04/04/2023

Performance Modeling of Computer Systems and Networks

Prof. Vittoria de Nitto Personè

Memoryless property
and
probability distributions

Università degli studi di Roma Tor Vergata
Department of Civil Engineering and Computer Science Engineering

1

Vediamo la memoryless non più come tempo di servizio rimanente, bensì come tempo di vita.

Memoryless property as lifetime

Probabilità che X duri più di 's+t' dato che è già durata 's'?
E' uguale alla probabilità che dal tempo 0 duri più di t. (E' come se traslassi il tempo passato a 0).

A random variable X is said to be memoryless if

$$Prob\{X > s+t \mid X > s\} = Prob\{X > t\} \quad \forall s,t > 0$$

Example
X is the lifetime of a lightbulb.
The property says that the probability that the lightbulb survives for at least another t seconds before burning out, given that the lightbulb has survived for s seconds already, is the same as the probability that the lightbulb survives at least t seconds independent of s.

Does this seem realistic for a lightbulb???

2

Lifetime and failure rate, esistono due modelli probabilistici:

Se probabilità di vita decresce, allora cresce probabilità di fallimento.
Distributions for which $Prob\{X > s+t \mid X > s\}$ goes down as s goes up are said to have *increasing failure rate*.
The device is more and more likely to fail as time goes on.

Example
A car's lifetime. The older a car is the less likely that it will survive another $t = 6$ years.

Se probabilità di vita cresce, allora decresce probabilità di fallimento.
Distributions for which $Prob\{X > s+t \mid X > s\}$ goes up as s goes up are said to have *decreasing failure rate*.
The device is less likely to fail as time goes on.

Example
- UNIX job CPU lifetimes. The more CPU a job has used up so far, the more it is likely to use up.
- The same for computer chips. If they're going to fail, they'll do it early. That's why chip manufacturers test them for a long while.

Tutto ciò ci porta a definire la frequenza istantanea di fallimento:

Hazard rate function

Let X be a continuous random variable with probability density function $f(t)$ and cumulative distribution function $F(t) = Pr\{X < t\}$.
Then $r(t)$ is formally defined as:

$$r(t) \equiv \frac{f(t)}{\overline{F}(t)}$$

where $\overline{F}(t) = 1 - F(t) = Pr\{X > t\}$

Consider the probability that a t-year old item will fail during the next dt seconds:

$$Pr\{X \in (t, t+dt) \mid X > t\} = \frac{Pr\{X \in (t, t+dt)\}}{Pr\{X > t\}} \approx \frac{f(t)dt}{\overline{F}(t)} = r(t)dt$$

the istantaneous failure rate
poichè c'è "dt" è nel continuo, quindi un valore istantaneo

Hazard rate function

If $r(t)$ is constant then $f(t)$ must be exponential
Indeed for the exponential

$$r(t) = \frac{f(t)}{F(t)} = \frac{\lambda \cancel{e^{-\lambda t}}}{\cancel{e^{-\lambda t}}} = \lambda$$

l'unica memoryless continua è l'esponenziale

We use the failure rate concept when we study scheduling

Why the remaining lifetime is so important?

CPU load balancing in a Network of Workstations

- It may help to migrate a job to a less-loaded workstation (one with fewer jobs) in order to improve mean response times
- migration can be expensive if the job has a lot of "state" that has to be migrated with the job (lots of memory).

Va bene migrare job verso un server meno carico, ma rendo tutto vano se ci sposto job che terminano appena dopo averli spostati. Devo pesare bene i job che sposto.

two types of migration used in load balancing techniques:

- non-preemptive migration (NP) — only relocates "newborn" processes (initial placement, or remote execution)
- preemptive migration (P) — migrates processes that are already active (running) (active process migration)

1. Should we bother with P migration, or is NP enough?
2. If we are going to bother with P migration, which processes are worth migrating?
   That is, what is a good migration policy?

terminology:

a job's "size": its total CPU demand
a job's "age": its total CPU usage thus far
a job's "lifetime" refers to its total CPU requirement (come la size)
a job's "remaining lifetime" refers to its remaining CPU requirement (ciò che rimane)

Observe that commonly, at any point in time, you don't know the job's remaining lifetime, just it's current CPU age.

