

# TUTOR CAPACITY PLANNING

21/03/2018

Tutor: Ing. Stefano Conoci [conoci.dis.uniroma1.it](http://conoci.dis.uniroma1.it)

Reliability  $\rightarrow R(t)$ : tiene conto dell'intervallo temporale  $(0, t)$

Availability  $\rightarrow A(t)$ : tiene conto solo dell'istante  $t$

DEPENDABILITY  $\leftarrow$  SPERIMENTALE (non lo consideriamo)  
 ANALITICO: basato su modelli

3 modelli che trattiamo sono di 2 tipi:

- COMBINATORI  $\rightarrow$  Availability, Reliability

- MARKOVIANI  $\rightarrow$  Reliability, Availability, Security, Perfromability

3 modelli COMBINATORI sono più semplici da trattare, ma come vedremo offrono meno criteri di valutazione rispetto ai MARKOVIANI

$$R(t) = e^{-\lambda t}$$

$\rightarrow$  nel caso ci mettiamo nella porta lineare delle BATHTUBE curve (faremo sempre quest'assunzione)

$$A(t) = \frac{MTTF}{MTTF + MTTR}$$

MTTF: mean time to failure

MTTR: mean time to repair

$$MTTF = \int_0^{\infty} t q(t) dt \stackrel{\text{dens. di rate di failure}}{=} - \int_0^{\infty} t \frac{dR(t)}{dt} dt \stackrel{\text{permetti}}{=} \int_0^{\infty} t R'(t) dt = - \frac{dR(t)}{dt}$$

$$= \int_0^{\infty} e^{-\lambda t} dt$$

$$\lambda = \frac{dR(t)}{dt} \quad \rightarrow \text{per } t=0 \quad \lambda = \frac{dR(0)}{dt} = -\frac{dR(t)}{dt}$$

$\rightarrow$  per  $t=\infty$   $R(t)$  va a 0 più rapidamente di  $t$  che va a  $\infty$

risolvo per sostituzione:  $-\lambda t = u$ ,  $dt = -du/\lambda$

$$\int_0^{\infty} e^{-\lambda t} dt \stackrel{\text{per sost.}}{=} \int_{-\infty}^0 e^u \left(-\frac{du}{\lambda}\right) = -\frac{1}{\lambda} \int_{-\infty}^0 e^u du = -\frac{1}{\lambda} [e^u]_{-\infty}^0 = \frac{1}{\lambda}$$

$$MTTF = 1/\lambda$$

Analogamente...

$$MTTR = 1/\mu$$

$\lambda$ : frequenza di fault istantanea o densità di probabilità di fault

$\mu$ : densità di probabilità di riparazione

## INTERCONNECTIONI

- Serie: se ho  $n$  sottosistemi in serie, affinché il sistema funzioni devono funzionare tutti i sottosistemi
- Parallello: è differente dal Serie, almeno un sottosistema deve funzionare
- Mount of  $N$ : come dice il termine, almeno  $m$  su  $N$  componenti devono funzionare.

Reliability e Availability nelle interconnessioni:

### SERIALE

$$R(t) = \prod_{i=1}^n R_i(t) \quad A(t) = \prod_{i=1}^n A_i(t)$$

### PARALLELO

$$R(t) = 1 - P(\text{nessuno funziona}) = 1 - (1-R_1(t))(1-R_2(t)) \dots (1-R_m(t))$$

$$A(t) = 1 - P(\text{nessuno disponibile}) = 1 - (1-A_1(t))(1-A_2(t)) \dots (1-A_m(t))$$

### Mount of $N$

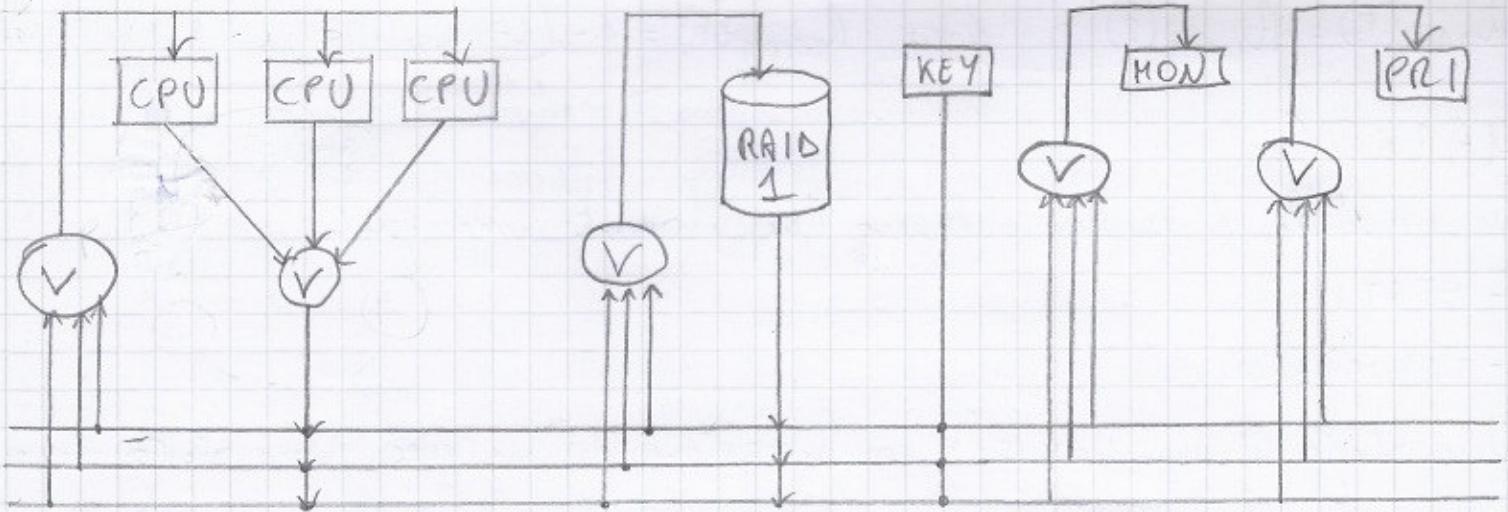
$$R(t) = \sum_{i=0}^K \binom{N}{i} R_c^{N-i}(t) (1-R_c(t))^i \quad \text{con } K=N-m$$

$$A(t) = \sum_{i=0}^K \binom{N}{i} A_c^{N-i}(t) (1-A_c(t))^i$$

cioè tutte le combinazioni in cui tutti funzionano (solo 1) + tutte le combinazioni in cui uno solo non funziona +...+ tutte le combinazioni in cui  $K=N-m$  non funzionano

Ese.

Evaluate the reliability and the availability of a system composed of three CPUs (working in parallel and whose output is given by a voter), a RAID 1 storage system with 6+6 disks, three buses (working in parallel and whose outputs are given by voters), one keyboard, one printer and one monitor. Assume that faults are exponentially distributed with rates equal to  $\lambda_{CPU}$ ,  $\lambda_{DISK}$ ,  $\lambda_{BUS}$ ,  $\lambda_{KEY}$ ,  $\lambda_{PR}$ ,  $\lambda_{MON}$ ,  $\lambda_{ROT-CPU}$ ,  $\lambda_{ROT-BUS}$ . The repair rate  $\mu$  is equal for all components.



### MONITOR

$$R_{\text{MON}}(t) = e^{-\lambda_{\text{MON}} t}$$

$$A_{\text{MON}}(t) = \frac{1/\lambda_{\text{MON}}}{1/\lambda_{\text{MON}} + 1/\mu}$$

### KEYBOARD

$$R_{\text{KEY}}(t) = e^{-\lambda_{\text{KEY}} t}$$

$$A_{\text{KEY}}(t) = \frac{1/\lambda_{\text{KEY}} t}{1/\lambda_{\text{KEY}} + 1/\mu}$$

$$R_{\text{tot}}(t) = R_{\text{CPU}}(t) \cdot R_{\text{SYS}}(t) \cdot R_{\text{BUS}}(t) \cdot R_{\text{RAID}}(t) \cdot R_{\text{MON}}(t) \cdot R_{\text{KEY}}(t)$$

$$R_{\text{CPU}}(t) = [R_{\text{CPU}}(t)^3 + 3 R_{\text{CPU}}(t)^2 (1 - R_{\text{CPU}}(t))] R_{\text{tot}}(t) \quad \text{dove } R_{\text{CPU}}(t) = e^{-\lambda_{\text{CPU}} t}$$

$$R_{\text{CPU-tot}}(t) = e^{-\lambda_{\text{CPU-tot}} t}$$

$$R_{\text{SYS}}(t) = R_{\text{ROT-BUS}}(t)^4 [R_{\text{BUS}}(t)^3 + 3 R_{\text{BUS}}(t)^2 (1 - R_{\text{BUS}}(t))] \quad \text{dove } R_{\text{ROT-BUS}}(t) = e^{-\lambda_{\text{ROT-BUS}} t}$$

$$R_{\text{BUS}}(t) = e^{-\lambda_{\text{BUS}} t}$$

Continuando l'esercizio...

