0, \frac{\lambda}{\mu}\right) = 1 \\ E_B\left(m, \frac{\lambda}{\mu}\right) = \frac{(\lambda/\mu)E_B(m-1,\lambda/\mu)}{m+(\lambda/\mu)E_B(m-1,\lambda/\mu)} & \text{if } m > 0 \end{cases}$$

Example

- We want to study an application with real-time requirements
- Jobs must not experience waiting time
- Jobs that cannot be processed immediately can be dropped
- Assume an arrival rate $\lambda = 10$ jobs/second
- The service time is $\mu^{-1} = 0.18$ seconds
- Under the exponential assumptions (Poisson arrival stream and exponential service time distribution) how many parallel cores to we need to have a loss rate below 1%?

The evaluation



- $m = 1$,
  $E(1, 1.8) = 0.6429$
- $m = 2$,
  $E(2, 1.8) = 0.3665$
- $m = 3$,
  $E(3, 1.8) = 0.1803$
- $m = 4$,
  $E(4, 1.8) = 0.0750$
- $m = 5$,
  $E(5, 1.8) = 0.0263$
- $m = 6$,
  $E(6, 1.8) = 0.0078$

The M/M/∞ queue

Model definition

- Poisson arrival process with rate λ
- Exponential service time
- Infinite number of servers
- The M/M/∞ queue usually models parallel independent computation
    - E.g. the thinking time
- The underlying stochastic process is still a birth&death process where $\mu(n) = n\mu$
- The queue is unconditionally stable
    - Why?

The steady-state distribution

- We can show that:

$$\pi(n) = \pi(0)\left(\frac{\lambda}{\mu}\right)^{n}\frac{1}{n!}$$

- For computing $\pi(0)$ we have:

$$\pi(0) = \left(\sum_{n=0}^{\infty}\left(\frac{\lambda}{\mu}\right)^{n}\frac{1}{n!}\right)^{-1} = e^{-\lambda/\mu}$$

where the series is always convergent

Performance measures

- Expected response time $\overline{R} = \frac{1}{\mu}$
- Expected number of customers $\overline{N} = \frac{\lambda}{\mu}$
- Throughput $X = \lambda$
- Notice that the customers do not have waiting time
- The results hold for any queueing discipline

The M/G/1 queue

Model description

- Poisson arrival process with rate λ
- General service time distribution with mean $1/mu$ and variance σ_s^2
- FCFS queueing discipline
- Infinite buffer
- Single server
- Stability condition $\lambda < \mu$, $\rho = \lambda/\mu < 1$

The Pollaczek-Khinchine (P-K) formula

- The formula states that the queue length depends not only on the ratio between the expected service time and the arrival rate, but also on the variance of the service time
- The formula:

$$E[n_q] = \frac{\rho^2 + \lambda^2 \sigma_s^2}{2(1-\rho)}$$

- For M/M/1 queue $\sigma_2^2 = 1/\mu^2$, and we can reproduce the well known result:

$$E[n_q] = \frac{\rho^2 + \rho^2}{2(1-\rho)} = \frac{2\rho^2}{2(1-\rho)} = \frac{\rho^2}{1-\rho}$$

Other performance measures

- Expected number of customers in queue:

$$\overline{N} = E[n] = \frac{\rho^2 + \lambda^2 \sigma_s^2}{2(1-\rho)} + \rho$$

- Expected waiting time:

$$E[w] = \frac{\rho + \lambda \mu \sigma_s^2}{2(\mu - \lambda)}$$

- Expected response time:

$$\overline{R} = E[r] = \frac{\rho + \lambda \mu \sigma_s^2}{2(\mu - \lambda)} + \frac{1}{\mu}$$

Exercise 1

- Derive the expected performance indices of the queue M/D/1 as a special case of the M/G/1 queue
- Compare the performance indices with those of the M/M/1 queue, which one performs better?

