10 Heavy Tails: The Distributions of Computing

We have studied several common continuous distributions: the Uniform, the Exponential, and the Normal. However, if we turn to computer science quantities, such as file sizes, job CPU requirements, IP flow times, and so on, we find that none of these are well represented by the continuous distributions that we've studied so far. To understand the type of distributions that come up in computer science, it's useful to start with a story. This chapter is a story of my own experience in studying UNIX jobs in the mid-1990s, as a PhD student at U.C. Berkeley. Results of this research are detailed in [37, 38]. The story serves as both an introduction to empirical measurements of computer workloads and as a case study of how a deeper understanding of computer workloads can inform computer system design. We end with results from 2020 measurements of workloads at Google from [72].

10.1 Tales of Tails

Back in the early 1990s, I was a PhD student happily studying computer science theory. Like many others in the theory area, I had avoided taking my graduate operating systems requirement for as long as possible. When I finally got up the guts to walk into the graduate operating systems class, I looked up at the blackboard (Figure 10.1) and thought, "Hmm ... maybe this isn't going to be so bad."

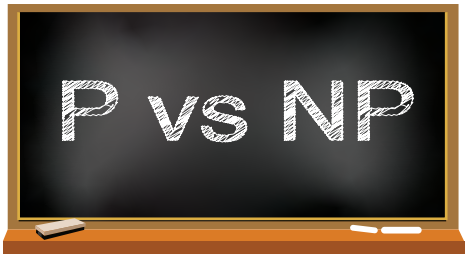


Figure 10.1 *The blackboard in my operating systems class.*

Sadly the professor wasn't referring to complexity theory. Instead, he was referring to migration for the purpose of CPU load balancing in a Network of Workstations – at U.C. Berkeley this project was coined the "N.O.W. project" [4]. The idea in *CPU load balancing* is that CPU-bound jobs (processes) might benefit from being *migrated* from a heavily loaded workstation to a more lightly loaded workstation (Figure 10.2).

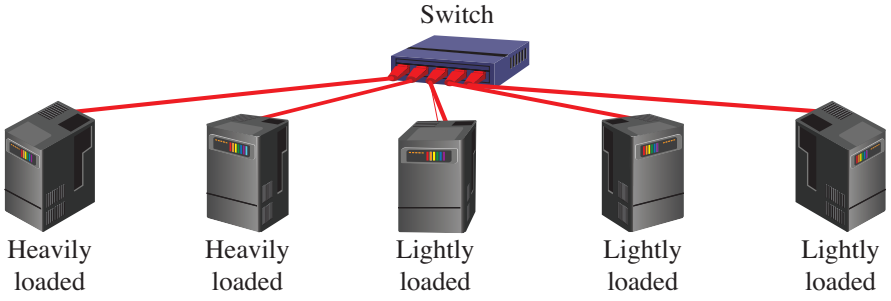


Figure 10.2 *"Network of Workstations." CPU load balancing migrates jobs from heavily loaded workstations to lightly loaded ones.*

CPU load balancing is still important in today's networks of servers. It is not free, however: Migration can be expensive if the job has a lot of "state" that has to be migrated with it (e.g., lots of open files associated with the job), as is common for jobs that have been running for a while. A job that has accrued a lot of state might not be worth migrating.

There are two types of migration used in load balancing techniques:

NP – non-preemptive migration This is migration of newborn jobs only – also called *initial placement* or *remote execution*, where you don't migrate a job once it has started running.

P – preemptive migration This is migration of jobs that are already active (running) – also referred to as *active process migration*.

In the mid-1990s it was generally accepted that migrating active processes was a bad idea, because of their high migration cost. Except for one or two experimental operating systems, like MOSIX [6], people only migrated newborn jobs.

First, some important terminology used in CPU load balancing:

Definition 10.1 *A job's* **size** *(a.k.a.* **lifetime***) refers to the job's total CPU requirement (measured in seconds or CPU cycles). A job's* **age** *refers to its total CPU usage thus far (also measured in seconds or CPU cycles). A job's* **remaining size** *(a.k.a.* **remaining lifetime***) refers to its remaining CPU requirement.*

What we really want to know is a job's remaining lifetime. If the job has a high remaining CPU requirement, then it may pay to migrate the job, even if it has accumulated a lot of state, because the job will get to spend its long remaining lifetime on a lightly loaded machine. Sadly, we do not know a job's remaining lifetime, just its current CPU age.