28/03/2018

Per le reliability del bus, abbiamo le reliability di 2 out of 3 bus in serie con i 4 rotori di output del bus, infatti il sottosistema bus contiene anche i rotori di output del bus.

$$R_{\text{SYS}}(t) = R_{\text{ROT-BUS}}(t)^4 [R_{\text{BUS}}(t)^3 + 3 R_{\text{BUS}}(t)^2 (1 - R_{\text{BUS}}(t))]$$

Per quanto riguarda il RAID 1 abbiamo un 4+4, dunque possiamo avere una serie di 4 copie, ogni coppia formata da un disco e dal suo gemello (se una delle copie non funziona, il sistema non funziona). Ogni coppia ha i due dischi in parallelo (basta che funziona almeno un disco di ogni coppia).

$$R_{\text{PAIR}}(t) = R_{\text{COPPIA}}(t)^4 \quad \text{dove } R_{\text{COPPIA}}(t) = 1 - \underbrace{(1 - R_{\text{DISK}}(t))^2}_{\text{0 dischi su 2 funzionano}}$$

## MARKOV

Per i modelli di Markov abbiamo due elementi importanti:

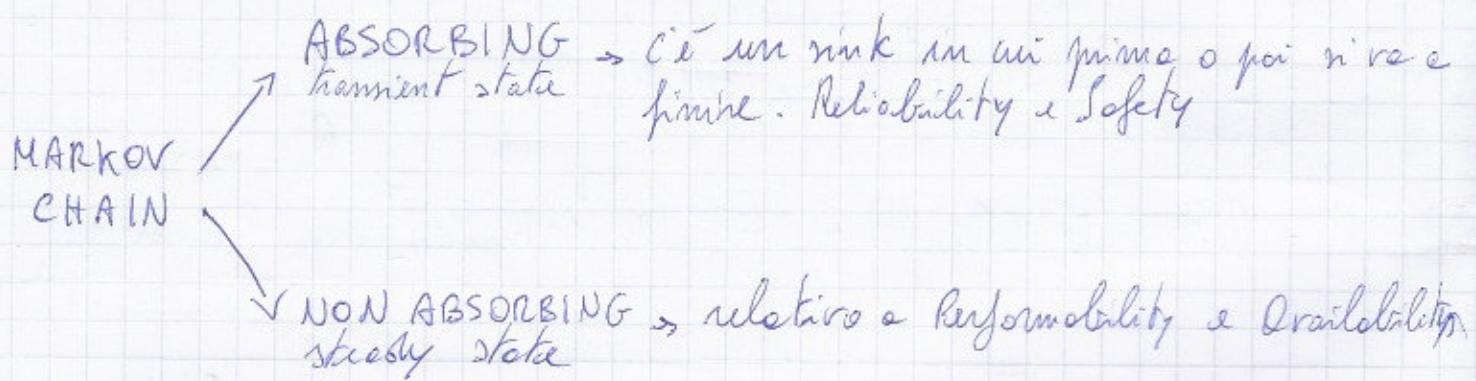
- STATI
- TRANSIZIONI

Uno stato è una possibile combinazione del sistema. Uno stato deve avere TUTTE le informazioni, e ciò che intendiamo è che uno stato deve contenere il suo percorso.

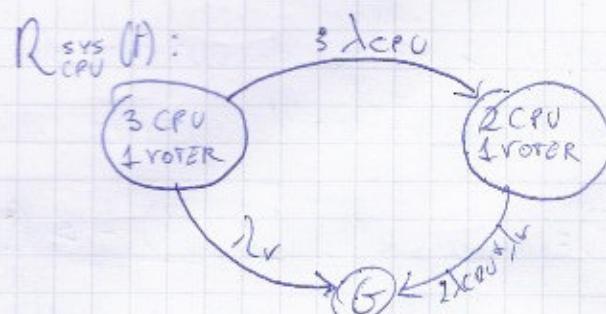
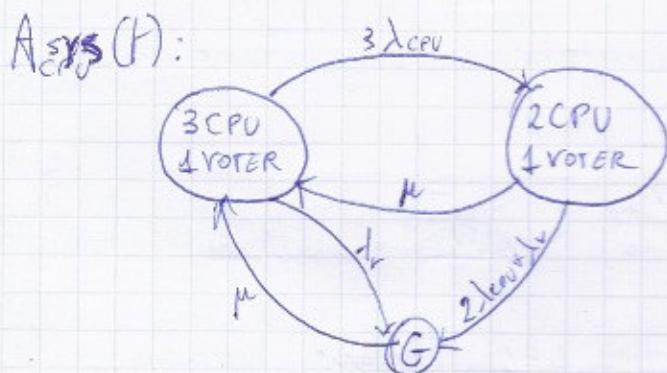
La prob. di stare in uno stato si compone di 2 parti:

$$P(\text{stato } Y \text{ in } t + \Delta t) = P(\text{stato } X \text{ in } t \text{ è in } Y) + P(\text{stato } Y \text{ in } t \text{ è in } X)$$

$$\text{Taylor: } 1 - e^{-\lambda \Delta t} \approx \lambda \Delta t$$



Es. precedente con modello MARKOVIANO



Note: quando si parla di availability si tiene conto anche della possibilità di riparare i componenti.

Qualitativamente:

### $R_{CPU}^{SYS}(t)$ :

$$\begin{cases} P_{3,1}(t+\Delta t) = P_{3,1}(t)[1 - (3\lambda_{CPU} \Delta t + \lambda_V \Delta t)] \\ P_{2,1}(t+\Delta t) = P_{2,1}(t)[1 - (2\lambda_{CPU} \Delta t + \lambda_V \Delta t)] + P_{3,1}(t) 3\lambda_{CPU} \Delta t \\ P_G(t+\Delta t) = P_{3,1}(t) \lambda_V \Delta t + P_{2,1}(t)[2\lambda_{CPU} \Delta t + \lambda_V \Delta t] + P_G(t) \cdot 1 \end{cases}$$

$$\begin{cases} \frac{\partial P_{3,1}(t+\Delta t)}{\partial t} = \lim_{\Delta t \rightarrow 0} \frac{P_{3,1}(t+\Delta t) - P_{3,1}(t)}{\Delta t} = [-3\lambda_{CPU} - \lambda_V] P_{3,1}(t) \\ \frac{\partial P_{2,1}(t+\Delta t)}{\partial t} = \lim_{\Delta t \rightarrow 0} \frac{P_{2,1}(t+\Delta t) - P_{2,1}(t)}{\Delta t} = [-2\lambda_{CPU} - \lambda_V] P_{2,1}(t) + P_{3,1}(t) 3\lambda_{CPU} \\ \frac{\partial P_G(t+\Delta t)}{\partial t} = \lim_{\Delta t \rightarrow 0} \frac{P_G(t+\Delta t) - P_G(t)}{\Delta t} = P_{3,1}(t) \lambda_V + P_{2,1}(t)[2\lambda_{CPU} + \lambda_V] \end{cases}$$

### $A_{CPU}^{SYS}(t)$ :

$$\begin{cases} P_{3,1}(t+\Delta t) = P_{3,1}(t)[1 - (3\lambda_{CPU} \Delta t + \lambda_V \Delta t)] + P_G(t) \cdot \mu \Delta t + P_{2,1}(t) \cdot \mu \Delta t \\ P_{2,1}(t+\Delta t) = P_{2,1}(t)[1 - (\mu \Delta t + \lambda_{CPU} \Delta t + \lambda_V \Delta t)] + P_{3,1}(t) 3\lambda_{CPU} \Delta t \\ P_G(t+\Delta t) = P_{3,1}(t) \lambda_V \Delta t + P_{2,1}(t)[2\lambda_{CPU} \Delta t + \lambda_V \Delta t] + P_G(t)[1 - \mu \Delta t] \end{cases}$$

$$\begin{cases} \frac{\partial P_{3,1}(t+\Delta t)}{\partial t} = \lim_{\Delta t \rightarrow 0} \frac{P_{3,1}(t+\Delta t) - P_{3,1}(t)}{\Delta t} = [-3\lambda_{CPU} - \lambda_V] P_{3,1}(t) + P_G(t) \mu + P_{2,1}(t) \mu \\ \frac{\partial P_{2,1}(t+\Delta t)}{\partial t} = \lim_{\Delta t \rightarrow 0} \frac{P_{2,1}(t+\Delta t) - P_{2,1}(t)}{\Delta t} = [-2\lambda_{CPU} - \lambda_V - \mu] P_{2,1}(t) + P_{3,1}(t) 3\lambda_{CPU} \\ \frac{\partial P_G(t+\Delta t)}{\partial t} = \lim_{\Delta t \rightarrow 0} \frac{P_G(t+\Delta t) - P_G(t)}{\Delta t} = P_{3,1}(t) \lambda_V + P_{2,1}(t)[2\lambda_{CPU} + \lambda_V] - P_G(t) \mu \end{cases}$$

## Comportamento degli Hard Disk

Sono dispositivi a memoria non volatili. La tecnologia è MAGNETICA.

