SAPIENZA
UNIVERSITÀ DI ROMA

"SAPIENZA" UNIVERSITY OF ROME
FACULTY OF INFORMATION ENGINEERING,
INFORMATICS AND STATISTICS
DEPARTMENT OF COMPUTER SCIENCE

Computer Network Performance

Lecture notes integrated with the book "Performance Modeling and Design of Computer Systems", M. Harchol-Balter

Author
Simone Bianco

October 29, 2025

Contents

Information and Contacts

Personal notes and summaries collected as part of the *Computer Network Performance* course offered by the degree in Computer Science of the University of Rome "La Sapienza".

Further information and notes can be found at the following link:
https://github.com/Exyss/university-notes. Anyone can feel free to report inaccuracies, improvements or requests through the Issue system provided by GitHub itself or by contacting the author privately:

- Email: **bianco.simone@outlook.it**
- LinkedIn: **Simone Bianco**

The notes are constantly being updated, so please check if the changes have already been made in the most recent version.

Suggested prerequisites:

Networks and Probability

1 Queueing theory

1.1 Introduction to performance evaluation

The modern digital infrastructure is based on networks formed of hundres of interconnected computer systems. To manage the ingoing and outgoing traffic, *queuing* is essential. **Queueing theory** is the study of what happens when many *jobs* compete for limited resources, often resulting in queues and delays due to the system not being capable of serving every incoming job at once. At its core, this theory seeks to explain why queues form, how they behave and what can be done to minimize or eliminate them.

Consider a simple example: a computer system, such as a web server, handling just one job. The job arrives, consumes some resources (like CPU and I/O), and then departs. Since there are no other jobs in the system, it's easy to predict exactly when it will finish: there's no waiting time and no queue. However, resources are often shared among many jobs, leading to contention and delays.

The first goal of queueing theory is **predicting** the system's performances that we're interested in. The second goal is to **improve design** in order to achieve a better performance. We'll start by giving some concrete examples of performance prediction and system analysis. First, we'll give some definitions.

Definition 1.1: Open and closed system

A system is said to be open if there is at at least one way for jobs to exit the system. If no way exists, the system is said to be closed.

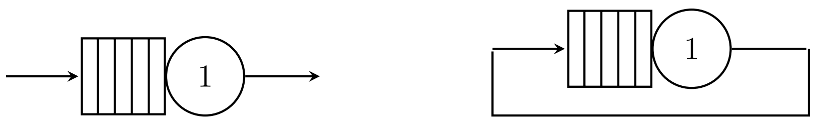


Figure 1.1: An open system and a closed system, both with one server and its queue.

We observe that closed systems may or may not contain a "thinking center", i.e. a subsystem of terminals that does some work before sending jobs to the system. Closed systems that don't contain such thinking center are called *batch systems*, while the others are called *interactive systems*.

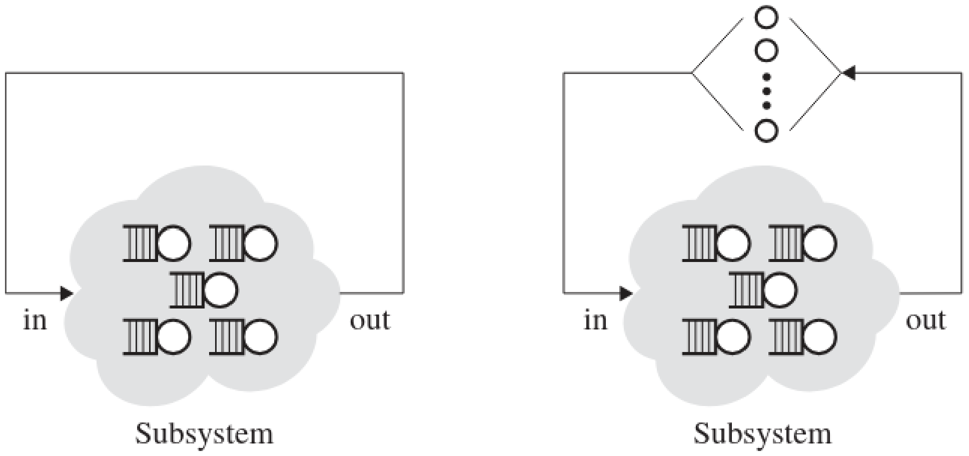


Figure 1.2: A batch system (left) and an interactive system (right)

In order to analyze and predict the behavior of a queueing system, we must first define a few key performance metrics. These metrics describe how jobs arrive, how quickly they are served and how efficiently the system operates overall. We start by defining three main parameters.

The **arrival rate**, typically denoted by λ and measured in jobs per unit of time, represents how frequently new jobs enter the system. For example, if an average of 10 jobs arrive every second, then $\lambda = 10$ jobs/second. In most queueing models, job arrivals are assumed to follow a random process – often a Poisson process – meaning that the times between successive arrivals are independent and exponentially distributed. The arrival rate thus characterizes the workload intensity imposed on the system.

The **service rate**, typically denoted by μ, measures how quickly the system can process jobs. Specifically, it represents the average number of jobs that can be completed per unit of time by a single server. For instance, if a server can complete 5 jobs per second on average, then $\mu = 5$ jobs/second.

The **throughput**, typically denoted by X, indicates the actual rate at which jobs are completed and leave the system. If a server processes 6 jobs per second on average, then $X = 6$ jobs/second.

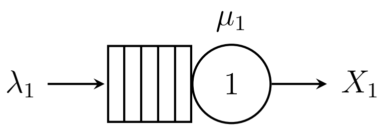


Figure 1.3: A device and its parameters.

To formally define these three metrics, we introduce three **elementary measures** based on elapsed time. Given a time instant t and a device i, let:

1. $A_i(t)$ denote the total number of jobs arrived at device i at time t
2. $C_i(t)$ denote the total number of jobs completed by device i at time t
3. $B_i(t)$ denote the total number of seconds the device i was busy working on jobs at time t

For the arrival rate and the throughput, we can express there two metrics as the ratio between the number of jobs of interest over the total amount of time t that elapsed (as $t \to +\infty$).

Definition 1.2: Arrival rate and Throughput

Given a device i, we define the arrival rate λ_i and the throughput X_i of device i as:

$$\lambda_i = \lim_{t \to +\infty} \frac{A_i(t)}{t} \qquad X_i = \lim_{t \to +\infty} \frac{C_i(t)}{t}$$

For the service rate, instead, we must restrict our interest only to the portion of time for which the device was actually busy working (otherwise, we would also be accounting for the time spent by jobs waiting in the queue).

Definition 1.3: Service rate

Given a device i, we define the service rate μ_i of device i as:

$$\mu_i = \lim_{t \to +\infty} \frac{C_i(t)}{B_i(t)}$$

These three fundamental measures can also be defined at **global** level, i.e. for the whole system. In this case, they are usually denoted with λ, μ, X and they are defined as:

$$\lambda = \lim_{t \to +\infty} \frac{A(t)}{t} \qquad \mu = \lim_{t \to +\infty} \frac{C(t)}{B(t)} \qquad X = \lim_{t \to +\infty} \frac{C(t)}{t}$$

where $A(t)$ and $C(t)$ are the total number of arrived and completed jobs in the system, while $B(t)$ is the total number of seconds that the system was busy working (i.e. with at least one device working). From now on, we'll always assume that the symbols λ, μ and X refer to these three parameters. When we have a system with multiple servers, we'll use pedices to distinguish the servers' parameters (e.g. λ_1, μ_1, X_1 are the parameters of Server 1). When the subscript is omitted, we're usually referring to the global metric, i.e. the metric over the whole system.

1.2 System stability

In order for performance measures to actually mean something, the system must be working at full capacity (hence why we consider $t \to +\infty$). When the system is closed, this hypothesis is not required since even a single job is able to make the system always operational (no idle time). We observe that these three parameters are strictly related to each other and may act as a bottleneck:

- If $\lambda < \mu$, the system receives at most as many jobs than those that it can process, meaning that it is able to serve them all. Hence, we get that $X = \lambda$.
- If $\lambda \geq \mu$, the system receives more jobs than those that it can process, meaning that it will be forced to drop some of them. Hence, we get that $X = \mu$.

We observe that the relationship between the arrival rate and the service rate not only defines the value of the throughput, but it also defined the **stability** of the system. A system is said to be *stable* when there is no queue build-up any device of the system, i.e. the size of every queue of each device doesn't "explode", becoming too large (thus forcing the devices to drop incoming jobs). It's easy to see that a system is stable if and only if each device is able to process jobs faster than they arrive in the queue, meaning that $\lambda_i < \mu_i$.

Proposition 1.1: System stability

A system is stable if and only if for every device i it holds that $\lambda_i \leq \mu_i$.

Proof. We prove both directions. First, it's easy to see that if $\lambda_i \leq \mu_i$ holds for every i then each device is able to process every incoming job without queue build-up, concluding that the system is stable.