What we're interested in is the **tail** of the job size, that is, $\mathbf{P}\{\text{Size} > x\}$. More specifically, we want to understand the conditional remaining lifetime given an age a:

$$\mathbf{P}\left\{\text{Size} > x + a \;\middle|\; \text{Size} > a\right\}.$$

Question: Suppose we have two jobs, one with age 2 seconds and the other with age 100 seconds, as in Figure 10.3. Which job is likely to have greater remaining lifetime?

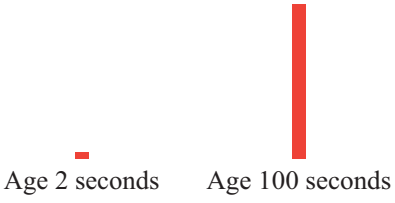


Figure 10.3 *Which job has greater remaining lifetime?*

Answer: We'll find out soon ...

10.2 Increasing versus Decreasing Failure Rate

The obvious question is, then, "How are UNIX job CPU lifetimes distributed?"

The common wisdom at the time, backed up by many research papers, suggested that UNIX job CPU lifetimes were *Exponentially distributed*.

Question: If UNIX job lifetimes are Exponentially distributed, what does that tell us about the question in Figure 10.3?

Answer: Recall from Section 7.1 that if Size is Exponentially distributed, then, by the memoryless property,

$$\mathbf{P}\left\{\text{Size} > x + a \;\middle|\; \text{Size} > a\right\} = \mathbf{P}\left\{\text{Size} > x\right\}.$$

Thus the conditional remaining lifetime is independent of the current age. This says that newborn jobs and older (active) jobs have the *same* expected remaining lifetime. Hence, since newborn jobs are much cheaper to migrate, it makes sense to favor migrating the newborn jobs and ignore the older jobs (NP beats P!).

One can imagine, however, that $\mathbf{P}\left\{\text{Size} > x + a \;\middle|\; \text{Size} > a\right\}$ might *not* be independent of a but rather might either decrease with a or might increase with a.

If $\mathbf{P}\left\{\text{Size} > x + a \;\middle|\; \text{Size} > a\right\}$ decreases with a, we call that **increasing failure rate** or **increasing hazard rate**. This is not a typo! The term "failure rate" refers to the probability that the job *terminates*. So we're saying that the older a job is, the sooner it will terminate, that is, the lower its probability of running an additional x seconds. Likewise, if $\mathbf{P}\left\{\text{Size} > x + a \;\middle|\; \text{Size} > a\right\}$ increases with a, we say that the Size has **decreasing failure rate** or **decreasing hazard rate**.

Colloquially, increasing failure rate says, "the older you are, the sooner you'll die," while decreasing failure rate says "the older you are, the longer you'll live."

Question: What are some real-world examples of random variables with increasing failure rate?

Answer: Here are a few:

- the lifetime of a car;
- the lifetime of a washing machine;
- the lifetime of a person.

Actually, almost anything you think of will have increasing failure rate. Aging leads to failing (ending) sooner.

Question: What are some real-world examples of random variables with decreasing failure rate?

Answer: This is a lot harder to think about because we're looking for an example where older is better in the sense of lasting longer. Here are some examples:

- The lifetime of a friendship. Generally, the longer you've been friends with someone, the longer you're likely to continue to be friends.
- The time you've lived in your home. If you've lived in your home for many years, you're more likely to continue to stay there.

To make the concept of failure rate more precise, we define the failure rate function.

Definition 10.2 *Given a continuous random variable (r.v.) X with probability density function (p.d.f.) $f_X(t)$ and tail $\overline{F}_X(t) = \mathbf{P}\{X > t\}$, the* **failure rate function***, $r_X(t)$, for X is:*

$$r_X(t) \equiv \frac{f_X(t)}{\overline{F}_X(t)}.$$

Question: $r_X(t)$ looks like a conditional density function. What is that density?