Distinguiamo diversi elementi:

- PIATTI
- TESTINA
- ...

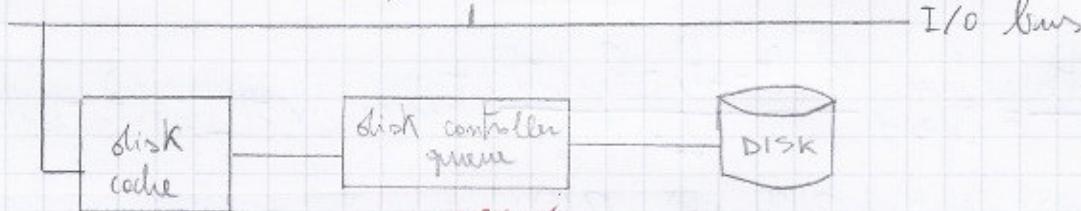
Si ha più di un piatto, e per ogni piatto vi usano entrambe le facce. In realtà vi hanno più testine, come se stesse braccia. Quindi vi sono diverse insieme. Per ogni piatto abbiamo 2 testine: una per ogni faccia. I movimenti sono meccanici e questo è un colpo di bottiglia.

Ottiamo 2 movimenti:

- Rotazione del piatto
- Spostamento delle testine.

Ogni piatto è composto da treccia (archi concentrici) e ogni treccia è composta da più settori.

Un disco è dotato anche di CACHE, più veloce e di dimensione molto ridotta. Se siamo fortunati troviamo il dato in cache, altrimenti abbiamo aspettare tutto il tempo di ricerca.



## NOMENCLATURA

$$S_d = \text{ControlTime} + P_{\text{min}} \quad (\text{Seek} + \text{Latency} + \text{Transfer})$$

ControlTime → tempo di risposta  
P<sub>min</sub> → tempo minimo per leggere o scrivere  
Seek → raggiungere il cilindro (traversa)  
Latency → raggiungere il settore

$$\text{Transfer} = \frac{\text{Block Size}}{\text{TransferRate}}$$

Il ControlTime include il tempo di ordinazione della richiesta via al tempo per leggere o scrivere sulla cache.

## Workload

È il carico di lavoro che avvia il disco. Lo rappresentiamo come una sequenza di richieste. Le chiamiamo descrittori.

Possiamo distinguere due tipi di Workload:

- Random Workload: 10, 201, 15, 1023, 45...
- Sequential Workload: 4, 102, 103, 104, 105, 106, 15, ...  
burst sequenziale

Le cache effettuano una politica 'look ahead', cioè cerca in cache il dato a un po' di cose sequenzialmente successive.

Sarà caso di Random Workload:

CT: control time

ST<sub>rand</sub>: seek time per workload random

RL: rotation latency

TT: transfer time  $\approx$  1ms

$$S_{d,rand} = \overline{CT} + ST_{rand} + RL + TT$$

ST<sub>rand</sub> ???

N = # tracce

$$T_{MAX} = N \cdot d \cdot v$$

$$\begin{aligned} T_{MEDIO} &= \sum_{x=0}^N \sum_{y=0}^N |x-y| \approx \int_{x=0}^N \int_{y=0}^N |x-y| dx dy = \int_{x=0}^N (x-y) dx + \int_{y=0}^N (y-x) dx = \\ &= \left[ \frac{x^2}{2} - xy \right]_y^N + \left[ \frac{y^2}{2} - xy \right]_y^N + \left[ yx - \frac{x^2}{2} \right]_0^y = \left[ \frac{N^2}{2} - Ny + \frac{y^2}{2} \right] + \left[ y^2 - \frac{y^2}{2} \right] = \\ &= \frac{N^2}{2} - Ny \dots \text{mon...} \end{aligned}$$

Tuizio: tutte le possibili combinazioni di spostamento da una qualsiasi traccia a qualsiasi altra

Il risultato deve venire  $\frac{N^3}{3}$  dove N è il numero delle tracce.

$$\text{Quindi } ST_{rand} = \frac{ST_{MAX}}{3}$$

$$TT = \frac{BS}{TR}$$
 dove BS è Block Size (taglia delle richieste) e TR è il Transfer Rate

$$RL \approx 0.5 \cdot \frac{60}{RS}$$
  $\rightarrow$  Rotation Time  
dove 0.5 significa che avviene di fare metà rotazione, RS è la velocità di rotazione (il cosiddetto rpm).

Ovviamente si fa metà rotazione perché ormai non il coso mette più in moto il disco, dopo averlo spostato sulla traccia giusta il motore si mette in moto.

### 3<sup>o</sup> caso di Sequential Workload:

NREQ = # richieste totali

NRUN = # run sequenziali

RLEN = # richieste in media in una RUN

$$\text{Ese: } WL = \{0, 10, 11, 12, 8, 60, 41, 39\}$$

$$\begin{aligned} NREQ &= 8 \\ NRUN &= 5 \end{aligned}$$

$$S_{d,seq} = NREQ \cdot CT + NRUN \cdot ST_{rand} + RL + NRUN \cdot TT$$

$$\begin{aligned} RL (\text{singole richieste}) &= [0.5(1-U_d) + 1 \cdot U_d] \cdot RS = \\ &= (1+U_d)/2 \cdot RS = (1+U_d)/2 \cdot \frac{60}{RPM} \end{aligned}$$

Gioi con prob. ( $1-U_d$ ) il disco è libero e peggio come nel caso del Random Workload, altrimenti sono costretto con prob.  $U_d$  a fare un giro intero.

$$RL (\text{singole RUN}) = [0.5 + \frac{1+U_d}{2}(RLEN-1)] \cdot \frac{60}{RPM}$$

$$RL_{tot} = \left[ 0.5 + \left( \frac{1+U_d}{2} \right) (RLEN-1) \right] \frac{60}{RPM} \cdot NRUN$$

$$S_{d,seq} = CT + \frac{ST_{rand}}{RLEN} + \frac{RL}{RLEN} + \frac{TT}{RLEN}$$

$U_d$  = Utilization Factor =  
= Prob. di trovare il disco occupato a servire richieste

In la prima mossa verso giro per il resto delle richieste nella run si prende dal fattore di utilizzazione

$$\text{Nota: } \frac{NRUN}{NREQ} = \frac{1}{RLEN}$$

Sic in caso di Random che Sequential,  $U_d$  viene calcolato nel caso Random e poi ri usso per il caso sequenziale.

### Phenomeno:

- WORKLOAD RANDOM

$$\overline{S_d} = CT + ST_{rand} + \frac{1}{2} \frac{60}{RPM} + \frac{BS}{TR}$$

- WORKLOAD SEQUENZIALE

$$\overline{S_d} = CT + \frac{ST_{rand}}{RLEN} + \frac{0.5 + \left( \frac{1+U_d}{2} \right) [Rlength-1] \cdot \frac{60}{RPM}}{RLEN} + \frac{BS/TR}{RLEN}$$

CT = Control Time    ST<sub>rand</sub> = Seek Time

BS = Block size    TR = transfer rate

## Esercizio

arrival rate = 20 req/sec

Workload:

20% random

80% sequential

average run length = 24

Block Size = 2,048 bytes

Disk Revolution Speed: 7,200 RPM

Seek Time: 7 msec

Transfer Rate = 20 MB/sec

Controller Time = 0.1 msec

Note: non abbiamo un set finito di richieste, ma in view solo delle percentuali di **RAND** e **SEQ**. Dunque per il **SEQUENZIALE** si calcola il ServiceTime in base alla **ALEN** (il ServiceTime medio, considerando la **RELEN** media) e infine il ServiceTime totale medio sarà:

$$S_{TOT} = \%RAND \cdot S_{RAN} + \%SEQ \cdot S_{SEQ}$$

C'è un motivo per cui partiamo da quello Random e tale motivo è che ci serve il fattore di utilizzazione  $U_d$  da usare quando calcoliamo il fattore di Workload Sequential.