Exercise 2

- A hard disk can start the data transfer after a *seek time* and a *rotation time*. This delay is called *access time*. Once in position, the hard disk reads a sector and transfers it to the main memory.
- Let us consider the Barracuda Seagate HD. From the datasheet we learn that it has an average of $8.5ms$ that we assume to be distributed uniformly between $2ms$ and $15ms$. The disk has a rotation speed of $7200rpm$ and hence the rotation delay is uniformly distributed between $0ms$ and $8.34ms$. We ignore the caching and we assume that there is no correlation between the requests of consecutive sectors. The transfer rate $6Gb/s$ and each sector has 4096 bytes, therefore the transfer time of a sector is deterministic and equal to $0.006ms$ and we ignore it.
- Compute the maximum arrival rate for the requests to the disk
- Assume that the arrival of the requests follow a Poisson process and compute the expected response time of the disk in case of heavy load (close to saturation), light load (almost empty), and average load.

The M/G/1/PS queue

Characteristics

- Poisson arrival process with intensity λ
- General service time with mean μ^{-1}
- Single server
- Jobs enter in service as soon as they arrive but they share the processor with the other job in service. In fact, each job receives the same amount of computational power from the processor
- The waiting time is defined as the difference between the response time and the service time (i.e., the response time if the job was in the system alone for all its residence)

Main result: insensitivity

- The expected performance indices of the M/G/1/PS queue are the same of the M/M/1
- This result is known as the **insensitivity** of the M/G/1/PS queue: the performance indices depend only on the mean on the service time and the following moments are irrelevant
- This is very different from the M/G/1 whose expected performance indeces depend on the variance of the service time

FCFS vs PS

- Assume we can choose between FCFS and PS scheduling
- If the service time has a variance smaller than the exponential distribution with the same mean, then FCFS perfoms better than PS
- Otherwise, PS performs better
- If the variance is not known, it is better to opt for PS since it guarantees the performance indices of the M/M/1 queue independently of the distribution of the service time

The G/G/1 system

Introduction to the system

- In the G/G/1 we allow arbitrary independent inter-arrival times and service times
- The intuition suggests that higher variances in the inter-arrival and/or service time should negatively affect the waiting time
- The exact analysis of G/G/1 and G/G/k is a holy Grail for queueing theorists, although most have lost the hope to find a closed form formula
- But we have approximations!

Why is the analysis challenging?

- We cannot use PASTA
- The queue has two clocks: one for the inter-arrival times and one for the service times
- When both clocks are exponentially distributed, the analysis is simply a Markov chain analysis
- When one clock is exponential and the other is general, we can build the *embedded Markov chain*
    - Consider the exponential clock, and build a discrete time chain with all possible transitions due to the other clock in the exponential time frame
    - You obtain a discrete time Markov chain
- The case of two general clocks is **really** challenging

Kingman's formula

- Also called Variability-Utilization-service Time (VUT) equation
- Published by John Kingman in 1961
- The formula is an approximation and work well in heavy traffic conditions
- Parameters:
    - $\rho = \lambda/\mu$ is the load factor, $\rho < 1$
    - c_a coefficient of variation of the inter-arrival time
    - c_s coefficient of variation of the service time

$$\overline{W} \simeq \left(\frac{\rho}{1-\rho}\right)\left(\frac{c_a^2 + c_s^2}{2}\right)\frac{1}{\mu}$$

Nice special cases

- M/M/1: in this case $c_a = c_s = 1$ and we get:

  $$\overline{W} \simeq \left(\frac{\rho}{1-\rho}\right)\frac{1}{\mu}$$

  which is the correct expression
- M/G/1: in this case $c_a = 1$ and we get:

  $$\overline{W} \simeq \left(\frac{\rho}{1-\rho}\right)\left(\frac{1+c_s^2}{2}\right)\frac{1}{\mu}$$

  which is again correct

Size based scheduling: SRPT

Size based scheduling

- The assumption of this class of schedulers is that we know the service time of the jobs as soon as they arrive at the system
- This assumption can be very difficult to accept in many contexts
- The key idea is that it can be adopted when we know in advance the job to be made
    - Transfer of many files (e.g., Dropbox sync)
    - Transfer of resources of known size (e.g., web server resource transfer)