Answer: If we write $\overline{F}_X(t) = \mathbf{P}\{X > t\} = \mathbf{P}\{X \geq t\}$, then we can see that:

$$r_X(t) = \frac{f_X(t)}{\overline{F}_X(t)} = f_{X|X \geq t}(t).$$

This is the density that $X = t$ given that $X \geq t$.

To further interpret $r_X(t)$, consider the probability that a t-year-old item will fail during the next dt seconds:

$$\begin{aligned}\mathbf{P}\{X \in (t, t+dt) \mid X > t\} &= \frac{\mathbf{P}\{X \in (t, t+dt)\}}{\mathbf{P}\{X > t\}} \\ &\approx \frac{f_X(t) \cdot dt}{\overline{F}_X(t)} \\ &= r_X(t) \cdot dt.\end{aligned}$$

Thus, $r_X(t)$ represents the *instantaneous failure rate* of a t-year-old item, whose lifetime distribution is X.

Definition 10.3 *If $r_X(t)$ is strictly decreasing in t, we say that X has* **decreasing failure rate***; if $r_X(t)$ is strictly increasing in t, we say that X has* **increasing failure rate***.*

In general, $r_X(t)$ is not necessarily going to always decrease with t or increase with t; it's common that $r_X(t)$ is decreasing for some t and increasing for others.

Question: Suppose $r_X(t)$ is constant. What do you know about X?

Answer: In Exercise 10.2, we prove that X *must* be Exponentially distributed.

Before we leave our discussion of the Exponential distribution, let's recall the notion of the squared coefficient of variation of a r.v. X. By Definition 5.6, this

is

$$C_X^2 = \frac{\mathbf{Var}(X)}{\mathbf{E}[X]^2}$$

and represents the normalized variance. It is the metric of choice for systems measurements because it is scale invariant.

Question: What is C_X^2, when $X \sim \text{Exp}(\lambda)$?

Answer: 1.

10.3 UNIX Process Lifetime Measurements

If UNIX process lifetimes (sizes) are Exponentially distributed, then there is no benefit to active process migration: all jobs have the remaining lifetime distribution, regardless of their age.

Refusing to believe that there were no benefits to active process migration, I decided to measure the distribution of job lifetimes. I collected the CPU lifetimes of millions of UNIX jobs on a wide range of different machines, including instructional, research, and administrative machines, over the course of many months, including only jobs whose size exceeded 1 second. Figure 10.4 shows the tail of my measured distribution.

At first glance Figure 10.4 looks like an Exponential distribution,

$$\overline{F}_{\text{Size}}(x) = e^{-\lambda x}.$$

But on closer examination you can see that it's not Exponential.

Question: How can you tell that job sizes are *not* Exponentially distributed?

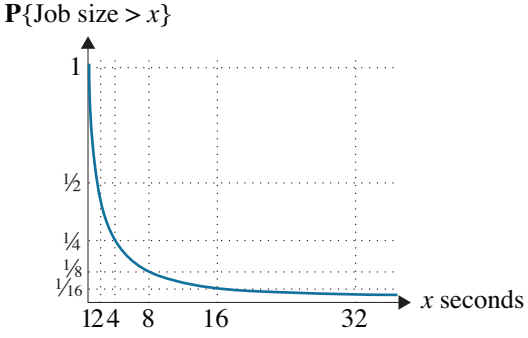


Figure 10.4 *Plot of measured distribution,* $\overline{F}_X(x) = \mathbf{P}\{\textit{Job size} > x\}$, *where* $x \geq 1$.

Answer: For an Exponential distribution, the fraction of jobs remaining should drop by a constant factor with each unit increase in x (constant failure rate). In Figure 10.4, we see that the fraction of jobs remaining decreases by a slower and slower rate as we increase x (decreasing failure rate). In fact, looking at the graph, we see that if we start with jobs of CPU age 1 second, half of them make it to 2 seconds. Of those that make it to 2 seconds, half of those make it to 4 seconds. Of those that make it to 4 seconds, half of those make it to 8 seconds, and so on.