Partiamo col modellare il workload random.

$$\begin{aligned} \overline{S}_{RAN} &= CT + P_{RAN} (S_{RAN} + T_{Ranf} + LR) = \text{per contr. in ms} \\ &= 0.1 \text{ ms} + 1 \left[ 7 \text{ ms} + 0.5 \left( \frac{60}{7200} \right) \cdot 1000 + \frac{2048 \text{ Byte}}{20 \text{ MB/s}} \cdot 1000 \right] = \\ &= 0.1 \text{ ms} + 7 \text{ ms} + 4.17 \text{ ms} + 0.1 \text{ ms} = \\ &\approx 13.4 \text{ ms} \quad U_d = \text{arrival rate} \cdot (S_{RAN} + L_n + T_{Ranf}) = 0.02 \cdot 13.4 = 0.27 \end{aligned}$$

In quanto riguarda il workload sequenziale:

$$\begin{aligned} \overline{S}_{SEQ} &= CT + \frac{\overline{S}_{RAN}}{ALEN} + \frac{1}{2} \frac{1 + [EN - 1](1 + U_d)}{nEN} \cdot \frac{60}{7200} + \frac{TT}{ALEN} \\ &= 0.1 \text{ ms} + \frac{7 \text{ ms}}{24} + 0.5 \frac{1 + 23(1 + 0.27)}{24} \cdot \frac{60}{7200} \cdot 1000 + \frac{(2048 \text{ Byte} \cdot 1000)}{20 \text{ MB/s} \cdot 24} = 5.73 \end{aligned}$$

Supponendo  $U_d = \emptyset$  la parte in mezzo viene  $\frac{1}{2} \frac{ALEN}{nEN} \cdot AL = 0.5 \text{ ms}$ 

(è una legge (Little's Law) che dice che:

 $U_d = \lambda \cdot \overline{S}_{RAN}$  cioè ArrivalRate per ServiceTime del wwo Randomdove  $\lambda = 20 \text{ req/sec}$  ma devo calcolare anche le probabilità del wwo Random:

$$U_{RAN} = P_{RAN} \lambda \cdot \overline{S}_{RAN} = 20\% \cdot 20 \text{ req/s} \cdot 13.4 = ...$$

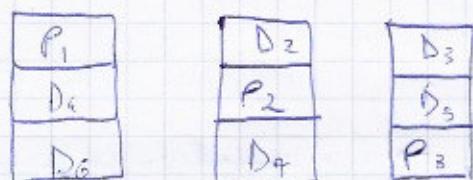
$$\overline{S}_{TOT} = P_{RAN} \cdot \overline{S}_{RAN} + P_{SEQ} \cdot \overline{S}_{SEQ} = 0.2 \cdot 13.4 + 0.8 \cdot 5.73 = 7.25$$

## Array di dischi: RAID

Leggiungo dischi per tollerare i guasti.

Utilizziamo il RAID 5: lavoro su stripe e il disco di parity è distribuito sugli altri dischi.

Ese:



Le scritture di un blocco di un disco, nel RAID 5, equivale a 2 scritture e 2 letture (assumendo di conservare la parità tramite gli xor).  
Questo perché, avendo  $P_1$  lo xor di  $D_2$  e  $D_3$ , se devo scrivere un nuovo  $D_2$  cito  $D_2$  allora:

$$P_1 = D_2 \oplus D_3 \quad \text{devo cioè leggere il vecchio valore } D_2 \text{ e il resto}$$
$$P'_1 = P_1 \oplus D_2 \oplus D'_2 \quad P_1 \text{ e aggiornarlo.}$$

Quindi per scrivere  $D'_2$  prendo  $P_1$  e  $D_2$  (2 letture), faccio lo xor tra  $P_1$  e  $D_2$  e infine lo xor con  $D'_2$ , infine scrivo  $D'_2$  e  $P'_1$  (2 scritture).

Per quanto riguarda la scrittura di 2 blocchi:

leggo  $P_1$   $D_2$  e  $D_3$

$$\text{calcolo } P'_1 = P_1 \oplus D_2 \oplus D_3 \oplus D'_2 \oplus D'_3$$

scrivo  $P'_1$   $D'_2$   $D'_3$

In totale sono 3 letture e 3 scritture. In realtà potrò direttamente calcolare  $P'_1$  come  $D'_2 \oplus D'_3$  in quanto per ogni riga ho solo 2 blocchi e il blocco di parità.

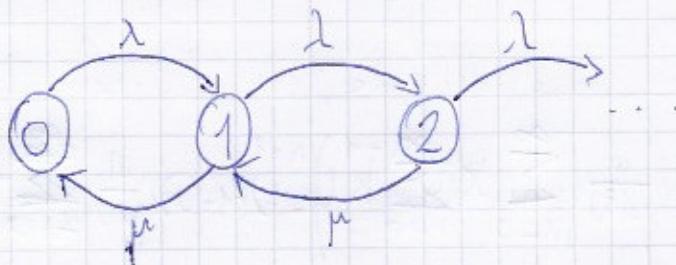
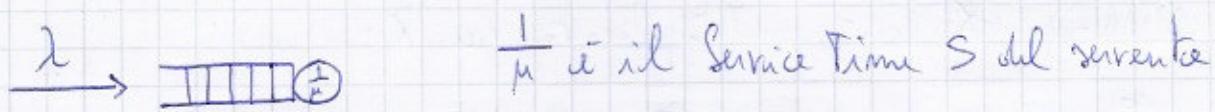
In generale se ho le scritture di  $m$  nuovi blocchi ho  $m+1$  letture e  $m+1$  scritture seguendo il procedimento standard, ma spesso posso ottimizzare come visto prima.

## Modello a code M/M/1

12/06/2018

Dobbiamo introdurre con licenzia. Abbiamo 1 servente e una coda infinita.

Vogliamo modellare la coda con Markov.



Ci interessa la prob. dello stato  $\emptyset$  (nessuno in coda)

$$\text{FLOW OUT} = \text{FLOW IN}$$

$$\begin{cases} \lambda P_0 = \mu P_1 \\ \lambda P_1 + \mu P_2 = \lambda P_0 + \mu P_2 \\ \vdots \\ \lambda P_k + \mu P_{k+1} = \lambda P_{k-1} + \mu P_{k+1} \end{cases}$$

Non basta, dobbiamo anche considerare le somme delle probabilità. Dunque il sistema completo è:

$$\begin{cases} \lambda P_0 = \mu P_1 \\ \lambda P_1 + \mu P_2 = \lambda P_0 + \mu P_2 \\ \vdots \\ \lambda P_k + \mu P_{k+1} = \lambda P_{k-1} + \mu P_{k+1} \\ P_0 + P_1 + \dots + P_k = 1 \end{cases} \rightarrow \begin{cases} P_1 = \frac{\lambda}{\mu} P_0 \\ P_2 = \frac{\lambda}{\mu} P_1 = \left(\frac{\lambda}{\mu}\right)^2 P_0 \\ \vdots \\ P_k = \frac{\lambda}{\mu} P_{k-1} = \left(\frac{\lambda}{\mu}\right)^k P_0 \\ \sum_{k=0}^{\infty} \left(\frac{\lambda}{\mu}\right)^k P_0 = 1 \end{cases}$$

Con l'assunzione che  $\frac{\lambda}{\mu} < 1$

Dalle somme delle probabilità, cioè  $\sum_{k=0}^{\infty} \left(\frac{\lambda}{\mu}\right)^k P_0 = 1$  ponendo fuori  $P_0$  ci si

$$P_0 \sum_{k=0}^{\infty} \left(\frac{\lambda}{\mu}\right)^k = 1 \quad \text{e segue che } P_0 = 1 / \sum_{k=0}^{\infty} \left(\frac{\lambda}{\mu}\right)^k \quad \text{e quindi:}$$

$$P_0 = \left[ \sum_{k=0}^{\infty} \left(\frac{\lambda}{\mu}\right)^k \right]^{-1} = \left[ \frac{1}{1 - \left(\frac{\lambda}{\mu}\right)} \right]^{-1} = 1 - \frac{\lambda}{\mu} \rightarrow \text{Prob. di trovare la macchina libera}$$

Dunque il coeff. di utilizzazione è  $1 - P_0$  cioè:

$$U = 1 - P_0 = 1 - \left(1 - \frac{\lambda}{\mu}\right) = \frac{\lambda}{\mu}$$

Quindi  $P_0 = 1 - U \rightarrow$  se  $U = 0$  vuol dire che il sistema non lavora e quindi  $P_0 = 1$  infatti non c'è nessuno nel sistema.

Quanti utenti in media mi aspetto nel sistema? Valore atteso!

Cioè  $\underbrace{0 \cdot P_0}_{\text{num. utenti}} + 1 P_1 + \dots + K P_K$

$$\sum_{K=0}^{\infty} K \cdot P_K = \sum_{K=0}^{\infty} K U^K P_0 = \frac{P_0}{(1-U)} \sum_{K=0}^{\infty} K U^K$$

Le sommatoria le saro come derrete, con poi  $\infty$  risolvere:

$$(1-U) \sum_{K=0}^{\infty} K U^K = (1-U) \sum_{K=0}^{\infty} K U \cdot U^{K-1} = (1-U) \cdot U \sum_{K=0}^{\infty} K U^{K-1} = (1-U) \cdot U \sum_{K=0}^{\infty} \frac{d}{dU} (U^K) =$$

$$= (1-U) \cdot U \cdot \frac{d}{dU} \left( \frac{1}{1-U} \right) = U(1-U) \frac{1}{(1-U)^2} = \frac{U}{1-U} \quad \begin{array}{l} \text{valore atteso del} \\ \text{num. di persone in} \\ \text{coda} \end{array}$$

$$R = \frac{N}{X} = \lambda \frac{U}{(1-U)} = \frac{\left(\frac{\lambda}{\mu}\right)}{\lambda(1-\frac{\lambda}{\mu})} = \frac{\frac{1}{\mu}}{1 - \frac{1}{\mu}} = \frac{1}{\mu(1-U)} = \frac{S}{1-U}$$

$X = \lambda$  perché cond. di equilibrio

Quanti utenti ci sono in coda?