Shortest Remaining Processing Time (SRPT)

- Preemptive discipline with resume
- At each epoch, the job with the smallest remaining processing (service) time will be served
- In this way, small jobs overtake large jobs
- This approach reduces the expected response time

G/G/1/SRPT queue

- Main result: the expected response time of the G/G/1/SRPT is optimal
- All other queueing disciplines have worse or equal response time to the SRPT!
- Fairness: large jobs may have strong delays in heavy load
- It is possible to compute the mean performance indices of M/G/1/SRPT but there is no closed-form formula
- We have an integral formula for the expected conditional response time

Applications of SRPT

- M. Harchol Balter et al. *Size based scheduling to improve web performance*. ACM Trans. on Comp. Syst. 21(2):207-233, 2003
- B. Schroeder et al. *Web servers under overload: how scheduling can help.*. ACM Trans. on Internet Tech. 6(1):20-52, 2006

The authors apply a patch to Linux kernel to allow Apache to perform SRPT scheduling

The All-Can-Win theorem: what is it about?

- The all-can-win theorem deals with the comparison of PS vs SRPT
- The intuition suggests that small jobs are happy with SRPT, but the big ones would prefer PS
- This is due to the fact that if small jobs can overtake big jobs, the latter will starve
- Actually, **big jobs will not starve**. If the queue is stable they will eventually be served because the discipline is work conserving
- Moreover, things are not that terrible: the priority of the job is dynamic, while it ages it also acquires priority!

Performing the comparison

- In the comparison of the expected response time, we know that SRPT performs better than PS:

$$\overline{R}^{SRPT} \leq \overline{R}^{PS}$$

- Suppose that we know the size x of the job. Let us call $\overline{R}^{d}(x)$ the expected response time under a discipline d of a job whose size is x
- The intuition of before suggested that for small x $\overline{R}^{SRPT}(x) \leq \overline{R}^{PS}(x)$ while for large x the opposite holds. This would create the feeling of *starvation*
- The all-can-win theorem states that this is not true!

The All-Can-Win theorem

Theorem

Given an M/G/1/PS and a M/G/1/SRPT queue with the same parameters, if $\rho < 0.5$, then, for all x:

$$\overline{R}^{SRPT}(x) \leq \overline{R}^{PS}(x)$$

- Notice that $\rho < 0.5$ is a sufficient condition for the theorem to hold, but not necessary
- For example, if the service time distribution is a bounded Pareto with $\alpha = 1.1$ and values between 332 and 10^{10}, we can show that the statement is true until $\rho = 0.96$

Age based scheduling

Age based disciplines: main idea

- At each epoch, we take the decision on the job(s) to serve based on the amount of work they have received up to that moment
- In general we tend to favor the jobs that have received the smallest amount of work: the bet is that those who have obtained a lot of work, the will ask even more
- These disciplines work well when the service times are heavily tailed

Least Attained Service (LAS)

- Also known with other names like Foreground-Background
- The discipline is with preemption and resume
- At each epoch, the scheduling discipline works as a PS only on the jobs that have recevied the least amount of service
- Difficult to implement (necessary an unbounded number of queues)

Hazard rate of distribution

- Let X be a r.v. with probability density function $f(t)$ and cumulative density function $F(t)$, $f(t) = \partial F(t)/\partial t$
- Suppose X measures the service time of a job
- What is the probability to complete the job in $[t, t + dt)$ given that the job has received t seconds of service? For $dx \to 0$

$$h(t) = Pr\{t \leq X < t + dt | X \geq t\}$$

$$h(t) = \frac{Pr\{t \leq X < t + dt \wedge X \geq t\}}{Pr\{X \geq t\}}$$

Hence:

$$h(t) = \frac{f(t)}{1 - F(t)}$$

Random variables with decreasing hazard rate

- If the hazard rate of a random variable is monotonically decreasing, it means that as time passes the time to completion increases
- In terms of service time, it means that if we have given a lot of service to a job, this will probably need even more work
- Expoenential random variables or uniform random variables do not have decreasing hazard rate!
- Heavily tailed distributions can have decreasing hazard rate