Let $A_i^{\text{queue}}(t)$ and $D_i^{\text{queue}}(t)$ respectively denote the number of jobs arrived to and departed from the queue of device i up to time t. Similarly, let $N_i^{\text{queue}}(t)$ be the number of jobs in the queue of device i at time t. We observe that:

$$N_i^{\text{queue}}(t) = A_i^{\text{queue}}(t) - D_i^{\text{queue}}(t)$$

By definition, we have that $A_i^{\text{queue}}(t) = A_i(t) = \lambda t$. Similarly, we have that $D_i^{\text{queue}}(t) \leq \mu t$ (due to the fact that some jobs may be dropped due to the queue being full). This implies that:

$$N_i^{\text{queue}}(t) = A_i^{\text{queue}}(t) - D_i^{\text{queue}}(t) \geq \lambda t - \mu t = (\lambda - \mu)t$$

By contrapositive, suppose that there is at least one device j such that $\lambda_j > \mu_j$. Then, as $t \to +\infty$ we get that $N_j^{\text{queue}}(t) = +\infty$, meaning that the system is unstable. $\square$

Clearly, we want to restrict our interest to systems that are stable since measuring the performance of an unstable system has no real use (every measure either goes to infinity or to zero as time elapses). Hence, from now on we'll assume to be working with stable systems. Since $\lambda_i = \mu_i$ holds for every device in stable systems, we know that the throughput is limited by the arrival rate, thus $X_i = \lambda_i < \mu_i$. In closed systems, the equality $X_i = \lambda_i$ always holds (even in unstable conditions) due to necessary packet drops.

Proposition 1.2

Given a device i, it holds that:

- If the system is closed then $X_i = \lambda_i$
- If the system is open then $X_i = \lambda_i$ if and only if the system is stable

Each time we want to change some part of the system, we have to ensure that stability is preserved. For instance, consider the following system with probabilistic routing.

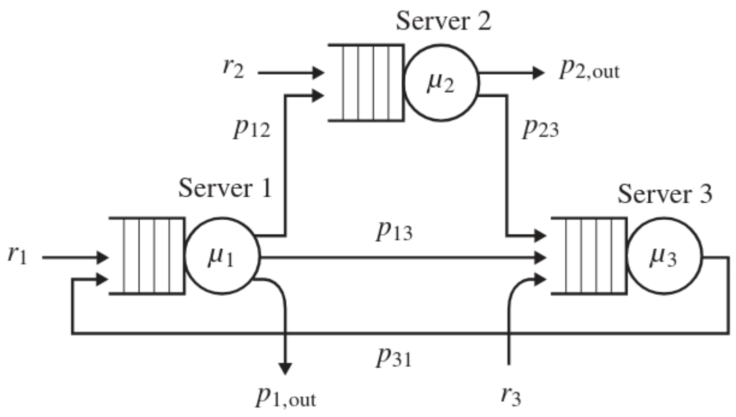


Figure 1.4: A system with probabilistic routing

Here, each device i receives external arrivals with rate r_i, but it also receives internal arrivals from some of the other servers. The sum of these two arrival rates corresponds to the total arrival rate λ_i. Each packet that is processed by device i is next routed to server j with probability $p_{i,j}$, where the device "out" represents the outside of the system.

Suppose that $\mu_1 = \mu_2 = \mu_3 = 10$ j/s (jobs/seconds) and that $r_2 = r_3 = 1$ j/s. The routing probabilities are set to the following values:

- $p_{1,2} = p_{2,out} = 0.8$
- $p_{2,3} = p_{1,3} = 0.2$
- $p_{3,1} = 1$

We want to find the maximum possible value for the external arrival rate r_1, preserving stability. First, we observe that:

$$\begin{cases} \lambda_1 = r_1 + p_{3,1}X_3 \\ \lambda_2 = r_2 + p_{1,2}X_1 \\ \lambda_3 = r_3 + p_{1,3}X_1 + p_{2,3}X_2 \end{cases}$$

Since we want stability to hold, we can assme that $X_i = \lambda_i$ holds for each device i. Substituting the probabilities, we obtain that:

$$\begin{cases} \lambda_1 = r_1 + \lambda_3 \\ \lambda_2 = r_2 + \frac{8}{10}\lambda_1 \\ \lambda_3 = r_3 + \frac{2}{10}\lambda_1 + p_{2,3}\lambda_2 \end{cases}$$

Solving the system in function of r_1, we obtain that:

$$\begin{cases} \lambda_1 = \frac{15}{8} + \frac{25}{16}r_1 \\ \lambda_2 = \frac{5}{4} + \frac{5}{4}r_1 \\ \lambda_3 = \frac{15}{8} + \frac{9}{16}r_1 \end{cases}$$

To impose stability, we know that $\lambda_i < \mu_i$ must hold for every device i. Therefore, we get that:

$$\begin{cases} 10 > \frac{15}{8} + \frac{25}{16}r_1 \\ 10 > \frac{5}{4} + \frac{5}{4}r_1 \\ 10 > \frac{15}{8} + \frac{9}{16}r_1 \end{cases}$$

Solving each inequality, we conclude that:

$$\begin{cases} r_1 < \frac{26}{5} \\ r_1 < 6 \\ r_1 < \frac{130}{9} \end{cases}$$

Thus, in order for the system to be stable the value of r_1 must be such that:

$$r_1 < \min\left(\frac{26}{5}, 6, \frac{130}{9}\right) = \frac{26}{5}$$

1.3 Utilization and time analysis

In the previous sections introduced rate metrics to analyze the speed of the system. But what about time metrics? In particular, we're interested in knowing performances such as how much is a device working, the time required by a job to be processed by the whole system and by a single device.

The first metric that we're going to introduce is the **utilization** of a device, written as ρ_i, which describes how busy a device is. Intuitively, this metric is represented as the ratio between the the time in which the device was busy working over the total elapsed time.

Definition 1.4: Utilization

Given a device i, we define the utilization ρ_i as:

$$\rho_i = \lim_{t \to +\infty} \frac{B_i(t)}{t}$$

Since the utilization is described as the percentage of time the device was busy working, it's easy to see that:

$$\begin{aligned}\rho_i &= \Pr[\text{device } i \text{ is busy}] \\ &= \Pr[N_i^{\text{server}} = 1] \\ &= 1 \cdot \Pr[N_i^{\text{server}} = 1] + 0 \cdot \Pr[N_i^{\text{server}} = 0] \\ &= \mathbb{E}[N_i^{\text{server}}]\end{aligned}$$

where N_i^{server} is the **server population** of device i, that being the number of jobs in the server of device i (thus not in the queue), which is at most 1.

The second metric that we're going to introduce is **service time**, typically written as S_i. Intuitively, this is the time spent by a device to process a job. By definition, the average service time of a device is nothing more than the inverse of its service rate.

Definition 1.5: Service time

Given a device i, we define the service time S_i of device i as the random variable such that:

$$\mathbb{E}[S_i] = \frac{1}{\mu_i}$$

Clearly, the above definition implies that:

$$\mathbb{E}[S_i] = \frac{1}{\mu_i} = \lim_{t \to +\infty} \frac{B_i(t)}{C_i(t)}$$

This allows us to formalize our first law, known as the **utilization law**.

Law 1.1: Utilization law

Given a device i, it holds that:

$$\rho_i = \mathbb{E}[S_i] X_i$$

Proof through definitions. Through easy algebraic manipulation we get that:

$$X_i = \frac{C_i(\tau)}{\tau} = \frac{C_i(\tau)}{B_i(\tau)} \cdot \frac{B_i(\tau)}{\tau} = \mu_i \rho_i$$

Thus, we conclude that $\rho_i = \mathbb{E}[S_i] X_i$. □

Proof through expected value. By definition, we have that:

$$\begin{aligned}X_i &= \mathbb{E}[\text{Completion rate of } i] \\ &= \mathbb{E}[\text{Completion rate of } i \mid i \text{ is busy}] \Pr[i \text{ is busy}] \\ &\quad + \mathbb{E}[\text{Completion rate of } i \mid i \text{ isn't busy}] \Pr[i \text{ isn't busy}] \\ &= \mu_i \rho_i + 0(1 - \rho_i) \\ &= \mu_i \rho_i\end{aligned}$$

Thus, we conclude that $\rho_i = \mathbb{E}[S_i]X_i$. □

To avoid confusion, we remark that service time referes only to the time spent by the job inside the device itself, thus without considering the time spent by it inside its queue. This latter metric is known as **waiting time**, written as W_i. When considering both portion sof time, the metric is known as **response time**, the time spent by a single job be fully served by a device.

Definition 1.6: Response time

Given a device i, we define the response time R_i of device i as the random variable such that:

$$\mathbb{E}[R_i] = \mathbb{E}[W_i] + \mathbb{E}[S_i]$$

where W_i is the time spent by a job inside the queue of device i.

Consider a single device i. When the system is stable, we know that jobs are being processed faster than their arrival in the queue ($\mu_i > \lambda_i$). In this case, the value $\mu_i - \lambda_i$ represents how much faster the server can serve customers than they arrive, namely the *net service capacity*. It's easy to see that the inverse of such metric is exactly the average response time of the device.

