

01/03/2022

MODelli DI SIMULAZIONE

Sono modelli: STOCASTICI, DINAMICI (= la variabile tempo ha un ruolo) e DISCRETI.

Come sviluppano un modello:

→ I passi 2 → 6 sono iterativi.

- 1) Definire gli obiettivi dello studio.
- 2) Definire il modello concettuale (M₁): riconoscere quali sono le variabili di stato (= descrizione istantanea del sistema + evoluzione del sistema nel tempo), come sono legate fra loro e quali possono essere rilevate e quali no.
- 3) Definire il modello di specifiche (M₂): riconoscere gli input del modello.
- 4) Definire il modello computazionale (il programma):
- 5) Verifica: stabilire se il modello computazionale corrisponde al precedente modello di specifiche (→ il modello è stato costruito in modo corretto?).
- 6) Validazione: capire quanto il modello computazionale sia rappresentativo del sistema (→ è stato costruito il modello giusto?). → DA NON FARE INSIEME ALLA VERIFICA

03/03/2022

- 7) Progettare gli esperimenti da fare.
- 8) Runnare il modello e registrare gli output ottenuti.
- 9) Analizzare gli output, osservando le medie, le deviazioni standard e i percentili, e producendo degliistogrammi, ecc.
- 10) Prendere delle decisioni in base agli output ottenuti.
- 11) Documentare i risultati.

MODelli ANALITICI

Riguardano la teoria delle code.

- TIPO 1: coda con un singolo server. ~~several stages~~

?

Esistono diverse politiche sull'ordine di servizio degli utenti (SCHEDULING, SERVICE ORDER): LIFO - FIFO - Random - con priorità - process sharing

Tipicamente Short Job First (SJF)

Definizione:

Un servizio è **Non-preemptive** se, una volta iniziato, non verrà mai interrotto fino al suo completamento (trattasi di un servizio atomico)

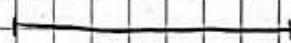
Definizione:

Un servente è **conservativo** se rimane sempre attivo, finché ci sono utenti in coda, o che stanno per essere serviti.

(non è mai idle)

Definizione:

TEMPO DI RISPOSTA = tempo che intercorre tra l'istante in cui l'utente (o il job) arriva ^{in coda} e l'istante in cui termina il suo servizio.



Introduciamo un altro tipo di coda:

- Tipo 2: coda con molteplici server.

↳ A parità di capacità totale, è preferibile utilizzare una coda con un singolo server a meno che il servizio sia Non-preemptive e il carico dei job abbia una varianza (varianza) elevata.

↳ Questa affermazione vale per l'obiettivo che consiste nel minimizzare il tempo di risposta medio; è chiaro che per obiettivi diversi vengono considerazioni diverse.

Poss - TIPO 3: Tante code con alberghi server.

↳ Qui ciascun job a quale coda deve essere assegnato? Esistono diverse possibilità:

► Random

► Round-Robin: l' i -esimo job va alla coda $\mod i$ mod n.

► Shortest-queue: ciascun job è assegnato alla coda con meno job in attesa.

► Size-Interval-Task-Assigment: ciascuna coda è relativa a un particolare range di dimensioni dei job. → più conveniente in caso di alta varianza del carico

► Least-Work-Left: ciascun job è assegnato alla coda relativa al server a cui è stato meno lavoro. → È EQUIVALENTE ALLA CODA CONTRARIA (tipo 2).

↳ più conveniente in caso di bassa varianza del carico

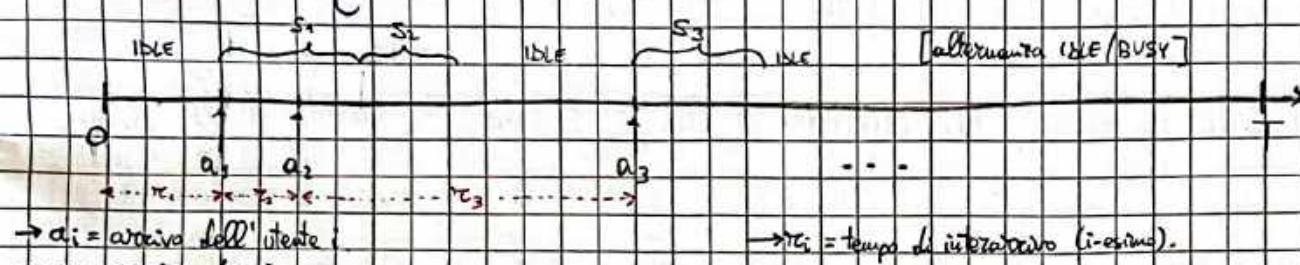
04/03/2022

Consideriamo la seguente coda:



► UTILIZZAZIONE (ρ): percentuale media di tempo in cui il servente è occupato.

Consideriamo il seguente scenario con una coda FIFO:



$\rightarrow a_i =$ arrivo dell'i-esimo intento

$\rightarrow s_i =$ servizio di i.

$\rightarrow r_i =$ tempo di istanziazione (i-esimo).

$$\rho = \lim_{T \rightarrow \infty} \frac{\text{Time Busy}}{T} = \lim_{T \rightarrow \infty} \frac{B}{T} = \lim_{T \rightarrow \infty} \frac{\sum_{i=1}^n s_i}{T} = \lim_{T \rightarrow \infty} \frac{\sum_{i=1}^n s_i}{\sum_{i=1}^n r_i} \approx$$

$$\approx \frac{E(\sum_{i=1}^n s_i)}{E(\sum_{i=1}^n r_i)} = \frac{\sum_{i=1}^n E[s_i]}{\sum_{i=1}^n E[r_i]} = \frac{m E[s]}{m E[r]} = \frac{\lambda}{\mu}$$

Possiamo DEFINIRE ρ nel seguente modo:

$\rho := \text{prob}\{N_s > 0\} = 1 - \text{prob}\{N_s = 0\}$, dove $N_s = \# \text{ UTENTI IN CODA} + \overset{N_s}{\text{L'EVENTUALE UTENTE IN SERVIZIO}}$

$$\Rightarrow \rho = 1 - \text{prob}(N_s = 1) + 1 \cdot \text{prob}(N_s = 2) + \dots$$

QUESTO È IL CALCOLO DELLA POPOLAZIONE MEDIA "ALL'INTERNO" DEL SERVENTE.

$\Rightarrow \rho =$ popolazione media nel server.

\rightarrow Definiamo $E[n]$ il numero di utenti serviti nell'intervallo di tempo t .

Allora $E[n]$ è il throughput (che indichiamo con X).

L'numero di utenti serviti nell'intervallo di tempo

RELAZIONI:

$$E[T_s] = E[T_b] + E[S]$$

$T_b =$ tempo di attesa
 $\rightarrow T_s =$ tempo di risposta

$$E[N_s] = E[N_a] + \rho$$



Migliorare il tempo di risposta di un servizio NON implica necessariamente migliorare

raccorre il throughput.

↳ Dato fatto, finché il tasso di servizio μ è maggiore del tasso di arrivo λ , il throughput X è sempre uguale a λ , indipendentemente dal valore preciso di μ (da cui dipende comunque il tempo di risposta).

Se invece $\lambda > \mu$, il throughput X diventa esattamente uguale a μ .

infatti CASO IN CUI LA CODA CRESCSE INDEFINITAMENTE NEL TEMPO

$$\hookrightarrow E[N_q \text{ nel tempo } T] \geq \lambda T - \mu T = T(\lambda - \mu) \xrightarrow{T \rightarrow +\infty} +\infty$$

Il sistema NON raggiunge mai la stazionarietà

Se invece la coda è finita, nel momento in cui si sazia, si hanno delle perdite.

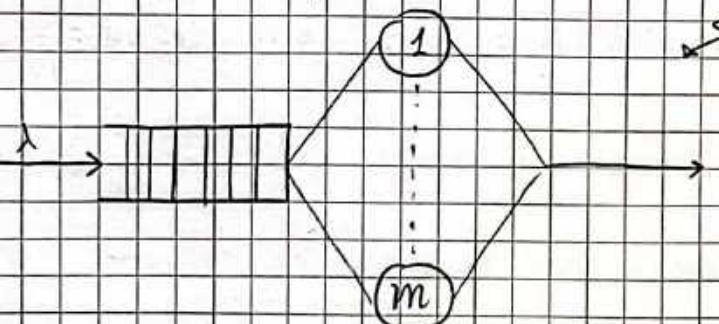
Di conseguenza, il throughput non può essere uguale a λ (anche se abbiamo $\lambda < \mu$), perché non è neanche vero che l'utilizzazione è pari a $\rho = \frac{\lambda}{\mu}$: invece, abbiamo $X < \lambda$.

↳ Basti pensare che non tutti quelli che arrivano vengono serviti.

→ Nel caso di coda finita si raggiunge sempre la stazionarietà in tempo finito.

Consideriamo ora un servere multiplo:

SE I SERVER SONO TUTTI UGUALI TRA LORO, IL SISTEMA È OMOGENEO.



$$N_s = \begin{cases} \{0, \dots, m\} & \text{se } N_a = 0 \\ N_a + m & \text{se } N_a > 0 \end{cases}$$

$$E[N_s] = \frac{1}{\mu} \quad \forall i = 1, \dots, m$$

$$E[N_s] = E[N_a] + mp$$

Qui dobbiamo distinguere l'utilizzazione globale ρ_{glob} dall'utilizzazione del singolo servere ρ_i :

$$\rightarrow \rho_i = \frac{\lambda}{m\mu}$$

$$\rightarrow \rho_{\text{glob}} = \frac{\lambda}{m\mu}$$

A Ogni SERVIZIO ARRIVA $\frac{\lambda}{m}$

GLOBALMENTE ↑
A OGNI SERVIZIO ARRIVA λ
ED ESCI $m\mu$

$$\Rightarrow P_{\text{att}} = p_i$$

Infine abbiamo distinguere il tempo di servizio s dal tempo che intercorre tra l'istante in cui un utente inizia il servizio e l'istante in cui qualunque altro utente termina il servizio.

$$\rightarrow E[s] = \frac{1}{m\mu}$$

$$E[\text{att}]: E[T_s] = \underbrace{E[T_Q]}_{\text{QUESTO È FUNZIONE DI } \lambda, \text{ di } m \text{ E DI } \mu} + E[s]$$

QUESTO È FUNZIONE DI λ , di m E DI μ .

Legge di Little:

Supponiamo che la disciplina della coda sia FIFO, e che la coda sia stabile.

Allora, la popolazione media del centro è data da $N = \lambda T$, dove T è il tempo di permanenza nel centro.

↳ In particolare, si ha:

- $\rho = \lambda E[s]$
- ~~$E[N_Q] = \lambda E[T_Q]$~~
- $E[N_s] = \lambda E[T_s]$

08/03/2022

Supponiamo di avere 150 macchine identiche che:

→ Operano 8 ore al giorno, 250 giorni all'anno (lo continuo).

→ Operano indipendentemente.

→ Vanno in riparazione se si rompono.

→ Ricavo: 50 €/h per le operazioni delle macchine.

→ Per quanto riguarda i guasti, i tecnici delle riparazioni vengono pagati 60.000 € ogni due anni.

→ Ciascun tecnico delle riparazioni lavora 8 ore al giorno, 230 giorni all'anno.

DOMANDA: quanti tecnici servono per massimizzare il profitto? a questo esprime il nostro obiettivo

↳ Abbiamo 2 casi limite:

• 1 solo tecnico \Rightarrow costo dei tecnici minimizzato MA inattività delle macchine in caso di guasto massimizzata.

• 1 tecnico per ogni macchina \Rightarrow costo dei tecnici massimizzato MA inattività delle macchine in caso di guasto minimizzato.

MODELLO CONCETTUALE:

Definiamo:

- Lo stato di ciascuna macchina (operativa vs guasta).
- Lo stato di ciascun tecnico (libero vs occupato).

Non abbiamo oggetti che entrano e oggetti che escono

Il modello è un sistema chiuso, con $JOB = MACCHINE GUASTE$ e $SERVER = TECNICI$.

MODELLO DELLE SPECIFICHE:

E domande simili:

Cosa si sa sulla distribuzione degli intervalli di tempo ogni cui una macchina si guasta.

MODELLO COMPUTAZIONALE:

Qui devono essere definite le code delle macchine guaste, la struttura per raccogliere dati statici, ecc.

VERIFICA:

Qui si controlla se c'è corrispondenza tra modello delle specifiche e modello computazionale (tipicamente ciò si fa mediante l'attività di testing dell'ingegneria del software).

VALIDAZIONE:

Qui, ad esempio, si può controllare se, all'aumentare del numero dei tecnici, il numero di macchine guaste diminuisce. E test simili.

PROGETTAZIONE DEGLI ESPERIMENTI:

È questa la fase in cui si deve trovare il numero stimato di tecnici per massimizzare il profitto. Bisogna dunque definire le condizioni iniziali (e quante macchine sono guaste all'inizio?) e i vari parametri (e.g. frequenza dei guasti), ed è opportuno seguire un certo numero (da fissare) di tecni per ciascuna delle configurazioni.

PRODUZIONE DELLE ESECUZIONI:

Attenzione a mai memorizzare i dati generati (di fatto, se vengono eseguite tante run, la gestione dei risultati di output diventa un problema).

ANALISI DEGLI OUTPUT:

Tenere conto che l'analisi statistica degli output di simulazione è spesso più complicata dell'analisi statistica classica: è una buona idea costruire (bene) un campione IID (di variabili indipendenti e identicamente distribuite).

FASE DECISIONALE:

Si stabilisce il numero di tecniche da assumere anche in funzione di politiche esterne.

Entriamo in qualche maggiore dettaglio.

Per un job i :

- a_i = TEMPO DI ARRIVO
- s_i = TEMPO DI SERVIZIO
- d_i = TEMPO DI ATTESA (= delay) \geq tempo come attesa nella coda
- $b_i = a_i + d_i$ = TEMPO IN CUI IL SERVIZIO INIZIA
- $w_i = \sum_{j=1}^{i-1} d_j + s_i$ = TEMPO DI RISPOSTA (= wait)
- $c_i = a_i + w_i$ = TEMPO DI COMPLETAMENTO (= departure time)

} Variabili di input

} Variabili di output

$$\rightarrow Y_i = a_i - a_{i-1} = \text{TEMPO DI INTERARRIVO}, \text{ dove, per definizione, } a_0 = 0.$$

Il primo arrivo è dato dall'istante a_0 .

\Rightarrow Se assumiamo di avere 2 solo arrivo per un istante temporale (a_0 o a_1), vor dire che non abbiamo i BULK.

DOMANDA: Dati i tempi di arrivo a_i e i tempi di servizio s_i , come possiamo calcolare i tempi di attesa d_i ?

\rightarrow Per alcuni modelli di code non è facile determinarlo, ma per una coda FIFO sì.

Infatti:

- CASO 1: $a_i < c_{i-1} \Rightarrow d_i = c_{i-1} - a_i$
- CASO 2: $a_i \geq c_{i-1} \Rightarrow d_i = 0$

Le STATISTICHE DI OUTPUT possono essere medicate sul numero di job oppure sul tempo.

STATISTICHE MEDICATE SUL NUMERO DI JOB:

$$\rightarrow \text{TEMPO DI INTERARRIVO MEDIO: } \bar{Y}_C = \frac{1}{m} \sum_{i=1}^n Y_i = \frac{a_n}{m}$$

$$\rightarrow \text{FREQUENZA MEDIA DI ARRIVO: } \frac{1}{\bar{Y}_C}$$

$$\rightarrow \text{TEMPO DI SERVIZIO MEDIO: } \bar{s} = \frac{1}{n} \sum_{i=1}^n s_i$$

$$\rightarrow \text{FREQUENZA MEDIA DI SERVIZIO: } \frac{1}{\bar{s}}$$

$$\rightarrow \text{TEMPO DI ATTESA MEDIO: } \bar{d} = \frac{1}{n} \sum_{i=1}^n d_i$$

$$\rightarrow \text{TEMPO DI RISPOSTA MEDIO: } \bar{w} = \frac{1}{n} \sum_{i=1}^n w_i = \bar{d} + \bar{s}$$

Quello che è opportuno fare nella pratica è ~~non~~ calcolare (in modo diretto) sia S , sia \bar{d} , sia \bar{w} , e utilizzare $\bar{w} = \bar{d} + \bar{S}$ come formula da verificare.

STATISTICHE MEDIE NEL TEMPO:

→ NUMERO TOTALE DI JOBS NEL SERVICE NODE = $l(t)$ ($= N_S$)

→ NUMERO DI JOBS NELLA CODA = $q(t)$

→ NUMERO DI JOBS IN SERVIZIO = $x(t)$

Per definizione $l(t) = q(t) + x(t) \quad \forall t$

→ NUMERO MEDIO DI JOBS NEL SERVICE NODE = $\bar{l} = \frac{1}{t} \int_0^t l(t) dt$

→ Analogamente per le altre 2 grandezze

Chiarimenti: $\bar{l} = \bar{q} + \bar{x}$ → UTILIZZAZIONE (dal momento in cui $l(t) = q(t) + x(t) \quad \forall t$)

Formulazione alternativa della legge di Little:

$$\cdot \int_0^{C_n} l(t) dt = \sum_{i=1}^n w_i$$

$$\cdot \int_0^{C_n} q(t) dt = \sum_{i=1}^n d_i$$

$$\cdot \int_0^{C_n} x(t) dt = \sum_{i=1}^n s_i$$

→ C_n = tempo di completamento dell'ultimo job
Job = fine dell'intervallo di osservaz.

COROLARIO: → PERCHÉ SI TRATTA DI UNA Coda infinita: $C_n \bar{l} = \int_0^{C_n} l(t) dt = \sum_{i=1}^n w_i = n \bar{w}$

$C_n \bar{l} = m \bar{w} \Rightarrow \bar{l} = \frac{m}{C_n} \bar{w}$, dove, di fatto, $\frac{m}{C_n}$ = THROUGHPUT MEDIO DEL SISTEMA.

Analogamente, $\bar{q} = \frac{m}{C_n} \bar{d}$, $\bar{x} = \frac{m}{C_n} \bar{s}$

→ Per una coda infinita è perfettamente uguale al tasso medio di arrivo (λ).

Definizione:

L'INTENSITÀ DI TRAFFICO è il rapporto tra il tasso d'arrivo e il tasso di servizio:

$$\frac{1/\bar{c}}{1/S} = \frac{\bar{s}}{\bar{t}} = \frac{\bar{s}}{\bar{a}/h} = \frac{C_n}{\bar{a}h} \bar{x}$$

$$\bar{x} = \frac{m}{C_n} \bar{s}$$

→ Quando $\frac{C_n}{\bar{a}h} \approx 1$, l'intensità di traffico e l'utilizzazione sono allo stesso tempo circa uguali. → QUESTO SICURAMENTE NON PUÒ ACCADERE PER CODE NON STABILI ($C_n > \bar{a}h$).
↳ Infatti in tal caso avremmo che $C_n > \bar{a}h \Rightarrow$ intensità di traffico $\gg 1$.

10/03/2022

Caso di studio 1:

Si vuole aggiungere un'opzione di gelato in una gelateria, ma ci si vuole assicurare

che questo non infici sui guadagni (e.g. in termini di coda troppo lunga).

Per semplicità, possiamo assumere l'uso di un single-server queue. \rightarrow CON DISCIPLINA FIFO

Supponiamo di avere 100 clienti caratterizzati da:

• TASSO MEDIO DI ARRIVO = $\frac{1}{10} = 0,10$ job/sec

→ VALORI (INSIEME A \bar{q} , \bar{W}) OTTENUTI FACILMENTE A PARTIRE DAI DATI INIZIALI SE, SÌ.

• TASSO MEDIO DI SERVIZIO = $\frac{1}{3} = 0,33$ job/sec

ALTRI TIPI, SE DOBBIANO DUNQUE ASSUMERLI NOTI, E DA ESTI DIPENDONO TUTTI I VALORI NUMERICI SFORNATI QUI.

→ Si possono calcolare \bar{l} , \bar{q} , \bar{x} , e viene fuori che:

• Il servere è libero il 28% del tempo ($\rightarrow 1 - \bar{x} = 0,28$)

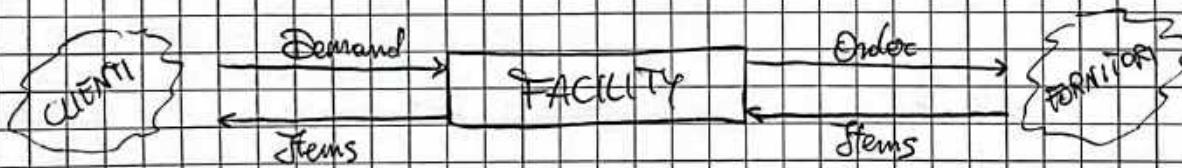
• La popolazione media nella coda è $\bar{q} = 2$

\bar{x} non vicino a 1

→ Notiamo che, nonostante il servere sia "abbastanza" libero, mediamente l'utente deve attendere in coda per un tempo non trascurabile. $\sim \bar{q} = 2$

Abbiamo una crescita (più che) esponentiale della popolazione media in coda, all'aumentare del tempo di servizio medio (e quindi, dell'utilizzazione). In particolare, se l'utilizzazione è già a 1, \bar{q} va all'infinito.

Caso di studio 2: \rightarrow GESTIONE DI UN MAGAZZINO



OBIETTIVI:

Investigare su quale politica di inventario ottimizza il rapporto tra richieste di articoli da parte dei clienti e rifornimenti. Su questo abbiamo due possibili approcci:

1) Reporting di ogni transazione.

2) Revisione periodica dell'inventario (\rightarrow viene fissato un valore minimo del livello di rifornimento e viene effettuato un controllo ogni lunedì; se il rifornimento è al di sotto di tale soglia minima si effettua l'ordine, altrimenti no).

\uparrow È QUESTA LA POLITICA ATTUATA NELLA PRATICA

Definendo S = livello minimo di rifornimento, S' = livello massimo di rifornimento, vogliamo stabilire qual è la coppia (S, S') che minimizza il costo

MODELLO CONCETTUALE:

Bisogna definire:

- Costo di mantenimento degli elementi nel magazzino.
- Costo dovuto al non-soddisfacimento ^{immediato} delle richieste degli utenti.
- Costo di setup (\equiv quanto costa istituire un ordine?)
- Costo di ordine ^{di} di ciascun item.
- Il livello dell'inventario l .
- L'ammontare degli ordini O .
- La quantità di item richiesta.

Assumiamo che il livello iniziale e finale dell'inventario sia S .

→ Per semplicità, assumiamo inoltre di non avere DELIVERY LAG (\rightarrow quando viene richiesto un fornimento, il magazzino viene riempito instantaneamente).

MODELLO SENZA SPECIFICHE:

- Gli istanti di tempo di revisione sono $t = 0, 1, 2, \dots$ (EVOLUZIONE DISCRETA)
- $l_{i-1} :=$ livello dell'inventario all'inizio dell'intervallo i -esimo (quello da $t=i-1$ a $t=i$).
- $O_{i-1} :=$ quantità di merce ordinata all'istante $t=i-1$.
- $d_i :=$ quantità di merce richiesta durante l' i -esimo intervallo.

CONSIDERAZIONI:

$$O_{i-1} = \begin{cases} 0 & \text{se } l_{i-1} \geq S \\ S - l_{i-1} & \text{se } l_{i-1} < S \end{cases} \rightarrow \text{se } l_{i-1} < S, \text{ il LIVELLO DI RIFORNIMENTO VIENE RIPORTATO A } S.$$

$$l_i = l_{i-1} + O_{i-1} - d_i \rightarrow \text{IL LIVELLO DI RIFORNIMENTO VARIA IN BASE AGLI ORDINI IN INGRESSO E ALLE DOMANDE DEI CLIENTI.}$$

→ d_i è l'unica traccia nel nostro modello.