Main result on LAS

- LAS is the optimal scheduling discipline if we do not know the job size in advance (or its distribution) but the job size has monotonically decreasing hazard rate
- In practice, when do we use LAS? We have many small jobs and a few huge jobs but we cannot say in advance who is small and who is big

The PS+PS queueing discipline

- This discipline is also known and the two-level PS discipline
- It is the practical implementation of LAS
- Preemption and Resume
- We maintain two PS queues. Queue 1 has strict priority on Queue 2
- We define a threshold T
- When a job arrives it stays in Queue 1 until: either it finishes its job and hence it leaves the queue, or it reaches the recevied work T. In the latter case it is moved to Queue 2
- If more than a job is in Queue 1, they are served in PS. Same for Queue 2.

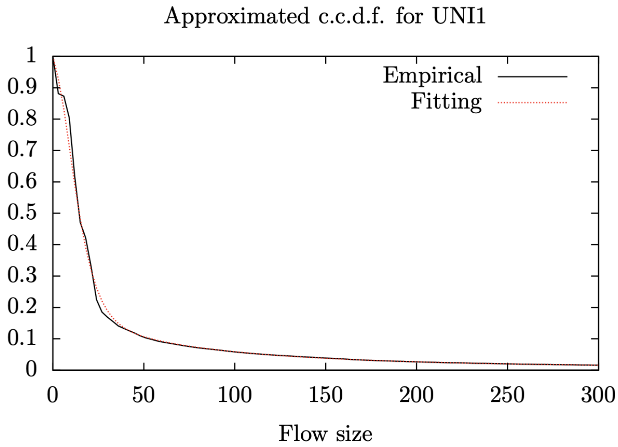
Main result on PS+PS

- If the job size has decreasing hazard rate, PS+PS cannot do worse than a simple PS whatever threshold T is chosen
- However, clearly the choice of T is strategic
- There are algorithms to determine the optimal value of T to minimize the expected response time
- With respect to LAS od SRPT, the problem of starvation is reduced

Example: scheduling of TCP jobs

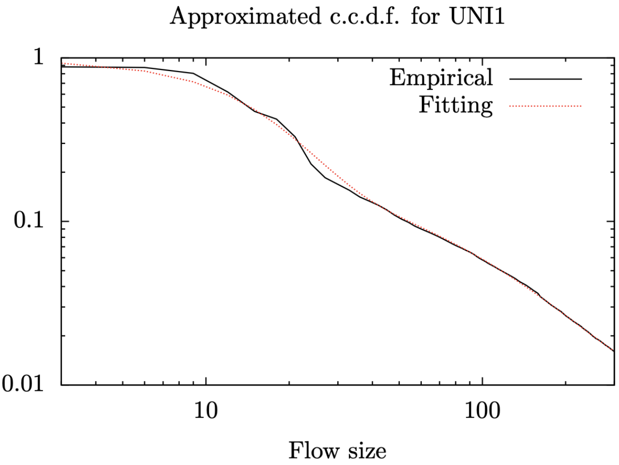
- The example is taken from the paper: Andrea Marin, Sabina Rossi, Carlo Zen: Size-based scheduling for TCP flows: Implementation and performance evaluation. Comput. Networks 183: 107574 (2020)
- Scenario: we have a router that receives the packet arriving from several TCP connections
- Research questions: can we use PS+PS to schedule the packets? Can we implement the discipline in a Linux kernel and test it?