Proposition 1.3

Consider a stable system. Given a device i, it holds that:

$$\mathbb{E}[R_i] = \frac{1}{\mu_i - \lambda_i}$$

Through the above result, we also conclude that:

$$\mathbb{E}[W_i] = \mathbb{E}[R_i] - \mathbb{E}[S_i] = \frac{1}{\mu_i - \lambda_i} - \frac{1}{\mu_i} = \frac{\lambda_i}{\mu_i(\mu_i - \lambda_i)}$$

Since $X_i = \lambda_i$ holds, through the utilization law, we conclude that:

$$\mathbb{E}[W_i] = \frac{\lambda_i}{\mu_i(\mu_i - \lambda_i)} = \frac{X_i}{\mu_i(\mu_i - \lambda_i)} = \rho\,\mathbb{E}[R_i]$$

In other words, the average time spend by a time in the queue is given by the average time required by a job to be fully processed by the device and the workload of the device itself.

1.4 Little's law

We introduce two new global metrics: the system traversal time and the system response time. The **system traversal time** can be informally described as the time required by a job to go from "out" of the system to "out" of the system. The **system response time**, instead, can be informally described as the time required by a job to go from "into" the system to "out" of the system.

For open systems, it holds that $\mathbb{E}[T] = \mathbb{E}[R]$ since there is no difference from coming out of the system or into the system due to all jobs coming from the outside. For closed system, instead, the traversal time may also contain time required by the system to "think" (e.g. we're working with an interactive system). Thus, we have that $\mathbb{E}[T] = \mathbb{E}[R] + \mathbb{E}[Z]$, where Z is the **thinking time** of the system. If the closed system has no thinking center, we can simply set $\mathbb{E}[Z] = 0$.

Definition 1.7: Traversal time

The traversal time T of a system is the random variable such that:

- $\mathbb{E}[T] = \mathbb{E}[R]$ if the system is open
- $\mathbb{E}[T] = \mathbb{E}[R] + \mathbb{E}[Z]$ if the system is closed

To compute the value of $\mathbb{E}[R]$, we usually use Little's Law, one of the most powerful and elegant results in queueing theory and operations management. It provides a simple relationship between three fundamental performance measures of any stable system. Formally, it states that $\mathbb{E}[T] = \frac{\mathbb{E}[N]}{X}$, where N is the number of jobs in the system. In order for such law to hold, the system must be **ergodic**.

Definition 1.8: Ergodic system

A system is said to be ergodic if it has the following three properties:

- *Irreducibility*: from every state (defined as the population in the servers) the system can reach every other state and back.
- *Positive recurrence*: starting from state, the probability to return in that same state over infinite time is 1.
- *Aperiodicity*: the system doesn't have a periodic behavior, meaning that it doesn't return in a specific state after a precise amount of time.

Law 1.2: Little's law

Given an ergodic system, it holds that:

$$\mathbb{E}[T] = \frac{\mathbb{E}[N]}{X}$$

Proof. Fix a generic time instant t. Let $\mathcal{A}(t)$ denote the area used by the jobs up to time t (see figure Figure 1.5). Interpreting the graph as a continuous function, we get that:

$$\mathcal{A}(t) = \int_0^t N(s)\,ds$$

where $N(s)$ denotes the number of jobs in the system at time s. For each job j, let T_j denote the time required to complete job j. Moreover, let $J_A(t)$, $J_C(t)$ denote, respectively, the set of arrived and completed jobs at time t. We observe that:

$$\sum_{j \in J_C(t)} T_j \leq \mathcal{A}(t) \leq \sum_{j \in J_A(t)} T_j$$

Dividing the whole inequality by t we get that:

$$\frac{1}{t} \sum_{j \in J_C(t)} T_j \leq \frac{1}{t} \int_0^t N(s)\,ds \leq \frac{1}{t} \sum_{j \in J_A(t)} T_j$$

Through algebraic manipulation we get that:

$$\frac{C(t)}{t} \cdot \frac{\sum_{j \in J_C(t)} T_j}{C(t)} \leq \frac{\int_0^t N(s)\,ds}{t} \leq \frac{A(t)}{t} \cdot \frac{\sum_{j \in J_A(t)} T_j}{A(t)}$$

By taking the limit as $t \to +\infty$, we obtain that:

$$\begin{aligned} &\lim_{t \to +\infty} \frac{C(t)}{t} \cdot \frac{\sum_{j \in J_C(t)} T_j}{C(t)} \leq \lim_{t \to +\infty} \frac{\int_0^t N(s)\,ds}{t} \leq \lim_{t \to +\infty} \frac{A(t)}{t} \cdot \frac{\sum_{j \in J_A(t)} T_j}{A(t)} \\ \implies & X \cdot \lim_{t \to +\infty} \frac{\sum_{j \in J_C(t)} T_j}{C(t)} \leq \mathbb{E}[N] \leq X \cdot \lim_{t \to +\infty} \frac{\sum_{j \in J_A(t)} T_j}{A(t)} \end{aligned}$$

since $\lim_{t \to +\infty} \frac{C(t)}{t} = X$ and $\lim_{t \to +\infty} \frac{A(t)}{t} = \lambda = X$. Moreover, we observe that as $t \to +\infty$ the difference between arrival and completion becomes negligible. Thus, we get that:

$$\lim_{t \to +\infty} \frac{\sum_{j \in J_C(t)} T_j}{C(t)} = \mathbb{E}[T] = \lim_{t \to +\infty} \frac{\sum_{j \in J_A(t)} T_j}{A(t)}$$

concluding that $X\,\mathbb{E}[T] \leq \mathbb{E}[N] \leq X\,\mathbb{E}[T]$. □

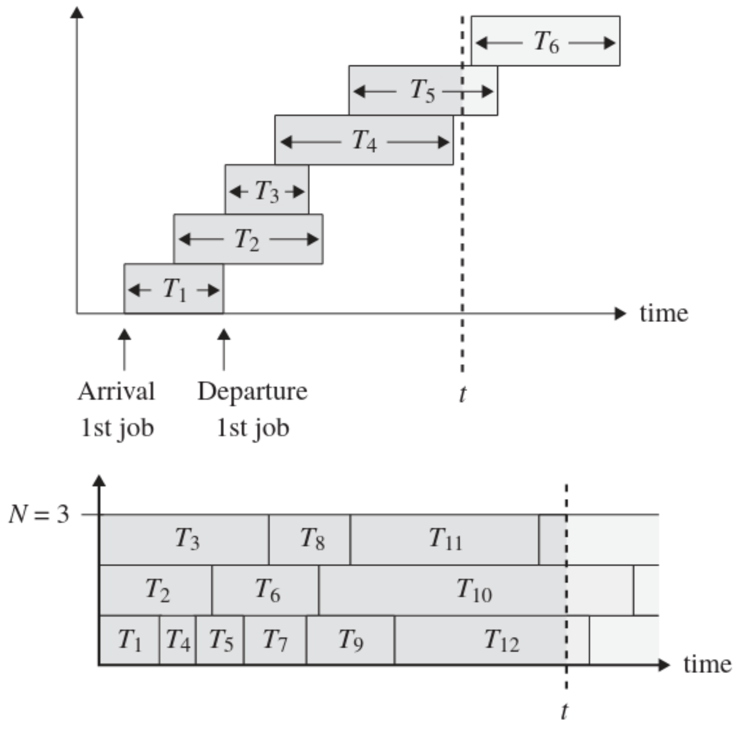


Figure 1.5: Graph of job system times in open system (above) and closed systems (below). The darker area corresponds to $\mathcal{A}(t)$.

After proving the law's correctness, we discuss its applications. First of all, we observe that the law can be made more explicit for open and closed systems by expanding the traversal time. For open systems we have that:

$$\mathbb{E}[R] = \mathbb{E}[T] = \frac{\mathbb{E}[N]}{X}$$

For closed systems, we also observe that $\mathbb{E}[N] = N$ since the number of jobs in the system is constant. Thus:

$$\mathbb{E}[R] = \frac{N}{X} - \mathbb{E}[Z]$$

The latter formula is also known as **response time law for closed systems**.

Nonetheless, the true power of the law comes from the fact that it can be applied not only to the whole system but also to sub-systems. The key idea here is that any sub-system can be viewed as a "box" with ingoing and outgoing jobs. Then, each box can be viewed as an open system on which the law can be applied.

For instance, consider the following interactive system with a central system deployed in the cloud (thus we don't know its interal connections).