↳ TRACCE = valore V che S può assumere nella realtà. \rightarrow le simulazioni che si basano su una o più tracce sono dette TRACE-DRIVEN.

STATISTICHE DA CALCOLARE (MODELLO COMPUTAZIONALE):

$$\bar{d} = \frac{1}{m} \sum_{i=1}^m d_i \quad \bar{O} = \frac{1}{m} \sum_{i=1}^m O_i$$

↳ Poiché (minimo o poi) tutte le richieste degli utenti devono essere soddisfatte, abbiamo $\bar{d} = \bar{O}$.

Per quanto riguarda il presentarsi di uno shortage, abbiamo due casi:

- 1) $l(t)$ resta non-negativo nell' i -esimo intervallo: $\rightarrow l(t)$ è il livello dell'inventario istante per istante \Rightarrow per conoscere tale funzione, è necessario sapere gli istanti precisi di tutti i d_i (è possibile comunque effettuare delle assunzioni esemplificative a riguardo).

- 2) $l(t)$ diventa negativo a un certo istante $t=\tau$:

$$l_i^+ = \int_{i-1}^{\tau} l(t) dt$$

$$l_i^- = - \int_{\tau}^{\tau} l(t) dt$$

dove l_i^+ è legato al costo di holding e l_i^- è legato al costo di shortage.

Definiamo in modo più puntuale i costi: (settimanali):

- Item cost: $C_{item} \cdot \bar{o}$
- Setup cost: $C_{setup} \cdot \bar{m}$ → dove \bar{m} = "order frequency" = $\frac{\# \text{ORDINI}}{n}$
- Holding cost: $C_{hold} \cdot \bar{l}^+$ → dove $\bar{l}^+ = \frac{1}{n} \sum_{i=1}^n l_i^+$ → LIVELLO MEDIO DELL'INVENTARIO:
- Shortage cost: $C_{short} \cdot \bar{l}^-$ → dove $\bar{l}^- = \frac{1}{n} \sum_{i=1}^n l_i^-$ $\bar{l} = \frac{1}{n} \int_0^{\tau} l(t) dt = \bar{l}^+ - \bar{l}^-$

Per calcolare la minimizzazione del costo, è opportuno far variare una sola variabile per volta tra s e S . (ad esempio se fissiamo S).

Sappiamo che $\bar{o} = \bar{d}$ e \bar{d} dipende solo dalle richieste \Rightarrow l'item cost è indipendente da (s, S) ; sono le restanti 3 parti del costo a dipendere da (s, S) .

PROIEZIONE DEGLI ESPERIMENTI:

Per ora fissiamo S . Finché s cresce:

→ Setup cost e holding cost aumenteranno.

→ Shortage cost diminuirà.

L'idea è minimizzare SETUP COST + HOLDING COST + SHORTAGE COST, che è praticamente una funzione a U.

11/03/2022

Ma cosa possiamo fare se abbiamo bisogno di più dati in input ma non li abbiamo disponibili? → RANDOM NUMBER GENERATOR (che genera valori reali compresi tra 0 e 1)

Esistono 3 tipi di random generator:

- Table look-up generator
- Hardware generator
- Software generator

Un buon generatore dovrebbe prendere un certo set di sisteme di valori compresi tra 0 e 1 ("tutti" sarebbero infiniti!) ed estrarre i numeri da tale sistema con probabilità UNIFORME.

→ COME SE FOSSE UNA V.A. CONTINUA UNIFORMEMENTE DISTRIBUITA.

In pratica, i numeri da prendere in considerazione, fissato un m grande, sono:

$$\frac{1}{m}, \frac{2}{m}, \dots, \frac{m-1}{m}$$

[0 e 1 sono esclusi].

PER PAGGIANTI PRATICHE, CONSIDEREREMO ESTRATTIONI SENZA RIMPIAZZAMENTO (MA SE m È GRANDE, NO PROBLEMI)

Generatore di Lehmer:

In tal caso gli scambi con riempimento e senza riempimento sono approssimabili tra loro.

È definito tramite due parametri:

• m , che è il modulo

• a , che è un moltiplicatore appartenente a $X_m = \{1, 2, \dots, m-1\}$

$\forall i$ vale che $g(x_i) = x_{i+1}$; in particolare: $g(x) = ax \bmod m$ e $x_0 \in \text{SEME}$ (scelto a priori - tipicamente $\overset{\text{(seed)}}{=}$ pari a 1).

ESEMPIO:

$$m=7 \Rightarrow X_7 = \{1, 2, 3, 4, 5, 6\}$$

$$a=3$$

$$x_0=1$$

Allora: $x_0=1; x_1=3; x_2=2; x_3=6$

$$x_4=4;$$

$$x_5=5$$

$$e \text{ poi basta perché } x_6=x_0=1 \text{ e il ciclo riunisce}$$

↳ Possiamo notare che la scelta di x_0 è indifferente (determina solo il punto di inizio della sequenza ma non la "randomità").

Ma se invece cambiasimo il moltiplicatore a (fissandolo a 2)?

$$x_0=1;$$

$$x_1=2;$$

$$x_2=4;$$

$$\boxed{x_3=1}$$

OOPS!

Non tutti i moltiplicatori sono in grado di generare tutti i valori (ovvero non sono tutti FULL PERIOD).

⇒ La lunghezza del generatore dipende dal parametro a .

Ma dipende anche da m , che deve essere un numero primo grande (tipicamente si sceglie $m=2^{31}-1$).

Primo perché così evitiamo di ottenere lo zero come risultato in qualche iterazione (e sappiamo che il prodotto per 0 mod m è sempre 0).

15/03/2022

Ma come facciamo a scegliere il parametro a (in modo tale che la sequenza generata, oltre a essere full period, abbia anche una parvenza casuale)?

Teniamo conto che, fissata la coppia (a, m) e fissato il seme x_0 , il valore x_0 occorre ridursi modularmente all'iterazione p -esima, dove p è uguale a $m-1$ [CASO FULL PERIOD] oppure a un sottomultiplo di $m-1$.

Vogliamo un'implementazione CORRETTA ed EFFICIENTE.

Tra l'altro, esiste anche un problema di implementazione del generatore random: ad esempio, è possibile avere un overflow nel prodotto ax (che può essere finitamente maggiore di m).

→ Effettuando il rapporto tra m e ax , otteniamo un quoziente q e il resto r .

Se $r < q \Rightarrow a$ ha la proprietà di MODULO-COMPATIBILITÀ.

Teorema:

Se $m = aq + r$ è un numero primo e a è modulo-compatibile \Rightarrow per un $x \in X_m$

$S(x) = 0 \vee S(x) = 1$, dove $S(x) := \lfloor x/q - \lfloor ax/m \rfloor \rfloor$. CONFESSIONE: $S(x) + m \delta(x)$

Inoltre,

$S(x) = 0 \Leftrightarrow Y(x) \in X_m$

\rightarrow caso $Y(x) > 0 \rightsquigarrow$ qui il valore restituito è $Y(x)$

$S(x) = 1 \Leftrightarrow -Y(x) \in X_m$

\rightarrow caso $Y(x) < 0 \rightsquigarrow$ qui il valore restituito è $Y(x) + m$

dove $Y(x) = \{a(x \bmod q) - r\} \lfloor x/q \rfloor$.

Calcolare ax mod m passando per $Y(x)$ è quello che si fa nella pratica perché, in tal modo, non si ottengono mai risultati interi che facciano overflow (dove che siano maggiori di m).

→ I moltiplicatori MC (modulo-compatibili) sono tutti minori o uguali a $\frac{m-1}{2}$.

→ Un moltiplicatore a è definito "piccolo" $\Leftrightarrow a^2 < m$.

→ Se a è piccolo $\Rightarrow a$ è MC.

→ $a = 48271$ è il moltiplicatore di default per $m = 2^{31} - 1$.

→ I generatori che presentano degli OUTLIERS non vanno scartati a priori.

→ Un outlier consiste nel fenomeno per cui vengono generati 15-20 numeri (consecutivi) e, ad esempio, sono tutti compresi tra 0,63 e 1.

→ Questo succede anche nella realtà.

Per intenderci, noi vogliamo un moltiplicatore a che sia FP (Full Period) e MC (Modulo-Compatibile).

→ TUTTI MOLTIPLICATORI SONO NETTI FP/MC.

Tra i moltiplicatori FP/MC ci piace quello che genera la sequenza apparentemente più randomica.

Torniamo al modello analitico...

Coda M/G/1 scheduling ciascuno:

Distribuzione interattiva
markoviana
(= poissoniana)

Distribuzione
servizio
generica

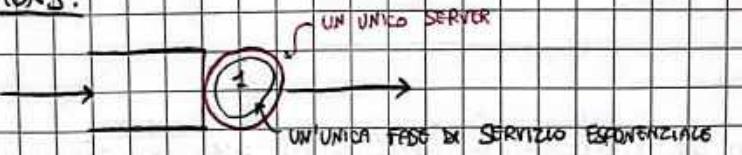
Servente
singolo

$$E(N_q) = \frac{P^2}{2(1-p)} \left[1 + \frac{\sigma^2(s)}{E^2(s)} \right] \rightarrow s \text{ è la distribuzione del servizio}$$

\downarrow
 $C^2 = \text{dispersione del tempo di servizio}$

PHASE-TYPE DISTRIBUTIONS:

► ESPONENZIALE
(M)



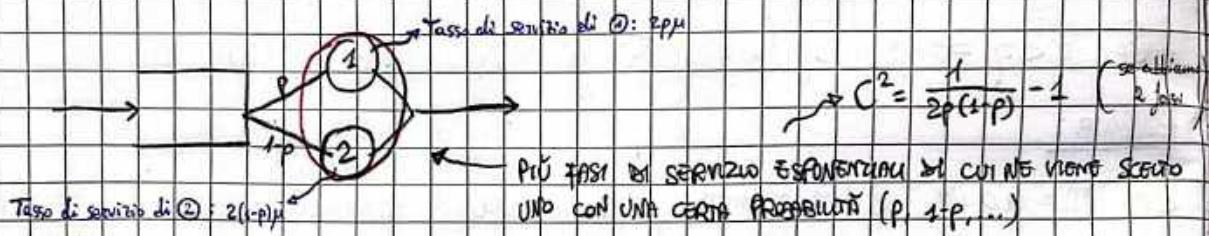
$$\rightarrow C^2 = 1$$

► K-ERLANG
(E_k)



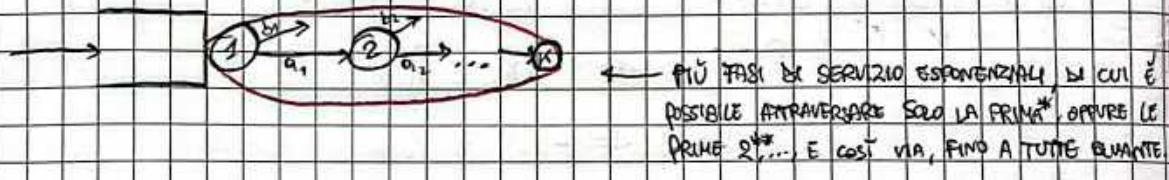
$$\rightarrow C^2 = 1/k$$

► IPERESPONENZIALE
(H_k)



$$\rightarrow C^2 = \frac{1}{2p(1-p)} - 1 \quad (\text{se abbiamo } 2 \text{ fasi})$$

► COXIANA



SENSITIVITÀ DEL TEMPO DI SERVIZIO:

$$E[N_q]_D \leq E[N_q]_{E_k} \leq E[N_q]_H \leq E[N_q]_{H_k}$$

$$\sigma^2[N_q]_D \leq \sigma^2[N_q]_{E_k} \leq \sigma^2[N_q]_H \leq \sigma^2[N_q]_{H_k}$$

} Chiaramente lo stesso vale per N_s .

$$E[T_q]_D \leq E[T_q]_{E_k} \leq E[T_q]_H \leq E[T_q]_{H_k}$$

↑ Questo ordinamento non vale per la varianza del tempo.

D'altra canto:

$$E[N_q]_{\text{FIFO}} = E[N_q]_{\text{LIFO}} = E[N_q]_{\text{RAND}} = E[N_q]_{\text{abstract}}$$

$$\sigma^2[N_q]_{\text{FIFO}} = \sigma^2[N_q]_{\text{LIFO}} = \sigma^2[N_q]_{\text{RAND}} = \sigma^2[N_q]_{\text{abstract}}$$

$$E[T_q]_{\text{FIFO}} = E[T_q]_{\text{LIFO}} = E[T_q]_{\text{RAND}} = E[T_q]_{\text{abstract}}$$

↑ Queste uguaglianze non valgono per la varianza del tempo.

$$\sigma^2[T_q]_{\text{FIFO}} \leq \sigma^2[T_q]_{\text{RAND}} \leq \sigma^2[T_q]_{\text{LIFO}}$$

Se ci pensi è anche intuitivo :)

17/03/2022

$$E[T_q] = \frac{P}{1-p} \cdot \frac{\lambda^2 + 1}{2} E[S] \rightarrow \text{è una funzione di } p \text{ e di } E[S]$$

Supponiamo che in una coda a un certo punto il tasso d'arrivo raddoppi ($\lambda' = 2\lambda$).

Se raddoppieremmo le prestazioni del server ($\mu' = 2\mu$), avremmo:

$$\rightarrow p' = \frac{\lambda'}{\mu'} = \frac{2\lambda}{2\mu} = p$$

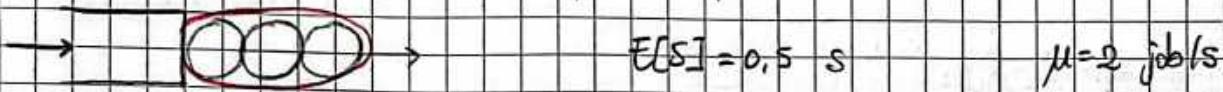
$$\rightarrow E[S'] = \frac{1}{\mu'} = \frac{1}{2\mu} = \frac{E[S]}{2}$$

Se poi riferimento alle formule delle distribuzioni specifiche, si vede bene che $E[T_q'] = \frac{1}{2} E[T_q]$

$$\rightarrow E[T'_s] = E[T_q'] + E[S'] = \frac{E[T_q]}{2} + \frac{E[S]}{2} = \frac{E[T_s]}{2}$$

K=3

Consideriamo un caso in cui il tempo di risposta del servizio ha distribuzione di Erlang:

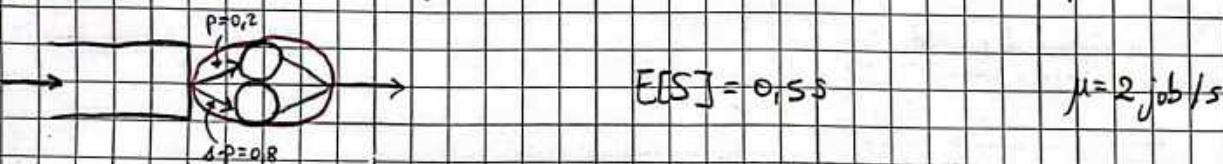


$$\rightarrow E[S_i] = \frac{0,5}{K} = \frac{0,5}{3} = 0,1666 \text{ s}$$

$$\rightarrow \sigma^2(S) = \frac{1}{K} \left(\frac{1}{\mu} \right)^2 \approx 0,083333 \text{ s}^2$$

↑ VARIANZA DELL'ESPONENTIALE

Se invece il tempo di risposta del servizio ha distribuzione iperespontenziale:



$$\rightarrow \sigma^2(S) = g(p) \left(\frac{1}{\mu} \right)^2 \quad \text{dove } g(p) = \frac{1}{2p(1-p)} - 1$$

Qui quello che succede è che il 20% degli utenti riceve un servizio + LENTO che ANDRA' A RIACCARE ANCHE GUELLI CHE SUCCESSIVAMENTE RICEVERANNO IL SERVIZIO + VELOCE

La maggiore variabilità del servizio provoca un aumento più marcato del tempo d'attesa.
L'effettivamente $g(p)$ non solo è maggiore di $\frac{1}{\mu}$ se $K > 1$, ma è anche maggiore o uguale a 1.

Esercizio:

La frequenza di arrivo è $\lambda = 15$. → MARKOVIANA

Il tempo di servizio medio per transazione è 58,37 ms. → GENERICA

Cosa succede al tempo di risposta medio se il tasso di arrivo aumenta del 10%?

Memoryless property:

È una proprietà di una variabile aleatoria per cui i suoi valori futuri dipendono esclusivamente dal tempo presente e non dal tempo passato.

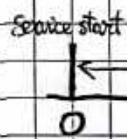
ESEMPIO:

Supponiamo che un negozi ~~apre~~ ^{aleatoria} apri alle 9:00 e il primo cliente arrivi solo alle 9:30. Vogliamo capire quale sia la distribuzione di probabilità degli arrivi successivi alle 9:30 dato che dalle 9:00 alle 9:30 non è arrivato nessuno.

↳ Se la distribuzione degli arrivi gode della memoryless property, non cambia tra il caso standard e il caso in cui dalle 9:00 alle 9:30 non arriva nessuno.

↳ Le uniche 2 distribuzioni che godono di tale probabilità sono l'**ESPOENZIALE** e la **GEOMETRICA**.

CASO DISTRIBUZIONE ESPOENZIALE:



$$\text{Prob}(X \leq t) = 1 - e^{-\mu t} \quad \text{dove } \mu = \text{tasso di servizio}$$



$$\text{Prob}(X \leq t_0 + t \mid X > t_0) = \frac{\text{Prob}(t_0 < X \leq t_0 + t)}{\text{Prob}(X > t_0)} = \frac{\text{Prob}(X \leq t_0 + t) - \text{Prob}(X \leq t_0)}{\text{Prob}(X > t_0)} =$$

$$= \frac{1 - e^{-\mu(t_0+t)}}{1 - (1 - e^{-\mu t_0})} = 1 - e^{-\mu t} \quad \rightarrow \begin{array}{l} \text{E' PERFETTAMENTE UGUALE} \\ \text{AL CASO PRECEDENTE.} \end{array}$$

" Prob(X ≤ t) "

DENSITÀ DELL'ESPOENZIALE:

$$\bullet f(0) = \lambda$$

$$\bullet f(2) = \lambda e^{-2\lambda} = \lambda e^{-1} e^{-1} = f(1) e^{-1}$$

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

$$\bullet f(1) = \lambda e^{-\lambda} = f(0) e^{-1}$$

$$\bullet f(3) = \lambda e^{-3\lambda} = f(2) e^{-1}$$

$$\Rightarrow f(n) = f(n-1) e^{-\lambda} \rightarrow \text{PER OGNI UNITÀ DI TEMPO, LA QUANTITÀ VIENE RIDOTTA DI UN FATTORE COSTANTE, CHE È } e^{-\lambda}$$

$\rightarrow \lambda$ viene detta COSTANTE DI DECADIMENTO.

$$f(1) = \lambda e^{-\lambda} = \lambda e^{-\lambda} \approx \lambda \cdot 0,36788 \rightarrow \text{è ridotta del 63% (circa) rispetto al valore in 0.}$$

DENSISSIMA CALCOLATA NELLA MEDIA DELL'ESPONZIALE

FUNZIONE DI DISTRIBUZIONE (CUMULATIVA) DELL'ESPONZIALE: $F(x) = 1 - e^{-\lambda x}$

$$F(1) = 1 - e^{-\lambda} \approx 1 - 0,36788 \approx 63,21\% = \text{Prob}(X \leq 1) = \text{Prob}(X \leq E[X])$$

Abbiamo visto le distribuzioni a fase (esponeziale, K-Erlang, iperesponeziale).

Queste sono importanti perché i singoli stadi del processo sovrastante del servizio sono esponenziali (\rightarrow godono della memoryless property).

► ESPONZIALE: $f(x) = \mu e^{-\mu x}$

$$E[X] = \frac{1}{\mu}$$

$$\sigma^2[X] = \left(\frac{1}{\mu}\right)^2$$

► K-ERLANG: $f_i(x) = K \mu e^{-\mu x}$

$$E[X_i] = \frac{1}{K\mu}$$

$$\sigma^2[X_i] = \left(\frac{1}{K\mu}\right)^2$$

Relativum al singolo studio

$$E[X] = \frac{1}{\mu}$$

$$\sigma^2[X] = \frac{1}{K} \left(\frac{1}{\mu}\right)^2$$

► IPERESPONZIALE: $f(x) = p f_1(x) + (1-p) f_2(x)$ dove $f_1(x) = 2\mu e^{-2\mu x}$, $f_2(x) = 2(1-p)\mu e^{-2(1-p)\mu x}$

* dove $g(p) = \frac{1}{2p(1-p)} - 1$

$$E[X] = \frac{1}{\mu}$$

$$\sigma^2[X] = g(p) \left(\frac{1}{\mu}\right)^2 *$$

► DISTRIBUZ. QUALSIASI (MODELATA CON UNA COXIANA): ciascuno studio ha una sua media $\frac{1}{\mu}$ e ha una distribuzione esponeziale.

Se t_1, t_2, \dots, t_K sono i tempi spesi in ciascuna fase, il tempo totale di servizio sarà:

$\rightarrow t_1$ con probabilità b_1 . \rightarrow Vedere figura di 2 pagine fa.

$\rightarrow t_1 + t_2$ con probabilità $a_1 b_2$.

...

$\rightarrow t_1 + t_2 + \dots + t_K$ con probabilità $a_1 a_2 \dots a_{K-1} b_K$.

Ma come facciamo a rappresentare una distribuzione qualsiasi con una coxiana?

CASO 1: $f(t)$ arbitraria con una trasformata di Laplace razionale \rightarrow

$C_K(t) = f(t)$ per un dato K (uguaglianza esatta o con una precisione approssimativa nota)

Nel caso si abbiano troppi studi

CASO 2: $g(t)$ arbitraria senza una trasformata di Laplace razionale \Rightarrow

$f(t) \approx g(t)$ (approssimazione con una precisione nota).

$\hookrightarrow C_K(t) \approx f(g(t))$ per un dato K .

\hookrightarrow UNICO ESATTAMENTE UGUALE A $f(t)$

22/03/2022

Supponiamo ora di avere una coda semplice in cui il tempo di servizio va da 1 minuto a 2 minuti; non potendo effettuare ulteriori ipotesi, possiamo solo considerare ~~una~~ uniforme la distribuzione del tempo di servizio.

In generale però, non è ragionevole assumere che TUTTI i valori equiprobabili; i valori più piccoli sono più probabili \Rightarrow in tal caso ricorriamo alla distribuzione esponenziale. Per generare (tramite il generatore di Lehmer) dei valori che seguono una legge esponenziale, è possibile ricorrere a una trasformazione non lineare che mappa i valori in $(0,0; 1,0)$ su dei valori in $(0,0; +\infty)$.

\hookrightarrow Questa cosa fa lo stesso per il tempo di intervento tra i job/i clienti.



ASSUMIAMO CHE:

$$\lambda = 0,5 \text{ job/s} \quad (\text{MARKOVIANA})$$

$$E[S] = 1,5 \text{ s} \quad (\text{Umf}(1,2)) \quad \rightarrow f(x) = \frac{1}{b-a} 1_{[a,b]}(x) = 1_{[1,2]}(x)$$

$$\rightarrow \bar{x} = \frac{a+b}{2} = 1,5$$

$$\rightarrow \sigma^2(x) = \frac{(b-a)^2}{12} = \frac{1}{12}$$

$$\Rightarrow p = \lambda E[S] = 0,75$$

$$E[T_q] = \frac{p}{1-p} \cdot \frac{C+1}{2} E[S] \quad \text{dove } C = \frac{\sigma^2(x)}{\bar{x}^2} = \frac{1}{12} \cdot \frac{1}{(1,5)^2} = 0,037037$$

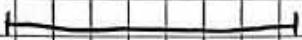
$$\Rightarrow E[T_q] = \frac{0,75}{0,25} \cdot \frac{1,037037}{2} \cdot 1,5 = 2,33 \text{ s}$$

$$E[T_s] = E[T_q] + E[S] = 2,33 + 1,5 = 3,83 \text{ s}$$

$$E[N_q] = \lambda E[T_q] = 1,1667$$

$$E[N_s] = E[N_q] + p = 1,1667 + 0,75 = 1,9167$$

→ Quando si effettuano le simulazioni, affinché i valori dei momenti del 1° ordine corrispondano ai valori teorici appena calcolati, è necessario che le simulazioni comprendano un numero sufficientemente elevato di job in ingresso.