Quelli totali sul sistema - quelli processati

$$Q = \sum_{K=1}^{\infty} (K-1) P_K = \frac{U^2}{1-U}$$

Le somme partono da  $K=1$  perché al termine quando  $K=0$  non ha effetto, mentre nella sommatoria lo  $(K-1)$  perché deve sempre togliere il fatto che uno è nel sistema mentre è one sarà ancora quelli in coda

Quindi ricapitolando

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

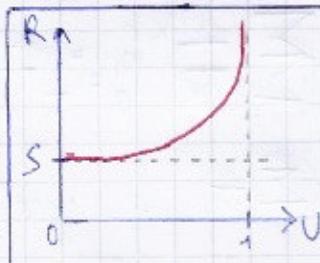
$$N = \frac{U}{1-U}$$

$$Q = \frac{U^2}{1-U}$$

$$R = \frac{1}{\mu(1-U)} = \frac{S}{1-U}$$

$$E[\text{wait}] = \frac{Q}{X} = \frac{Q}{\lambda}$$

L'idea  $\frac{1}{X}$  è in media i secondi per ogni utente, moltiplicando per il numero di utenti in coda faccio il tempo medio attesa minima.



Al crescere del fattore di utilizzazione c'è il response time. All'inizio  $R=S$  perché nel sistema non c'è nessuno, quindi entra e vengo subito senza attesa in coda e quindi ci metto  $R=S=1/\mu$ .

# Network Service Time

26/04/2018

Nel modello OSI seppiamo che quando si va a livello più basso viene aggiunto dell'overhead

TCP: protocollo a livello di trasporto con connessione

Se client invia il SYN, il server risponde con SYN+ACK e il client risponde con ACK. Da qui in poi la connessione è instanciata.

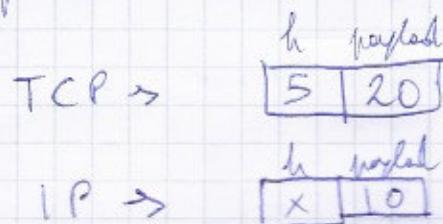
Nello strato TCP e IP i pacchetti hanno una taglia minima.

IP → more orientato alla connessione quindi il servizio è unreliable

## Frammentazione

C'è quando un pacchetto dello strato K+1 non entra nel payload dello strato K

Esempio:



Se la frammentazione viene fatta da TCP succede che vengono inviati a IP con l'header di TCP sul payload di IP e quindi dovrà fare 3 frammenti. I frammenti saranno:

IP Header → TCP HEADER → TCP PAYLOAD

X	S	S
X	S	S
X	S	S
X	S	S

in pratica l'header viene replicato in ogni pacchetto

Se invece la frammentazione viene svolta da IP:

ricorda IP mette l'header del TCP soltanto una volta.

Note: queste cose vengono fatte quando TCP non conosce la Max Transfer Unit (MTU) della local Network

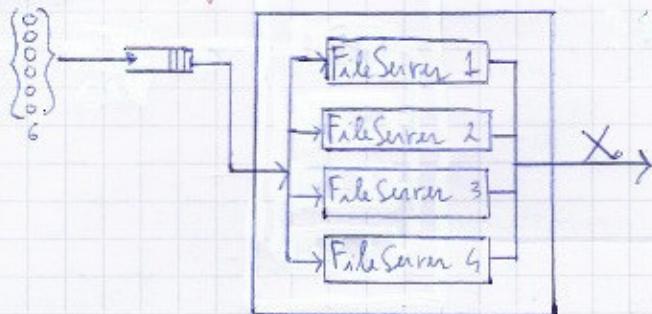
X	H	5
X	I	10
X	5	

$$\text{Service Time} = \frac{\# \text{ byte messaggio}}{\text{bandwidth}}$$

se le bande sono in bit per sec dobbiamo mettere a numero di messaggi in bit e sempre ri moltiplicare per

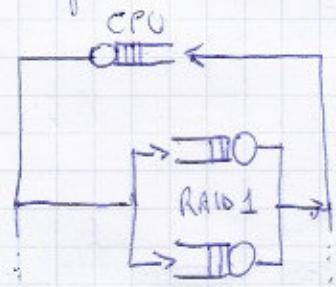
# Exercise Performance

10/05/2018



Siccome ho una rete che non faccio le MVA

Un singolo Server è fatto nel modo seguente:



Ogni Server ha un FileSystem fatto da un RAID 1 = 6 + 6

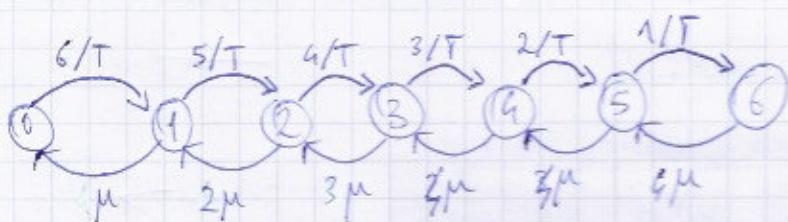
$$MTTF_{CPU} = \frac{1}{1000h} \quad MTTR_{CPU} = \frac{1}{10h} \rightarrow \alpha_{CPU} = \frac{MTTF_{CPU}}{MTTF_{CPU} + MTTR_{CPU}}$$

$$MTTF_{DISK} = \frac{1}{500h} \quad MTTR_{DISK} = \frac{1}{50h} \rightarrow \alpha_{DISK} = \frac{MTTF_{DISK}}{MTTF_{DISK} + MTTR_{DISK}}$$

$N_{ALL}$  (numero di utenti) = 6  $\Rightarrow$  #utenti possibili

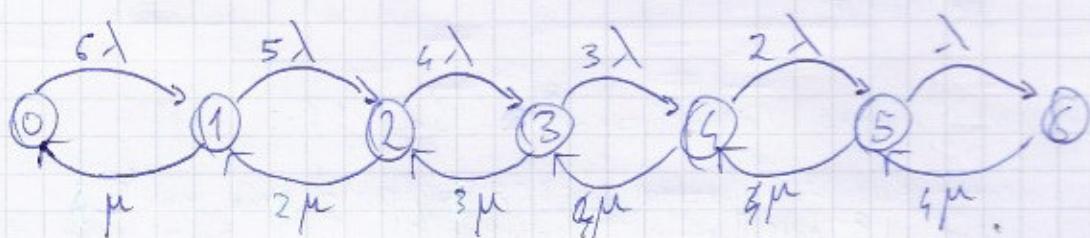
$$T = \frac{1}{\lambda} \rightarrow \text{thinking time}$$

$$\mu = \frac{1}{5s} \rightarrow \text{service rate di un FS} \rightarrow S_{FS} = \frac{1}{\mu} = 5s$$



← Modello Markoviano su tutto il sistema

ormai:



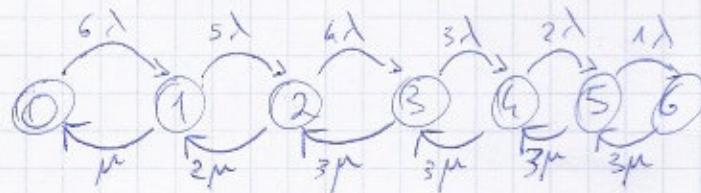
3 valori di T e μ sono i seguenti:

$$T = 10 \rightarrow (\text{quindi } \lambda = 1 \text{ req ogni 10 s per ogni utente})$$

$$\mu = \frac{1}{5s}$$

Dunque NON utilizziamo la Mean Value Analysis

Facciamo il caso in cui un Server è GOASTO:



$$\text{FLOW-IN} = \text{FLOW-OUT}$$

$$\text{stato 0: } 6\lambda P_0 = 1 \cdot \mu P_1$$

$$\text{stato 1: } 5\lambda P_1 + \mu P_1 = 6\lambda P_0 + 2\mu P_2$$

$$\text{stato 2: } 4\lambda P_2 + 2\mu P_2 = 5\lambda P_1 + 3\mu P_3$$

... ...

Dalle 1<sup>o</sup> eq. vede che nelle seconde posso semplificare, dalle 2<sup>o</sup> vede che nelle 3<sup>o</sup> posso semplificare e così via...