Measurements of CPU requirements of Unix jobs

jobs lifetime for 3 months, only those jobs with CPU lifetimes greater than one second

$\overline{F}(x) = \Pr\{X > x\}$ probabilità che tempo di vita > x

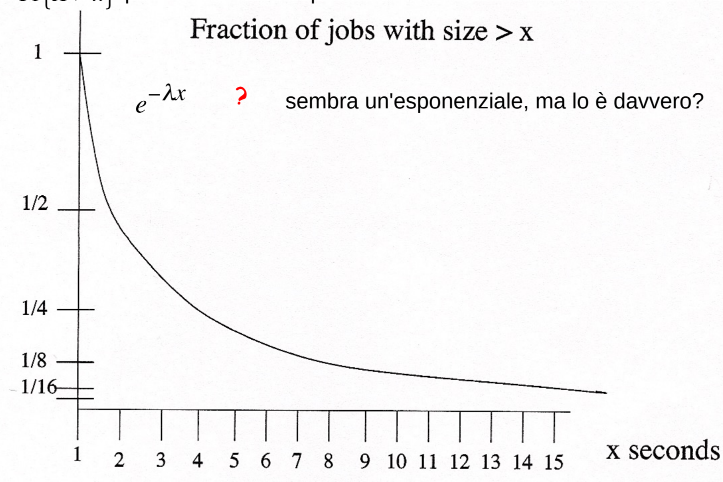
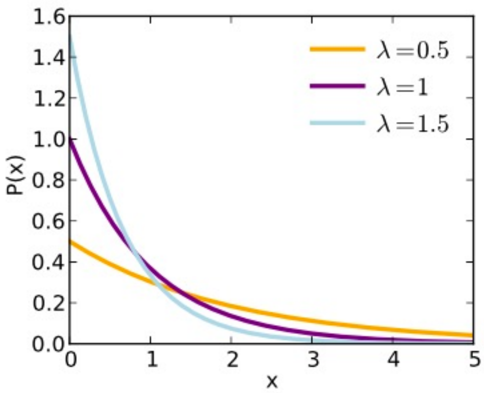


Figure 1: *Plot of measured distribution on non-log axes.*

sembra appartenere a questa famiglia di funzioni



$$f(x) = \lambda e^{-\lambda x}$$

1. For each unit step, the "quantity" is reduced of $e^{-\lambda}$

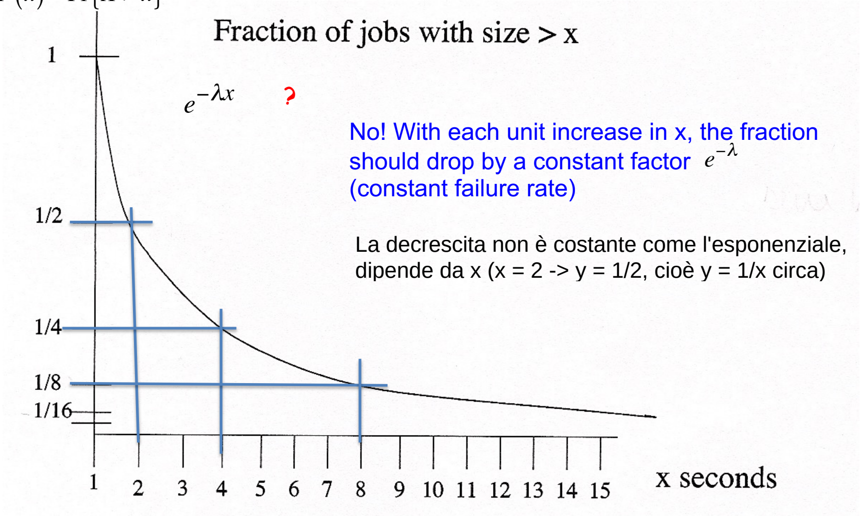
λ is the *decay constant*

9

Measurements of CPU requirements of Unix jobs

jobs lifetime for 3 months, only those jobs with CPU lifetimes greater than one second

$\overline{F}(x) = \Pr\{X > x\}$



No! With each unit increase in x, the fraction should drop by a constant factor $e^{-\lambda}$ (constant failure rate)

La decrescita non è costante come l'esponenziale, dipende da x (x = 2 -> y = 1/2, cioè y = 1/x circa)