I tipici modelli di simulazione hanno molte componenti stocastiche, ma noi vorremmo avere un'unica sorgente di randomizate, ad tal proposito, possiamo usare un Multi-Stream Lehmer RNG, che ci permette di distribuire i valori generati pseudo-randomicamente tra le componenti stocastiche.

↳ In particolare, stiamo usando uno stesso generatore (uno stream) a partire dal seguente teorema:

Teorema:

Dati $g(x) = ax \bmod m$ e un intero j tale che $1 < j < m-1$:

→ JUMP FUNCTION: $g^j(x) = [(a^j \bmod m) \cdot x] \bmod m$

→ JUMP MULTIPLIER: $a^j \bmod m$ z, è necessario che sia modulo-compatibile

Se $g()$ genera $x_0, x_1, x_2 \Rightarrow g^j()$ genera $x_0, x_{0j}, x_{2j}, \dots$

→ Così facendo si generano diversi flussi per un singolo RNG.

→ j dovrebbe essere uguale al più grande intero minore di L^{m-1} , tale che $a^j \bmod m$ sia modulo-compatibile, dove s è il numero di flussi.

↳ In pratica, se abbiamo M valori che possono essere generati in tutto, col multi-stream l'insieme degli M valori viene suddiviso in un certo numero di sottinsiemi DISGIUNTI ciascuno dei quali è relativo a un particolare flusso di generazione (a un particolare stream).

↳ Ad esempio, è possibile utilizzare uno stream per gli arrivi e uno stream per i servizi, dove arrivi e servizi sono le due componenti stocastiche del nostro sistema.

Se invece abbiamo 2 tipi di job ciascuno dei quali con una diversa distribuzione degli interarrivi e una diversa distribuzione dei servizi, abbiamo bisogno di 4 stream diversi del nostro generatore: 2 stream per la prima tipologia di job e 2 stream per la seconda tipologia di job.

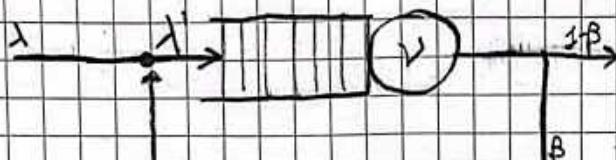
*) Se invece avessimo avuto un unico stream di un unico Lehmer RNG e avessimo usato due seed come punti di partenza per generare i tempi di arrivo e i tempi di servizio, nessuno ce ne avrebbe garantito che i due seed sarebbero stati distanti dal punto di vista del numero di chiamate a Random() di differente. Per seed sufficientemente vicini, avremmo avuto un overlap fra i due flussi di generazione e, quindi, arrivi e servizi sarebbero stati accoppiati.

24/03/2022

Simple Server Queue con feedback:

È un Simple Server Queue dove un Job, quando completa il servizio, non è detto che abbandoni il centro: potrebbe rimettersi in coda per ricevere un ulteriore servizio.

↪ COMPLETAMENTO DEL SERVIZIO \neq PARTENZA



Un job/utente può ricevere il servizio un numero arbitrario di volte.

$\beta = \text{probabilità di rimettersi in coda dopo aver completato il servizio}$

Dal punto di vista del servente, i job in coda hanno una frequenza maggiore di 1 (verro λ')

BUANCIAIMENTO DEL FLUSSO:

$$\begin{aligned} &\rightarrow \text{Job che arrivano dall'esterno} = \lambda \\ &\rightarrow \text{Job che escono dal centro} = (\bar{x}(1-\beta))\nu \end{aligned}$$

$$\Rightarrow \lambda = \bar{x}(1-\beta)\nu$$

SATURAZIONE:

$$\text{La si ha quando } \bar{x} \rightarrow 1, \text{ ovvero quando } \lambda = (1-\beta)\nu \Rightarrow 1-\beta = \frac{\lambda}{\nu} \Rightarrow \beta = 1 - \frac{\lambda}{\nu}$$

UTILIZZAZIONE:

$$p = \frac{\lambda'}{\nu}, \text{ dove } \lambda' = \frac{\lambda}{1-\beta}$$



Il feedback, chiaramente, rappresenta una componente stocastica in più, per cui necessita di un ulteriore stream del generatore pseudo-random.

Memoryless property vista dal punto di vista della lifetime:

Una v.a. X è detta memoryless se:

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

ESEMPIO:

X è il tempo di vita di una lampadina.

La proprietà ci dice che, la lampadina viverà per un altro tempo t dopo aver vissuto per un tempo s con la stessa probabilità con cui viverà per un tempo t senza altre assunzioni.

↪ Ma questo è irrealistico per una lampadina!

Inoltre, il tempo di servizio medio è aumentato rispetto a ν perché, per alcuni job, bisogna sommare + servizi.

- Le distribuzioni per cui $\text{Prob}\{X > s+t | X > s\}$ diminuisce all'aumentare di s sono dette a failure rate CRESCENTE.
- ESEMPIO: tempo di vita di una lampadina.
- Le distribuzioni per cui $\text{Prob}\{X > s+t | X > s\}$ aumenta all'aumentare di s sono dette a failure rate DECRESCENTE.
- ESEMPIO: tempo di uso della CPU di un job UNIX.

HAZARD RATE FUNCTION:

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

dove $f(t)$ = densità della v.a.
 $F(t)$ = funz. distribuz. della v.a.

Noi denotiamo $1-F(t)$ anche con $\bar{F}(t)$.

FAILURE RATE
(STANTANEO)

$$\rightarrow \text{Prob}\{X \in (t, t+dt) | X > t\} = \frac{\text{Prob}\{X \in (t, t+dt)\}}{\text{Prob}\{X > t\}} = \frac{f(t) dt}{1-F(t)} = r(t) dt$$

= \rightarrow PROBABILITÀ CHE UN ITEM VECCHIO t FALLIRÀ NEL PROSSIMO INTERVALLO DI DURATA dt .

29/03/2022

Se $r(t)$ è costante, allora $f(t)$ deve essere necessariamente esponenziale:

$$r(t) = \frac{f(t)}{1-F(t)} = \frac{\lambda e^{-\lambda t}}{e^{-\lambda t}} = \lambda$$

→ È IL CASO INTERMEDIO TRA LE DISTRIBUZIONI A FAILURE RATE CRESCENTE E QUELLE A FAILURE RATE DECRESCENTE (DI FATTO È UNA DI STRATEGIA A FAILURE RATE COST.)

Ma perché il tempo di vita pienamente è così importante?

→ BILANCIMENTO DEL CARICO: può aiutare a migrare un job verso un server meno carico
 → al fine di migliorare i tempi di risposta medi. → che deve essere trasferito insieme al job
 (chiamati Etiamo parlando di job che sono già in servizio)

→ Diciamo che il tempo di uso della CPU di un job UNIX ha una DECREASING FAILURE RATE $\Rightarrow 1-F(x) = x^{-\alpha}$, $0 < \alpha < 2$, dove $F(t)$ è la funzione di distribuzione di Pareto.

$$\text{Se } \alpha > 1 \Rightarrow E[X] = \frac{\alpha K}{\alpha - 1} \quad \text{in ottica del tempo di servizio} \quad || \quad \text{Se } \alpha > 2 \Rightarrow \sigma^2[X] = \frac{\alpha^2 K^2}{(\alpha - 2)^2 (\alpha - 1)}$$

$$\text{DENSITÀ DI PARETO: } f(x) = \alpha K x^{-\alpha-1}$$

$$K \leq x < +\infty, \quad 0 < \alpha < 2 \quad \text{Per } \alpha > 2 \text{ non abbia senso delle di}$$

→ α misura la variabilità della distribuzione e la "heavy-tailedness":

$\alpha \rightarrow 0 \Rightarrow +$ Variabilità, coda più alta.

$\alpha \rightarrow 2 \Rightarrow -$ Variabilità, coda meno alta. (PER NON PARLARE DI $\alpha > 2$)

→ Se $\alpha = 1.1 \Rightarrow$ l'1% dei job + grandi costituisce la metà del carico.

Esiste anche una versione della distribuzione di Pareto TRONCATA a destra ed è:

$$f(x) = \alpha x^{-\alpha-1} \frac{K^\alpha}{1-(K/p)^\alpha}, \quad K < x \leq p, \quad 0 < \alpha < 2$$

→ ATTUALMENTE GLI UTENTI SONO FINITI INDEPENDENTI DA α .

→ Nel nostro contesto impone un tempo di vita massimo ai job.

In conclusione, sembra aver senso migrare i job che si trovano in CPU da tanto tempo dato che è molto probabile che stiano in CPU ancora per tanto tempo (\rightarrow non conviene spendere tempo e risorse per migrare job che termineranno l'esecuzione fra 100 ms!).

Tempo di attesa e tempo di servizio riemannante:

Supponiamo che un job entri in una coda in un istante del tempo in cui non è vuota.

$$T_Q = t_c + S_{rem} \rightarrow \text{IL TEMPO DI ATTESA È UNA COMBINAZIONE TRA IL TEMPO DI SERVIZIO RICHIEGTO DEL JOB IN SERVIZIO E IL TEMPO DI SERVIZIO DI TUTTI I JOBS CORRENTEMENTE IN CODA (t_c)}$$

$$E[S_{rem}] = \frac{\lambda}{2} E[S^2]$$

\hookrightarrow Se il tempo di servizio è esponenziale $\Rightarrow E[S_{rem}] = \rho E[S]$ (perché stiamo considerando anche i casi in cui il servizio è 0)

INFATI: $E^2[S] = E[S^2] - E^2[S] = E^2[S] \Rightarrow E[S^2] = 2E^2[S] \Rightarrow E[S_{rem}] = \lambda E^2[S] = \frac{\lambda}{\mu^2} = \frac{\rho}{\mu} = PES[S]$

Nel caso generale: $E[T_Q] = \frac{\lambda}{1-\rho} E[S^2] = \frac{1}{1-\rho} E[S_{rem}] \quad (\text{PER UNA CODA M/G/1})$

$\rightarrow \frac{1}{1-\rho}$ rappresenta il tempo medio per completare i job ^{che sono} in coda nell'istante in cui arriva il job di riferimento.

Per calcolare $E[T_s]$ basta sommare $E[S]$ a $E[T_Q]$.

Inoltre, se $E[S_{rem}] = 0$, vuol dire che non c'è ^(mai) coda \Rightarrow ha senso che $E[T_Q] = 0$.

Se invece $\frac{1}{1-\rho} = 1$, vuol dire che la componente "job in coda" non dà contributo (potrebbe non esserci nessuno in coda). Questo ha senso, perché $\frac{1}{1-\rho}$ va a 1 per $\rho \rightarrow 0$ (utilizzazione trascurabile \Rightarrow probabilità trascurabile di trovare qualcuno nel centro).

\hookrightarrow Le condizioni $E[S_{rem}] = 0$, $\frac{1}{1-\rho} = 1$ vanno abbastanza a braccetto.

Teorema (Conway, Maxwell, Miller):

Tutti gli scheduling astratti e non-preemptive hanno la stessa distribuzione della popolazione nel sistema.

\downarrow Non fanno uso delle dimensioni dei job

\rightarrow Noi non consideriamo fare costringere tutti i job ad attendere lo stesso tempo indipendentemente dalle dimensioni del servizio che chiedono; è meglio far attendere una quantità di tempo proporzionale alla dimensione del servizio richiesto.

\downarrow Per questo motivo, introduciamo lo SLOW DOWN CONDIZIONATO:

$$E[Sd(x)] = \frac{E[T_s(x)]}{x} \rightarrow \text{RAFFORZO TRA TEMPO DI RISPOSTA MEDIO E DIMENSIONE (x) DEL JOB}$$

\hookrightarrow PER UNA KP: $E[Sd(x)] = 1 + \frac{\frac{1}{2} E[S^2]}{x(1-\rho)}$ \rightarrow Qui si vede che:

$E[Sd(x)] > 1$

$E[Sd(x)]$ è più grande per i job più piccoli.

INFATI: $E[T_s(x)] = x + \frac{\frac{1}{2} E[S^2]}{1-\rho} = \text{tempo di servizio} + E[T_Q]$

31/03/2022

Ci piacerebbe rendere $E[T_s]$ piccolo per i job piccoli. Ma come facciamo se non conosciamo le dimensioni dei job?

Processor sharing:

È una tecnica che consente di servire i job piccoli presto anche se non se ne conoscono le dimensioni. L'idea è che i vari job siano serviti contemporaneamente ma con un tasso di servizio ridotto di un fattore λ_s , dove λ_s è la popolazione totale correntemente nel centro; di conseguenza, non esiste più il tempo in coda.

$$\Pr\{N_s = m\} \stackrel{M/G/1/PS}{=} p^m (1-p) = \Pr\{N_s = m\} \stackrel{M/M/1/FIFO}{=}$$

Processor sharing

Stessa cosa con
 $\rightarrow E[N_s]$ e $E[T_s]$
(sono uguali tra PS e FIFO).

→ Il fatto che il processor sharing (PS) sia in grado di offrire le stesse (buone) prestazioni di un'esponentiale FIFO vuol dire che riesce a superare i costi (fatti) della variabilità dei job (vedi le prestazioni nel caso ^(piatto) dello scheduling FIFO con distribuzione del tempo di servizio di Pareto!). → In particolare, PS è migliore di FIFO quanto $C^2 >$

→ DAL PUNTO DI VISTA NOTAZIONALE, IL PROCESSOR SHARING SI DISSENA COSÌ:

$$E[T_s(x)] \stackrel{M/G/1/PS}{=} \frac{x}{1-p} \Rightarrow E[S_d(x)] \stackrel{M/G/1/PS}{=} \frac{1}{1-p} \rightarrow \begin{array}{l} \text{Lo slow down condizionato non} \\ \text{dipende dalle dimensioni dei job?} \end{array}$$

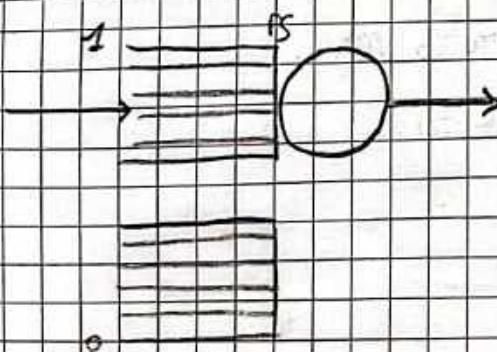
È per questo motivo che PS è considerato lo scheduling + fair.

NB: Processor sharing è necessariamente uno scheduling PREEMPTIVE.

→ È TIPO UNO SCHEDULING ROUND ROBIN (VISTO A SISTEMI OPERATIVI) CON QUANTUM DI TEMPO TENDENTE A ZERO.

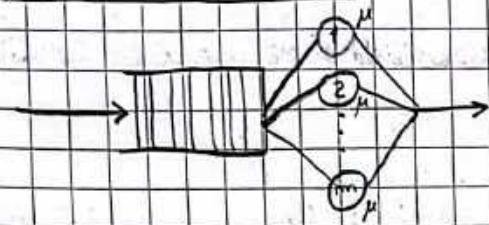
Al di là del PS, un modo per valutare le dimensioni dei job senza conoscerle a priori consiste nel vedere la loro latency (+ l'AGE è elevata, + dovrebbe essere elevato il lifetime).

SCHEDULUNG IN UNIX: FOREGROUND → BACKGROUND SCHEDULUNG:



- I job inizialmente appartenevano alla coda 1 (con maggiore priorità) dove vengono serviti con politica PS; quando la loro age supera un certo soglia t , vengono trasferiti nella coda 2, dove rimangono in attesa fino a quando la coda 1 non si svuota.

Coda multi-server:



Consideriamo il caso M/M/m.

$$P(n) = \begin{cases} \frac{1}{m!} (\lambda\mu)^m P(0) & \text{se } 0 \leq n \leq m \\ \frac{m^m}{m!} \mu^m P(0) & \text{se } n > m \end{cases}$$

Probabilità che nel centro ci siano n job.

FORMULA ERLANG-C:

$$P_Q = P_Q(n \geq m) = \sum_{n=m}^{+\infty} P(n) = \sum_{n=m}^{+\infty} \frac{m^m}{m!} \mu^n P(0) = \frac{m^m}{m!} P(0) \mu^m \sum_{n=0}^{+\infty} \mu^n = \frac{m^m}{m!} P(0) \mu^m \frac{1}{1-\mu}$$

(qui sono stati saltati dei passaggi)

$$\rightarrow E[N_q]_{\text{ERLANG}} = P_Q \frac{\mu}{1-\mu}$$

$$\rightarrow E[N_s] = P_Q \frac{\mu}{1-\mu} + (m\mu)$$

Giustamente è la probabilità media nei servizi.

$$\rightarrow E[T_q] = \frac{E[N_q]}{\lambda} = \frac{P_Q E[S]}{1-\mu}$$

dove $E[S] = \frac{E[S_i]}{m} = \frac{1}{m\mu}$

Definiamo $C := \# \text{server occupati}$:

$$E[C] = \sum_{n=0}^{m-1} n P(n) + \sum_{n=m}^{+\infty} m P(n) = m P \quad \Rightarrow P = \sum_{n=0}^{m-1} \frac{n}{m} P(n) + \sum_{n=m}^{+\infty} P(n) = \sum_{n=0}^{m-1} \frac{n}{m} P(n) + P_Q$$

$$\Rightarrow P < P_Q \Rightarrow E[T_q]_{\text{ERLANG}} = \frac{P_Q E[S]}{1-\mu} < E[T_q]_{\text{KP}} = \frac{P E[S]}{1-\mu}$$

Attesa minore rispetto al server singolo

Supponiamo di avere:

- 1) m server separati, ciascuno con una propria coda con tasso di arrivo $\frac{\lambda}{m}$ e tasso di servizio μ . \rightarrow SCENARIO TIPO: frequency-division multiplexing (FDM).
- 2) m server con una coda unica, con tasso di arrivo λ e tasso di servizio $m\mu$ per ogni server.
- 3) Server singolo con tasso di arrivo λ e tasso di servizio $m\mu$.

In tutti e tre i casi abbiamo $P = \frac{\lambda}{m\mu}$.

SCENARIO TIPO: statistical multiplexing (SM).

Concentriamoci sul tempo di risposta.

Per una M/M/1 vale:

PROBABILITÀ CHE IL SISTEMA SIA COMPL. TUTTO

$$\text{dove } P(0) = \left[\sum_{i=0}^{m-1} \frac{(\lambda\mu)^i}{i!} + \frac{(\lambda\mu)^m}{m!(1-\mu)} \right]^{-1}$$

P_Q È LA PROBABILITÀ CHE TUTTI I SERVIZI SIANO BUSY.

$$E[T_s] = E[S] = \frac{1}{\mu - \lambda} = \frac{1}{\mu(1 - \frac{\lambda}{\mu})}$$

$$\Rightarrow \begin{cases} E[T_s]^{FDI} = \frac{m}{m\mu - \lambda} = \frac{1}{\mu - \lambda/m} \\ E[T_s]^{SM} = \frac{1}{m\mu - \lambda} \end{cases}$$

Quindi si hanno
pro o contro tra le
due configurazioni.

Inoltre, il merge di tutti i busi relativamente
a una variabile porta a un unico flusso con
una variabilità + alta (e sappiamo che la variabi-
lità non è un indice positivo).

\rightarrow m volte più piccola del frequency-division multiplexing.
Tuttavia, a differenza di FDI, non garantisce alcuna garanzia sul servizio
(vedi Fatal?).

01/04/2022

Next-event simulation:

È un approccio più generale e più flessibile per la simulazione discrete-event.

Definizione:

Lo STATO è una caratterizzazione completa del sistema che lo ~~s~~ descrive in un certo istante
di tempo.

\hookrightarrow Nel caso di un single server queue, lo stato è in numero di job correntemente nel nido.

Definizione:

Un EVENTO è un'occorrenza che cambia lo stato del sistema.

Lo stato del sistema non può cambiare se non occorre un evento. Tuttavia, possono esistere
degli eventi che non modificano lo stato; un esempio di questi eventi è la chiusura degli arrivo-
ri nel sistema (si fa in modo che non arriverà più alcun job). *Detti utili.

Definizione:

Il CLOCK DI SIMULAZIONE rappresenta il valore corrente del tempo simulato.

Definizione:

Lo SCHEDULER rappresenta un meccanismo che faccia scorrere il tempo, in modo tale
da poter fare in modo che alcuni eventi accadano prima di altri. \rightarrow INVERSO CHE GLI EVENTI OCCORRENNO
NELL'ORDINE CORRETTO.

\hookrightarrow Lo scheduling che usiamo noi è il NEXT-EVENT time.

Definizione:

L'EVENT LIST è una struttura dati contenente il tempo della prossima occorrenza di
ogni tipo di evento.

Per costruire il next-event simulator:

- 1) Definire lo stato del sistema (costruendo l'insieme delle variabili di stato).
- 2) Individuare i tipi di evento.
- 3) Per ciascun tipo di evento, generare un algoritmo che lo gestisca.

ALGORITMO PER LA NEXT-EVENT SIMULATION:

- 1) INIZIALIZZAZIONE: imposta il clock di simulazione e il primo istante di tempo della prima occorrenza per ogni tipo di evento.
- 2) PROCESSAMENTO DELL'EVENTO CORRENTE: scansiona l'event list per determinare l'evento più imminente; fa' avanzare il clock di simulazione; aggiorna lo stato corrente del sistema.
- 3) SCHEDULAZIONE DI NUOVI EVENTI: posiziona eventuali nuovi eventi all'interno della event list.
- 4) TERMINAZIONE: continua a fare avanzare il clock di simulazione e a gestire gli eventi finché non potrà soddisfatta la condizione di terminazione.

05/04/2022

Fattore di scala:

Supponiamo di avere un servizio multiplo e che, a un traffico, il tasso di arrivo passi da ~~λ~~ a $\alpha\lambda$ e il tasso di servizio di tutti i server passi da μ a $\alpha\mu$.