To see the distribution more easily it helps to view it on a log-log plot, as shown in Figure 10.5. The bumpy line shows the data, and the straight line is the best curve-fit. From Figure 10.5 it is apparent that the tail of the distribution of job lifetimes decays like $\frac{1}{x}$. That is, the distribution is well approximated by

$$\mathbf{P}\{\text{Size} > x\} = \frac{1}{x}, \qquad x \geq 1.$$

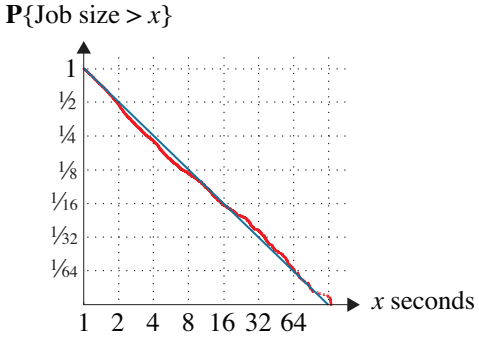


Figure 10.5 *Log-log plot of measured distribution,* $\overline{F}_X(x) = \mathbf{P}\{\textit{Job size} > x\}$, $x \geq 1$.

10.4 Properties of the Pareto Distribution

It turns out that the distribution that I had measured has a name in economic theory. It is called the Pareto distribution, or "power-law distribution," and is named after Vilfredo Pareto, who was an economist in the early 1900s.

Definition 10.4 *We say that X follows a* **Pareto distribution** *with parameter α, written $X \sim Pareto(\alpha)$, if*

$$\overline{F}_X(x) = \mathbf{P}\{X > x\} = x^{-\alpha}, \quad \textit{for } x \geq 1,$$

where $0 < \alpha < 2$.

Question: So job sizes are distributed as Pareto($\alpha = 1$). What does this say about $\mathbf{E}[\text{Size}]$? Also, does the job size distribution exhibit increasing failure rate, or decreasing failure rate, or neither?

Answer:

It's easy to see that $\mathbf{E}[\text{Size}] = \infty$ and that the failure rate is decreasing. We derive this below for general $0 < \alpha < 2$.

Let $X \sim \text{Pareto}(\alpha)$. Then:

$$\overline{F}_X(x) = \mathbf{P}\{X > x\} = x^{-\alpha}, \qquad x \geq 1$$

$$\Rightarrow F_X(x) = \mathbf{P}\{X < x\} = 1 - x^{-\alpha}, \qquad x \geq 1$$

$$\Rightarrow f_X(x) = \frac{dF_X(x)}{dx} = \alpha x^{-\alpha-1}, \qquad x \geq 1$$

$$\Rightarrow r_X(x) = \frac{f_X(x)}{\overline{F}_X(x)} = \frac{\alpha x^{-\alpha-1}}{x^{-\alpha}} = \frac{\alpha}{x}, \qquad x \geq 1.$$

Because $r_X(x) = \frac{\alpha}{x}$ decreases with x, the Pareto distribution has decreasing failure rate (DFR). Thus the older a job is (the more CPU it has used up so far), the greater its probability of using another second of CPU.

The Pareto($\alpha = 1$) distribution has an interesting *doubling property*.

Question: Given that Job size $\sim$ Pareto($\alpha = 1$), what is the probability that a job of age $t > 1$ survives to age $\geq 2t$?

Answer:

$$\mathbf{P}\{\text{Size} > 2t \mid \text{Size} \geq t\} = \frac{\frac{1}{2t}}{\frac{1}{t}} = \frac{1}{2}.$$

Question: For $X \sim \text{Pareto}(\alpha)$, with $0 < \alpha \leq 1$, what are the moments of X?

Answer: The calculations are straightforward, by integration over the density function. It is easy to see that all moments are infinite.

Question: For $X \sim \text{Pareto}(\alpha)$, with $1 < \alpha < 2$, what are the moments of X?

Answer: The mean of X is now finite. Higher moments are still infinite.

But something doesn't seem right here. How can our distribution of job sizes have infinite mean? Although the data fits a Pareto($\alpha = 1$) distribution very

well, the moments of job size are still finite. To see this we need to introduce the Bounded-Pareto distribution.

10.5 The Bounded-Pareto Distribution

When fitting a curve to *measured (empirical) data*, the data has a *minimum* job lifetime, k, and a *maximum* job lifetime, p. In particular, the measured data has *finite* moments, not infinite ones. To model the empirical data, we therefore want a distribution with a Pareto shape, but that has been truncated between k and p. We refer to such a distribution as a *Bounded-Pareto* distribution.