Dunque alla fine:

$$\begin{cases} 6\lambda P_0 = 1 \cdot \mu P_1 \\ 5\lambda P_1 = 2\mu P_2 \\ 4\lambda P_2 = 3\mu P_3 \\ 3\lambda P_3 = 3\mu P_4 \\ 2\lambda P_4 = 3\mu P_5 \\ \lambda P_5 = 3\mu P_6 \end{cases} \rightarrow \begin{cases} P_1 = P_0 \left( \frac{6\lambda}{\mu} \right) \\ P_2 = P_0 \left( \frac{6\lambda}{\mu} \right) \cdot \frac{5}{2} \left( \frac{\lambda}{\mu} \right) \\ P_3 = P_0 \left( \frac{6\lambda}{\mu} \right) \cdot \frac{5}{2} \left( \frac{\lambda}{\mu} \right) \cdot \frac{4}{3} \left( \frac{\lambda}{\mu} \right) \\ P_4 = P_0 \left( \frac{6\lambda}{\mu} \right) \cdot \frac{5}{2} \left( \frac{\lambda}{\mu} \right) \cdot \frac{4}{3} \left( \frac{\lambda}{\mu} \right) \cdot \frac{3}{3} \left( \frac{\lambda}{\mu} \right) \\ P_5 = P_0 \left( \frac{6\lambda}{\mu} \right) \cdot \frac{5}{2} \left( \frac{\lambda}{\mu} \right) \cdot \frac{4}{3} \left( \frac{\lambda}{\mu} \right) \cdot \frac{3}{3} \left( \frac{\lambda}{\mu} \right) \cdot \frac{2}{3} \left( \frac{\lambda}{\mu} \right) \\ P_6 = P_0 \left( \frac{6\lambda}{\mu} \right) \cdot \frac{5}{2} \left( \frac{\lambda}{\mu} \right) \cdot \frac{4}{3} \left( \frac{\lambda}{\mu} \right) \cdot \frac{3}{3} \left( \frac{\lambda}{\mu} \right) \cdot \frac{2}{3} \left( \frac{\lambda}{\mu} \right) \cdot \frac{1}{3} \left( \frac{\lambda}{\mu} \right) \end{cases}$$

$$\begin{cases} P_1 = P_0 \left( \frac{\lambda}{\mu} \right) \cdot 6 \\ P_2 = P_0 \left( \frac{\lambda}{\mu} \right)^2 \cdot 6 \cdot \frac{5}{2} \\ P_3 = P_0 \left( \frac{\lambda}{\mu} \right)^3 \cdot 6 \cdot \frac{5}{2} \cdot \frac{4}{3} \\ P_4 = P_0 \left( \frac{\lambda}{\mu} \right)^4 \cdot 6 \cdot \frac{5}{2} \cdot \frac{4}{3} \cdot \frac{3}{3} \\ P_5 = P_0 \left( \frac{\lambda}{\mu} \right)^5 \cdot 6 \cdot \frac{5}{2} \cdot \frac{4}{3} \cdot \frac{3}{3} \cdot \frac{2}{3} \\ P_6 = P_0 \left( \frac{\lambda}{\mu} \right)^6 \cdot 6 \cdot \frac{5}{2} \cdot \frac{4}{3} \cdot \frac{3}{3} \cdot \frac{2}{3} \cdot \frac{1}{3} \end{cases}$$

Possiamo notare un pattern ripetitivo.

Le linee sono divise dal punto di sopra in cui i denominatori avranno del peso netto in cui rimangono costanti dopo l'ultima riga nel peso di sopra.

Le eq. dei due casi le possiamo generalizzare:

- Punto di sopra:

$$P_i = P_0 \left(\frac{\lambda}{\mu}\right)^i \cdot \binom{6}{i}$$

- Punto di sotto:

$$P_i = P_0 \left(\frac{\lambda}{\mu}\right)^i \cdot \frac{6!}{(6-i)! n! k^{i-k}}$$

Quindi riassumendo:

$$P_i = \begin{cases} P_0 \left(\frac{\lambda}{\mu}\right)^i \binom{6}{i} & i \leq k \\ P_0 \left(\frac{\lambda}{\mu}\right)^i \frac{6!}{(6-i)! n! n^{i-k}} & i > k \end{cases}$$

dove  $k$  è il numero di server funzionanti

In realtà avremo una equazione:

$$\sum_{i=0}^6 P_i = 1$$

Analizziamo le performance: Response Time e Throughput

Sappiamo che:

$$R(K) = \frac{N(K)}{X(K)}$$

dove  $N(K)$  è il num. di utenti con  $K$  server  
e  $X(K)$  è il throughput con  $K$  server

$$N(K) = \sum_{i=0}^6 i \cdot P_i(K)$$

$$X(K) = \begin{cases} \sum_{i=0}^6 P_i(K) \cdot i \cdot \mu & i \leq k \\ \sum_{i=0}^6 P_i(K) \cdot K \cdot \mu & i > k \end{cases}$$

→ Quando  $K$  server attivo, il throughput medio è quello ponderato sulla prob. degli utenti attivi, quindi prob. di avere  $i$  utenti nel sistema moltiplicata per il throughput con  $i$  utenti che sono  $i \cdot \mu$  se ho meno di  $K$  server, altrimenti  $K \mu$  che è il massimo throughput. Il throughput totale sarà poi quello ponderato sulle prob. di quanti server sono attivi.

Il throughput medio TOTALE quindi sarà:

$$\bar{X} = q_1 \boxed{X(1)} + q_2 \boxed{X(2)} + \dots + q_n X_n$$

↓  
Prob. 1 server attivo

Punto da 1 perché con 0 utenti ho  $X(0) = 0$

Le  $q_i$  possiamo calcolarle. Date AFS l'arrivals rate di ogni server ha:

$$q_i = A_{FS}$$

$$q_3 = 6 A_{FS}^3 (1-A_{FS})$$

$$q_2 = 6 A_{FS}^2 (1-A_{FS})^2$$

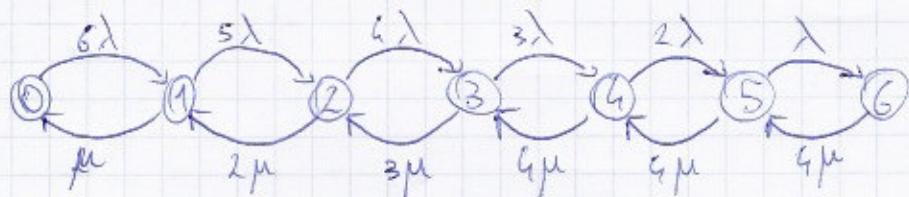
$$q_1 = 6 A_{FS} (1-A_{FS})^3$$

$q_0 = (1-A_{FS})^4 \rightarrow$  non c'è nulle zone per il Marginal Netto  $X(0) = \emptyset$

Rifine  $\bar{R} = \frac{N}{X}$

Finiamo l'esercizio delle scorse lezioni

17/05/2018



Modello Markoviano complementare

Stavolta assumiamo 4 Server funzionanti

Stiamo facendo le PERFORMABILITY  $\Rightarrow$  "mettere insieme AVAILABILITY e PERFORMANCE"

$$X(k) = \sum_{s=1}^6 P_s(k) X_s(k) \quad \Rightarrow \text{throughput medio}$$

$$N(k) = \sum_{s=1}^6 P_s(k) \cdot s \quad \Rightarrow \text{numero utenti medio}$$

Portiamo a 1 perché con 0 il valore è 0 quindi non ha importanza in una somma

$$R(k) = \frac{N(k)}{X(k)} \quad \Rightarrow \text{Response Time medio}$$

Per calcolare  $R(k)$  dovrà calcolare l'AVAILABILITY

AVAILABILITY di un solo server:

$$A_{FS} = A_{CPU} \cdot A_{RAID}$$

per calcolare usiamo il Combinatorio e non Markov perché proviamo usare MTTR e MTTF per l'Availability di ogni singola componente

$$A_{CPU} = 0,99$$

$$A_{DISK} = 0,9$$

$$A_{COPPIA} = 1 - (1 - A_{DISK})^2 = 0,99$$

$$A_{RAID} = (A_{COPPIA})^4 = 0,96$$

$$A_{SERVER} = A_{CPU} A_{RAID} = 0,95$$

Availability Totale

$$A = 1 - P(\text{nessun server funziona}) =$$

$$= 1 - (1 - A_{SERVER})^4 = 0,99$$

$$\bar{X} = \sum_{k=1}^4 q_k X(k) \cdot k$$

$$\bar{N} = \sum_{k=1}^4 q_k R(k)$$

$$q_4 = A_{FS}^4 \quad q_3 = 4A_{FS}(1-A_{FS}) \quad q_2 = 6A_{FS}^2(1-A_{FS})^2$$

$$q_1 = 4A_{FS}(1-A_{FS})^3$$

Il Response Time finale  $\Rightarrow$  lo calcolo come

$$\bar{R} = \frac{\bar{N}}{\bar{X}}$$

oppure calcolo gli  $R(k) = \frac{N(k)}{X(k)}$  per i vari  $k$  e infine

$$\bar{R} = \sum_{k=1}^4 q_k R(k)$$

Per quale lo s'intende un thought sulla questione del thought medio. Valendo ottenere sempre una matrice che non considera il caso q=0 allora:

$$\overline{X} = \frac{1}{1-q_0} \sum_{K=1}^q q_K X(K) \cdot K$$

$q_2 + q_1 + q_0 = 1$  ma non voglio i prob. normalizzati  
le prob.  $q_2$  e  $q_1$  b.c.

$$q_2 + q_1 = 1 - q_0 \Rightarrow \frac{q_2 + q_1}{1 - q_0} = \frac{1 - q_0}{1 - q_0} \Rightarrow \frac{q_2}{1 - q_0} + \frac{q_1}{1 - q_0} = 1$$