Allora:

$$\rightarrow p \text{ rimane uguale: } \frac{\lambda}{m\mu}$$

che \rightarrow PASSA DA $\frac{1}{m\mu}$ A $\frac{1}{\alpha m\mu}$

$$\rightarrow E[S_i] \text{ passa da } \frac{1}{\mu} \text{ a } \frac{1}{\alpha\mu}, \text{ con } E[S] = \frac{E[S_i]}{m} \quad (m = \# \text{ server}).$$

$$\rightarrow E[TT_q]_{m,\alpha} = \frac{P_q E[S]_{m,\alpha}}{1-p} = \frac{P_q}{m\alpha\mu(1-p)} = \frac{1}{\alpha} \frac{P_q E[S]_{m,1}}{1-p} = \frac{1}{\alpha} E[TT_q]_{m,1}$$

$$\hookrightarrow \text{Da qui, si vede banalmente che } E[TT_q]_{m,\alpha} = \frac{1}{\alpha} E[TT_q]_{m,1} \quad \text{infatti: } E[TT_q]_{m,\alpha} = \frac{1}{\alpha} E[TT_q]_{m,1} + \frac{1}{\alpha} E[TT_q]_{m,1} = \frac{1}{\alpha} [E[TT_q]_{m,1} + E[TT_q]_{m,1}] = \frac{1}{\alpha} E[TT_q]_{m,1}$$

→ Usando Little, invece, si dimostra che tra i due casi le popolazioni medie non cambiano.

$$\hookrightarrow \text{e.g. } E[N_q]_{m,\alpha} = (\alpha\lambda) E[TT_q]_{m,\alpha} = \alpha\lambda \frac{1}{\alpha} E[TT_q]_{m,1} = \lambda E[TT_q]_{m,1} = E[N_q]_{m,1}$$

Esempio [caso $\downarrow - m$ CENTRI CON m CODE DISTINTE]:

$$\lambda = 4 \text{ j/s}$$

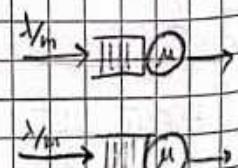
$$m = 4$$

$$\mu = 1,5 \text{ j/s} \Rightarrow E[S] = 0,66667$$

$$p = \frac{\lambda}{m\mu} = 0,666667$$

$$[E[TT_q]] = \frac{1}{\mu - \lambda} = 2 \text{ s}$$

$$[E[TT_q]] = \frac{p E[S]}{1-p} \approx 1,336 \text{ s}$$



CASO 2 - UN UNICO CENTRO CON M SERVENTI:

$P(0) = 0,059857$ (vedere formula di Erlang) \rightarrow PROBABILITÀ CHE I 4 SERVENTI SIANO TUTTI VUOTI

$$P_a = \frac{(4p)^4}{4! (1-p)} P(0) = 0,37847 \rightarrow$$

PROBABILITÀ CHE I 4 SERVENTI SIANO TUTTI PIENI

$$E[T_s] = \frac{P_a E[S]}{1-p} + E[S_i] = 0,855992 \text{ s}$$

dove $E[S_i] = \frac{1}{\mu} = 0,666667$

$$E[T_q] = 0,189292 \text{ s}$$

CASO 3 - UN UNICO CENTRO CON UN SERVENTE:

$$E[T_s] = \frac{1}{m\mu - \lambda} = 0,5 \text{ s}$$

$$E[T_q] = 0,3334 \text{ s} = \frac{P E[S]}{1-p} = \frac{P}{1-p} \cdot \frac{1}{m\mu}$$

\rightarrow Riforce i calcoli con $p = 0,533334$ e $P = 0,8$

Possiamo notare che il caso 2 presenta un tempo di accodamento (di attesa) più breve e il caso 3 presenta un tempo di risposta più breve.

PROCESSI DI MARKOV

Definizione:

Un PROCESSO STOCASTICO è un insieme di variabili random che indicano un'evoluzione di una certa variabile X nel tempo.

$$\rightarrow \{X(t_1), X(t_2), \dots\}$$

Definizione:

Lo SPAZIO DEGLI STATI è l'insieme dei valori ammissibili per una variabile X .

$$\hookrightarrow E = \{s_0, s_1, \dots\}$$

\rightarrow Se siamo in grado di individuare un numero degli stati FINITO o NUMERABILE, in realtà, non parliamo di processo, bensì di CATENA.

Definizione:

$$(*) P\{X(t_{n+1}) = x_{n+1} \mid X(t_n) = x_n, X(t_{n-1}) = x_{n-1}, \dots, X(t_0) = x_0\} = P\{X(t_{n+1}) = x_{n+1} \mid X(t_n) = x_n\}$$

Una CATENA DI MARKOV è una catena la cui evoluzione nel futuro NON dipende dalla sua storia passata, bensì solo dallo stato attuale. \rightarrow MEMORYLESS PROPERTY

Definizione:

La DISTRIBUZIONE DI PROBABILITÀ ISTANTANEA indica la probabilità che la variabile X assuma un certo valore a un istante di tempo t :

$$P\{X(t) = s_i\} = \pi(s_i, t)$$

↳ Se lo spazio degli stati è finito e il processo è IRRIDUCIBILE ed ERGODICO, allora:

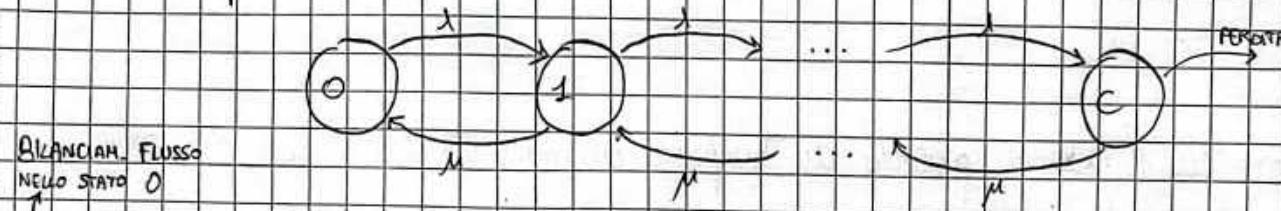
$$\lim_{t \rightarrow \infty} \pi(s_i, t) = \pi(s_i) = \text{PROBABILITÀ CHE } X \text{ ASSUMA IL VALORE } s_i \text{ IN GENERALE (INDIPENDENTEMENTE DAL TEMPO) AVVERO NELLO STAZIONARIO}$$

Tuttasi di una probabilità stazionaria.

Catino con servente singolo e buffer finito:



→ λ' ce lo possiamo calcolare solo con una catena di Markov:



$$\pi_0 \lambda = \pi_1 \mu \Rightarrow \pi_1 = \frac{\lambda}{\mu} \pi_0$$

$$\pi_1 (\lambda + \mu) = \pi_0 \lambda + \pi_2 \mu \Rightarrow \pi_2 = \left(\frac{\lambda}{\mu}\right)^2 \pi_0$$

$$\rightsquigarrow \text{In generale: } \pi_i = \left(\frac{\lambda}{\mu}\right)^i \pi_0$$

$$\rightsquigarrow \text{Inoltre, trattandosi di probabilità, vale: } \sum_{i=0}^c \pi_i = 1$$

Dai tutte queste equazioni, siamo in grado di calcolare $\pi_0, \pi_1, \dots, \pi_c$.

$$\rightarrow P\{X(t) \text{ loss}\} = p_{\text{loss}} = \pi_c = \left(\frac{\lambda}{\mu}\right)^c \pi_0 = \text{percentuale di tempo in cui il sistema è sotto}$$

di questo punto:

$$\lambda' = \lambda(1 - p_{\text{loss}}) = \lambda(1 - \pi_c) \quad ; \quad p = \frac{\lambda'}{\mu} = \frac{\lambda(1 - \pi_c)}{\mu}$$

Esempio:

$$C=4; \quad \lambda=\mu=5 \text{ j/s}$$

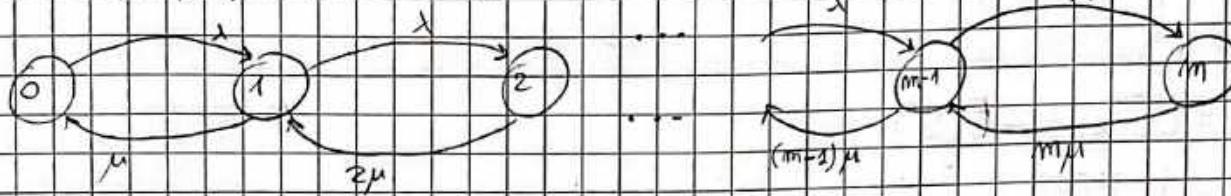
$$\pi_i = \left(\frac{\lambda}{\mu}\right)^i \pi_0 \quad ; \quad \sum_{i=0}^4 \pi_i = 1 \Rightarrow \pi_0 = \pi_1 = \pi_2 = \pi_3 = \pi_4 = \frac{1}{5}$$

$$p_{\text{loss}} = \pi_4 = \frac{1}{5} \quad ; \quad \lambda' = 5(1 - \frac{1}{5}) = 4 = X \quad ; \quad p = \frac{\lambda'}{\mu} = \frac{4}{5} = 0,8$$

Assume una
distribuzione stazionaria

M-server loss system:

$\sum_{k=0}^m \frac{1}{k!} \mu^k M/M/m/m$ (senza coda):



$$\text{Da qui viene che: } \pi_i = \left(\frac{\lambda}{\mu}\right)^i \frac{1}{i!} \pi_0$$

$$\pi_m = \left(\frac{\lambda}{\mu}\right)^m \frac{1}{m!} \pi_0$$

< Per del caso della Erlang-C [MULTISERVER CON CODA INFINTA]

Notiamo che π_0 è MAGGIORRE del p_0 del caso della Erlang-C: se mai abbandoniamo la coda, abbiamo molta più perdita, per cui è più difficile che un job che abbandona un server venga istantaneamente rimpiazzato da un altro.

07/04/2022

Priority scheduling:



I job nella coda 2 vengono serviti solo se la coda 1 è vuota e così via.

Tra l'altro, è possibile avere prelazione opposta. Se c'è prelazione, un job con priorità j viene interrotto dall'arrivo di un altro job con priorità i < j.

PRIORITÀ ASTRATTA (NON SIZE-BASED) SENZA PRELAZIONE:

$$E[S_k] = \frac{1}{\mu_k}$$

$$E[S] = E[S_k] = \frac{1}{\mu_k} = \frac{1}{\mu}; \quad p = \sum_{k=1}^m p_k$$

↳ Analogamente, $\sigma^2[S_k] = \sigma^2[S] \forall k$

TIENI CONTO CHE PIÙ LA PRIORITÀ HA INDICE MINORE, PIÙ È UNA PRIORITÀ ALTA.

→ È chiaro che non abbiamo fatto assunzioni sulle dimensioni dei job tra le diverse code, per cui possiamo considerarle tutte uguali fra loro.

Quando un job con priorità k arriva nel centro, deve attendere:

→ Il job che è già in servizio per un tempo S_{rem} . $\rightarrow E[S_{\text{rem}}] = \frac{1}{2} E[S^2]$

→ Il job che sono già nel centro e con priorità tra 1 e $\geq k$. $\rightarrow \frac{1}{1 - \sum_{i=k+1}^m p_i}$

→ Il job che avverranno successivamente con priorità strettamente minore di $\geq k$.

$$\text{IN DEFINITIVA: } E[CT_{Q_k}] = \frac{\frac{1}{2} E[S^2]}{(1 - \sum_{i=1}^k p_i)(1 - \sum_{i=k+1}^{m-1} p_i)}$$

È abbastanza ovvio che $E[T_{q_k}] \leq E[T_{q_{k+1}}]$ (più la priorità è bassa e maggiore sarà il tempo di attesa).

Dopodiché:

$$E[TS_k] = E[T_{q_k}] + E[S]$$

$$E[N_{q_k}] = \lambda_k E[T_{q_k}]$$

$$E[N_{S_k}] = \lambda_k E[TS_k] = E[N_{q_k}] + p_k$$

Ma per quanto riguarda le performance globali?

$$E[T_q]^{NP+priority} = E[E[T_{q_k}]] = \sum_{k=1}^n p_k E[T_{q_k}] \quad \text{dove } p_k = \frac{\lambda_k}{\lambda}, \quad n = \# \text{ classi di priorità}$$

$$E[TS]^{NP+priority} = E[T_q]^{NP+priority} + E[S]$$

$$\rightarrow p_k = \lambda_k E[S_k] = \lambda_k E[S] = \lambda p_k E[S] = \rho p_k$$

Rispetto a uno scheduling che non usa code di priorità:

→ I job con priorità alta (pari a $\frac{1}{2}$) sperimentano un tempo di attesa minore.

→ I job con priorità bassa (pari a $\frac{1}{n}$) sperimentano un tempo di attesa maggiore.

→ Le prestazioni complessive NON VARIANO: $E[TS]^{NP+priority} = E[T_q]^{NP+priority} = \frac{1}{2} \frac{E[S^2]}{1-\rho}$

$$\therefore E[TS]^{NP+priority} = E[T_q]^{NP+priority}$$

12/04/2022

Preemptive priority:

È possibile applicare la priorità nel momento in cui arriva un job con priorità più elevata di quella che è attualmente in servizio.

Anche in questo caso, quando un job con priorità k arriva nel centro, deve attendere le stesse tre cose elencate nella pagina precedente. D'altra parte, il tempo medio di attesa è definito come:

$$E[T_{q_k}]^{P+priority} = \frac{\frac{1}{2} \sum_{i=1}^k \lambda_i E[S^2]}{(1 - \sum_{i=1}^k p_i)(1 - \sum_{i=1}^{k-1} p_i)}$$

$$\text{Globalmente: } E[T_q]^{P+priority} \leq E[T_q]^{NP+priority} = E[T_q]^{KP}$$

(di fatto, per ogni classe di priorità)

Stavolta abbiamo guadagnato qualcosa globalmente rispetto alla KP.

Definizione:

IL TEMPO DI SERVIZIO VIRTUALE $S_{vrt,k}$ è il tempo di servizio di un job considerando anche i momenti in cui il suo servizio viene interrotto (e, quindi, l'ATTESA DA PREEMPTION).

$$E[S_{vrt,k}] = \frac{E[S]}{1 - \sum_{i=1}^{k-1} p_i}$$

$$\rightarrow E[T_{S_k}]^{\text{P-priority}} = E[T_{Q_k}]^{\text{P-priority}} + E[S_{vrt,k}]$$
$$\rightarrow E[T_{S_k}]^{\text{NP-priority}} = E[T_{Q_k}]^{\text{NP-priority}} + E[S]$$

{ Non è possibile fare un confronto preciso
sul tempo di risposta.
} HA

Per l'esponentiale, grazie alla MEMORYLESS PROPERTY, esattamente fatto ciò che si guadagna nel tempo di attesa con la preemption lo si perde nel tempo di servizio.

↳ PERCIÒ, PER L'ESPONENZIALE: $E[T_{S_k}]^{\text{P-priority}} = E[T_{S_k}]^{\text{NP-priority}} = E[T_{S_k}]^{\text{KP}}$ (e vbb)

In relazione alle slide EX9...

$$E[T_{Q_1}] = \frac{E[S]}{1 - p_1} = "QoS"$$

← L'espressione è questa (con $pE[S]$) perché stiamo assumendo un tempo di servizio esponentiale.

$$p_1 = \frac{1}{P} - \frac{E[S]}{QoS} = 1,25 - 0,8889 \approx 0,36 \quad (\text{con } E[S] = 0,4, "QoS" = 0,4s, P = 0,8)$$

$$\text{Se invece "QoS" = 0,39} \Rightarrow p_1 \approx 0,22$$

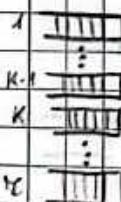
Ora vediamo capire quante classi di priorità sperimentano un attesa minore rispetto al caso delle KP, ovvero per quali K vale: \rightarrow Sembra che qui siamo tornati allo scheduling non-preemptive.

$$E[T_{Q_k}] = \frac{E[S_{vrt}]}{(1 - \sum_{i=1}^k p_i)(1 - \sum_{i=1}^{k-1} p_i)} \leq \frac{E[S_{vrt}]}{1 - p}$$

$$\Rightarrow (1 - \sum_{i=1}^k p_i)(1 - \sum_{i=1}^{k-1} p_i) > 1 - \sum_{i=1}^k p_i$$

$$\Rightarrow 1 - \sum_{i=1}^{k-1} p_i - \sum_{i=1}^k p_i + \sum_{i=1}^k p_i \sum_{i=1}^{k-1} p_i > 1 - \sum_{i=1}^k p_i - \sum_{i=k+1}^{\infty} p_i$$

$$\Rightarrow \sum_{i=k+1}^{\infty} p_i > \sum_{i=1}^{k-1} p_i (1 - \sum_{i=1}^k p_i)$$



← Questo dimostrerebbe indicando che il tasso di occupazione delle classi con priorità minore deve essere maggiore di una porzione ($\sum_{i=1}^{k-1} p_i$ è minore di 1) del tasso di occupazione delle classi con priorità maggiore.

PRIORITÀ MINORE = PRIORITÀ DI INIZIO PIÙ ALTA (e viceversa).

11/06/2022

Esercizio per casa:

$$E[S] = 0,4 \text{ s} \quad \lambda = 0,8 \text{ J/s}$$

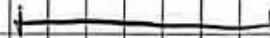
GUADAGNO SE $T_a < 0,1$

$$C_1 = 5$$

PERDITA SE $T_a > 0,1$

$$C_2 = 3$$

Si vogliono calcolare le probabilità p_1, p_2 di appartenere alla coda con più priorità e di appartenere alla coda con meno priorità affinché $R = p_1 C_1 - p_2 C_2$ sia massimizzato.



Altro scenario:

→ Tasso di arrivo $\lambda = 1 \text{ J/s}$ (randic).

→ La latenza media è $Z = 6 \cdot 10^5$ operazioni e ha una caratterizzazione esponenziale.

Vogliamo esaminare due configurazioni possibili:

1) Un unico server di capacità $C = 10^6 \text{ operat./s}$

2) Un dual-core di capacità C_2 per ciascun core.

Requisiti di QoS:

a) Tempo di attesa medio $T_a < 0,15 \text{ s}$.

b) Per almeno il 35% degli arrivi, $T_s < 0,5 \text{ s}$.

→ Chiaramente $E[S] = Z/C = 0,4 \text{ s}$; $\rho = \lambda E[S] = 0,4$

$$1) E[T_a] = \frac{\rho E[S]}{1-\rho} \approx 0,26 \text{ s}$$

caso con 1 A CODA SINGOLA

$$E[T_{a_1}] = \frac{\rho_1 E[S]}{1-\rho_1} \approx 0,065 \text{ s}$$

dove $\rho_1 = 35\% \text{ di } 0,4$

$$E[T_{a_2}] = \frac{\rho_2 E[S]}{(1-\rho_1)(1-\rho)} = 0,3 \text{ s}$$

$$\rightarrow E[T_a]_{\text{totale}} = 0,22 \text{ s}$$

CASO CON
LT CODA
DI PRIORI
TA (con
prioritazione)

$$2) E[S_i] = \frac{Z}{C_2} = 2E[S] = 0,8 \text{ s}$$

$$E[T_a]_{\text{totale}} = \frac{\rho_i E[S]}{1-\rho}$$

$$\rightarrow \rho(0) = \left[1 + 2\rho + \frac{(2\rho)^2}{2(1-\rho)} \right]^{-1} = 0,42857 \quad (\text{qui } m=2)$$

$$\rightarrow \rho_0 = \frac{(2\rho)^2}{2(1-\rho)} \quad \rho(0) = 0,22857 \Rightarrow E[T_a]_{\text{totale}} \approx 0,15238 \text{ s}$$

↳ Come abbiamo visto, in nessuna configurazione considerata, si riesce ad avere un tempo di attesa minore a 0,15 s.

⇒ Potremmo dover ricorrere a un MULTISERVER CON CLASSI DI PRIORITY.

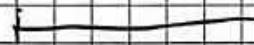
Multiserver con classi di priorità:

Supponiamo di avere priorità. Allora:

$$E[T_q] = p_1 \frac{P_1 E[S]}{1-p_1} + p_2 \frac{P_2 E[S]}{(1-p_1)(1-p_2)}$$

dove P_i viene calcolato in funzione di p_i .

Qui abbiamo fatto l'ipotesi di avere 2 classi di priorità e 2 server. Lo scenario è facilmente estendibile a m server ma, per quanto riguarda le classi di priorità, è molto più complesso (bisognerebbe ricorrere alla considerazione di probabilità).



→ IN EFFETTO CON EQUAL-CORE GIÀ IL SECO TEMPO DI SERVIZIO MEDIO E[S] SARA' APPROSSIMATIVAMENTE A 0,15 T < 0,5 S.

Tornando all'esercizio di prima, si può verificare che si riesce a soddisfare il requisito di QoS (b) solo col servente singolo (uff oh?).

→ A SECONDA DELL'OGGETTIVO, IL SISTEMA DA IMPLIMENTARE PUÒ ESSERE COMPLETAMENTE DIVERSO (SENZA UN OGGETTIVO NON SI PUÒ DARE UNA RISPOSTA).

15/01/2022

STATISTICA CAMPIONARIA

Prevede l'interpretazione dei dati (inferire delle informazioni) su un campione che sia rappresentativo per la popolazione globale.

Nella nostra trattazione, le statistiche campionarie possono essere:

→ Interne a una Single Run (within-run).

→ Between-runs.

Definizione:

→ ROOT-MEAN-SQUARE (RMS)

$$d(x) = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2}$$

è una funzione che misura la dispersione da qualunque valore x .

Per $x = \bar{x}$ (media), $d(x)$ diviene uguale alla deviazione standard⁽³⁾ e assume il valore minimo.

Consideriamo l'intervallo $[\bar{x} - k_s, \bar{x} + k_s]$, dove \bar{x} = media campionaria e s = dev. standard campionaria.

Definiamo l'insieme $S_K = \{x_i \mid \bar{x} - Ks < x_i < \bar{x} + Ks\}$.

Sia $p_K = \frac{|S_K|}{n}$ la (probabilità) proporziona degli x_i che cadono nell'intervallo.

Per la disegualanza di Chebyshev: $p_K \geq 1 - \frac{1}{K^2}$. \rightarrow È UNA DISEGUALANZA MOLTO + BLANDA DI QUELLO CHE VALE NELLA REALITÀ.

Trasformazioni lineari:

Spesso vengono usate per convertire le grandezze da un'unità di misura a un'altra.

Sono del tipo: $x'_i = ax_i + b$.

$$\begin{aligned} \bar{x}' &= a\bar{x} + b \\ (s')^2 &= a^2 s^2 \Rightarrow s' = |a|s \end{aligned}$$

ESEMPIO:

x_1, x_2, \dots, x_n sono misurati in secondi e vogliamo trasformarli in minuti.

$$x'_i = \frac{x_i}{60} \quad \forall i = 1, \dots, n \quad (a = \frac{1}{60}, b = 0)$$

$$\Rightarrow \bar{x}' = \frac{\bar{x}}{60}; \quad s' = \frac{s}{60}$$

ALTRÒ ESEMPIO:

Standardizzazione dei dati: $a = \frac{1}{s}$, $b = -\bar{x}/s \Rightarrow \bar{x}' = \frac{\bar{x}}{s}; \quad s' = \pm 1$.

Trasformazioni non lineari:

Soltanente danno luogo a un risultato baleano.

Sia A un insieme fissato; allora: $x'_i = \begin{cases} 1 & \text{se } x_i \in A \\ 0 & \text{altrimenti.} \end{cases}$

$$\begin{aligned} \bar{x}' &= p, \quad s' = \sqrt{p(1-p)} \\ \text{dove } p &= \#\{x_i \in A\} / n \end{aligned}$$

SE CI TAI CASO SONO PROBABILMENTE STANDARD DI UNA V.R. BERNOULLI

ESEMPIO: TEMPO DI ATTESA

Sia x_i il "ritardo" per il job i , e sia $A^+ \equiv \mathbb{R}^+$; allora $x'_i = 1 \Leftrightarrow x_i > 0$.

\hookrightarrow In questo esempio, ci preoccupiamo solo dei job che aspettano in coda senza di soluzione sul tempo di attesa.

19/04/2022

Fa uso delle seguenti relazioni: $\bar{x}_i = \bar{x}_{i-1} + \frac{1}{i} (x_i - \bar{x}_{i-1})$; $s_i = \sqrt{\frac{1}{i} ((x_i - \bar{x}_{i-1})^2)}$

Algoritmo di Welford:

L'idea è quella di calcolare media campionaria e deviazione standard campionaria ~~parzialmente~~ sicuramente onde evitare overflow e la necessità di memorizzare l'intero campione per portare a termine la campionazione.

Per quanto riguarda invece le statistiche time-averaged...