Definition 10.5 *The* **Bounded-Pareto**(k, p, α) *distribution has density function*

$$f(x) = \alpha x^{-\alpha-1} \cdot \frac{k^\alpha}{1 - \left(\frac{k}{p}\right)^\alpha},$$

for $k \leq x \leq p$ *and* $0 < \alpha < 2$.

The factor $\frac{k^\alpha}{1-(k/p)^\alpha}$ in Definition 10.5 is a normalization factor needed to make the integral of the density function between k and p come out to 1. For the Bounded-Pareto distribution, obviously all of the moments are finite.

For the UNIX job sizes that I measured, the squared coefficient of variation, C^2, was finite, ranging between $C^2 = 25$ and $C^2 = 49$, which was considered extremely high in the 1990s.

10.6 Heavy Tails

The following are three properties of the Pareto distribution:

1. **Decreasing failure rate (DFR)** – The more CPU you have used so far, the more you will continue to use.
2. **Infinite variance**
3. **"Heavy-tail property"** – A minuscule fraction of the very largest jobs comprise 50% of the total system load. (Note that this is much more biased than the often quoted 80–20 rule.)

The "heavy-tail property" comes up in many other settings. For example, in economics, when studying people's wealth, it turns out that the richest 1% of all people have more money between them than all the remaining 99% of us combined. The heavy-tailed property is often referred to as "a few big elephants (big jobs) and many, many mice (little jobs)," as illustrated in Figure 10.6. For comparison, in an Exponential distribution, the largest 1% of the jobs comprise only about 5% of the total demand.

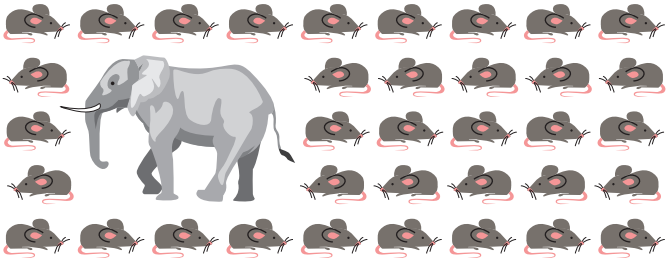


Figure 10.6 *Heavy-tailed property: "Elephants and mice."*

The parameter α can be interpreted as a measure of the variability of the distribution and the heavy-tailedness: $\alpha \to 0$ yields the most variable and most heavy-tailed distribution, whereas $\alpha \to 2$ yields the least variable, and least heavy-tailed distribution. These properties are explored in more depth in the exercises.

These properties largely hold for the Bounded-Pareto distribution as well as the Pareto, although clearly the Bounded-Pareto has finite moments. Also the Bounded-Pareto cannot have strict DFR because there is an upper bound on job size.

10.7 The Benefits of Active Process Migration

Let's return to the original question of CPU load balancing.

Question: What does the DFR property of the Pareto distribution tell us about whether it pays to migrate older jobs?

Answer: DFR says that the older jobs have higher expected remaining lifetimes. This leads us to think that it may pay to migrate *older* jobs. Although an old job may have a high migration cost because it has accumulated a lot of state (memory), if the job is really old then it has a high probability of using a lot more CPU in the future. This means that the cost of migration can be amortized over

a very long lifetime, as the job gets to spend its long remaining lifetime running on a lightly loaded machine.

Question: What does the heavy-tail property of the Pareto distribution tell us?

Answer: By the heavy-tail property, it is only necessary to migrate the 1% biggest jobs, because they contain most of the work [38].

10.8 From the 1990s to the 2020s

At this point you might be wondering whether these Pareto distributions still apply to jobs today. To answer this, we look at the jobs scheduled by the Borg scheduler [73], which serves jobs in Google data centers.

Question: How do you imagine that jobs look different today than they did in the 1990s?