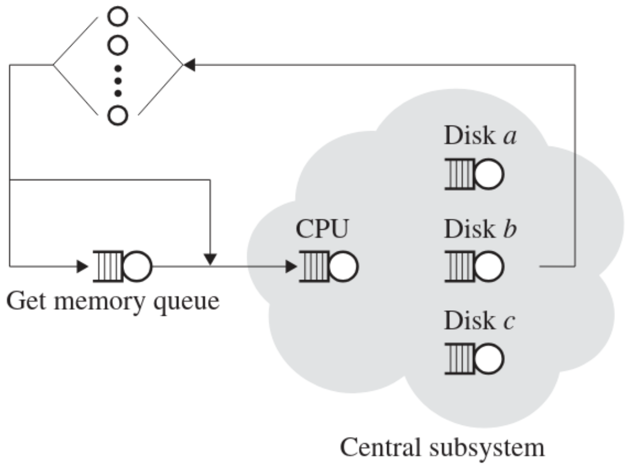


Figure 1.6: An interactive system with a cloud central system

The following information about the system is known:

- $\mathbb{E}[N] = 23$
- $\mathbb{E}[Z] = 21$ secs
- $X = 0.45$ jobs/secs
- $\mathbb{E}[N_{\text{mem}}] = 11.65$

We want to compute the response time of the central cloud system, namely R_{cloud}. With the known information, we can easily apply the response time law to the whole system, obtaining that:

$$\mathbb{E}[T] = \frac{\mathbb{E}[N]}{X} - \mathbb{E}[Z] = 23 \cdot \frac{100}{45} - 21 = \frac{271}{9} \text{ sec}$$

Now, we use the real power of Little's law. Consider the following sub-system described by the red box. We refer to this sub-system as *mem-sys*.

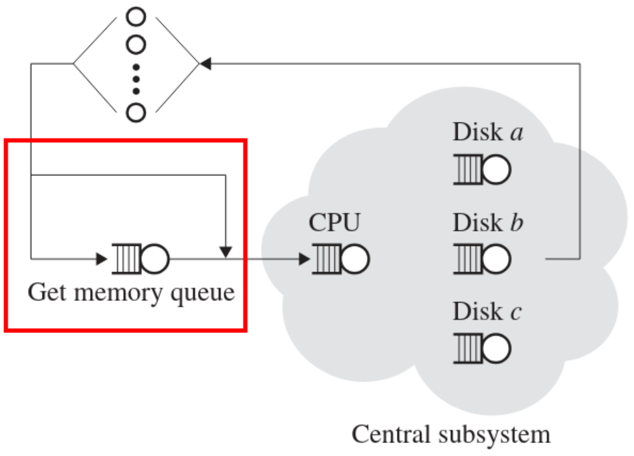


Figure 1.7: The mem-sys sub-system.

Similarly, the cloud center can also be viewed as a sub-system. We observe that

$$\mathbb{E}[R_{\text{cloud}}] = \mathbb{E}[T] - \mathbb{E}[R_{\text{mem-sys}}]$$

Since this mem-sys sub-system can be viewed as an open system, by Little's law we have that:

$$\mathbb{E}[R_{\text{mem-sys}}] = \frac{\mathbb{E}[N_{\text{mem-sys}}]}{X_{\text{mem-sys}}}$$

Now, we observe that $X_{\text{mem-sys}} = X$ since every job of the system enters to and exits from mem-sys. Moreover, we also have that $\mathbb{E}[N_{\text{mem-sys}}] = \mathbb{E}[N_{\text{mem}}]$, concluding that:

$$\mathbb{E}[R_{\text{mem-sys}}] = \frac{\mathbb{E}[N_{\text{mem-sys}}]}{X_{\text{mem-sys}}} = \frac{\mathbb{E}[N_{\text{mem}}]}{X} = \frac{1165}{100}\frac{100}{45} = \frac{233}{9} \text{ sec}$$

Therefore, we conclude that:

$$\mathbb{E}[R_{\text{cloud}}] = \mathbb{E}[T] - \mathbb{E}[R_{\text{mem-sys}}] = \frac{271}{9} - \frac{233}{9} = \frac{38}{9}$$

On a simpler case, Little's law can also be used to derive the utilization law. Given a system made of a single device, we can consider the sub-system composed only by the server. Since there is no queue in the system, we know that $\mathbb{E}[R_i^{\text{server}}] = \mathbb{E}[S_i^{\text{server}}]$. Then, byLittle's law we conclude that:

$$\mathbb{E}[S_i^{\text{server}}] = \mathbb{E}[R_i^{\text{server}}] = \frac{\mathbb{E}[N_i^{\text{server}}]}{X_i^{\text{server}}} = \frac{\rho}{X_i}$$

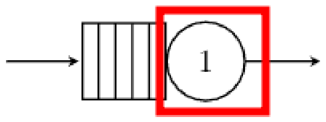


Figure 1.8: The sub-system considered to derive the utilization law

1.5 Forced flow law

Consider the following interactive system with a CPU device. The device has two outgoing links: one that goes back to the terminal center and one that loops back into the device itself. Each job exiting the device has $\frac{1}{2}$ probability of looping back and $\frac{1}{2}$ probability of going back to the terminal.

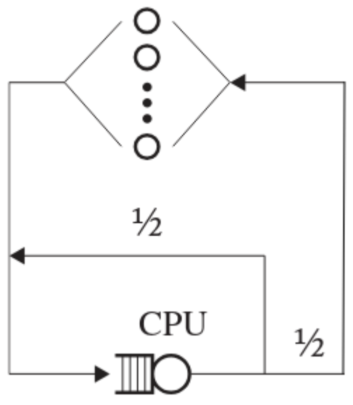


Figure 1.9: An interactive system with a fork with 0.5 probability of going back to the terminals.

Consider now the sub-system formed of the CPU device and the looping link (represented by the red box in Figure 1.10). Since every job of the system enters and exits from this sub-system, we know that the throughput of the latter must be equal to the system throughput. Thus, we have that $X_{\text{sub-sys}} = X$.

But what about the throughput of the sub-system formed only by the CPU device (represented by the green box in Figure 1.10)? We notice that every job that exits the CPU and goes back to the terminal center gets accounted both for the CPU throughput and the global system throughput. Since each exiting job has $\frac{1}{2}$ probability of going back to the terminal center, we can expect that $X = \frac{1}{2}X_i$. Indeed, this is the case!

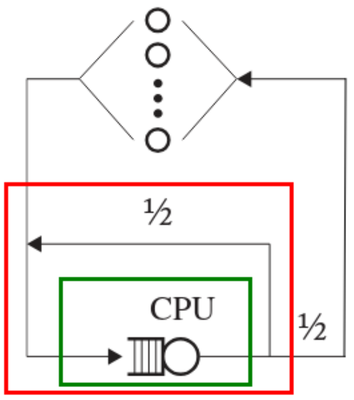


Figure 1.10: The two sub-systems.

In general, the relationship between the overall system throughput and the throughput of each individual service center within the system is described by the **forced flow law**, which formally states that $X_i = \mathbb{E}[V_i]X$. Here, V_i is the **visit ratio** of the device, that being the number of visits any job makes to the device.

Definition 1.9: Visit ratio

Given a device i, we define the visit ratio S_i of device i as:

$$V_i = \lim_{t \to +\infty} \frac{C_i(t)}{C(t)}$$

Law 1.3: Forced flow law

Given a device i, it holds that:

$$X_i = \mathbb{E}[V_i]X$$

Proof. Let $V_i^{(j)}$ be the number of visits that the j-th job makes to device i. Let $J_C(t)$ be the set of completed jobs at time t. We observe that:

$$C_i(t) = \sum_{j \in J_C(t)} V_i^{(j)}$$

Dividing the whole equality by t and rewriting the right-hand side, we get that:

$$\frac{C_i(t)}{t} = \frac{\sum_{j \in J_C(t)} V_i^{(j)}}{t} = \frac{\sum_{j \in J_C(t)} V_i^{(j)}}{C(t)} \cdot \frac{C(t)}{t}$$

By taking the limit as $t \to +\infty$, we obtain that:

$$\lim_{t \to +\infty} \frac{C_i(t)}{t} = \lim_{t \to +\infty} \frac{\sum_{j \in J_C(t)} V_i^{(j)}}{C(t)} \cdot \frac{C(t)}{t} \implies X_i = \mathbb{E}[V_i]X$$

□

Computing the value of $\mathbb{E}[V_i]$ is not an easy task since it highly depends on the structure of the system and the probabilities of each link. For instance, consider the following generalization of the previous example.

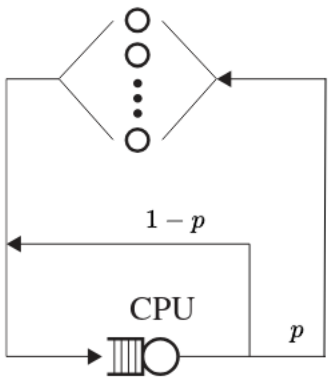


Figure 1.11: An interactive system with a fork with p probability of going back to the terminals.