\rightarrow MEDIA DEL SAMPLE-PATH: $\bar{x} = \frac{1}{T} \int_0^T x(t) dt$

$$\rightarrow \text{VARIANZA DEL SAMPLE-PATH: } S^2 = \frac{1}{T} \int_0^T (X(t) - \bar{x})^2 dt = \frac{1}{T} \int_0^T X^2(t) dt - \bar{x}^2$$

→ Il sample-path rappresenta i valori assunti dalla variabile x (x_1, x_2, \dots).

QUESTO PERCHÉ NELLA PRACTICA, LE STATISTICHE TIME AVERAGED (IL PERCENTILE VENDETTA FUNZIONE DI DISTRIBUZIONE)

$$\begin{aligned} &\rightarrow \text{Anche i valori del sample path lo calcoliamo tramite roll-off:} \\ &S_i = t_i - t_{i-1} \quad \text{roll-off} \\ &S_i = \bar{x}_{i-1} + \frac{S_i}{t_i} (x_i - \bar{x}_{i-1}) \\ &y_i = v_i + \frac{S_i}{t_i} (t_i - t_{i-1})^2 \end{aligned}$$

Generazione di variabili aleatorie discrete:

Definizione:

Sia X una r.a. discreta e sia F la sua funzione cumulativa (cdf). Allora, l'inversa di F (la idf F^*) è definita come:

$$F^*: (0,1) \rightarrow X$$

Jusione dei valori
che possono essere assunti
da X

$$F(u) = \min_x \{x : u < F(x)\}$$

(di distribuzione)

cdf = FUNZIONE DI DISTRIBUZIONE
idf = FUNZIONE INVERSA DI DISTRIB.
pdf = FUNZIONE DI DENSITÀ

È il numero reale tale per cui la funz. di distribuz. $F(x)$ è più grande di u .

In alcuni casi la $F^*(u)$ può essere scritta esplicitamente. Ad esempio, per una bernulliana:

$$F^*(u) = \begin{cases} 0 & \text{se } 0 < u < p \\ 1 & \text{se } p \leq u \leq 1 \end{cases}$$

Teorema:

Sia X una r.a. discreta con idf $F^*(\cdot)$. Sia U una r.a. continua uniforme definita in $(0,1)$. Sia Z una r.a. definita come $Z = F^*(U)$. Allora:

X, Z sono identicamente distribuite.

Grazie a questo teorema, ci basta sapere com'è fatta la idf $F^*(\cdot)$ della nostra variabile aleatoria discreta e sfruttare le seguenti due linee di codice:

$u = \text{Random();}$

ritorna $F^*(u)$;

→ IN TAL MODO RIUSCIRAMO A SIMULARE LE VARIABILI ALEATORIE DISCRETE.

In particolare, si riesce a generare la Bernulliana, la Discreta Uniforme e la Geometrica (le altre possono richiedere meccanismi diversi perché magari non è mica la idf $F^*(\cdot)$).

↳ comunque per motivi di efficienza.

Noncamento:

A volte i valori realistici di una r.a. sono ristretti a un sottoinsieme.

Sia X una r.a. con valori possibili $\mathcal{X} = \{0, 1, 2, \dots\}$ e con $F(x) = P(X \leq x)$.

Supponiamo di voler restringere i valori possibili per X a un intervallo discreto $[a, b]$.

Immagistolo ci chiediamo quante masse di probabilità stanno buttando:

$$\alpha = P(X < a), \beta = P(X > b) \Rightarrow \text{Nel sistema buttando } \alpha + \beta = F(a-1) + 1 - F(b)$$

$$\Rightarrow F(a \leq X \leq b) = F(b) - F(a-1) \rightsquigarrow \text{ESSENZIALMENTE, LA NOSTRA RESTRIZIONE E' CORRETTA SE} \\ F(b) \approx 1,0 \wedge F(a-1) \approx 0,0.$$

(INSIGNIFICANTE)

• Se conosciamo $a, b \Rightarrow$ possiamo calcolare:

$$\alpha = P(X < a) = F(a)$$

\rightarrow LA TRASFORMAZ. È ESATTA.

• Se conosciamo $\alpha, \beta \Rightarrow$ possiamo calcolare:

$$a = F^*(\alpha); \quad b = F^*(1-\beta)$$

\rightarrow LA TRASFORMAZ. È NON ESATTA

perché X è DISCRETA e $F^*(\cdot)$ È UN'APPROXIMAZ. DELLA FUNZIONE INVERSA DI $F(\cdot)$ (vai a vedere lo spazio).

(in generale)

EFFETTI DEL TRONCAMENTO:

Spesso il troncamento non porta a particolari conseguenze; chiaramente è un'operazione utile per motivi di efficienza.

↳ Alcune altre volte però il troncamento è significativo e dà luogo a una v.a. differente (per cui bisogna stare attenti a troncare in maniera corretta e a redistribuire i valori che sono stati troncati).

↳ IN QUESTO TALI DA RICORRERE ALLA RIDISTRIBUZIONE DI PROBABILITÀ

SE VOGLIANO TRONCARE UNA DISTRIBUZIONE COSTRIGENDOLA TRA ZONE VALORI a, b , ABBIANO CHE:

$$f_t(d) = \frac{f(d)}{F(b)-F(a)}$$

ove $f_t(d)$ indica la densità della v.a. troncata.

21/06/2022

Generazione di Variabili aleatorie continue:

Qui la inverse distribution function (idf) è a tutta gli effetti la funzione inversa delle $F(x)$.

Teorema:

Sia X una v.a. continua con idf $F^{-1}(\cdot)$. Sia U una v.a. continua uniforme definita in $(0,1)$. Sia Z una v.a. definita come $Z = F^{-1}(U)$. Allora:

X, Z sono identicamente distribuite. → Di conseguenza, per simulare le v.a. continue, basta inserire:

$u = \text{Random}();$
 $\text{return } F^{-1}(u);$

è il SUPPERO

Troncamento:

Sia X una v.a. continua che può assumere i valori appartenenti a un insieme \mathcal{X} e con funzione di distribuzione ~~$F(x) = P(X \leq x)$~~ $F(x) = P(X \leq x)$. Supponiamo di voler troncare la v.a. nell'intervallo $(a, b) \subset \mathcal{X}$.

→ $X \leq a$ con probabilità $F(a)$.

→ $X \geq b$ con probabilità $1 - F(b)$.

→ X assume valori in (a, b) con probabilità $F(b) - F(a)$.

$$\alpha = P(X \leq a) = F(a)$$

$$\beta = P(X \geq b) = 1 - F(b)$$

In questo caso, sia se partiamo da a, b per calcolare α, β , sia se partiamo da

α, β per calcolare a, b , abbiamo delle trasformazioni ESATTE.

$$\rightarrow a = F^{-1}(\alpha) ; \quad b = F^{-1}(1-\beta)$$

Size-based priority scheduling:

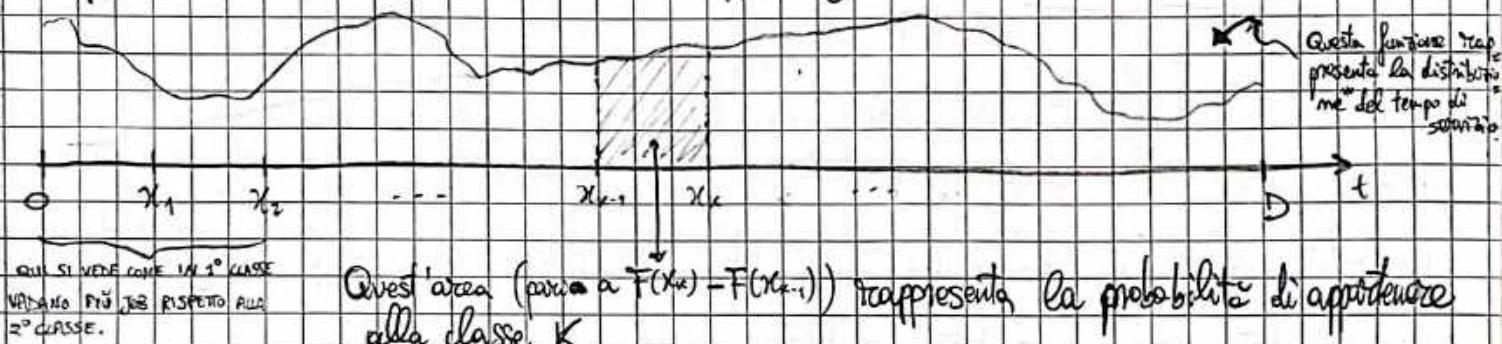
Ricorda che i job di dimensioni minori abbiano più priorità e viceversa.

→ Se h, K sono due priorità tali che $h < K \Rightarrow E[S_h] < E[S_K]$.

$$\sim E[S_K] = \frac{1}{\mu_K} ; \quad P_K = \lambda \cdot E[S_K] ; \quad \lambda = \sum_{i=1}^n \lambda_i ; \quad \rho = \sum_{i=1}^n p_i$$

→ Sia gli ordini che i servizi dipendono dalla forma della distribuzione?

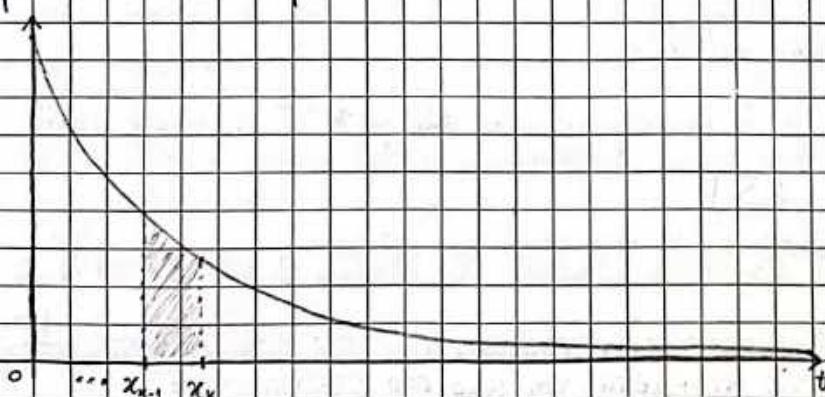
Supponiamo che il tempo di servizio s appartenga all'intervallo $(0, D]$



$$E[S_K] = \int_{x_{k-1}}^{x_k} t f^n(t) dt , \text{ dove } f^n(t) = \frac{f(t)}{F(x_k) - F(x_{k-1})} \quad [\text{è la } f(t) \text{ normalizzata all'intervallo } (x_{k-1}, x_k)]$$

26/04/2022

Supponiamo che il tempo di servizio abbia una distribuzione di probabilità esponenziale:



$$\cdot \lambda_K = \lambda P_K = \lambda \Pr\{\text{appartenere alla classe } K\} = \lambda (F(x_k) - F(x_{k-1}))$$

$$\cdot P_K = \lambda_K E[S_K]$$

$$\cdot \sum_{i=1}^n p_i \text{ è la somma dei prodotti di due serie: } \lambda_1, \lambda_2, \dots, \lambda_n \text{ e } E[S_1], E[S_2], \dots, E[S_n]$$

↪ La somma di prodotti minima è data da due serie tali che una è monotona crescente e l'altra è monotona decrescente.

Di fatto, nel caso nostro:

- $\lambda_1 > \lambda_2 > \dots > \lambda_n$ ar Nel caso astratto non c'è una relazione.
- $E[S_1] \geq E[S_2] \geq \dots \geq E[S_n]$ ar Nel caso astratto sono uguali.

$$\Rightarrow \sum_{i=1}^n p_i^{\text{SB}} \leq \sum_{i=1}^n p_i^{\text{Abstract}}$$

dove $\text{SB} = \text{Size-Based}$.

Tornando a p_k : $p_k = \lambda_k E[S_k] = \lambda_k (F(x_k) - F(x_{k-1})) \int_{x_{k-1}}^{x_k} t f(t) dt = \lambda_k \int_{x_{k-1}}^{x_k} t f(t) dt$

- Quindi il tempo di attesa (per una classe k) si calcola esattamente come nel caso delle classi di priorità astratte. In particolare:

$$E[T_{Q_k}]^{\text{SB-NP-priority}} = \frac{\lambda_k E[S^2]}{(1-\lambda) \int_0^{x_k} t f(t) dt (1-\lambda) \int_0^{x_{k-1}} t f(t) dt}$$

$$\int_0^{x_k} t f(t) dt = \sum_{i=1}^n \lambda_i \int_{x_{i-1}}^{x_i} t f(t) dt = \sum_{i=1}^n p_i$$

Tuttavia, $E[T_{Q_k}]^{\text{SB-NP}} \leq E[T_{Q_k}]^{\text{abstract-NP}}$

↳ Infatti, vale: $[(1 - \sum_{i=1}^k p_i)(1 - \sum_{i=1}^{k-1} p_i)]^{\text{SB-NP}} \geq [(1 - \sum_{i=1}^k p_i)(1 - \sum_{i=1}^{k-1} p_i)]^{\text{abstract-NP}}$

$$\Rightarrow \left[\sum_{i=1}^k p_i \right]^{\text{SB-NP}} \leq \left[\sum_{i=1}^k p_i \right]^{\text{abstract-NP}} \quad \forall k$$

QUESTO È VERO PERCHÉ, COME DICCEVANO PRIMA, NEL CASO SIZE-BASED: $\lambda_1 > \lambda_2 > \dots > \lambda_n$; $E[S_1] \geq E[S_2] \geq \dots \geq E[S_n]$

- Per quanto riguarda il tempo di risposta, invece, alcune classi nel size-based ci guadagnano rispetto al caso astratto, ma altre classi ci perdono. ↳ Non si possono tuttavia fare delle considerazioni generali.

Dal punto di vista delle prestazioni globali:

$$\rightarrow E[T_Q]^{\text{SB-NP}} \leq E[T_Q]^{\text{abstract-NP}}$$

poiché la disegualanza vale per tutte le singole classi.

$$\rightarrow E[S]^{\text{SB-NP}} = E[S]^{\text{abstract-NP}} = E[S]$$

$$\rightarrow E[T_s]^{\text{SB-NP}} \leq E[T_s]^{\text{abstract-NP}}$$

↳ ANDIAMO A GUADAGNARE! E QUESTO VALE PER TUTTE LE DISTRIBUZIONI DI PROBABILITÀ MONOTONE DECRESCENTI, NON SOLO PER L'ESPONENZIALE.

- Lo scheduling che stiamo considerando ora è della anche SHORTEST JOB FIRST nel momento in cui il numero delle classi tende all'infinito: ordiniamo ogni singola classe in base alla sua dimensione.

$$\bullet E[T_Q]^{\text{SJF}} = \frac{1}{2} E[S^2] \int_0^\infty \frac{dF(x)}{(1-\lambda) \int_0^x t f(t) dt)^2}$$

↳ DI FATTO: $E[T_Q]^{\text{SJF}} = \frac{1}{2} E[S^2] \sum_{k=1}^n \frac{F(x_k) - F(x_{k-1})}{(1-\lambda) \int_0^{x_{k-1}} t f(t) dt (1-\lambda) \int_0^{x_k} t f(t) dt}$

SB PREEMPTIVE PRIORITY:

Con la prelazione dobbiamo stare attenti a una cosa: se a un certo punto c'è in servizio un job di classe h e arriva un job di classe $K < h$, non vale la pena fare la prelazione se il job di classe h a ha quasi finito il suo servizio: quello che si fa è confrontare il tempo di servizio t_h del job di classe K col tempo di servizio t_h rimanente del job di classe h ; nel caso in cui $t_K < t_h$, il job di servizio viene messo nella coda di classe $h' < h$ indipendentemente da qual è il suo tempo di servizio rimanente t_h .

- Tempo di servizio rimanente dei job che non possono essere interrotti perché di classe K o inferiore:

$$E[S_{\text{rest},K}] = \frac{\lambda}{2} \int_0^{x_K} t^2 dF(t) \quad \underbrace{1-F(x_K)}$$

- Tempo di servizio rimanente degli altri job: $\text{Prob}\{\text{il job è intercomponibile}\} \frac{1}{2} x_K^2$

$$\Rightarrow E[T_{Q_K}]^{SB-P} = \frac{\frac{\lambda}{2} \left[\int_0^{x_K} t^2 dF(t) + (1-F(x_K)) x_K^2 \right]}{(1 - \sum_{i=1}^K p_i)(1 - \sum_{i=1}^{K-1} p_i)} \quad \begin{array}{l} \hookrightarrow \text{Secondo che questi "altri job" sono quelli di} \\ \text{classe superiore a } K \text{ il cui tempo rimanente} \\ \text{è diverso da } x_K. \end{array}$$

- $E[T_{Q_K}]^{SB-P} \leq E[T_{Q_K}]^{SB-NP}$ (ma questo ormai è chiaro).

- $E[T_{S_K}]^{SB-P} = E[T_{Q_K}]^{SB-P} + E[S_{\text{rest},K}]$, dove $E[S_{\text{rest},K}] = \frac{E[S_K]}{1 - \sum_{i=1}^{K-1} p_i}$

PER SULLE SUE
STANNO LE MISURE
DI DELLE PERFORMANCE
GLOBALI

Tra SB PREEMPTIVE PRIORITY e SHORTEST JOB FIRST:

- Se la variabilità dei job è alta potrebbe essere migliore SJF. \rightsquigarrow PER VIA DELL'ORDINAMENTO COMPLETO DEI JOB
- Se la variabilità dei job è bassa potrebbe essere migliore SB-P. \rightsquigarrow PER VIA DELLA PRELACIONE
- Ma, qui non c'è una regola generale (il discorso sarebbe da approfondire ulteriormente).

\hookrightarrow L'ultimo già lo si avrebbe se introducessimo la prelazione nello scheduling Shortest Job First, ottienendo così la SHORTEST REMAINING PROCESSING TIME.

~~SHORTEST REMAINING PROCESSING TIME:~~

Qui potrebbe essere utile considerare il tempo di attesa medio dei job di size x :

$$E[T_Q(x)] = \frac{\frac{\lambda}{2} \int_{t=0}^x t^2 f(t) dt + \frac{\lambda}{2} x^2 (1-F(x))}{(1-p_x)^2}$$

$$\Rightarrow E[T_{S(x)}] = E[T_Q(x)] + \int_{t=0}^x \frac{dt}{1-p_t} \quad \text{dove } p_x = \lambda \int_0^x t f(t) dt$$

Questo termine decresce col passare del tempo (cioè decresce p_t).

\hookrightarrow DI FATTO RAPPRESENTA IL TEMPO DI SERVIZIO RIMANENTE.

\hookrightarrow QUESTO A SUA VOLTA PERCHÉ SI ESTENDE L'INTERVALLO DI INTEGRAZIONE.

28/04/2022

$$\begin{aligned} A_i &= A_{i-1} + R_i \\ \text{con } A_0 &= 0 \\ A_0 < A_1 < A_2 < \dots \end{aligned}$$

Tornando alla generazione di v.a. continue..

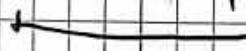
Definizione:

Se R_1, R_2, \dots è una sequenza di tempi di interrivo iid con $E[R_i] = \lambda_i > 0$, allora la sequenza corrispondente di tempi di arrivo A_1, A_2, \dots è un \rightarrow PROCESSO DI ARRIVO STAZIONARIO con tasso λ .

ogni R_i è stazionario

→ Se invece il tasso di arrivo λ varia nel tempo, allora il processo di arrivo è NON STAZIONARIO.

Noi comunque ci concentreremo sui processi di arrivo stazionari: di fatto, hanno molto senso quando prendiamo in considerazione i piccoli intervalli di tempo (\equiv fasi scorse)



Se R_1, R_2, R_3, \dots sono variabili aleatorie esponenziali $\Rightarrow A_i = R_1 + R_2 + \dots + R_i$ è una v.a. di Erlang $[Erlang(i, \lambda)]$.

Somma di i esponenziali
di parametro λ

→ ANCHEGLIAMENTE POTREMO DIRE CHE LA SEQUENZA A_1, A_2, A_3, \dots È UN PROCESSO DI ARRIVO STAZIONARIO DI POISSON.

Teorema:

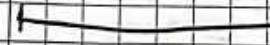
Stato:

- A_1, A_2, \dots è una sequenza di v.a. uniformi che cadono nell'intervalle $(0, t)$.
- X è una v.a. discreta \rightarrow che rappresenta il numero di arrivi A_i che cado in un sotto-intervalle di $(0, t)$ di lunghezza τ .

Allora, X è indistinguibile da una v.a. di Poisson con parametro (media) $\lambda\tau$, dove λ è il tasso medio degli arrivi (a patto che n sia grande e τ/λ sia piccolo).

Teorema:

Se gli arrivi sono modellati da una v.a. di Poisson e hanno tasso medio $\lambda \Rightarrow$
 \Rightarrow il tempo di interrivo è una v.a. esponenziale di media λ^{-1} . → C'è una breve dimostrazione sulle slide.

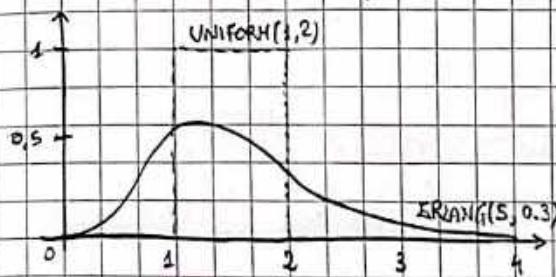


La moda ($=$ il valore "più probabile") di una v.a. esponenziale è $0 \Rightarrow$ un processo di arrivo stazionario di Poisson tende a esibire dei cluster ($=$ gruppi di arrivi che adorano praticamente insieme).

→ Se invece il tempo di interrivo è modellato con una Erlang (che è una variabile) gli arrivi non presentano clustering, le cui sono più distribuiti nel tempo.

Modelli del tempo di servizio:

Diversamente dal caso dei processi di arrivo, non ci sono delle vere e proprie linee guida di default per modellare i processi di servizio, che sono più dipendenti dall'applicazione.



← QUI POSSIAMO AD ESEMPIO NOTARE COME LA DISTRIBUZIONE DI ERLANG ABbia UNA VARIANZA MAGGIORA DELLA DISTRIBUZIONE UNIFORME (CALCHENO NEL NOSTRO CASO).
⇒ IN QUESTO CASO, LA DISTRIBUZIONE DI ERLANG PER IL TEMPO DI SERVIZIO, PER QUANTO SIA PIÙ REALISTICA, RISULTA ESSERE PIÙ SPATIAGGIOSA.

Supponiamo ora invece di voler modellare il tempo di servizio con una v.a. normale di media 1,5; consideriamo ad esempio $N \sim \text{Normal}(1,5; 2)$. Qui è necessario truncare N per eliminare i valori negativi e quelli troppo grandi (e.g. quelli maggiori di 4).

Prendiamo dunque $a=0, b=4$.

$$\Rightarrow \alpha = \Pr(N < a) = 0,2265 ; \beta = \Pr(N > b) = 0,1056$$

Abbiamo ottenuto così la variabile aleatoria truncata \hat{N} di media 1,85 e varianza 1,07.

Ma noi voleremo una media pari a 1,5, proprio come avevamo fissato all'inizio!

Da qui subentra la CONSTRAINED INVERSION, che ci permette di truncare una v.a. in modo asimmetrico (come è avvenuto nel nostro caso) senza cambiare la media; consiste nei seguenti due comandi:

$$\begin{aligned} u &= \text{Uniform}(a, 1-\beta); \\ \text{return } F^{-1}(u); \end{aligned}$$

03/05/2022

Intervallo di confidenza:

Consideriamo ad esempio il tempo di risposta, e supponiamo di volerlo effettuare a seguito di un certo numero n di run. Ciascuna run fornisce i seguenti tempi di risposta medi:

$$\bar{x}_1, \dots, \bar{x}_i, \dots, \bar{x}_n$$

Non vogliamo costruire un intervallo di confidenza $[\bar{x} \pm w]$ dove, con un certo livello di confidenza, possiamo dire che ciascun valore \bar{x}_i cade all'interno dell'intervallo.