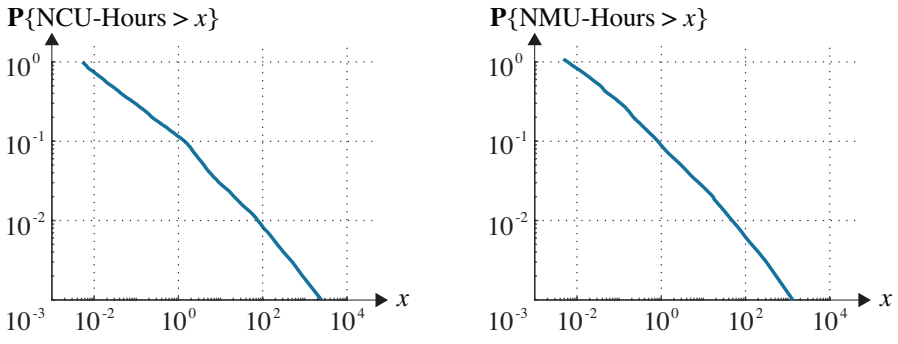
Answer: There are many differences, but an important one is that back in the 1990s a job ran on a single CPU. The job's *size* was the time it needed on a single CPU. By contrast, the Google jobs today are all parallel jobs. We can think of a job as holding onto a certain number of processors (CPUs) for an amount of time. The **size of a job** is then measured in CPU-hours (number of CPUs occupied times the number of hours).

Jobs today also often utilize a lot of memory (think about machine learning jobs). We can also view the **size of a job** as measured in memory-unit-hours (number of memory units times hours held).

Question: If you had to guess, would you guess that the distribution of compute usage today is more variable or less variable than in the 1990s? Would you guess that the distribution is more heavy-tailed or less heavy-tailed than in the 1990s?

Answer: The answer to both is "more," but the degree to which the answer is "more" is quite shocking.

Figure 10.7(a) shows the distribution of compute usage, and Figure 10.7(b) shows the distribution of memory usage [72]. Because Google doesn't like to reveal exact numbers, it uses normalized units in expressing compute and memory usage. Thus, per-job compute usage is expressed in units of NCU-hours (normalized CPU times hours) and per-job memory usage is expressed in units of NMU-hours (normalized memory units times hours). Note that a 100 NCU-hour job might have consumed 100 machines for 1 hour, or 5 machines for 20 hours, or various other combinations.



(a) Per-job NCU-hours (b) Per-job NMU-hours

Figure 10.7 *Tail of resource usage based on a trace of millions of jobs run at Google in May 2019 [72, 77]. NCU-hours denotes normalized CPU-hours used. NMU-hours denotes normalized memory-unit-hours used.*

The distribution for compute usage at Google's data centers fits a Pareto($\alpha = 0.69$) distribution, which is much more heavy-tailed than what we saw in the 1990s measurements. We find that, while the mean NCU-hours used per job is about 1.2, the variance is 33,300, which means that the squared coefficient of variation is

$$C^2 = \frac{\text{variance}}{\text{mean}^2} = 23{,}000,$$

which is huge! The heavy-tailed property is also much more extreme than what we saw in the 1990s: The largest (most compute-intensive) 1% of jobs comprise about 99% of the compute load.

Memory usage follows much the same patterns as compute usage, obeying a Pareto($\alpha = 0.72$) distribution with astronomical variability: $C^2 \approx 43{,}000$. Again we see an extremely strong heavy-tailed property, with the top 1% of jobs comprising 99% of the total memory usage. Memory and compute usage are also correlated.

10.9 Pareto Distributions Are Everywhere

It is not just computing jobs that fit a heavy-tailed Pareto distribution. Pareto job size distributions are everywhere in computer science and in nature! Here are some more practical and interesting stories:

Web file size: Around 1996–1998, Mark Crovella, Azer Bestavros, and Paul Barford at Boston University were measuring the sizes of files on websites. They

found that these file sizes obeyed a Pareto distribution with $\alpha \approx 1.1$. They also found similar results for the sizes of files requested from websites. Their SURGE web workload generator is based on these findings [7, 18, 19].

Internet node degrees: Around the same time, the three Faloutsos brothers were observing a similar distribution when looking at the Internet topology. They observed, for example, that most nodes have low out-degree, but a very few nodes have very high out-degree, and the distribution of the degrees follows a Pareto distribution. Their beautiful 1999 paper won the Sigcomm Test of Time award [25].