$q_2'$  e  $q_1'$  sono le nuove prob. normalizzate

$$\overline{R} = \sum_{K=1}^q \frac{q_K}{1-q_0} R(K)$$

## Esercizio con la Mean Value Analysis

5 server

$\lambda = 25$  req/sec

Dcpu = 20 ms

Ddisk = 50 ms

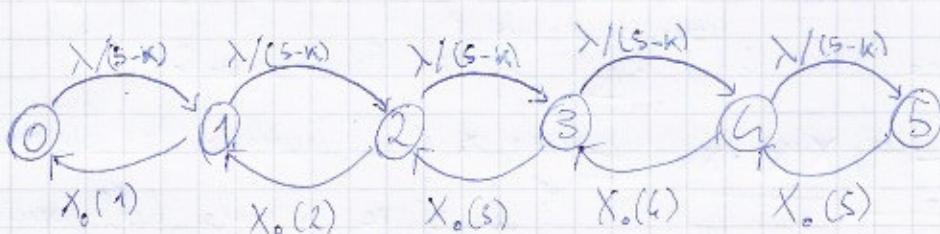
Max concurrent reqs per server = 5

MTTF = 3600. 1000 s

MTTR = 3600. 10 s

Mean Value Analysis perché abbiamo un massimo numero di utenti per ogni server (una coda finita per server). Se ci fosse una coda infinita (load balancer) allora la MVA non va fatta

K = # server NON funzionanti



Modello Markoviano  
del singolo Server

Dobbiamo fare il calcolo del Mean Value Analysis:

$$\left\{ \begin{array}{l} P_i^{'}(m) = D_x [1 + m_s(m-1)] \\ X_0(m) = \frac{m}{P_0(m)} \\ M_i(m) = X_0(m) P_i^{'}(m) \end{array} \right. \quad \left. \begin{array}{l} \\ \\ \end{array} \right\} \begin{array}{l} \\ \\ \end{array}$$

M=1

$$P_{0,0}(1) = D_{cpu} = 20 \text{ ms} \quad P_{0,disk}(1) = D_{disk} = 50 \text{ ms}$$

$$P_0^{'}(1) = P_{0,cpu}(1) + P_{0,disk}(1) \approx 70 \text{ ms}$$

$$X_o(1) = \frac{1}{R_o(1)} = 1/70 \text{ ms}$$

$$M_{CPU}(1) = X_o \cdot R'_{CPU}(1) = \frac{20 \text{ ms}}{70 \text{ ms}}$$

$$M_{DISK}(1) = X_o \cdot R'_{DISK}(1) = \frac{50 \text{ ms}}{70 \text{ ms}}$$

• M=2

$$R'_{CPU}(2) = D_{CPU} [1 + M_{CPU}(1)]$$

$$R'_{DISK}(2) = D_{DISK} [1 + M_{DISK}(1)]$$

$$X_o(2) = \frac{2}{R_o(2)}$$

$$M_{CPU}(2) = X_o(2) R'_{CPU}(2)$$

$$M_{DISK}(2) = X_o(2) R'_{DISK}(2)$$

• M=3

Una volta ottenuti  $X_o(1), X_o(2), \dots, X_o(5)$  poniamo calcolare  $p_0, \dots, p_5$

$$\begin{cases} \text{Flow IN} = \text{Flow OUT} \\ \sum_{i=0}^5 p_i = 1 \end{cases}$$

→

$$X(K) = \sum_{i=1}^5 p_i X_o(i)$$

$$N(K) = \sum_{i=1}^5 i \cdot p_i$$

$$R(K) = \frac{N(K)}{X(K)}$$

↑ SINGOLO SERVER

Queste sono le performance di un Single Server quando ne sono funzionanti 5. Ripeto le procedure con 4, 3, 2, 1 Server Funzionanti.

$$\text{Sopra } \bar{X} = \sum_{h=1}^5 \frac{1}{1-q_h} q_h X(h) \cdot h$$

Prob. h Server      Throughput  
 Sistema con h server

dove  $h$  è il numero di Server funz.

È la stessa formula dell'esercizio precedente

$$\bar{N} = \sum_{h=1}^5 h \cdot N(h) \cdot \frac{q_h}{1-q_h}$$

$$\bar{R} = \frac{\bar{N}}{\bar{X}}$$

Poi il numero di richieste rifiutate al secondo è  $25 - \bar{X}$  e lo percentuale è

$$\frac{25 - \bar{X}}{25} \cdot 100$$

Le  $q_i$  vengono calcolate tramite le MTTF e MTTR dei Server

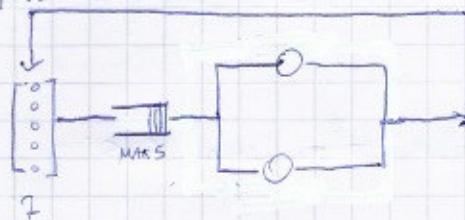
2 Server in parallelo

$$MTTF = 24 \text{ min}$$

max 5 utenti

$$MTTR = 4 \text{ h}$$

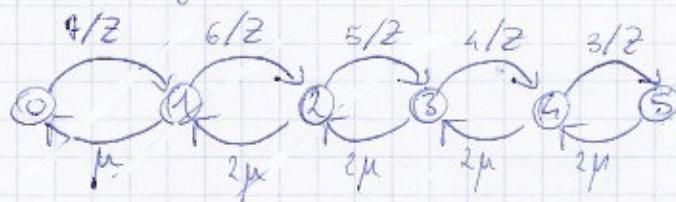
# utenti pass = 4



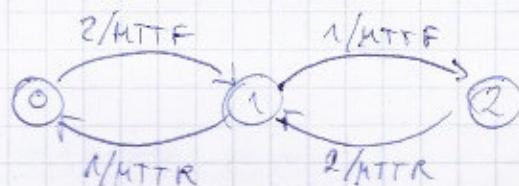
Z - blocking time = 10 h

$$\mu = 6 \text{ h}$$

Il sistema ha il seguente diagramma Markoviano:

 $\bar{x}, \bar{x}?$ 

Per quanto riguarda l'Availability:



$P_i = \text{Prob. di } i \text{ Server guasti}$

Facciamo  $\text{Flow}_i - \text{N} = \text{Flow}_i^{\text{out}}$  per trovare i  $P_0, P_1, \dots$ 

Poi troviamo  $\bar{N} = P_0 N_0 + P_1 N_1 + \dots$  e lo stesso per  $\bar{X}$  e infine  $\bar{Q} = \frac{\bar{N}}{\bar{X}}$  per la律得律得  
Non è MVA perché ho una sola coda. Se mi fosse stato detto "bilanciatore di carico"  
allora avrei avuto più code e quindi MVA