First of all, we observe that $\Pr[V_i = k] = \Pr[\text{the job exits after looping } k-1 \text{ times}]$ since one visit is accounted when the job exits the loop. Moreover, probabilistically speaking, it is possible for a job to loop indefinitely in the system. Thus, we have that:

$$\mathbb{E}[V_i] = \sum_{k=0}^{+\infty} k \Pr[V_i = k] = \sum_{k=0}^{+\infty} k \Pr[\text{the job exits after looping } k-1 \text{ times}]$$

Now, it's easy to see that the event of looping $k-1$ times inside the device is described by a geometric process: in order to exit at the k-th iteration, the job must first loop for $k-1$ times, meaning that:

$$\Pr[\text{the job exits after looping } k-1 \text{ times}] = (1-p)^{k-1}p$$

Thus, we get that:

$$\mathbb{E}[V_i] = \sum_{k=0}^{+\infty} k(1-p)^k p$$

concluding that V_i is an r.v. with geometric distribution of parameter p. Identifying the distribution type of V_i allows us to immediately compute its expected value through known results. In particular, since $V_i \sim \text{Geo}(p)$, we immediately get that $\mathbb{E}[V_i] = \frac{1}{p}$. In other words, the averagae number of visits received by device i is the inverse of the probability of a job exiting the loop. Going back to our original example, we get that $X_i = 2X$ as expected by our observation.

The forced flow law also gives us an additional way to express the average system response time in terms of the average response time of the devices composing the system.

Proposition 1.4

Given a system with n devices, it holds that:

$$\mathbb{E}[R] = \sum_{i=1}^{n} \mathbb{E}[V_i]\,\mathbb{E}[R_i]$$

Proof. First, we observe that $\mathbb{E}[N] = \mathbb{E}[N_1] + \ldots + \mathbb{E}[N_n]$. By applying Little's law on each device and the whole system, we get that:

$$\mathbb{E}[R]X = \mathbb{E}[R_i]X_1 + \ldots + \mathbb{E}[R_n]X_n$$

By dividing both sides by X and by applying the forced flow law, we conclude that:

$$\mathbb{E}[R] = \mathbb{E}[V_1]\,\mathbb{E}[R_1] + \ldots + \mathbb{E}[V_n]\,\mathbb{E}[R_n]$$

□

1.6 Demand law and bottleneck law

The last fundamental metric that we're going to introduce is **device demand**, that being the effective load per job placed on a device. This metric is defined as the sum over all the single service time intervals per visit of a job.

Definition 1.10: Demand

Given a device i, the demand of i, written as D_i, is defined as:

$$D_i = \sum_{j=1}^{V_i} S_i^{(j)}$$

where $S_i^{(j)}$ is the service time of i for the j-th job.

Intuitively, even if a device is fast (small S_i), it may have an high visit count (high V_i), making the effective load higher. The relationship between these three metrics is described by the **demand law**.

Law 1.4: Demand law

Given a device i, it holds that:

$$\mathbb{E}[D_i] = \mathbb{E}[V_i]\,\mathbb{E}[S_i]$$

Proof. Before proving the result, we discuss its non-trivialness: even tought $S_i = \sum_{j=1}^{+\infty} S_i^{(j)}$, we cannot directly establish the result since V_i is a random variable – which may be dependent on other variables. In fact, an independence assumption will be made throughout the proof. First, we observe that:

$$\mathbb{E}[D_i] = \mathbb{E}\left[\sum_{j=1}^{V_i} S_i^{(j)}\right] = \sum_{n=0}^{+\infty} \mathbb{E}\left[\sum_{j=1}^{n} S_i^{(j)} \mid V_i = n\right] \Pr[V_i = n]$$

Here, we assume that the number of visits a job makes to device i is not affected by its service demand at the device (it's easy to see that this assumption is realistic in almost every case). Thus, we get that:

$$\begin{aligned}\mathbb{E}[D_i] &= \sum_{n=0}^{+\infty} \mathbb{E}\left[\sum_{j=1}^{n} S_i^{(j)}\right] \Pr[V_i = n] \\ &= \sum_{n=0}^{+\infty} n\,\mathbb{E}[S_i]\Pr[V_i = n] \\ &= \mathbb{E}[S_i]\,\mathbb{E}[V_i]\end{aligned}$$

□

Through the demand law we also get another formulation of demand in terms of elementary measures:

$$\mathbb{E}[D_i] = \mathbb{E}[V_i]\,\mathbb{E}[S_i] = \lim_{t\to+\infty} \frac{C_i(t)}{C(t)}\frac{B_i(t)}{C_i(t)} = \lim_{t\to+\infty} \frac{B_i(t)}{C(t)}$$

Out of all the metrics that we have discussed so far, demand is one of the most important ones to analyze in a system. This property is described by the **bottleneck law**

Law 1.5: Bottleneck law

Given a device i, it holds that:

$$\rho_i = \mathbb{E}[D_i]X$$

Proof. By applying the demand law, the forced flow law and the utilization law, we get that:

$$\mathbb{E}[D_i] = \mathbb{E}[V_i]\,\mathbb{E}[S_i] = \frac{\mathbb{E}[V_i]}{X}\,\mathbb{E}[S_i] = \frac{\rho_i}{X}$$

□

The bottleneck law establishes a deep connection between the system throughput and the device with the highest expected demand. Fix any device i. Since $0 \leq \rho \leq 1$, through the bottleneck law we obtain that:

$$X = \frac{\rho_i}{\mathbb{E}[D_i]} \leq \frac{1}{\mathbb{E}[D_i]} \leq \frac{1}{D_{\max}}$$

where $D_{\max} = \max_i \mathbb{E}[D_i]$. This inequality establishes that the throughput of the system is "choked" by the device with the highest expected demand, referred to as the *bottleneck* (hence the name of the law). Through Little's law, we can also obtain a lower bound for the response time. For closed systems, we have that:

$$\mathbb{E}[R] = \frac{N}{X} - \mathbb{E}[Z] \geq ND_{\max} - \mathbb{E}[Z]$$

while for open systems we have that:

$$\mathbb{E}[R] = \frac{\mathbb{E}[N]}{X} \geq ND_{\max}$$

Another lower bound for the response time is given by the case of lowest possible congestion, i.e. when only one job is in the system. Let $R(n)$ denote the response time when n jobs are in the system. By definition, we have that $\mathbb{E}[R] = \mathbb{E}[R(N)]$. When only one job is in the system, we expect the job to pass through every device of the system on average. Thus, we conclude that:

$$\mathbb{E}[R] = \mathbb{E}[R(N)] \geq \mathbb{E}[R(1)] = D_{\text{tot}}$$

where $D_{\text{tot}} = \sum_i \mathbb{E}[D_i]$. Again, through Little's law this can be turned into another higher bound for the throughput.

$$X = \frac{N}{\mathbb{E}[R] + \mathbb{E}[Z]} \leq \frac{N}{D_{\text{tot}} + \mathbb{E}[Z]}$$

Law 1.6: Asymptotic bounds

For any system it holds that:

$$X \leq \min\left(\frac{\mathbb{E}[N]}{D_{\text{tot}} + \mathbb{E}[Z]}, \frac{1}{D_{\max}}\right) \qquad R \geq \max\left(D_{\text{tot}}, \mathbb{E}[N]D_{\max} - \mathbb{E}[Z]\right)$$

Note: the above bounds are correct both for open and closed systems (in the former $\mathbb{E}[Z] = 0$, while in the latter $\mathbb{E}[N] = N$)

We observe that the above bounds are **tight** for small values of N only in closed systems since it is always pushing the throughput to the maximum ($X_i = \mu_i$), while in open systems this is not the case ($X_i = \lambda_i < \mu_i$).

We denote with N^* the average job population where the two metrics switch bound (e.g. X goes from the first bound to the second). For the throughput, this value is given by:

$$\frac{N_X^*}{D_{\text{tot}} + \mathbb{E}[Z]} = \frac{1}{D_{\max}} \implies N_X^* = \frac{D_{\text{tot}} + \mathbb{E}[Z]}{D_{\max}}$$

For the response time, instead, the value is given by:

$$D_{\text{tot}} = N_R^* D_{\max} - \mathbb{E}[Z] \implies N_R^* = \frac{D_{\text{tot}} + \mathbb{E}[Z]}{D_{\max}}$$

thus $N_X^* = N_R^*$. When the number of jobs N is below N^*, we know that both the system throughput and the response time are bounded by the total demand of the system. This case is known as **low population**. When $N > N^*$, instead, they are both bounded by $D_{\max}$. This case is known as **high population**.

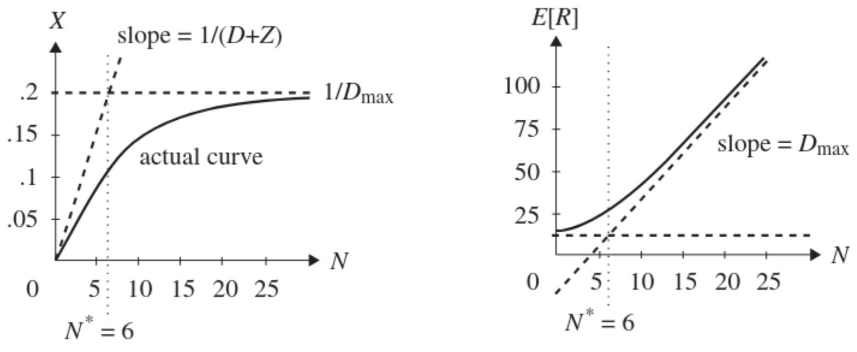


Figure 1.12: Performances of the system as functions of N

From this analysis, we get that reducing the demand of any device in low population will improve both metrics. When in high population, instead, we are forced to reduce the demand of the *bottleneck device* in order to improve both metrics. Through the demand law, we know that the demand of a device is given by the average number of visits it receives and the average service time. Thus, to change the demand we can either change the routing policy of the router (thus changing the number of visits for the device) or change the device itself (thus changing its service time).