IP flow durations: In 1999, Jennifer Rexford, Anees Shaikh, and Kang Shin at AT&T were working on routing IP flows to create better load balancing. Their goal was to reroute only 1% of the IP flows. Would that be enough? Fortunately, their measurements showed that the number of packets in IP flows follows a heavy-tailed Pareto distribution. Consequently, the 1% largest IP flows (those with the most packets) contain about 50% of the bytes in all flows. By rerouting only 1% of the flows, they were able to redistribute half the load. Their paper appeared in Sigcomm 99 [69] and generated a large group of follow-up papers dealing with sampling methods for how to detect which flows are large, based on using the DFR property and the knowledge of how many packets the flow has sent so far.

Implications for designing scheduling policies: Around this same time, my students and I, in collaboration with Mark Crovella at Boston University, started a project called SYNC (Scheduling Your Network Connections). The goal was to improve the performance of web servers by changing the order in which they scheduled their jobs to favor requests for small files over requests for large files. Clearly favoring requests for small files over large ones would decrease mean response time. However, people had not tried this in the past because they were afraid that the requests for large files would "starve" or at least be treated unfairly compared to requests for small files. Using the heavy-tailed property of web file sizes, we were able to prove analytically and in implementation that this fear is unfounded for the distribution of web files. The crux of the argument is that, although short requests do go ahead of long requests, all those short requests together make up very little load (more than half the load is in the top 1% of long requests) and hence do not interfere noticeably with the long requests [5, 17, 39]. In 2004, Ernst Biersack, Idris Rai, and Guillaume Urvoy-Keller extended the SYNC results to TCP flow scheduling by exploiting the DFR property of the Pareto distribution to discern which flows have short remaining duration [58, 59].

Wireless session times, phone call durations, wealth, natural disasters: There are many, many more examples of the Pareto distribution in measured

distributions involving jobs created by humans. Wireless session times have been shown to follow a Pareto distribution [8]. Phone call durations have been shown to follow a distribution similar to a Pareto. Human wealth follows a Pareto distribution. Natural phenomena too follow Pareto distributions. For example, John Doyle at Caltech has shown that the damage caused by forest fires follows a Pareto distribution, with most forest fires causing little damage, but the largest few forest fires causing the majority of the damage. The same property holds for earthquakes and other natural disasters.

Given the prevalence of the Pareto distribution, there has been a great deal of research interest in **why** the Pareto distribution comes up everywhere. Ideally, we would like to prove something similar in nature to the Central Limit Theorem (CLT), which explains the ubiquity of the Normal distribution, but this time for the Pareto distribution. If you recall, CLT assumed that we are taking the average of many i.i.d. random variables, each with *finite* variance. Suppose that we're taking the average of i.i.d. random variables, where these have infinite variance. Does that lead to a different distribution than a Normal? Does it lead to a Pareto? If you are interested in this question, and, more generally in the question of why the Pareto distribution comes up, I recommend a book, *The Fundamentals of Heavy Tails* [55].

10.10 Summary Table for Continuous Distributions

At this point, we have seen several continuous distributions. Just as we summarized the mean and variance of our discrete distributions in Table 5.1, it is worth taking the time to do the same for the continuous distributions. Table 10.1 summarizes the common continuous distributions.

10.11 Exercises

10.1 **How variable is a Uniform distribution really?**
The Uniform distribution feels highly variable, particularly when its endpoints are far apart. Consider $X \sim \text{Uniform}(0, b)$, and assume that b is large. What is C_X^2 as a function of b? Do you still think the Uniform is highly variable?

10.2 **Failure rate**
Let X be a continuous random variable with p.d.f. $f_X(t), t \geq 0$ and c.d.f.

Distribution	p.d.f. $f_X(x)$	Mean	Variance
Exp(λ)	$f_X(x) = \lambda e^{-\lambda x}, x \geq 0$	$\frac{1}{\lambda}$	$\frac{1}{\lambda^2}$
Uniform(a, b)	$f_X(x) = \frac{1}{b-a}$, if $a \leq x \leq b$	$\frac{b+a}{2}$	$\frac{(b-a)^2}{12}$
Pareto(α), $0 < \alpha < 2$	$f_X(x) = \alpha x^{-\alpha-1}$, if $x > 1$	$\begin{cases} \infty & \text{if } \alpha \leq 1 \\ \frac{\alpha}{\alpha-1} & \text{if } \alpha > 1 \end{cases}$	∞
Normal(μ, σ^2)	$f_X(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$, $-\infty < x < \infty$	μ	σ^2

Table 10.1 *Common continuous distributions.*

$F_X(t) = \mathbf{P}\{X < t\}$. We define the failure rate of X to be $r_X(t)$, where

$$r_X(t) \equiv \frac{f_X(t)}{\overline{F}_X(t)}.$$

Thus, $r_X(t)dt$ represents the probability that a t-year-old item will fail in the next dt seconds.