Teorema del limite centrale:

Se X_1, X_2, \dots, X_n è una sequenza di v.a. iid con una stessa media μ e una stessa

deviazione standard σ , allora la media campionaria $\bar{X} = \frac{1}{n} \sum_{i=1}^n X_i$ tende alla ^{stessa} seguente distribuzione: $\text{Normal}(\mu, \frac{\sigma^2}{n})$ per $n \rightarrow +\infty$.

t-statistica:

$$t_j = \frac{x_j - \mu}{s/\sqrt{n}}, \quad j = 1, 2, 3, \dots$$

È possibile standardizzare i valori x_1, x_2, \dots, x_n di X_1, X_2, \dots, X_n calcolando $z_j = \frac{x_j - \mu}{\sigma/\sqrt{n}} \quad \forall j = 1, 2, 3, \dots$. Da qui è possibile calcolare la media campionaria \bar{z} che per $n \rightarrow +\infty$ tende a $\text{Normal}(0, 1)$.

Per n sufficientemente grande, tende a una v.a. di Student con $n-1$ gradi di libertà.

È una statistica che serve a correggere la normalizzazione della varianza (che altrimenti risulrebbe maggiore del suo valore nella realtà).

Teorema:

Se x_1, x_2, \dots, x_n è un campione ^{INDIPENDENTE} estratto da una "sorgente" di media μ e varianza s^2 , se \bar{x} , s sono risp. media e deviazione std del campione, e se n è grande, allora è

APPROXIMATIVAMENTE vero che

$$t_m = \frac{\bar{x} - \mu}{s/\sqrt{n-1}} \sim \text{Student}(n-1).$$

(PER $n \rightarrow +\infty$ DIVENTA INDISTINGUIBILE DA UNA GAUSSIANA STANDARD)

Sia $T \sim \text{Student}(n-1)$ e sia α un parametro di confidenza compreso tra 0 e 1.

Allora esiste un valore reale positivo t^* tale che $P(-t^* \leq T \leq t^*) = 1 - \alpha$.

In altre parole: se μ è ignoto, dato che $T \sim \text{Student}(n-1)$, $-\frac{t^* s}{\sqrt{n-1}} \leq \bar{x} - \mu \leq \frac{t^* s}{\sqrt{n-1}}$

è vero con probabilità $1 - \alpha$.

→ DALLA DISUGUAGLIANZA DI DESTRA: $\bar{x} - \left(\frac{t^* s}{\sqrt{n-1}} \right) \leq \mu$

QUESTO È IL NOSTRO -W DELL'INTERVALLO DI CONFIDENZA

→ DALLA DISUGUAGLIANZA DI SINISTRA: $\mu \leq \bar{x} + \left(\frac{t^* s}{\sqrt{n-1}} \right)$

QUESTO È IL NOSTRO +W DELL'INTERVALLO DI CONFIDENZA

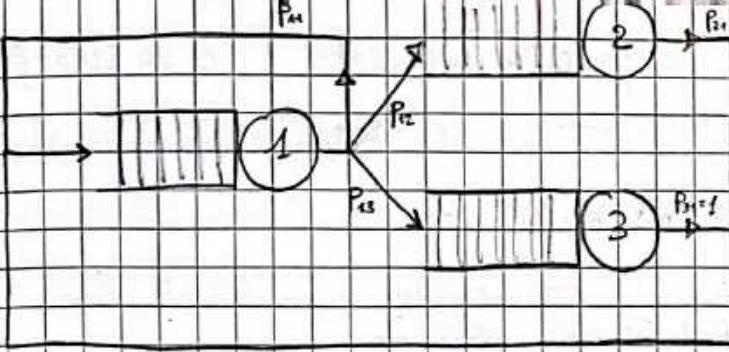
→ Con confidenza $1 - \alpha$ possiamo dire che $\bar{x} - \frac{t^* s}{\sqrt{n-1}} \leq \mu \leq \bar{x} + \frac{t^* s}{\sqrt{n-1}}$

→ t^* è detto valore critico di T .

RETE DI CONNESSIONI

È una mappa ad alto livello di un processo di Markov.

Consideriamo una RETE CHIUSA a tre centri:



$$\bullet M = \text{num. centri} = 3$$

$$\bullet N = \text{popolazione totale} = 2 ;$$

il fatto che la rate è chiusa determina che i due job circolano indefinitamente nel sist.

$$\bullet \mu_1 = 4$$

$$\mu_2 = \mu_3 = 2$$

$$\bullet P = \|P_{ij}\| = \text{matrice di routing}$$

Lo stato è dato dal numero di job che si trovano in ciascuno dei tre centri e può essere pari a:

$$(2 \ 0 \ 0) \rightarrow S_1$$

$$(0 \ 2 \ 0) \rightarrow S_2$$

$$(1 \ 1 \ 0) \rightarrow S_0$$

$$(0 \ 1 \ 1) \rightarrow S_5$$

$$(1 \ 0 \ 1) \rightarrow S_3$$

$$(0 \ 0 \ 2) \rightarrow S_6$$

L'insieme che compone queste triple lo denotiamo con E :

$$E = \{S \mid n_i \geq 0, \sum_{i=1}^M n_i = N\}$$

qui abbiamo supposto $S = (n_1, n_2, n_3)$.

$$N_S = |E| = \binom{N+M-1}{M-1}$$

A ciascuno degli stati s_i associano un tempo di vita t_{v_i} che corrisponde all'arco di tempo che va dall'istante in cui il processo entra nello stato s_i all'istante in cui ne esce.

Definizione:

Definiamo un processo Markoviano un processo i cui stati sono mutuamente esclusivi e collettivamente esaurienti, e la cui evoluzione dipende esclusivamente dallo stato corrente e non dalla storia passata.

05/05/2022

Nell'ipotesi semplificativa (sufficiente ma non necessaria) per cui i serventi hanno tutti tempo di servizio esponenziale, allora t_{v_i} ha una distribuzione esponenziale, con $E[t_{v_i}] = \frac{1}{\alpha_i}$, dove $\alpha_i = \text{TASSO DI USCITA DALLO STATO } i$.

Relativamente all'esempio dell'altra volta:

$$\rightarrow S_1 = (2, 0, 0) \rightarrow E[t_{v_1}] = \frac{1}{\mu_1} \quad (\alpha_1 = \mu_1)$$

$\rightarrow S_2 = (0, 1, 1, 0)$; in questo stato si può arrivare da uno dei seguenti tre stati:

$$(0, 2, 0) \quad (0, 0, 1)$$

D'altra parte, da s_2 si esce o da una partenza dal centro 1, o da una partenza dal centro 2. Perciò si ha: $E[t_{v_2}] = \frac{1}{\mu_1 + \mu_2}$ ($\alpha_2 = \mu_1 + \mu_2$)

DEFINIZIONE:

$\pi_{ij} :=$ probabilità di transizione dallo stato s_i allo stato s_j .

Consideriamo la seguente transizione: $(1 \ 1 \ 0) \rightarrow (2 \ 0 \ 0)$

$$\text{Allora } \pi_{21} = \frac{\mu_2}{\mu_1 + \mu_2} P_{21} = \frac{\mu_2}{\mu_1 + \mu_2}$$

→ Informalmente, π_{ij} possiamo definirlo come:

Tasso di uscita dal centro che provoca la transizione

Tasso di uscita $\xrightarrow{\alpha_i \rightarrow \text{nel nostro caso } \alpha_2}$

(probabilità di andare nello stato j a partire dallo stato i considerando il solo centro che provoca la transizione)

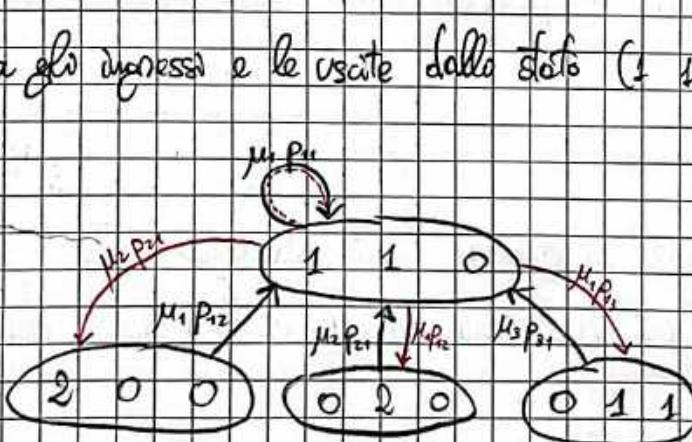
DEFINIZIONE:

$q_{ij} :=$ frequenza di transizione dello stato s_i allo stato s_j = $\alpha_i \pi_{ij} =$

= (Tasso di uscita dal centro che provoca la transizione) · (Probabilità di andare nello stato j a partire dallo stato i considerando il solo centro che provoca la transizione)

Consideriamo ora gli ingressi e le uscite dallo stato $(1 \ 1 \ 0)$:

Questi sono TASSI →



→ Questo è un pezzo del GRAFO DI TRANSIZIONE tra stati.

Nell'equilibrio si ha una condizione di BLANCIAMENTO DEL FLUSSO; per ogni stato s_i :

Tasso di uscita da s_i = Tasso di ingresso in s_i



Proposizione:

Per poter essere stazionario un processo deve essere:

→ IRRIDUCIBILE: è possibile passare da uno stato a un qualsiasi altro in un numero finito di passi. → Ciò sicuramente prevede che non ci siano centri isolati o una separazione tra gruppi di centri.

→ APERIODICO: il tempo di ritorno in uno stesso stato non è periodico.

→ RICORRENTE POSITIVO: il tempo di ritorno in uno stesso stato è finito.

Le prime due condizioni garantiscono l'esistenza di $\pi(s_i) = \lim_{t \rightarrow \infty} \pi(s_i, t)$, dove $\pi(s_i, t)$ è la probabilità istantanea di trovarsi nello stato s_i nell'istante di tempo t .

Tutte e tre le condizioni garantiscono anche l'unicità di $\pi(s_i) = \lim_{t \rightarrow \infty} \pi(s_i, t) \neq 0$.

↪ Quindi questo processo è anche detto ERGODICO.

Il bilanciamento globale del flusso possiamo descriverlo come:

$$\pi(s) \cdot (\text{flusso in uscita}) = \sum_{\substack{\text{da } s \\ \text{da } s'}} \pi(s') \cdot (\text{flusso in entrata in } s) \quad \forall s \in E$$

$$\Rightarrow \pi(\bar{s}) = \pi(s) [1 - (\text{flusso in uscita})] + \sum_{\substack{\text{da } s \\ \text{da } s'}} \pi(s') \cdot (\text{flusso in entrata in } \bar{s})$$

Mettendo insieme tutte le equazioni per tutti gli stati s , possiamo scrivere il bilanciamento globale del flusso in forma matriciale: $\bar{\pi} = \bar{\pi} Q$

$$\text{dove } \bar{\pi} = [\pi(s_1), \pi(s_2), \dots, \pi(s_{|E|})]$$

$$Q = \begin{bmatrix} q_{11} & q_{1,2,3} & \cdots & q_{1,|E|} \\ q_{21} & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \vdots \\ q_{|E|1} & q_{|E|,2,3} & \cdots & q_{|E|,|E|} \end{bmatrix} \quad \leftarrow \begin{array}{l} \text{Tassi di uscita da } 1 \\ \text{Tassi di ingresso in } 1 \end{array}$$

Sulla diagonale principale, nella riga i , abbiamo $1 - \sum_{j \neq i} q_{ij}$

UN PO' DI SPERI

$$\rightarrow \bar{\pi} = \bar{\pi} Q \Rightarrow \bar{\pi} Q - \bar{\pi} = \phi \Rightarrow \bar{\pi} (\underbrace{Q - I}_{S}) = \phi$$

S = generatore del processo

→ A partire da S , noi generiamo una matrice S' sostituendo una qualsiasi colonna di S con una colonna di 1.

$$\Rightarrow \bar{\pi} S' = [1 \ 0 \ \dots \ 0] \quad (\text{qui abbiamo sostituito la prima colonna})$$

↑ Questo rappresenta la condizione di normalizzazione delle probabilità a 1.

$$\Rightarrow \bar{\pi} = [1 \ 0 \ \dots \ 0] S'^{-1} \quad \rightarrow Ecco LA SOLUZIONE :)$$

10/05/2022

Definizione:

(STAZIONARIA)

• PROBABILITÀ MARGINALE CHE NEL CENTRO CI SIANO m Job = $p_i(n) = \sum_{s: n_i=s} \pi(s)$

• POPOLAZIONE MEDIA NEL CENTRO $i = E[n_i] = \sum_{n_i=0}^N n_i p_i(n_i)$

- UTILIZZAZIONE DEL CENTRO i (CASO SERVENTE SINGOLO) = $U_i = 1 - P_i(0)$
 - THROUGHTPUT DEL CENTRO i (CASO SERVENTE SINGOLO) = $X_i = U_i \mu_i$
 - UTILIZZAZIONE DEL CENTRO i (CASO MULTISERVER) = $U_i = \frac{E[C_i]}{m_i}$
- dove $C_i = \#$ canali occupati, $m_i = \#$ serventi del centro i
- UTILIZZAZIONE DEL CENTRO i (CASO MULTISERVER) = $\left[\sum_{j=1}^{m_i-1} J P_i(j) + \sum_{j=m_i}^N M_i P_i(j) \right] \frac{1}{m_i}$
- THROUGHTPUT DEL CENTRO i (CASO MULTISERVER) = $X_i = U_i m_i \mu_i = \frac{E[C_i]}{m_i} m_i \mu_i =$
 $= \sum_{j=1}^{m_i-1} P_i(j) \cdot j \mu_i + \sum_{j=m_i}^N P_i(j) M_i \mu_i$

→ Un INFINITE SERVER è praticamente un MULTISERVER con N serventi (dove $N =$ popolazione totale nella rete) : di conseguenza, non genera mai una coda.

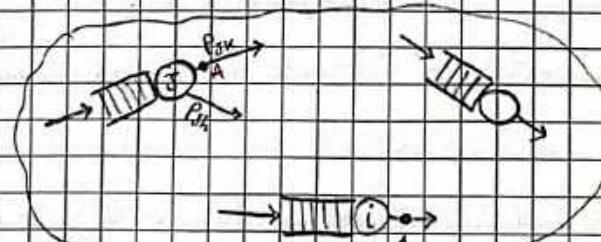
- UTILIZZAZIONE DEL CENTRO i (CASO INFINITE SERVER) = $U_i = \frac{E[C_i]}{N} = \sum_{j=1}^N J P_i(j)$
- THROUGHTPUT DEL CENTRO i (CASO INFINITE SERVER) = $X_i = U_i N \mu_i$

→ Il TEMPO DI RESIDENZA del centro i per il job j è la somma di tutti i tempi di risposta del centro i per il job j ; in effetti, finché j rimane all'interno della rete, può passare più volte per uno stesso centro.

- MEDIA DEL TEMPO DI RESIDENZA DEL CENTRO i = $E[t_i] = \frac{E[n_i]}{X_i}$ (per Little)

Quelli che abbiamo elencato ora sono gli INDICI LOCALI del sistema.

Supponiamo di avere una rete chiusa generica:



prendiamo questo come punto di riferimento

Per quanto riguarda il throughput della rete:

- $X_{rispetto a i} = X_i$
- $X_{rispetto ad n} = X_j P_{jk}$

Tempo (medio) che passa da quando il job esce dal centro i e quando ritorna nel centro i .

NUMERO MEDIO DI VISITE DEL CENTRO j RISPETTO AL CENTRO i

→ LEGGE DEL TEMPO DI RISPOSTA. $[E[t_{q_i}]] = \sum_{\substack{j=1 \\ j \neq i}}^N (V_{ji}) E[t_j]$

Questo sembrerebbe essere un tempo di risp.

Sia \bar{y} il vettore dei THROUGHPUT RELATIVI. Allora: $\bar{y} = \bar{y} P$, dove P è la matrice di routing.

In particolare, $\bar{y} = [y_1, y_2, \dots, y_H]$ e $V_{j/i} = \frac{y_j}{y_i}$

Supponiamo ora che la rete in esame sia APERTA:

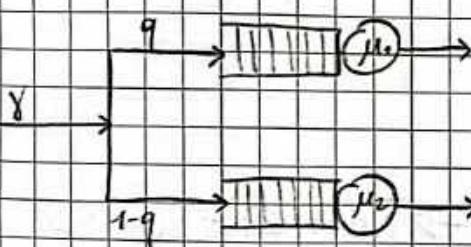
• PROBABILITÀ DI USCIRE DALLA RETE A PARTIRE DAL CENTRO $i = p_{i,j} = 1 - \sum_{j=1}^H p_{ij}$
 ↳ dall'centro i i job possono anche uscire dall'esterno (con un tasso γ_i). Però, il tasso di arrivo dal punto di vista del centro i è dato da:

$$\lambda_i = \gamma_i + \sum_{j=1}^H \lambda_j p_{ji} \quad \leftarrow \text{Questo è proprio il throughput: } \lambda_i = \chi_i$$

FOSSIANO INVECE DEFINIRE χ IL TASSO DI ARRIVO NELLA RETE COMPLESSIVA, ovvero: $\chi = \sum_{i=1}^H \gamma_i$

• NUMERO MEDIO DI VISITE CHE I JOBS FANNO A UN CENTRO j DALL'ISTANTE IN CUI ENTRA NELLA RETE ALL'ISTANTE IN CUI NE ESCE = $V_j = \lambda_j / \chi$

Esempio:



• $20 \text{ req/s} = \chi$

• Disk 1 $\rightarrow 30 \text{ ms} \Rightarrow \mu_1 = 33,3 \text{ req/s}$

• Disk 2 $\rightarrow 46 \text{ ms} \Rightarrow \mu_2 = 21,7391 \text{ req/s}$

↳ CALCOLARE q , $1-q$ AFFINCHÉ IL TEMPO DI RISPOSTA GLOBALE SIA MINIMIZZATO.

Concettualmente parlando, per minimizzare il tempo di risposta globale, è opportuno fare in modo che i due centri abbiano la stessa utilizzazione $p_i = \frac{\lambda_i}{\mu_i}$.

↳ Ili possono essere calcolati tramite le equazioni di traffico:

• $\lambda_1 = q \cdot \chi$

• $\lambda_2 = (1-q) \chi$

Possiamo anche considerare le visite:

• $V_1 = \frac{\lambda_1}{\chi} = q$

• $V_2 = 1-q$

Comunque sia, l'ugualanza da impostare sarà: $\frac{q}{\mu_1} = \frac{(1-q)}{\mu_2}$

DA QUI È POSSIBILE
LE ESPlicitare =

↳ LA q , CHE VIENE:

$q = 0,6052$

12/05/2022

Consideriamo il caso minimale di rete aperta: \Rightarrow



→ Poiché la probabilità ~~che~~ ^{che} l'arrivo sia compresa, non è fissa o finita (come invece accade nelle reti chiuse), lo spazio degli stati è INFINITO.

→ Il primo centro può essere trattato come una KP:

$$P(N_s = m) \xrightarrow{M/M/1/FIFO} = \rho^m (1-\rho)$$

$$\xrightarrow{\text{no case losses between}} P(m_1 = K) = \rho^K (1 - \rho)$$

Teorema di Burke:

Dato un sistema M/M/1 stabile con processo di Poisson di parametro λ , il processo di partenza è anch'esso un processo di Poisson di parametro λ .

→ Anche per il centro 2 vale che: $P(m_2 = K) = \rho^K (1 - \rho)$

(\hookrightarrow due centri si comportano come se fossero isolati \Rightarrow sono indipendenti)

$$\Rightarrow P(m_1 = k, m_2 = h) = P(m_1 = k) P(m_2 = h)$$



Complichiamo un po' il sistema:



Con questo feedback il teorema di Burke non vale più \rightarrow i due centri NON sono indipendenti.

Tuttavia, se andiamo a guardare il diagramma degli stati, esso ha una struttura a lattice (che è regolare). È stato dimostrato che questo posto a posto sarebbe comunque che: $P(m_1 = k, m_2 = h) = P(m_1 = k) P(m_2 = h)$. \hookrightarrow FORMA PRODOTTO

Qui però:

$$\begin{cases} \lambda_1 = \lambda + (1-p)\lambda_2 \\ \lambda_2 = \lambda_1 \end{cases} \Rightarrow \lambda_1 = \lambda_2 = \frac{\lambda}{p}$$

$$\Rightarrow \nu_1 = \nu_2 = \frac{\lambda_1}{\lambda_0} = \frac{\lambda_2}{\lambda_0} = \frac{1}{p}$$

\hookrightarrow effettivamente possiamo notare che ogni visita al centro 1 comporta invariabilmente una visita al centro 2.

Rete separabile:

\hookrightarrow Mentre nel caso semplice di prima avevamo $\lambda = \lambda_1 = \lambda_2 \Rightarrow \nu_1 = \nu_2 = 1$

Una rete è Separabile se è vero che $P(m_1 = k_1, m_2 = k_2, \dots) = P(m_1 = k_1) P(m_2 = k_2) \dots$

Per avere una rete separabile devono ~~essere~~ valere le seguenti ipotesi:

1) Bilanciamento del flusso.

2) Comportamento "one-step" (\rightarrow non possono avvenire + transizioni di stato contemporaneamente).

3) Componenti di:

• ROUTING (il passaggio da un centro a un altro non ~~può~~ dipendere dallo stato dei vari centri).

• DISPOSITIVI (un dispositivo non può dipendere dallo stato di un altro centro).

• ARRIVI ESTERNI (che devono essere indip. dello stato del sistema).

Teorema ROLL:

La forma prodotto vale per:

→ Reti aperte, chiuse e miste.

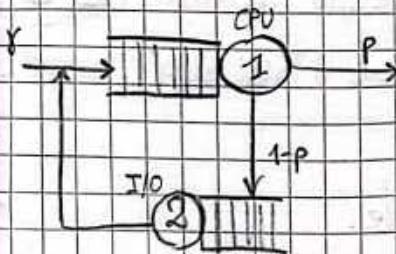
↳ sono APERTE PER alcune CLASSI, CHIUSE PER ALTRE

→ Routing probabilistico, senza memoria e indipendente dallo stato.

→ Servizi esponenziali se lo scheduling prevede attese.

→ Servizi generali se lo scheduling è immediato.

Esempio:



μ_1, μ_2 sono esponenziali.

Supponiamo che λ_1 sia TUTTO il flusso che parte dal centro 1.

$$\begin{cases} \lambda_1 = \gamma + \lambda_2 \\ \lambda_2 = (1-p)\lambda_1 \end{cases} \Rightarrow \begin{cases} \lambda_1 = \frac{\gamma}{p} \\ \lambda_2 = \frac{\gamma}{p}(1-p) \end{cases} \Rightarrow \begin{cases} \nu_1 = \frac{1}{p} \\ \nu_2 = \frac{1-p}{p} = \nu_1 - 1 \end{cases}$$

$$\pi(\nu_1, \nu_2) = \pi_1(\nu_1) \pi_2(\nu_2)$$

$$\text{dove } \pi_i(\nu_i) = \underbrace{\pi_i(0)}_{1-p} p^{\nu_i} = (1-p_i) p_i^{\nu_i}$$

Supponiamo che $\gamma = 1,3 \text{ job/s}$, $p = 0,05$, $\mu_1 = 30 \text{ job/s}$, $\mu_2 = 25 \text{ job/s}$

$$\Rightarrow \lambda_1 = 26 \text{ job/s} < \mu_1 \quad ; \quad \lambda_2 = 24,7 \text{ job/s} < \mu_2$$

$$\nu_1 = 20$$

$$\nu_2 = 19$$

(mediamente)

$$p_1 = 0,8666667$$

$$p_2 = 0,988$$

$$\text{dove: } E[t_1] = \frac{\nu_1 E[S_1]}{1-p_1} + E[S_1] = 0,25 \text{ s}$$

$$E[t_2] = \nu_2 E[t_1] + \nu_2 E[t_2] \approx 68,3333 \text{ s}$$

$$E[t_2] = \frac{\nu_2 E[S_2]}{1-p_2} + E[S_2] = 3,3333 \text{ s}$$

TEMPO DI RISPOSTA DELL'INTERO CENTRO

Nella forma prodotto delle reti aperte abbiamo: $\pi(n_1, \dots, n_M) = \pi_1(n_1) \cdots \pi_M(n_M)$,
dove $\pi_i(n_i) = p_i(n_i)$ è proprio la probabilità marginale. → PROBABILITÀ CHE NEL CENTRO i
CI SIA IL n_i ; I.C.