For instance, consider the following interactive system connected to three cloud devices: a CPU and two disk called A and B. The known information is the following: $\mathbb{E}[Z] = 18$ secs, $\mathbb{E}[D_{\text{CPU}}] = 5$ secs, $\mathbb{E}[D_A] = 4$ secs, $\mathbb{E}[D_B] = 3$ secs and $\mathbb{E}[N] = 4$.

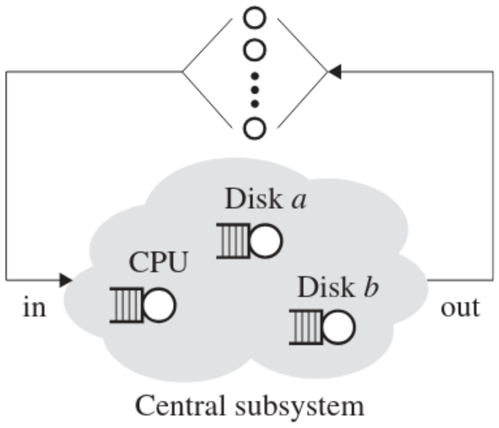


Figure 1.13: An interactive system with three cloud devices.

We're asked to find out the current maximum throughput, minimum response time and if replacing disk A with new disk that is twice as fast would improve such bounds. First, we compute the bounds:

$$X \leq \min\left(\frac{\mathbb{E}[N]}{D_{\text{tot}} + \mathbb{E}[Z]}, \frac{1}{D_{\max}}\right) = \min\left(\frac{N}{30}, \frac{1}{5}\right)$$

$$R \geq \max\left(D_{\text{tot}}, \mathbb{E}[N]D_{\max} - \mathbb{E}[Z]\right) = \max\left(12, 5N - 18\right)$$

then the knee-point of the bounds:

$$N^* = \frac{D_{\text{tot}} + \mathbb{E}[Z]}{D_{\max}} = \frac{12 + 18}{5} = 6$$

concluding that:

$$X \leq \begin{cases} \frac{1}{30}N & \text{if } N < 6 \\ 0.2 & \text{if } N \geq 6 \end{cases} \qquad R \geq \begin{cases} 12 & \text{if } N < 6 \\ 5N - 18 & \text{if } N \geq 6 \end{cases}$$

We observe that $D_{\max} = D_{\text{CPU}} = 5$, hence the CPU is the bottleneck of the system. However, we cannot directly conclude that changing disk A with a faster one wouldn't improve the system. In fact, since $\mathbb{E}[N] < N^*$, we're in low population, meaning that we have to reduce the total demand in order to improve the bounds. Hence, our change to disk A would effectively reduce the system. Since disk A' is twice as fast as disk A, it holds that $\mu_{A'} = 2\mu_A$ and thus that $\mathbb{E}[S_{A'}] = \frac{1}{2}\mathbb{E}[S_A]$.

$$\mathbb{E}[D_{A'}] = \mathbb{E}[S_{A'}]\,\mathbb{E}[V_{A'}] = \frac{1}{2}\mathbb{E}[S_A]\,\mathbb{E}[V_A] = \frac{1}{2}\mathbb{E}[D_A] = 2$$

Thus $D'_{\text{tot}} = 10$, improving the bound for low population:

$$X' \leq \min\left(\frac{\mathbb{E}[N]}{D'_{\text{tot}} + \mathbb{E}[Z]}, \frac{1}{D'_{\text{max}}}\right) = \left(\frac{N}{28}, \frac{1}{5}\right)$$

$$R' \geq \max\left(D'_{\text{tot}}, \mathbb{E}[N]D'_{\text{max}} - \mathbb{E}[Z]\right) = \max\left(10, 5N - 18\right)$$

1.7 Load balancing

Consider an interactive system with a CPU, a fast disk and a slow disk. We don't know the structure of the system, but we know that $\mathbb{E}[Z] = 15$ secs and $N = 20$. At time instant τ, we registered the following data:

- $C(\tau) = 200$
- $C_{\text{CPU}}(\tau) = 200$ and $B_{\text{CPU}}(\tau) = 400$ sec
- $C_{\text{FD}}(\tau) = 20000$ and $B_{\text{CPU}}(\tau) = 600$ sec
- $C_{\text{SD}}(\tau) = 2000$ and $B_{\text{CPU}}(\tau) = 100$ sec

We want to know if replacing the CPU with a faster one would improve the performances of the system. First, we compute the demands of the various devices by using the registered data:

$$D_{\text{CPU}} = \frac{B_{\text{CPU}}(\tau)}{C(\tau)} = \frac{400}{200} = 2 \text{ sec}$$

$$D_{\text{FD}} = \frac{B_{\text{FD}}(\tau)}{C(\tau)} = \frac{600}{200} = 3 \text{ sec}$$

$$D_{\text{SD}} = \frac{B_{\text{SD}}(\tau)}{C(\tau)} = \frac{100}{200} = 0.5 \text{ sec}$$

obtaining $D_{\text{tot}} = 5.5$ sec and $D_{\text{max}} = D_{\text{FD}}$, meaning that the CPU isn't the bottleneck of the system. Hence, changing the CPU with a faster one would improve the performances only if we're in low population (since it would reduce the value of D_{tot}). However, we observe that:

$$N^* = \frac{\mathbb{E}[Z] + D_{\text{tot}}}{D_{\text{max}}} = \frac{15 + 5.5}{3} = \frac{20.5}{3} < 20 = N$$

hence we're in high population, concluding that this change wouldn't improve the system at all. Are there other ways to improve the system without changing the fast disk with a faster one? The answer is yes! In fact, we have two methods to improve the system:

change the bottleneck with a faster device (thus reducing the value of $S_{\max}$) or reduce the number of jobs that reach the devices (thus reducing the value of $V_{\max}$).

Clearly, we cannot force the system to send less jobs to the bottleneck since all the re-routed jobs would still need to be served. The solution here is to **balance the load** of the system, i.e. re-routing jobs from the bottleneck to other devices. For instance, in our example we could try to re-route jobs from the fast disk to the slow disk, preserving the total number of visits.

It's easy to see that this operation not only reduces D_{FD} but also increases D_{SD}. However, if we reduce the demand of the fast disk by too much, the slow disk may become the new bottleneck. Hence, the minimum value of the maximum demand is reached when both devices have the same demand.

With these two constraints, we can model the following system of equations to find out the new values of $\mathbb{E}[V'_{\text{FD}}]$ and $\mathbb{E}[V'_{\text{SD}}]$:

$$\begin{cases} \mathbb{E}[V'_{\text{FD}}] + \mathbb{E}[V'_{\text{SD}}] = \mathbb{E}[V_{\text{FD}}] + \mathbb{E}[V_{\text{SD}}] \\ \mathbb{E}[V'_{\text{FD}}]\,\mathbb{E}[S'_{\text{FD}}] = \mathbb{E}[V'_{\text{SD}}]\,\mathbb{E}[S'_{\text{SD}}] \end{cases} \implies \begin{cases} \mathbb{E}[V'_{\text{FD}}] + \mathbb{E}[V'_{\text{SD}}] = \frac{C_{\text{FD}}(\tau)}{C(\tau)} + \frac{C_{\text{SD}}(\tau)}{C(\tau)} \\ \mathbb{E}[V'_{\text{FD}}]\frac{B_{\text{FD}}(\tau)}{C_{\text{FD}}(\tau)} = \mathbb{E}[V'_{\text{SD}}]\frac{B_{\text{SD}}(\tau)}{C_{\text{SD}}(\tau)} \end{cases}$$

$$\implies \begin{cases} \mathbb{E}[V'_{\text{FD}}] + \mathbb{E}[V'_{\text{SD}}] = 110 \\ \frac{3}{100}\,\mathbb{E}[V'_{\text{FD}}] = \frac{5}{100}\,\mathbb{E}[V'_{\text{SD}}] \end{cases} \implies \begin{cases} \mathbb{E}[V'_{\text{FD}}] = \frac{275}{4} \\ \mathbb{E}[V'_{\text{SD}}] = \frac{165}{4} \end{cases}$$

With these new values, we obtain that:

$$D'_{\text{FD}} = \mathbb{E}[V'_{\text{FD}}]\,\mathbb{E}[S'_{\text{FD}}] = \frac{275}{4}\frac{3}{100} = \frac{33}{16} \approx 2.06$$

$$D'_{\text{SD}} = \mathbb{E}[V'_{\text{SD}}]\,\mathbb{E}[S'_{\text{SD}}] = \frac{165}{4}\frac{5}{100} = \frac{33}{16} \approx 2.06$$

concluding that we improved the performance of the system without spending money to buy a faster disk.