(a) Prove that for the Exponential distribution the failure rate is a constant.

(b) Prove that the Exponential distribution is the only non-negative distribution with constant failure rate.

10.3 **Modeling distributions with low variability: the Erlang-*k***

The Erlang-k distribution is often used to model distributions, X, where $0 < C_X^2 < 1$. An Erlang-k distribution is a sum of k Exponentially distributed "stages." Formally, we say that $X \sim$ Erlang-$k(\mu)$ if

$$X = X_1 + X_2 + \cdots + X_k,$$

where the X_i's are i.i.d., with $X_i \sim \text{Exp}(k\mu)$.

(a) What is $\mathbf{E}[X]$?

(b) What is $\mathbf{Var}(X)$?

(c) What is C_X^2?

(d) What happens to X as $k \to \infty$?

10.4 **Hyperexponential distribution and DFR**

We say that X follows a two-phase **Hyperexponential** distribution (written H_2) if:

$$X \sim \begin{cases} \text{Exp}(\mu_1) & \text{w/prob } p \\ \text{Exp}(\mu_2) & \text{w/prob } 1-p \end{cases},$$

where $\mu_1 \neq \mu_2$.

(a) Prove that the Hyperexponential distribution has DFR. [Hint: Take the derivative of the failure rate.]

(b) Explain intuitively why the Hyperexponential has DFR.

10.5 **Squared coefficient of variation for the Hyperexponential**

Consider three different distributions:

(i.) $X \sim \text{Exp}(\mu = 1)$

(ii.) $X \sim \text{Exp}(\mu = .01)$

(iii.)

$$X \sim \begin{cases} \text{Exp}\,(1) & \text{w/prob. } 0.99 \\ \text{Exp}(\mu = 0.01) & \text{w/prob. } 0.01 \end{cases}.$$

For each distribution:

(a) What is $\mathbf{E}[X]$?

(b) What is $\mathbf{Var}(X)$?

(c) What is C_X^2?

10.6 **Why the Hyperexponential is good for modeling high variability**

The Hyperexponential is good at modeling high-variability distributions. To gain some intuition for why this is true, let us analyze the simple case of a *Degenerate Hyperexponential* distribution, where one of the phases is identically zero:

$$X \sim \begin{cases} \text{Exp}\,(p\mu) & \text{w/prob } p \\ 0 & \text{w/prob } 1-p \end{cases}.$$

(a) What is $\mathbf{E}[X]$?

(b) What is C_X^2?

(c) What values of C_X^2 are possible?

10.7 **Bounded-Pareto with negative parameter**

A Pareto(α) distribution is defined with $0 < \alpha < 2$. But what happens if you set $\alpha = -1$? Let $X \sim \text{BoundedPareto}(k, p, \alpha)$, where $\alpha = -1$. What is the density function $f_X(x)$? What does this tell you about the distribution of X?

10.8 **The heavy-tail property**

We explore three distributions for job size, all with mean $3,000$:

(a) Exponential distribution with rate $\frac{1}{3,000}$.

(b) BoundedPareto($k = 0.0009, p = 10^{10}, \alpha = 0.5$).

(c) BoundedPareto($k = 332.067, p = 10^{10}, \alpha = 1.1$).

In each case, compute the fraction of load, q, made up by just the top (largest) 1% of all jobs. For a non-negative job size distribution, X, with density $f_X(\cdot)$,

$$q = \frac{\int_{[t \text{ in top } 1\%]} t f_X(t) dt}{\int_0^\infty t f_X(t) dt}.$$

Also report the size cutoff, x, defining the top 1% of jobs. It may help to use a symbolic math package to do this calculation.