Distribution of the flow size


Approximated c.c.d.f. for UNI1

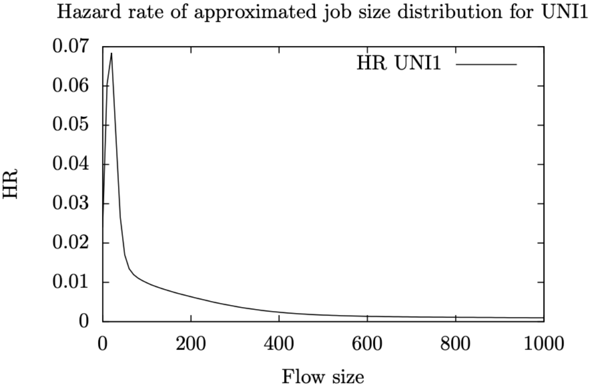
(a) c.c.d.f. of the TCP flow size distribution and its fitting for the dataset UNI1.

Distribution of the flow size



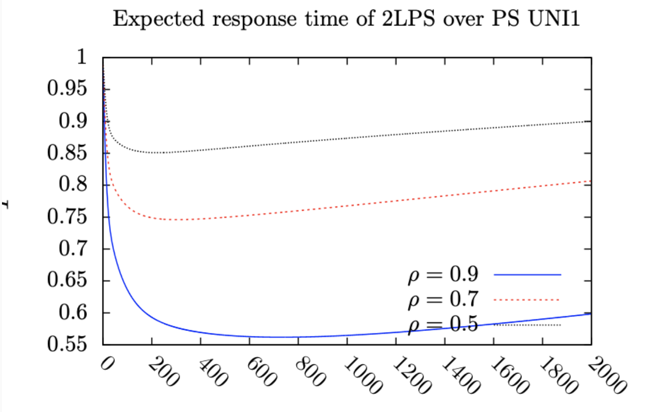
(b) c.c.d.f. of the TCP flow size distribution and its fitting for the dataset UNI1 with logarithmic scale.

Hazard rate

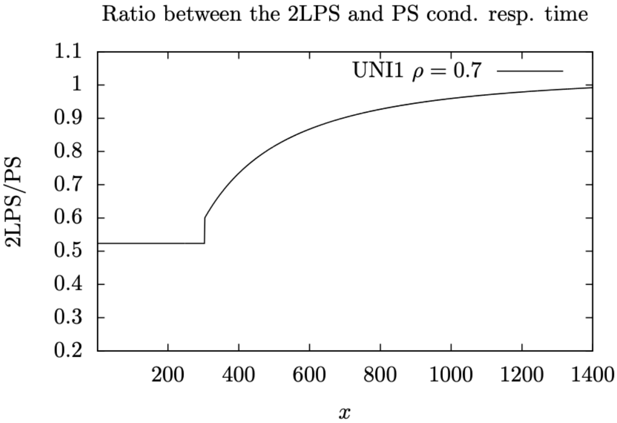


(a) Hazard rate of the distribution of the TCP flow sizes for the dataset of UNI1.

Example: PS+PS vs PS

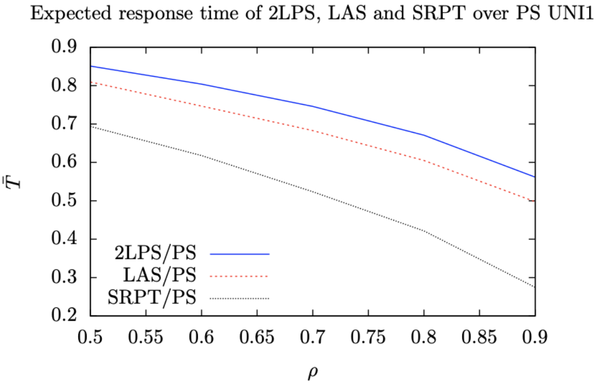


Example: Conditioned response time



(d) Conditioned response time of 2LPS over PS with a load factor of 0.7 for UNI1 dataset.

Example: Comparison with LAS and SRPT



(a) Dataset UNI1: Ratio between the 2LPS, LAS and SRPT expected response time and the PS expected response time as function of the load factor ρ.

Conclusion of the research

- PS+PS is very good to schedule TCP flows
- It can be implemented in Linux using hash tables to detect the flows
- Experiments with benchmarking show the benefits of PS+PS in real scenarios