1.8 Solved exercises

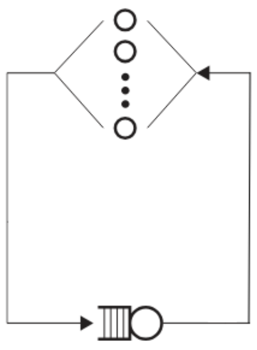
Problem 1.1

Consider a multi-tier system made of an application layer and a database layer. The first layer queries the second one, which then returns the asked data after retrieving it. We know that the system is running 400 simultaneous jobs and that the database layer manages 55 jobs/sec. Moreover, we know that the application layer requires an average think time of 150 ms per job.

- Model the systems and formalize the given parameters.
- Compute the response time of the database layer during peak hours.
- Does the response time improve if the application layer is replaced with one twice as fast? What's the reasoning behind this result?

Solution:

The system can be modeled as the following interactive system, where the terminal center corresponds to the application layer and the device corresponds to the database layer. The system's parameter are $N = 400$, $\mathbb{E}[Z] = 1.5$ msec and $\mu_{\text{DB}} = 55$ job/sec.



During peak hours, we know that the utilization of the the device is maximum, i.e. that $\rho_{\text{DB}} = 1$. Through the utilization law, we get that

$$X_{\text{DB}} = \frac{\rho_{\text{DB}}}{\mathbb{E}[S_{\text{DB}}]} = \mu_{\text{DB}}$$

Moreover, since we have only one device, we know that $X = X_{\text{DB}}$. By applying the response time law, we conclude that:

$$\mathbb{E}[R] = \frac{N}{X} - \mathbb{E}[Z] = \frac{400}{55} - 1510 = \frac{127}{22} \approx 5.77$$

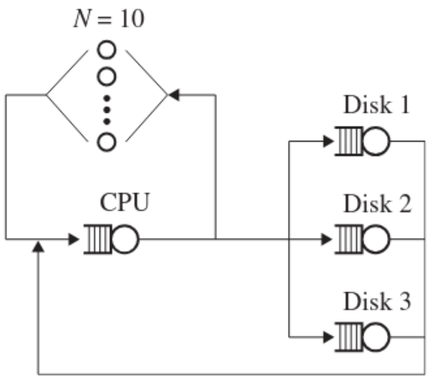
If we make the first layer twice as fast, i.e. $\mathbb{E}[Z'] = \frac{1}{2}\mathbb{E}[Z] = 0.75$ msec, we get that:

$$\mathbb{E}[R'] = \frac{N}{X} - \mathbb{E}[Z'] = \frac{400}{55} - 75100 = \frac{287}{44} \approx 6.52$$

Therefore, this change would worsen the response time. Intuitively, this comes from the fact that a faster application layer implies that jobs are being server more quickly to the database, increasing the average queue size and thus the response time.

Problem 1.2

Given the following system. We know that the throughput of disk 3 is 40 requests/sec, that the service time of an average request at disk 3 is 0.0225 sec and that average number of jobs in disk 3 is 4. Find the utilization and the waiting time of disk 3.



Solution:

Through the utilization law we know that:

$$\rho_3 = \mathbb{E}[S_3]X_3 = \frac{225}{10000}40 = \frac{9}{10}$$

The waiting time can be computed in two ways. The first method is to subtract the service time of disk 3 from its response time, which can be computed through Little's law:

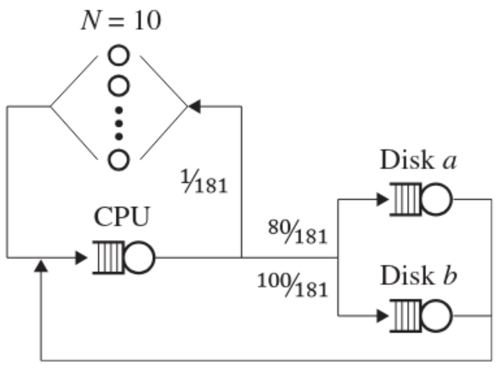
$$\mathbb{E}[W_3] = \mathbb{E}[R_3] - \mathbb{E}[S_3] = \frac{\mathbb{E}[N_3]}{X_3} - \mathbb{E}[S_3] = \frac{4}{40} - \frac{225}{10000} = \frac{1}{400}$$

The second method, instead, is to apply Little's law directly to the queue (as if it were a sub-system):

$$\mathbb{E}[W_3] = \mathbb{E}[R_3^{\text{queue}}] = \frac{\mathbb{E}[N_3^{\text{queue}}]}{X_3^{\text{queue}}} = \frac{\mathbb{E}[N_3] - \mathbb{E}[N_3^{\text{server}}]}{X_3^{\text{queue}}} = \frac{\mathbb{E}[N_3] - \mathbb{E}[4 - 0.9]}{40} = \frac{1}{400}$$

Problem 1.3

Given the following system, compute $\mathbb{E}[V_i]$ for each device.



Solution:

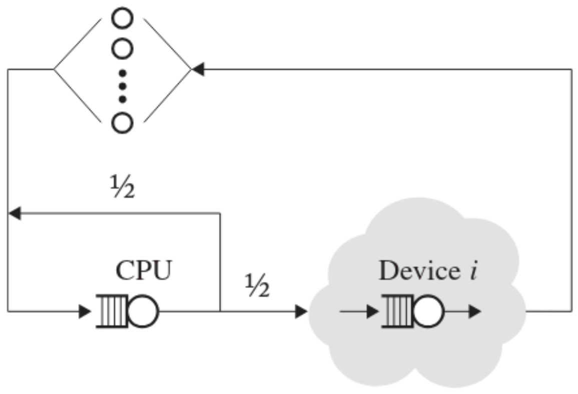
First, we notice that $\mathbb{E}[V_a] = \frac{80}{181}, \mathbb{E}[V_{\text{CPU}}]$ and $\mathbb{E}[V_b] = \frac{100}{181}\mathbb{E}[V_{\text{CPU}}]$. Thus computing $\mathbb{E}[V_{\text{CPU}}]$ gives us all the required values. Now, we observe that each job exiting the CPU can loop back into it by passing through one of the disk. This concludes that:

$$\begin{aligned}
\mathbb{E}[V_{\text{CPU}}] &= \sum_{k=0}^{+\infty} k \left(\frac{80}{181} + \frac{100}{181}\right)^{k-1} \frac{1}{181} \\
&= \sum_{k=0}^{+\infty} k \left(1 - \frac{180}{181}\right)^{k-1} \frac{1}{181} \\
&= 181
\end{aligned}$$

This concludes that $\mathbb{E}[V_{\text{CPU}}] = 181, \mathbb{E}[V_a] = 80$ and $\mathbb{E}[V_b] = 100$.

Problem 1.4

Consider the following interactive system with average think time of 5 seconds and average traversal time of 50 seconds. The CPU is known to be working at 50% capacity, while the device i is working at 30% capacity. Moreover, we know that the device's server processes each job with an average of 0.01 seconds and that it is visited 10 times per visit of the CPU. What is the value of $\mathbb{E}[N_{\text{CPU}}^{\text{queue}}]$ if the device is cloud sub-system processes an average of 20 jobs?



Solution:

First, we formalize the known information by setting $\mathbb{E}[Z] = 5$ sec, $\mathbb{E}[T] = 50$, $\rho_{\text{CPU}} = 0.5$, $\rho_i = 0.3$, $\mathbb{E}[S_i] = 0.01$ sec, $\mathbb{E}[V_i] = 10\,\mathbb{E}[V_{\text{CPU}}]$ and $\mathbb{E}[N_{\text{cloud}}] = 20$. Then, we observe that:

$$\mathbb{E}[N_{\text{CPU}}^{\text{queue}}] = \mathbb{E}[N_{\text{CPU}}] - \mathbb{E}[N_{\text{CPU}}^{\text{server}}] = \mathbb{E}[N_{\text{CPU}}] - \rho_{\text{CPU}} = \mathbb{E}[N_{\text{CPU}}] - \frac{1}{2}$$

Let *CPU-sys* be the sub-system formed by the CPU device and the backwards link. Since $\mathbb{E}[N_{\text{CPU}}] = \mathbb{E}[N_{\text{CPU-sys}}]$ and $X = X_{\text{CPU-sys}}$, by Little's law we have that:

$$\mathbb{E}[N_{\text{CPU}}] = \mathbb{E}[N_{\text{CPU-sys}}] = \mathbb{E}[T_{\text{CPU-sys}}]X_{\text{CPU-sys}} = \mathbb{E}[T_{\text{CPU-sys}}]X$$

Thus we need to compute both $\mathbb{E}[T_{\text{CPU-sys}}]$ and X. By the bottleneck law and the demand law, we know that:

$$X = \frac{\rho_i}{D_i} = \frac{\rho_i}{\mathbb{E}[V_i]\,\mathbb{E}[S_i]} = \frac{0.3}{10\,\mathbb{E}[V_{\text{CPU}}] \cdot 0.01}$$