↳ Questo però non vale per le reti chiuse (per cui qui è necessario calcolare le marginali tramite la loro definizione).

Definizione:

Le statistiche dello stato stazionario (se esiste lo stato stazionario) sono statistiche calcolate su una simulazione con un "orizzonte infinito", ovvero con una durata tale per cui, se aumentata, non porta ad alcun cambiamento nei risultati.

D'altra parte, le statistiche transienti = vengono calcolate su una simulazione con un "orizzonte breve" (finito), ovvero sul breve periodo.

→ Le statistiche dello stato stazionario non risentono della variazione dello stato iniziale del sistema. Inoltre, una simulazione con un orizzonte infinito richiede che l'ambiente del sistema sia statico (i parametri come i tassi debbano rimanere costanti).

→ Tutto questo non vale per il caso transienti.

13/05/2022

Repliche:

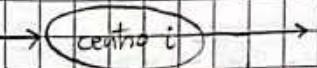
Sono simulazioni ripetute che differiscono solo nel semme selezionato; insieme costituiscono il cosiddetto ENSEMBLE. Scorreremo su ~~ognuna~~ una stessa statistica transiente.

↳ Quello che conviene fare però è mantenere lo stesso seed tra tutte diverse e iniziare la $i+1$ -esima run a partire dal valore finale del generatore nell' i -esima run: è importante che le repliche non si sovrappongano fra loro.

14/05/2022

Tornando alle reti chiuse..

Supponiamo di avere M centri e N utenti.



t_i(N) è tempo di risc. del centro i

Per l'algoritmo MEAN VALUE ANALYSIS, nel caso di tempo di servizio indipendente dal carico:

$$E[t_i(N)] = \text{tempo speso in servizio} + \text{tempo speso in attesa che gli "altri" utenti terminino il servizio}$$

$$= E[S_i] + E[a_i(N)] \cdot E[S_c]$$

È IL NUMERO DI UTENTI DA ATTENDERE

Teorema degli orocivi:

$$E[a_i(N)] = E[n_i(N-1)] \Rightarrow E[a_i(N)] \text{ è uguale alla popolazione media nel centro se nell'intera rete ci fosse un utente in meno.}$$

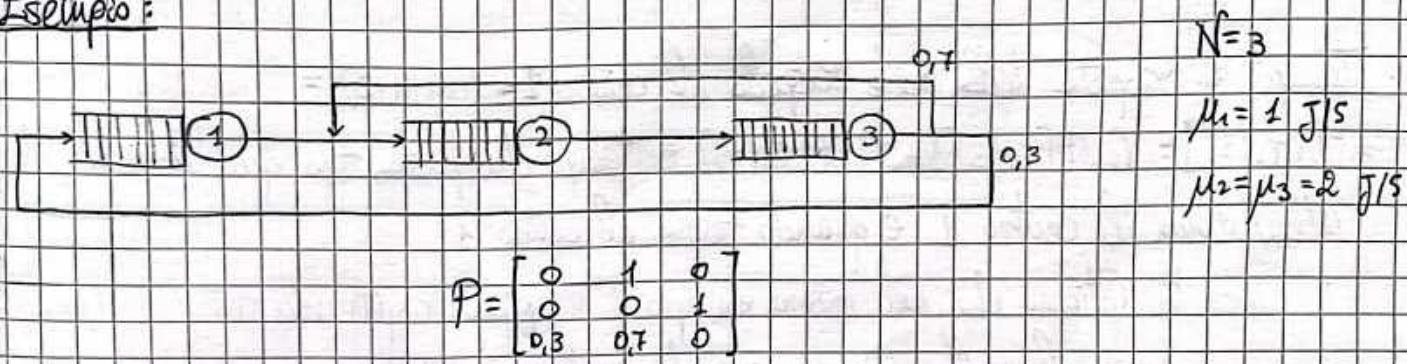
Questo va aggiornato vale per le reti separabili.

$$\Rightarrow E[t_i(N)] = E[S_i] + E[n_i(N-1)] E[S_c] = E[S_i] (1 + E[n_i(N-1)])$$

$$\text{THROUGHPUT DEL CENTRO } i = \lambda_i(N) = \frac{N}{\sum_{j=1}^N \gamma_{ij} E[t_j(N)]} \quad \begin{array}{l} \text{(ma lo sto calcolando come throughput di rete nel punto i)} \\ \text{infatti questo è il tempo medio richiesto ai job per passare dal punto i al treno allo stesso punto i.} \end{array}$$

Per Little: $E[n_i(N)] = \lambda_i(N) E[t_i(N)]$

Esempio:



EQUAZ. DI TRAFFICO:

$$\begin{cases} y_1 = 0,3 y_3 \\ y_2 = y_1 + 0,7 y_3 \\ y_3 = y_2 \end{cases}$$

Se fissiamo $y_3 = 1$ \Rightarrow

$$\begin{cases} y_1 = 0,3 \\ y_2 = 1 \\ y_3 = 1 \end{cases}$$

Applichiamo ora MVA:

$$\rightarrow E[n_i(0)] = 0 \quad i = 1, 2, 3 \quad (\text{canale})$$

$$\rightarrow E[t_{i(0)}] = 1 \quad \rightarrow E[t_{i(1)}] = \frac{1}{2}$$

$$\rightarrow \gamma_{1/1} = \frac{y_1}{y_1} = 1 \quad \rightarrow \gamma_{2/1} = \frac{y_2}{y_1} = 3,333333$$

$$\rightarrow \lambda_1(1) = \frac{1}{1+3,333333 \cdot 0,5 + 3,333333 \cdot 0,5} = 0,230769$$

$$\rightarrow \lambda_2(1) = \frac{1}{0,3 + 1 \cdot 0,5 + 1 \cdot 0,5} = 0,709231$$

$$\rightarrow E[t_{3(1)}] = \frac{1}{2}$$

$$\rightarrow \gamma_{3/1} = \frac{y_3}{y_1} = 3,333333$$

$$\rightarrow \text{Analogamente, } \lambda_3(1) = 0,769231$$

$$\rightarrow E[n_1(1)] = \lambda_1(1) \cdot E[t_1(1)] = 0,230769$$

$$E[m_2(1)] = \lambda_2(1) \cdot E[t_2(1)] = 0,384615$$

$$E[m_3(1)] = \lambda_3(1) \cdot E[t_3(1)] = 0,384615$$

Si procede con calcoli iterativamente fino ad arrivare ai seguenti risultati:

$$\rightarrow E[t_1(3)] = 1,421052 \quad \rightarrow E[t_2(3)] = 0,894737$$

$$\rightarrow E[t_3(3)] = 0,894737$$

$$\rightarrow \lambda_1(3) = 0,406176 \quad \rightarrow \lambda_2(3) = 1,353919$$

$$\rightarrow \lambda_3(3) = 1,353919$$

$$\rightarrow E[m_1(3)] = 0,577197 \quad \rightarrow E[m_2(3)] = 1,211602$$

$$\rightarrow E[m_3(3)] = 1,211602$$

$$\rightarrow U_1(3) = \lambda_1(3) E[S_1] = 0,406176$$

$$\rightarrow U_2(3) = \lambda_2(3) E[S_2] = 0,67696$$

$$\rightarrow U_3(3) = \lambda_3(3) E[S_3] = 0,67696$$

RICORDIAMO
 $E[t_{r_i}] = \sum_{j=1}^M v_{j,i} \cdot t_j$

NB: Tempo di risposta della rete rispetto al centro 1 (con $N=3$) =

$= E[t_{r_1}(3)] = V_{2,1} E[t_2(3)] + V_{3,1} E[t_3(3)]$ = tempo che passa tra quando un job abbandona il centro 1 e quando torna al centro 1.

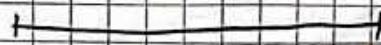
Notiamo che nel nostro esempio è maggiore rispetto al centro 2 (non ci vuole molto a convincersi che mediamente si torna nel centro 2 più spesso e, quindi, più velocemente).

Per completezza riportiamo: $E[t_{r_2}(3)] = V_{3,2} E[t_3(3)] + V_{1,2} E[t_1(3)]$

Notiamo anche che tale tempo di risposta è diverso dal ~~tempo di ciclo~~ tempo di ciclo della rete rispetto al centro 1. Infatti:

Tempo di ciclo della rete rispetto al centro 1 =

= ~~tempo di risposta della rete rispetto al centro 1 + tempo di risposta del centro 1~~



In una rete chiusa, se abbiamo un centro i di capacità finita che si riempie completamente, quello che succede è che gli altri centri non inviano più job a i finché non

si libera qualche posto.

↳ Reti chiuse con centri di capacità finite sono più complesse da trattare.

19/05/2022

I modelli analitici visti finora effettuano delle assunzioni piuttosto stringenti (i job sono indipendenti, i parametri sono costanti, il processo sottostante è Markoviano, ...).

Quello che si fece è:

→ Sostituire i valori teorici dei parametri con valori operazionali (osservati direttamente).
Da qui si dimostrò che le equazioni ottenute dal modello analitico continuano a valere.

ANALISI OPERAZIONALE

Definizione:

Le ipotesi testabili operativamente sono ipotesi la cui veridicità può essere stabilita tramite una misura.

Componenti dell'analisi operazionale:

→ Un sistema.

→ Periodo di tempo finito (PERIODO DI OSSERVAZIONE - OP).

In particolare consideriamo:

• T = durata dell'op.

• A = numero di arrivi durante l'op.

• B = tempo totale in cui il sistema è occupato ($\leq T$). }

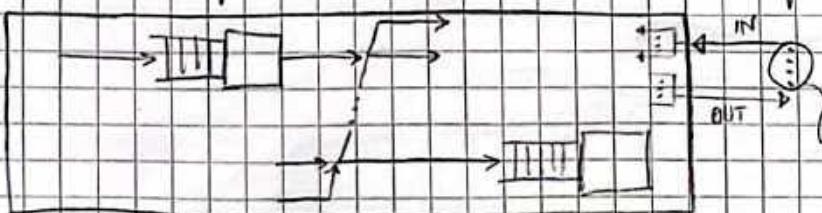
• C = numero di completamenti durante l'op.

Tali quantità possono essere relative anche a dispositivi d'uno stesso tipo.
Es. diretta il n. volte in cui un job va al disp. J subito dopo aver effettuato il n. disp. e.

Queste sono quantità che devono essere misurate nel campo.

→ SISTEMA OPERAZIONALMENTE CONNESSO = dispositivo viene visitato almeno uno volta da almeno un job durante l'op.

→ Nell'analisi operazionale consideriamo sistemi operaz. connessi che vengono trappes. così:



↳ Una chiusura qui l'abbiamo se il sistema è chiuso.

A partire da A, B, C, T possiamo definire:

$$\rightarrow \frac{A}{T} = \# \text{ arrivi nell'unità di tempo} = \lambda.$$

$$\rightarrow \frac{C}{T} = \text{frequenza di uscita} = X.$$

$$\rightarrow \frac{B}{T} = \text{utilizzazione} = U.$$

$$\rightarrow \frac{B}{C} = \text{tempo di servizio medio per job} = S.$$

Consideriamo l'utilizzazione:

$$U = \frac{B}{T} = \frac{B}{T} \cdot \frac{C}{C} = XS$$

Se $A=C$ (abbiamo una condizione di job flow balance) $\Rightarrow U = \lambda S$ (e $X=1$)

Questa è la legge di utilizzazione che vale ALL'EQUILIBRIO, ovvero quando si ha bilanciamento del flusso.

\hookrightarrow In condizioni generali, invece, possiamo solo dire che $U = XS = \frac{B}{T}$.

\rightsquigarrow Nella realtà, possiamo assumere una condizione di job flow balance (Jfb) se $\frac{A}{C}$ è molto piccolo (i.e. trascurabile).

N.B.: Non è detto che $A \geq C$: l'osservazione può partire anche da un sistema non vuoto.

A partire da A_i, B_i, C_{ij}, T possiamo definire:

$$\rightarrow C_i = \sum_{j=0}^k C_{ij} = \# \text{ completamenti del sistema } i \text{ dispositivo } j. \quad \rightarrow k = \# \text{ dispositivi}$$

$$\rightarrow A_o = \sum_{j=1}^k A_{oj}$$

$$\rightarrow C_o = \sum_{i=0}^k C_{oi} \quad \text{dove l'indice } o \text{ indica il mondo esterno; se il sistema è chiuso} \Rightarrow A_o = C_o.$$

$$\rightarrow U_i = \frac{B_i}{T}$$

$$\rightarrow S_i = \frac{B_i}{C_i}$$

$$\rightarrow X_i = \frac{C_i}{T} \quad \text{Dato più probabilità}$$

Le componenti della matrice di routing sono definite come:

$$P_{ij} = \begin{cases} \frac{C_{ij}}{C_i} & \text{se } i = 1, \dots, k \\ \frac{A_{oi}}{A_o} & \text{se } i = 0 \end{cases}$$

$$P_{10} + \sum_{j=1}^k P_{ij} = 1$$

$$\rightarrow X_0 = \frac{C_0}{T} = \sum_{i=1}^k \frac{C_{i,0}}{T} = \sum_{i=1}^k \frac{C_i}{T} = \sum_{i=1}^k X_i P_{i,0}$$

$$\rightarrow U_i = \frac{B_i}{T} = \frac{B_i}{T} \frac{C_i}{C_i} = X_i S_i$$

$$\rightarrow \bar{n}_i = \frac{W_i}{T} \quad \text{dove} \quad \bar{n}_i = \text{POPOLAZIONE MEDIA IN } T.$$

W_i = AREA DEL GRAFICO RELATIVO ALLA POPOLAZIONE NEL SISTEMA AL VARIARE DEL TEMPO.

$$\rightarrow R_i = \frac{W_i}{C_i} = \text{tempo di risposta medio del dispositivo } i.$$

$$\rightarrow \bar{n}_i = \frac{W_i}{T} \cdot \frac{C_i}{C_i} = X_i R_i \quad \leftarrow \text{LEGGI DI LITTLE che, come vediamo, vale anche in assenza dell'ipotesi di job flow balance.}$$

Se job flow balance lo si ha se:

$$C_j = A_j = \underbrace{A_{0,j}} + \sum_{i=1}^k C_{i,j}$$

PER DEFINIZ. DI JOB FLOW BALANCE È UGUALE A $C_{0,j}$ $\Rightarrow C_j = \sum_{i=0}^k C_{i,j}$

$$\Rightarrow C_j = \sum_{i=0}^k C_i P_{i,j} \Rightarrow \frac{C_j}{T} = \sum_{i=0}^k \frac{C_i}{T} P_{i,j} \Rightarrow X_j = \sum_{i=0}^k X_i P_{i,j}$$

Definiamo inoltre:

$$\rightarrow V_i = \frac{X_i}{X_0} = \# \text{ medio di visite a un dispositivo } i.$$

Ma, dalla definizione della frequenza in output (X_i), possiamo anche dire che $V_i = \frac{X_i}{C_0}$.

Comunque sia, dalla definizione di V_i , ottieniamo l'EQUAZIONE DEL FLUSSO FORZATO:

$$X_i = V_i X_0$$

$$\rightarrow V_j = \sum_{i=0}^k V_i P_{i,j} \quad \begin{matrix} \text{DEI RAPPORTI TRA} \\ \text{VISITE} \end{matrix} \quad (\text{ottenuta da } X_j = \sum_{i=0}^k X_i P_{i,j} \text{ e da } X_i = V_i X_0)$$

Da cui:

$$\sum V_i = 1$$

$$\left\{ \begin{array}{l} V_j = P_{0,j} + \sum_{i=1}^k V_i P_{i,j} \\ \end{array} \right. \quad \rightarrow \text{queste sono K equazioni a K incognite che hanno una soluzione unica.}$$

Nota che valgono solo in caso di bilanciamento del flusso.

20/05/2022

Esempio 1:

Supponiamo di avere un server che riceve 5 richieste al disco e che il throughput del disco è pari a 10 req/s. Qual è il throughput del sistema?

→ Consideriamo la legge del flusso forzato: $X_i = V_i X_0$

Di fatto, noi abbiamo:

$$V_{\text{disk}} = 5 \text{ req/sec}$$

$$X_{\text{disk}} = 10 \text{ req/s}$$

$$\Rightarrow X_0 = \frac{X_{\text{disk}}}{V_{\text{disk}}} = 2 \text{ job/s}$$

Esempio 2:

$$\cdot A_i = 7 \text{ job}$$

$$\cdot B_i = 16 \text{ s}$$

$$\cdot C_i = 10 \text{ job}$$

$$\cdot m_i(0) = 3 \text{ job}$$

$$\cdot T = 20 \text{ s}$$

$$\cdot W_i = 40 \text{ job} \cdot \text{s}$$

$$\Rightarrow n_i(20) = m_i(0) + A_i - C_i = 0 \text{ job}$$

$$U_i = \frac{B_i}{T} = \frac{16 \text{ s}}{20 \text{ s}} = 0,8$$

$$S_i = \frac{B_i}{C_i} = \frac{16 \text{ s}}{10 \text{ job}} = 1,6 \text{ s}$$

$$X_i = \frac{C_i}{T} = \frac{10 \text{ job}}{20 \text{ s}} = 0,5 \text{ job/s}$$

$$A_i (-1) = \frac{7 \text{ job}}{20 \text{ s}} = 0,35 \text{ job/s}$$

$$\bar{n}_i = \frac{W_i}{T} = \frac{40 \text{ job} \cdot \text{s}}{20 \text{ s}} = 2 \text{ job}$$

$$R_i = \frac{W_i}{C_i} = \frac{40 \text{ job} \cdot \text{s}}{10 \text{ job}} = 4 \cdot \text{s}$$

Sappiamo che vale Little: $m_i = X_i R_i$. Infatti: $2 \text{ job} = 0,5 \text{ job/s} \cdot 4 \cdot \text{s}$ ✓

Supponiamo di avere un sistema composto da K dispositivi e supponiamo di voler calcolare il tempo di risposta di tutto il sistema:

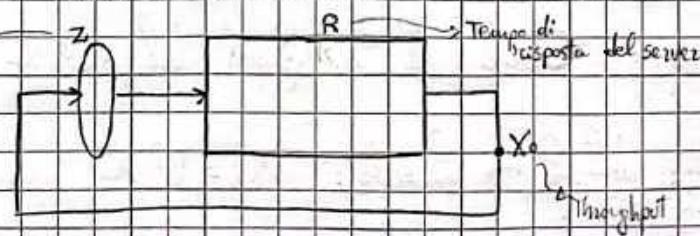
$$R = \frac{\bar{N}}{X_0} = \frac{1}{X_0} \sum_{i=1}^K \bar{n}_i = \sum_{i=1}^K \frac{X_i}{X_0} R_i = \sum_{i=1}^K V_i R_i$$

→ QUESTA LEGGE È VACUA ANCHE IN ASSenza DI EQUILIBRIO DEL FLUSSO.

Esempio 3:

Supponiamo di avere dei terminali che inviano delle richieste a un server:

Think time (= tempo in cui i terminali pensano prima di inviare richieste al server)



TRATTASI DI UN SISTEMA CHIUSO

$$M = (Z + R) X_0 \Rightarrow R = \frac{M}{X_0} - Z \rightarrow \text{PER LITTLE}$$

\uparrow
Balloone totale del sistema.

Esempio 4:

Supponiamo di avere un server che riceve 20 richieste al disco.

$$\rightarrow \text{Utilizzazione del disco} = 50\% = U_{\text{disk}}$$

$$\rightarrow \text{Servizio medio al disco} = 25 \text{ ms} = S_{\text{disk}}$$

$$\rightarrow \text{Num. terminali} = M = 25$$

$$\rightarrow \text{Think Time} = 18 \text{ s} = Z$$

$$R = \frac{M}{X_0} - Z$$

\rightarrow Dalla legge dell'utilizzazione: $U_i = X_i S_i$

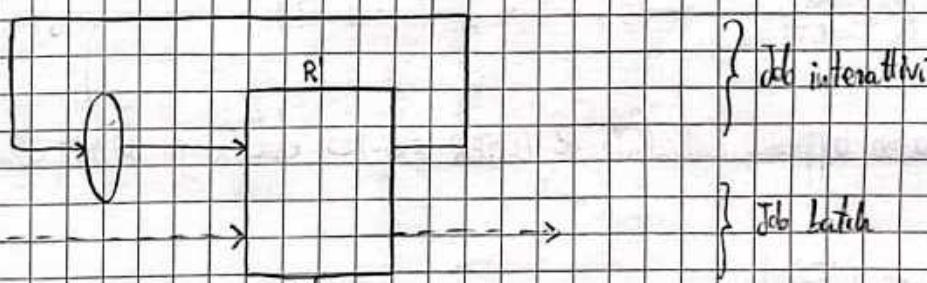
\rightarrow Dalla legge del flusso fisato: $X_0 = \frac{X_i}{V_i} = \frac{U_i}{(V_i S_i)}$ \rightarrow TEMPO DI RESIDENZA ($= D_i$)

$$\Rightarrow X_0 = \frac{0.5}{20 \cdot 0.025 \text{ s}} = 1 \text{ job/s}$$

$$\Rightarrow R = \frac{25}{1 \text{ job/s}} - 18 \text{ s} = 7 \text{ s}$$

Esempio 5:

$$\cdot 40 \text{ terminali} = M$$



$$\cdot 15 \text{ s} = \text{Think time} = Z$$

$$\cdot 5 \text{ s} = \text{tempo risposta} = R^i \text{ interattivo}$$

$$\cdot 40 \text{ ms} = \text{tempo servizio} \text{ medio disco} = S_{\text{disk}}$$

$$\cdot 10 \text{ req al disco} \wedge \text{job interattivo} = V_{\text{disk}}^i$$

$$\cdot 5 \text{ req al disco} \wedge \text{job batch} = V_{\text{disk}}^b$$

$$\cdot 90\% = \text{utilizzazione disco} = U_{\text{disk}}$$

$$\cdot X_0^b = ?$$

- Minimo tempo di risposta per il sistema interattivo = ?