Figure 1: *Plot of measured distribution on non-log axes.*

10

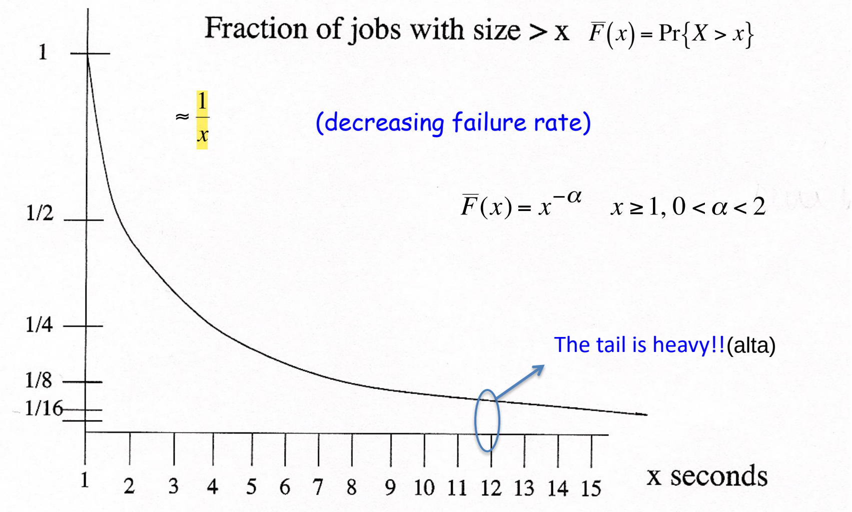


Figure 1: *Plot of measured distribution on non-log axes.*

11

Pareto distributions

in realtà, anche se formalmente x < infinito, non lo userò mai.

$$f(x) = \alpha k^{\alpha} x^{-\alpha-1} \quad k \leq x < \infty, 0 < \alpha < 2$$

α a measure of the distribution variability and of the "heavy-tailedness":

$\alpha \to 0$ +variability, +heavy

$\alpha \to 2$ -variability, -heavy

Problem: i-th moment is finite just for $\alpha > i$

$$E[X] = \frac{\alpha k}{\alpha - 1} \quad \alpha > 1$$

$$var[X] = \frac{\alpha k^2}{(\alpha-1)^2(\alpha-2)} \quad \alpha > 2$$

alfa deve essere maggiore del momento che voglio calcolare. Se voglio momento di ordine 2, alfa deve essere 2, ma cosi perdo la coda spessa.

12

Properties of Pareto distributions

Decreasing Failure Rate

The more cpu you have used so far, the more you will continue to use.

completely different from the exponential distribution, where your cpu usage after any point in time is completely independent of the amount of cpu used up to that point (memoryless property)

Infinite Variance

"Heavy-Tail Property"

A miniscule fraction of the very largest jobs comprise half of the load on the system.

For example, when $\alpha = 1.1$, the largest 1% of the jobs comprise 1/2 of the load.

Una piccola frazione dei job grandi produce una grande quantità di carico. Questo è importante per le politiche di migrazione, perchè basterebbe migrare questi pochi job grandi per alleggerire un server molto carico.

Bounded Pareto distributions

the measured data has a *minimum* job lifetime and a *maximum* job lifetime.

Thus the measured data has all finite moments.
To model the measured data, we therefore want a Pareto distribution which has been truncated.

sottoclasse della Pareto di prima, qui x è molto più limitato.

$$f(x) = \alpha x^{-\alpha-1} \frac{k^\alpha}{1-\left(\frac{k}{p}\right)^\alpha} \quad k \le x \le p, \; 0 < \alpha < 2$$

all of the moments are finite

The actual measured squared coefficient of variation values were (obviously) finite and were between 25 and 49 !!!

$25 < C^2 < 49$

$C^2 = \frac{var}{mean^2}$, expo=1

Torniamo alle domande di prima? ha senso migrare job vecchi? come?

1. Should we bother with P migration, or is NP enough?
2. If we are going to bother with P migration, which processes are worth migrating?
   That is, what is a good migration policy?

the DFR property leads us to think that it may pay to migrate old jobs.
The reasoning is that although an old job may have high migration cost, because it has accumulated a lot of memory, if the job is really old then it has a high probability of using a lot more cpu in the future, which means that the cost of migration can be amortized over a very long lifetime.

Ciò che abbiamo visto è che potremmo avere vantaggi nel migrare vecchi job se siamo sotto caratteristica DFR.