Since $V_{\text{CPU}} = \text{Geo}(0.5)$, we know that $\mathbb{E}[V_{\text{CPU}}] = 2$, concluding that $X = 1.5$ jobs/sec. To compute $\mathbb{E}[T_{\text{CPU-sys}}]$, we observe that:

$$\mathbb{E}[T_{\text{CPU-sys}}] = \mathbb{E}[T] - \mathbb{E}[Z] - \mathbb{E}[T_{\text{cloud}}] = 50 - 5 - \frac{\mathbb{E}[N_{\text{cloud}}]}{X_{\text{cloud}}} = 45 - \frac{20}{1.5} = \frac{95}{3}$$

Putting everything together, we conclude that:

$$\mathbb{E}[N_{\text{CPU}}^{\text{queue}}] = \mathbb{E}[T_{\text{CPU-sys}}]X - \frac{1}{2} = \frac{95}{3}\frac{15}{10} - \frac{1}{2} = 47$$

Problem 1.5

Consider an interactive system with a CPU and a disk with high utilization. It is known that $N = 50$ and $\mathbb{E}[Z] = 10$ sec. At time instant τ, the following data is registered:

- $C(\tau) = 100$
- $C_{\text{CPU}_1}(\tau) = 300$ and $B_{\text{CPU}_1}(\tau) = 600$ sec
- $C_{\text{disk}_1}(\tau) = 400$ and $B_{\text{CPU}_1}(\tau) = 1200$ sec

To increase the throughput, we're proposed the following approaches:

1. Add a new CPU two times as fast to the system along with the old one and split their load optimally
2. Add a new disk three times as fast to the system along with the old one and split their load optimally

Assuming that the new devices have the same cost, which solution is the best? What is the maximum throughput of the best approach?

Solution:

We start by computing the metrics of the two intial devices:

$$\mathbb{E}[S_{\text{CPU}_1}] = \frac{B_{\text{CPU}_1}(\tau)}{C_{\text{CPU}_1}(\tau)} = \frac{600}{300} = 2 \text{ sec} \qquad \mathbb{E}[S_{\text{disk}_1}] = \frac{B_{\text{disk}_1}(\tau)}{C_{\text{disk}_1}(\tau)} = \frac{1200}{400} = 3 \text{ sec}$$

$$\mathbb{E}[V_{\text{CPU}_1}] = \frac{C_{\text{CPU}_1}(\tau)}{C(\tau)} = \frac{300}{100} = 3 \qquad \mathbb{E}[V_{\text{disk}_1}] = \frac{C_{\text{disk}_1}(\tau)}{C(\tau)} = \frac{400}{100} = 4$$

$$\mathbb{E}[D_{\text{CPU}_1}] = \mathbb{E}[V_{\text{CPU}_1}]\,\mathbb{E}[S_{\text{CPU}_1}] = 6 \qquad \mathbb{E}[D_{\text{disk}_1}] = \mathbb{E}[V_{\text{disk}_1}]\,\mathbb{E}[S_{\text{disk}_1}] = 12$$

Then, we compute the knee-point of the asymptotic bounds:

$$N^* = \frac{D_{\text{tot}} + \mathbb{E}[Z]}{D_{\max}} = \frac{18 + 10}{12} = \frac{7}{3}$$

Since $N^* < N$ and $D_{\max} = \mathbb{E}[D_{\text{disk}_1}]$, we conclude that the first approach wouldn't change since it doesn't improve the demand of the bottleneck. Hence, the second approach is the best one. Since the new disk is three times as fast as the old disk, we know that $\mathbb{E}[S'_{\text{disk}_2}] = \frac{1}{3}\,\mathbb{E}[S_{\text{disk}_1}] = 1$ sec.

To balance the load of the two disks, we solve the following system of equations:

$$\begin{cases} \mathbb{E}[V'_{\text{disk}_1}] + \mathbb{E}[V'_{\text{disk}_2}] = \mathbb{E}[V_{\text{disk}_1}] \\ \mathbb{E}[V'_{\text{disk}_1}]\,\mathbb{E}[S'_{\text{disk}_1}] = \mathbb{E}[V'_{\text{disk}_2}]\,\mathbb{E}[S'_{\text{disk}_2}] \end{cases} \implies \begin{cases} \mathbb{E}[V'_{\text{disk}_1}] = 1 \\ \mathbb{E}[V'_{\text{disk}_2}] = 3 \end{cases}$$

Next, we compute the new demands of the disks:

$$\mathbb{E}[D'_{\text{disk}_1}] = \mathbb{E}[V'_{\text{disk}_1}]\,\mathbb{E}[S'_{\text{disk}_1}] = 3 \qquad \mathbb{E}[D'_{\text{disk}_2}] = \mathbb{E}[V'_{\text{disk}_2}]\,\mathbb{E}[S'_{\text{disk}_2}] = 3$$

concluding that the CPU is the new bottleneck and that:

$$X' \leq \min\left(\frac{N}{D'_{\text{tot}} + \mathbb{E}[Z]}, \frac{1}{D'_{\max}}\right) = \left(\frac{N}{22}, \frac{1}{6}\right)$$

Problem 1.6

In the world of power distribution, one reasonable approximation is that the power that is allocated to a machine is proportional to the speed at which that machine can run. In this problem we assume that, if a machine is allocated power w, then that machine processes jobs at speed w jobs/sec.

Consider a closed batch system with two servers and N users, where N is assumed to be high. Assume that each job, with probability p, is routed to server 1 for processing and, with probability $1-p$, is routed to server 2. You are given a total power budget W , which you need to distribute between the two machines. You can choose any way of dividing the power budget W between the two machines, and you can also choose any value you want for p, the routing probability.

1. What choice for dividing W and for picking p will maximize the throughput of your system?
2. Suppose that N was small. Would your answer still be the same? If so, explain why. If not, derive the optimal strategy.

Solution:

The problem asks us to find the optimal values of μ_1 and p under the constraints $W = \mu_1 + \mu_2$, $V_i = p$ and $V_2 = 1 - p$. We start by rewriting every metric in terms of μ_1:

$$\mathbb{E}[S_1] = \frac{1}{\mu_1} \qquad \mathbb{E}[S_2] = \frac{1}{W - \mu_1}$$

$$\mathbb{E}[D_1] = \mathbb{E}[V_1]\,\mathbb{E}[S_1] = \frac{p}{\mu_1} \qquad \mathbb{E}[D_2] = \mathbb{E}[V_2]\,\mathbb{E}[S_2] = \frac{1-p}{W - \mu_1}$$

When N is high, we know that X is bounded by $X \leq 1/D_{\max}$. Thus, the choice of μ_1 and p must minimize the value of $D_{\max}$. We easily observe that this is achieved when $\mathbb{E}[D_1] = \mathbb{E}[D_2]$.

$$\frac{p}{\mu_1} = \frac{1-p}{W - \mu_1} \implies Wp - \mu_1 p = \mu_1 - \mu_1 p \implies \mu_1 = Wp$$

Therefore, we conclude that μ_1 and p must be such that $\mu_1 = Wp$ (giving us infinite solutions). Moreover, we observe that when this is true we also have that:

$$\mathbb{E}[D_1] = \frac{p}{\mu_1} = \frac{p}{Wp} = \frac{1}{W}$$

$$\mathbb{E}[D_2] = \frac{1-p}{W - \mu_1} = \frac{1-p}{W - Wp} = \frac{1}{W}$$

When N is low, instead, we know that X is bounded by $X \leq N/D_{\text{tot}}$ (notice that $\mathbb{E}[Z] = 0$ since we're in a batch system). Thus, the choice of W and p must minimize the value of D_{tot}. We observe that:

$$D_{\text{tot}} = \mathbb{E}[D_1] + \mathbb{E}[D_2] = \frac{p}{\mu_1} + \frac{1-p}{W-\mu_1}$$

Through some algebraic manipulation, we can rewrite D_{tot} as a linear function of p.

$$D_{\text{tot}} = \left(\frac{1}{\mu_1} - \frac{1}{W-\mu_1}\right) p + \frac{1}{W-\mu_1}$$

Since D_{tot} is a linear function of p, we know that it reaches its minimum value when p is as low as possible or as high as possible, depending on the positivity of the angular coefficient (recall that $0 \leq p \leq 1$ since it is a probability). When $\mu_1 \leq W/2$, the angular coefficient is non-negative, thus the minimum value is reached when $p = 0$, concluding that:

$$D_{\text{tot}_{\text{min}}} = \left(\frac{1}{\mu_1} - \frac{1}{W-\mu_1}\right) \cdot 0 + \frac{1}{W-\mu_1} = \frac{1}{W-\mu_1}$$

which is minimized when $\mu_1 = 0$. When $\mu_1 > W/2$, instead, the angular coefficient is negative, thus the minimum value is reached when $p = 1$, concluding that:

$$D_{\text{tot}_{\text{min}}} = \left(\frac{1}{\mu_1} - \frac{1}{W-\mu_1}\right) \cdot 1 + \frac{1}{W-\mu_1} = \frac{1}{\mu_1}$$

which is minimized when $\mu_1 = W$. Therefore, we conclude that W and p must be such that either $p = 0$ and $\mu_1 = 0$ or when $p = 1$ and $\mu_1 = W$.