\rightarrow NELL'ASSUNZIONE CHE X_0^b TRIPLOCHI

$$X_0^b = \frac{X_{\text{disk}}^b}{V_{\text{disk}}^b}$$

$$X_{\text{disk}}^b = X_{\text{disk}} - X_{\text{disk}}^i$$

$$\text{dove: } U_{\text{disk}} = X_{\text{disk}} S_{\text{disk}}$$

$$X_{\text{disk}}^i = X_0^i V_{\text{disk}}^i$$

$$\rightarrow X_{\text{disk}} = \frac{V_{\text{disk}}}{S_{\text{disk}}} = 22,5 \text{ jobs}$$

$$\rightarrow X_0' = \frac{M}{Z+R'} = 2 \text{ jobs/s} \quad \Rightarrow X_{\text{disk}}' = X_0' V_{\text{disk}}' = 20 \text{ jobs/s}$$

$$\Rightarrow X_{\text{disk}}^b = X_{\text{disk}} - X_{\text{disk}}' = 2,5 \text{ job/s}$$

$$\Rightarrow X_0^b = \frac{X_{\text{disk}}^b}{V_{\text{disk}}'} = \frac{2,5 \text{ jobs}}{5} = 0,5 \text{ jobs/s} \quad \checkmark$$

$$R' = \frac{M}{X_0'} - Z$$

\hookrightarrow Tale valore è minimizzato se X_0' è massimizzato.

$$\rightarrow X_0' = \frac{X_{\text{disk}}'}{V_{\text{disk}}'}$$

\hookrightarrow Tale valore è massimizzato se X_{disk}' è massimizzato.

$$\rightarrow \max X_{\text{disk}}' = \max (X_{\text{disk}} - X_{\text{disk}}^b)$$

$$\rightarrow \max X_{\text{disk}}' = M_{\text{disk}} = \frac{1}{0,04 \text{ s}} = 25 \text{ job/s}$$

$$X_{\text{disk}}^b = X_0^b V_{\text{disk}}' = 1,5 \text{ job/s} \cdot 5 = 7,5 \text{ jobs/s}$$

$$\Rightarrow \max X_{\text{disk}} = 25 \text{ job/s} - 7,5 \text{ job/s} = 17,5 \text{ job/s}$$

$$\Rightarrow \max X_0' = \frac{17,5 \text{ job/s}}{10} = 1,75 \text{ job/s}$$

$$\Rightarrow R' \leq \frac{40}{1,75 \text{ job/s}} - 15 \text{ s} = 7,9 \text{ s} \quad (= \min R')$$

Ora abbiamo appena calcolato il LOWER BOUND del tempo di risposta interattivo.

BATCH MEAN

È un altro metodo di simulazione, che si basa sul bias della simulazione rispetto ai valori teorici elaborati allo stato iniziale del sistema. L'idea è effettuare un'unica run molto lunga che poi viene suddivisa in più parti (batch); in tal modo, per i batch non iniziali, non si arriva uno stato iniziale diverso dallo stato "tipico" del sistema (per cui evitiamo di partire ad esempio da un sistema vuoto se la popolazione media del sistema è $m > 0$; in questo caso il sistema vuoto porta a tempi di risposta più brevi e così via).

Nella pratica, si suddivide la simulazione in K batch lunghi b .

Attenzione nel selezionare K, b : c'è bisogno di un trade-off:

→ Se b è troppo piccolo, i batch diventano troppo dipendenti l'uno dall'altro.

→ Se b è troppo grande, la variabilità viene aumentata, rendendo la simulazione poco realistica. Tra l'altro ne consegue un K molto piccolo, che porta a un intervallo di confidenza troppo grande.

Comunque sia, esistono delle linee guida per selezionare K, b . → raccomandato $K=64$

N.B.: Le dimensioni dell'intervallo di confidenza delle misurazioni diminuiscono al diminuire di b e viceversa; notiamo che il centro dell'intervallo di confidenza è indipendente da b, K .

24/05/2022

Come linee guida, K deve essere ≥ 32 . Il valore raccomandato è 64.

Tornando al modello analitico...

CALCOLO DEI BOUND

I bound danno delle misure ottimistiche e/o pessimistiche per il sistema.

→ Ricordiamo che il tempo di residence è dato da $D_i = V_i \cdot S_i$:

$$\Rightarrow D_i = V_i \frac{B_i}{C_i} = \frac{C_i}{C_0} \cdot \frac{B_i}{C_i} = \frac{B_i}{C_0}$$

→ Inoltre l'utilizzazione è data da $U_i = X_i \cdot S_i$:

$$\Rightarrow U_i = X_i V_i S_i = \lambda V_i S_i = \lambda D_i$$

→ Il bottleneck del sistema è il sistema col D_i più grande.

$$\hookrightarrow D_{\text{bottleneck}} = D_b = V_b S_b = \max \{ V_1 S_1, V_2 S_2, \dots, V_n S_n \}$$

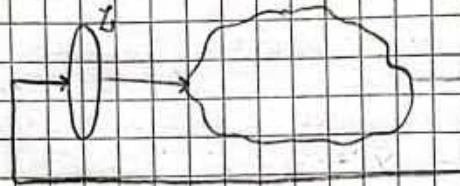
Di conseguenza: $\lambda_{\text{saturation}} = \lambda_{\text{sat}} = \frac{1}{V_b S_b} = \frac{1}{D_b} \rightarrow$ il throughput non può superare tale valore.

⇒ $\frac{1}{D_b}$ è il bound ottimistico per il throughput (chiaramente quello pessimistico è 0).

Per quanto riguarda il tempo di risposta, il caso ottimo è dato dalla somma dei tempi di risposta nei singoli centri: $D = \sum_{i=1}^n D_i$

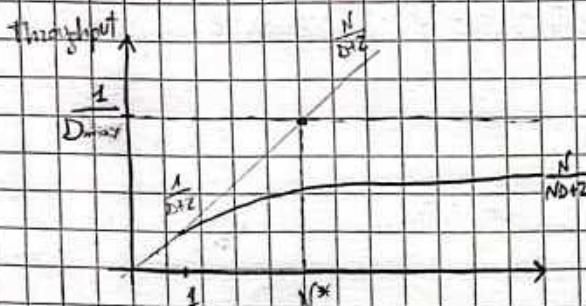
Il caso pessimo lo si ha nel caso in cui nel sistema arriva un burst di n job insieme che poi visiteranno gli stessi centri nello stesso ordine.

In realtà nei sistemi aperti il tempo di risposta massimo è un'informazione poco significativa. In un sistema chiuso invece:



$$X_0 = \frac{X_b}{V_b} = \frac{1}{D_{\max}}$$

poiché $D_b = V_b S_b$; $X_b = V_b X_0$



$\rightarrow \frac{1}{D+Z}$ è il throughput di un job da solo nel sistema.

$\rightarrow \frac{N}{D+Z}$ è il throughput di N job che non interferiscono. \rightarrow THROUGHPUT MAX

$\rightarrow N^* = \frac{D+Z}{D_{\max}} \rightarrow$ PUNTO DI SATURAZIONE

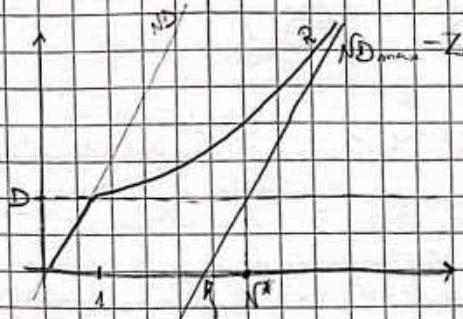
$\rightarrow \frac{N}{D+Z}$ è il throughput di N job che si interfondono sempre (vedere curva in nero).

\rightarrow Il throughput "realistico" è rappresentabile con una curva compresa tra $\frac{1}{D_{\max}}$ e $\frac{N}{D+Z}$.

Per quanto riguarda il tempo, ricordiamo che:

$$\rightarrow R = N X_0 - Z \rightarrow$$
 Per i job interattivi.

$$\hookrightarrow R \geq \max\{D, ND_{\max} - Z\} \quad \& \quad R \leq ND$$



Proviamo a vedere cos'è questo punto (che chiameremo N_b):

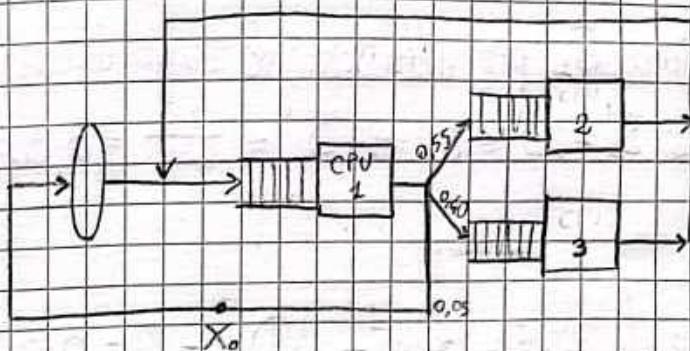
$$N_b D_{\max} - Z = 0 \Rightarrow N_b = \frac{Z}{D_{\max}} \Rightarrow N^* = \frac{D+Z}{D_{\max}} = \frac{Z}{D_{\max}} + \frac{D}{D_{\max}}$$

\hookrightarrow Quando il sistema è saturo, ci sono $\frac{D}{D_{\max}}$ job nel sistema stesso e N_b job in fase di thinking.

N.B.: N è il numero totale di job (n° verchio H).

26/05/2022

Esempio di analisi del bottleneck:



$$\begin{aligned} Z &= 20 \text{ s} \\ S_1 &= 0,05 \text{ s} \\ S_2 &= 0,08 \text{ s} \\ S_3 &= 0,04 \text{ s} \end{aligned}$$

Fissiamo arbitrariamente $V_0 = 1$.

$$\rightarrow V_0 = 0,05 V_1$$

$$\rightarrow V_1 = V_0 + V_2 + V_3$$

$$\rightarrow V_2 = 0,55 V_1$$

$$\rightarrow V_3 = 0,40 V_1$$

$$\Rightarrow \begin{cases} V_1 = 20 \\ V_2 = 11 \\ V_3 = 8 \\ V_0 = 1 \end{cases}$$

$$D_1 = V_1 S_1 = 1 \text{ s}$$

$$D_2 = V_2 S_2 = 0,88 \text{ s}$$

$$D_3 = V_3 S_3 = 0,32 \text{ s}$$

\Rightarrow Il colo del bottiglia è la CPU (1).

$$\rightarrow D = D_1 + D_2 + D_3 = 2,2 \text{ s}$$

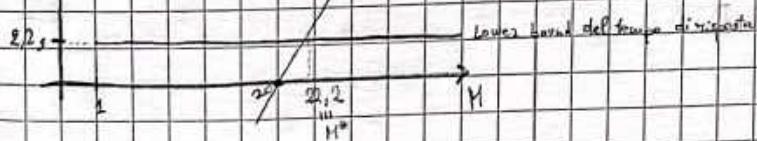
R₀ ↑

$$MD_{max} - Z \equiv M-20$$

- LEGGE DELL'UTILIZZAZIONE: $U_i = X_i S_i$

$$U_i = X_0 D_i$$

- LEGGE DEL FLUSSO FORZATO: $X_i = X_0 V_i$



\rightarrow Applichiamo Little al "centro" di thinking: $M = X_0 Z = \frac{1}{V_0 S_0} Z = 20$ terminali

$$\rightarrow 2,2 = M^* - 20 \Rightarrow M^* = 22,2 \quad \leftarrow$$
 Intersezione tra $D = 2,2$ e $MD_{max} - Z \equiv M-20$

DOMANDA 1: Se il throughput misurato è 0,715 j/s e il tempo di risposta del sistema è $R_0 = 5,2 \text{ s}$, quanti sono i ~~30~~ terminali collegati?

$$M = \frac{(R_0 + Z)X_0}{Z} = 18 \text{ terminali}$$

DOMANDA 2: È possibile che con 30 terminali il tempo di risposta sia di 8 secondi?

Se no, come bisogna migliorare la CPU affinché il tempo di risposta possa scendere a 8 secondi?

$$\rightarrow R_0 \geq M D_{\max} - Z = 30 \cdot 1 - 20 \text{ s} = 10 \text{ s} > 8 \text{ s}$$

\hookrightarrow In realtà il tempo di risposta non può scendere sotto i 10 s.

Calcoliamo come deve cambiare S_1 per abbassare il tempo di risposta a 8 s:

$$M \cdot S_1 - Z \leq 8 \text{ s} \Rightarrow 30 \cdot 20 S_1' - 20 \leq 8 \Rightarrow S_1' = \frac{28}{600} = 0,067$$

$$\Rightarrow \text{SPEEDUP} := \frac{S_1}{S_1'} = 1,07$$

\hookrightarrow Ora il bound per il tempo di risposta diventa $M D_{\max} - Z = 0,93 M - 20$

\rightarrow Il bound relativo alla risorsa 2 è dato da $0,88 M - 20$.

\rightarrow Il bound relativo alla risorsa 3 è dato da $0,32 M - 20$.

Chiaramente, se miglioriamo solo la CPU, il bound comunque non può andare oltre quello relativo alla risorsa 2.

Esercizio 1:

$$\cdot OP = 5 \text{ min}$$

$$\cdot \text{Utilizzazione CPU} = 30\% = U_{\text{CPU}}$$

$$\cdot \text{Domanda DISK} = 0,4 = D_{\text{disk}}$$

$$\cdot \# \text{operazioni I/O per transazione} = 10 = V_{\text{disk}}$$

$$\cdot \text{Utilizzazione DISK} = 40\% = U_{\text{disk}}$$

$$\cdot \text{Tempo di risposta} = 15 \text{ s/transazione} = R$$

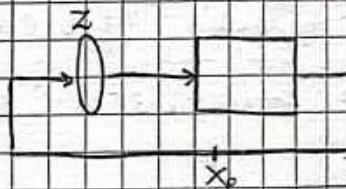
$$\cdot \# \text{utenti} = 50 = M$$

$$\cdot Z = ?$$

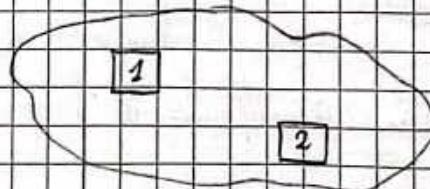
$$R = \frac{M}{X_0} - Z \Rightarrow Z = \frac{M}{X_0} - R$$

$$X_0 = \frac{U_{\text{disk}}}{D_{\text{disk}}} = 1 \text{ transaz./s}$$

$$\Rightarrow Z = \frac{50}{1} - 15 = 35 \text{ s}$$



Esercizio 2:



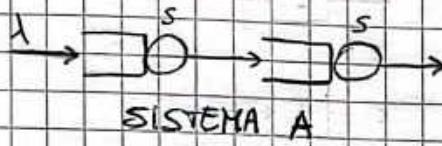
- Tempo di risposta di 1 = 10 s = R_1
- Tempo di risposta di 2 = 1 s = R_2
- Throughput di 1 = 4 transaz./s = X_1
- Throughput di 2 = 8 transaz./s = X_2
- Throughput del sistema = 16 transaz./s = X_0
- $R = ?$

$$R = V_1 R_1 + V_2 R_2$$

$$X_i = X_0 V_i \Rightarrow V_i = \frac{X_i}{X_0} \Rightarrow V_1 = 1; V_2 = 2$$

$\Rightarrow R = 12 \text{ s}$

Esercizio 3:



$$\lambda = 0,4 \text{ transaz./s}$$

$$S = 0,5 \text{ s}$$

$$R_A/R_B = ?$$

$$P_A = \cancel{\lambda} S = 0,2$$

$$P_B = \lambda \cdot 2S = 0,4$$

Dalle formule della KP: $R = \frac{S}{1-\lambda S}$

$$\Rightarrow \begin{cases} R_A = 2 \cdot \frac{S}{1-\lambda S} = \frac{5}{4} \text{ s} \\ R_B = \frac{2S}{1-\lambda \cdot 2S} = \frac{5}{3} \text{ s} \end{cases}$$

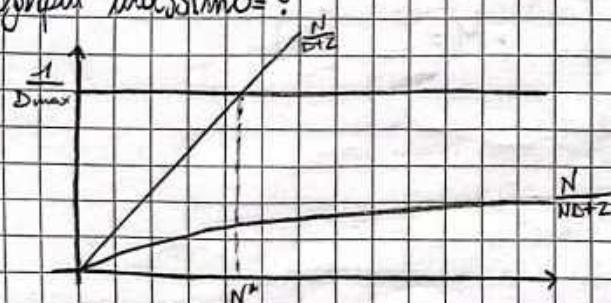
$$\Rightarrow \frac{R_A}{R_B} = \frac{3}{4}$$

Esercizio 4:

$$D_1 = 1 \text{ s} \quad D_2 = 1 \text{ s} \quad D_3 = 2 \text{ s}$$

$$\text{Urate con } Z = 6 \text{ s}$$

Throughput massimo?



$$N^* = \frac{D+Z}{D_{\max}} = \frac{4s + 6s}{2s} = 5 \rightarrow \text{da 5 utenti in più il sistema diventa saturo.}$$

\rightarrow Se abbiamo 4 utente solo $\Rightarrow X_{\max} = \frac{N}{D+Z} = \frac{1}{10} \text{ job/s}$

\rightarrow Da 5 utenti in più la formula sarebbe diventata $X_{\max} = \frac{1}{D_{\max}}$

Esercizio 5:

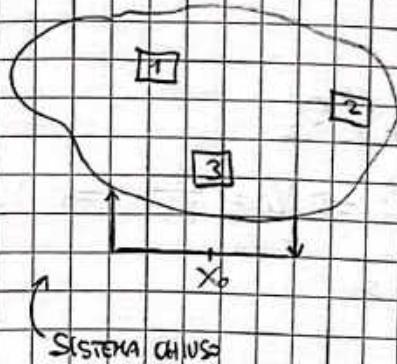
- $D_1 = 1s$

- $D_2 = 2s$

- $D_3 = 2s$

- $N = 2s$

- $X_0 = ?$



$$E[t_i^{(n)}] = E[s_i] \left(1 + E[n_i^{(N-1)}] \right)$$

$$\Rightarrow v_i E[t_i^{(n)}] = v_i E[s_i] \left(1 + E[n_i^{(N-1)}] \right)$$

$$\Rightarrow R_i^{(n)} = D_i \left(1 + Q_i^{(N-1)} \right)$$

$\underbrace{\phantom{R_i^{(n)}}}_{E[n_i]}$

FASSO BASE: $Q_1(0) = Q_2(0) = Q_3(0) = 0$

$$\Rightarrow R_1(1) = 1s \quad ; \quad R_2(1) = 2s \quad ; \quad R_3(1) = 2s$$

$$\Rightarrow X_0(1) = \frac{1}{5} \text{ transaz./s}$$

$\underbrace{}_{\text{posta da } M = (R+Z)X_0}$

$$Q_1(1) = X_0(1) R_1(1) = 0,2$$

$$Q_2(1) = X_0(1) R_2(1) = 0,4$$

$$Q_3(1) = X_0(1) R_3(1) = 0,4$$

$$\Rightarrow R_1(2) = D_1 \left(1 + Q_1(1) \right) = 1,2s \quad ; \quad R_2(2) = 2,8s \quad ; \quad R_3(2) = 2,8s$$

$$\Rightarrow X_0(2) = \frac{2}{R_1(2) + R_2(2) + R_3(2)} = \frac{2}{6,8} \approx 0,2941 \text{ transaz./s}$$

$\underbrace{}_{M = (R+Z)X_0}$

Perciò:

- Domanda CPU = 4 sec/trans.

- $M = ? \quad (5)$

- Utilizzaz. CPU = 50%

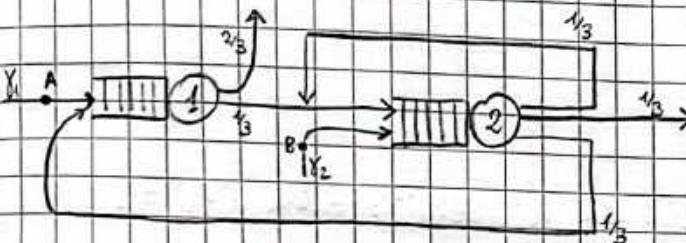
- Tempo Misp. = 15 s

- Think time = 25 s

- $O = 1 \text{ min}$
- $M = 80$
- Tempo risp. medio = 5 s
- # transat. complete = 60
- $U_{CAV} = 80\%$
- $U_{DISK1} = 50\%$
- $U_{DISK2} = 50\%$

27/05/2022

Esercizio 1:



$$\cdot \mu_1 = 3 \text{ pkt/s}$$

$$\cdot \mu_2 = 5 \text{ pkt/s}$$

$$\cdot \gamma_2 = 1 \text{ pkt/s}$$

$\cdot \gamma_1 \text{ MAX prima della sottrazione} = ?$

$$\begin{cases} \lambda_1 = \gamma_1 + \frac{1}{3} \lambda_2 \\ \lambda_2 = \gamma_2 + \frac{1}{3} \lambda_1 + \frac{1}{3} \lambda_2 \end{cases} \Rightarrow \begin{cases} \lambda_1 = \frac{6}{5} \gamma_1 + \frac{3}{5} \\ \lambda_2 = \frac{3}{5} \gamma_1 + \frac{9}{5} \end{cases}$$

$$\rho_1 = \frac{\lambda_1}{\mu_1} < 1 \Rightarrow \frac{1}{\mu_1} \left(\frac{6}{5} \gamma_1 + \frac{3}{5} \right) < 1 \Rightarrow \gamma_1 < 2 \text{ pkt/s}$$

$$\rho_2 = \frac{\lambda_2}{\mu_2} < 1 \Rightarrow \frac{1}{\mu_2} \left(\frac{3}{5} \gamma_1 + \frac{9}{5} \right) < 1 \Rightarrow \gamma_1 < 5,3 \text{ pkt/s}$$

$\Rightarrow \gamma_1$ deve essere strettamente minore di 2 pkt/s.

—————

Se $\gamma_1 = 1,8 \text{ pkt/s}$, qual è il tempo di risposta per un pacchetto che entra nel sistema, partendo dal punto A?

$$\rightarrow \lambda_1 = 2,76 \text{ pkt/s} ; \quad \lambda_2 = 2,88 \text{ pkt/s}$$

$$\rightarrow \gamma = \gamma_1 + \gamma_2 = 2,8 \text{ pkt/s}$$

$$\rightarrow N_{1/A} = \frac{\lambda_1}{\gamma_1} = 1,533333 ; \quad N_{2/A} = \frac{\lambda_2}{\gamma_1} = 1,6$$

↑ # visite al centro 1 per i pacchetti che entrano nel sistema a partire dal punto A.

$$\rightarrow \rho_1 = \frac{\lambda_1}{\mu_1} = 0,92 ; \quad \rho_2 = \frac{\lambda_2}{\mu_2} = 0,576$$

$$\rightarrow E[t_1] = \frac{1}{\mu_1 - \lambda_1} \Rightarrow E[t_1] = 4,166667 \text{ s} ; \quad E[t_2] = 0,471698 \text{ s}$$

$$\rightarrow E[m_i] = \frac{p_i}{1-p} \Rightarrow E[m_1] = 11,5 ; \quad E[m_2] = 1,35849$$

$$\rightarrow E[t_{R_A}] = E[t_1] \cdot \gamma_{1/n} + E[t_2] \cdot \gamma_{2/n} = 7,143605 \text{ s}$$

→ Analogamente, è possibile calcolare $E[t_{R_B}] = 12,85849 \text{ s}$

NB: Il tempo di risposta globale è più piccolo sia di $E[t_{R_A}]$ sia di $E[t_{R_B}]$:

$$E[t_R] = 4,59237 \text{ s} = E[t_{R_A}] \cdot \frac{\gamma_1}{\gamma} = E[t_{R_B}] \cdot \frac{\gamma_2}{\gamma}$$

→ Applichiamo Little:

$$\bar{N} = \gamma E[t_R] = 12,8585$$

$$\bar{N} = \gamma_1 E[t_{R_A}] = 12,8585$$

$$\bar{N} = \gamma_2 E[t_{R_B}] = 12,8585$$

} Tutto torna γ

Per casa:

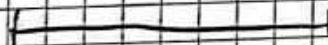
$$\cdot U_{CPU} = 0,5$$

• Domande di servizio alla CPU = ?

$$\cdot \text{Tempo risp.} = 15 \text{ s}$$

$$\cdot Z = 5 \text{ s}$$

$$\cdot M = 100$$



$$\cdot T = 60 \text{ min}$$

$$\cdot \# frames completati = 900 = C$$

$$\cdot M = 60$$

$$\cdot \# media utenti mossi in attesa di risposta (stanno pensando) = 57,5$$

$$\cdot R = ? \text{ (0 s)}$$