Esercizio

3 CPU + 1 ROTER

RAID 5

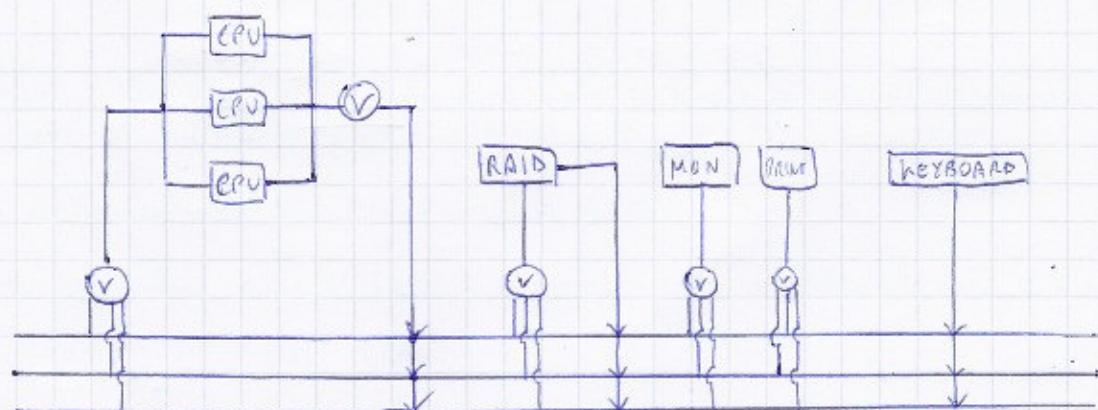
8 DISCHI

1 KEYBOARD

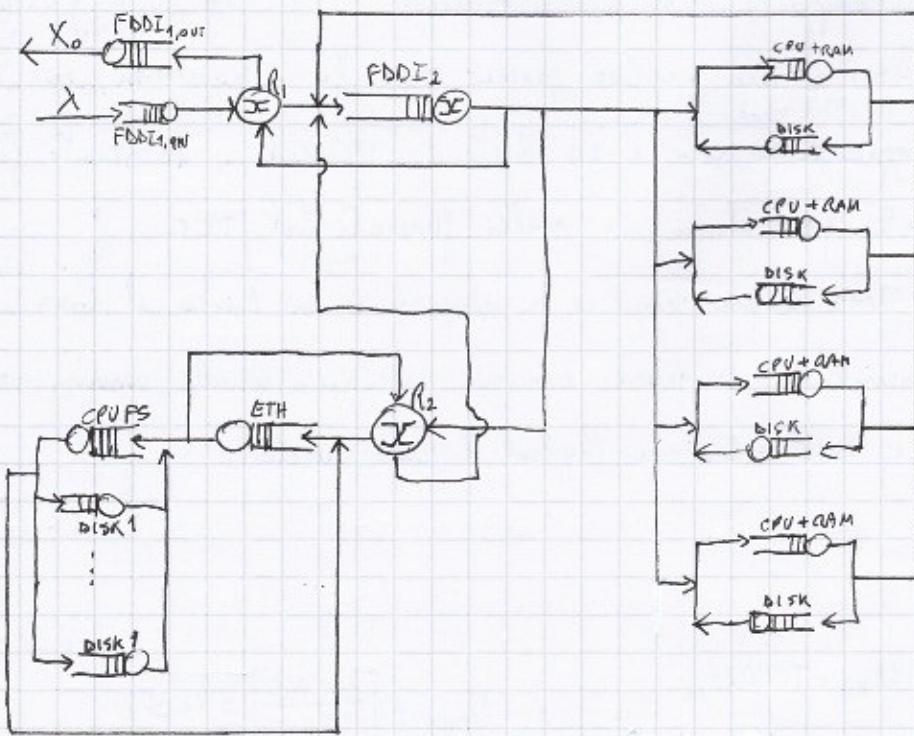
1 PAINTER

1 MONITOR

3 BUS = 1 ROTER



Di per ogni componente.  
Analizziamo la SAFETY.  
Abbriamo un ripostiglio separato per ogni tipo di componente, che può riportare max 2 componenti insieme.  
Facciamo la Safety sul sottoinsieme SOS:



100 file in totale: ogni richiesta chiude uno specifico file  $\Rightarrow$  100 tipi di richieste

FDDI<sub>1</sub>: Full Duplex

FDDI<sub>2</sub>: Half Duplex

RAM: contiene i TOP5 files

DISCO: contiene i successivi TOP10

Quindi:

RAM  $\Rightarrow$  F<sub>1</sub>, F<sub>2</sub>, F<sub>3</sub>, F<sub>4</sub>, F<sub>5</sub>

DISCO  $\Rightarrow$  F<sub>6</sub>, F<sub>7</sub>, F<sub>8</sub>, F<sub>9</sub>, F<sub>10</sub>, F<sub>11</sub>, F<sub>12</sub>, F<sub>13</sub>, F<sub>14</sub>, F<sub>15</sub>

FILESYSTEM  $\Rightarrow$  F<sub>16</sub>, ..., F<sub>100</sub>

RAIDS (7 DISCHI) nel FS

HTTP<sub>RA</sub> = 400 Byte

IF<sub>i,1</sub> = 240 KByte

FDDI<sub>1</sub> Speed = 1 Gbps      FDDI<sub>2</sub> Speed = 1,44 Gbps

$$\lambda = 100 \text{ req/s}$$

$$\text{ETH Speed} = 0,5 \text{ Gbps}$$

Ogni file ha una prob. di essere richiesto parie  $P_i = \frac{K}{i}$  (il file F<sub>1</sub> ha popolarità K, il file F<sub>2</sub> ha popolarità K/2 ecc...). Ormai anche  $\sum_i P_i = 1$

La RAM di ogni Server può ospitare 5 file, mentre il disco 10 file. Il fronte di una richiesta vede prima se il file è nella RAM, altrimenti vede nel DISCO, altrimenti deve fare un forward della richiesta nel Filesystem.

Note: se trovo il file in RAM o sul DISCO ritorno nello CPU come suggerisce il diagramma.

$D_{CPU}^{HIT} = 5 \text{ ms}$   $\rightarrow$  Comprendo che il caso di file in RAM che il caso del DISCO

$D_{DISK} = 10 \text{ ms}$   $\rightarrow$  Per fare il rettangolo di 20 KByte. Quindi  $D_{DISK} = 10 \text{ ms}$ .  $\frac{240 \text{ KByte}}{20 \text{ KByte}} = 12 \text{ ms}$

$D_{CPU}^{MISS} = 30 \text{ ms}$   $\rightarrow$  In tal caso dovrò andare a chiedere al Filesystem.

Verso

$$D_{DISK} = 12/7 \times 10 \text{ ms} = 13 \text{ ms}$$

In media, nel File System, per prendere un file devo fare 12 volte ai DISCHI. Dunque un unico disco del RAID serve in media  $12/7 = 1,7$  richieste.

Questo perché ogni file è distribuito sui 7 dischi del RAID.

L'esercizio lo possiamo modellare come MULTICASS in cui ho sostanzialmente 3 domi:

- Richiesta di un file tra F<sub>1</sub> ... F<sub>5</sub> (RAM)
- Richiesta di un file tra F<sub>6</sub> ... F<sub>15</sub> (DISK del Server)
- Richiesta di un file tra F<sub>16</sub> ... F<sub>100</sub> (RAIDS)

Pur quanto riguarda la Rete, dobbiamo analizzare lo specchettamento. Dobbiamo fare l'operazione sulla conoscenza del TCP nei confronti della rete sottostante.

Se assumiamo che TCP conosca l'MTU è più semplice perché TCP fa i pacchetti, poi IP li lascia così come sono aggiungendo solo i 20 Byte di IPOverhead e lo strato finisce aggiungere il suo Overhead. I pacchetti però riempiono quelli generati dal TCP.

Questa soluzione crea più Overhead ma è più semplice e risparmia ai Router il lavoro di specchettamento, dunque è più performante. In questo esercizio facciamo questa assunzione.  
L'MTU è 1500 Byte  $\rightarrow$  Il TCP vedrà un Payload di 1460 perché:

TCP	18	20	20	1460
IP				

$$\text{TCP Overhead}_{\text{RA}} = [1 + \frac{1}{3}] \cdot 20 \quad \text{IP Overhead}_{\text{RA}} = \text{TCP Overhead}_{\text{RA}}$$

$$N_{\text{PACK}} = \left\lceil \frac{260000}{1460} \right\rceil = 165$$

$$\text{TCP Overhead}_{\text{RES}} = [3 + \left\lceil \frac{260000}{1460} \right\rceil] \cdot 20 \quad \text{IP Overhead}_{\text{RES}} = \text{TCP Overhead}_{\text{RES}}$$

$$\text{FDDI Overhead}_{\text{RA}} = [1 + 3] \cdot 28 \text{ Byte}$$

$$\text{FDDI Overhead}_{\text{RES}} = [3 + 165] \cdot 28 \text{ Byte}$$

$$\text{ETH Overhead} = [1 + 6 + 165] \cdot 18 \text{ Byte}$$

Nelle RES il 3 sta per la chiusura delle connessioni da inglobiamo nella risposta.

$$\text{Dunque il numero di vanta in un Router è } V_R = 165 + \left\lceil \frac{7}{7} \right\rceil = 172$$

La FDDI<sub>1,IN</sub> vede solo le richieste mentre la FDDI<sub>1,OUT</sub> vede solo le risposte.

$$D_{\text{FDDI},\text{IN}} = \frac{(\text{RA} + \text{TCP Overhead}_{\text{RA}} + \text{IP Overhead}_{\text{RA}} + \text{FDDI Overhead}_{\text{RA}}) \times 8}{1 \text{ Gbps}} = \frac{(600B + 20B + 80B + 112B) \times 8}{1 \text{ Gbps}} \approx 5,36 \mu\text{s}$$

$$D_{\text{FDDI},\text{OUT}} = \frac{(\text{FI} + \text{TCP Overhead}_{\text{RES}} + \text{IP Overhead}_{\text{RES}} + \text{FDDI Overhead}_{\text{RES}}) \times 8}{1 \text{ Gbps}} = \frac{(260000 + 168 \cdot 20 + 168 \cdot 20 + 168 \cdot 28)B \times 8}{1 \text{ Gbps}} \approx 2 \text{ ms}$$

Quello che vede la FDDI<sub>2</sub> è la somma di ciò che vede FDDI<sub>1,IN</sub> e FDDI<sub>1,OUT</sub> e in aggiunta vede le richieste delle CPU del Server in caso di miss diretta al File System. Tutto ciò va ponderato.

$$D_{\text{FDDI},\text{HIT}} = D_{\text{FDDI},\text{IN}} + D_{\text{FDDI},\text{OUT}}$$

$$D_{\text{FDDI},\text{MISS}} = 2 D_{\text{FDDI},\text{IN}} + D_{\text{FDDI},\text{OUT}}$$

$$D_{\text{FDDI},\text{2}} = D_{\text{FDDI},\text{HIT}} \times P_{\text{HIT}} + D_{\text{FDDI},\text{MISS}} \times P_{\text{MISS}}$$

In caso di miss vede le richieste che mi arriva dal client  
quelle che la CPU fa al FS perché ha un miss e poi vede il file che arriva dal FS.

Per quanto riguarda l'ETH del FileSystem essa vede la richiesta fatta dalla CPU del Server, il file di intorno, e in aggiunta vede gli overhead relativi:

$$D_{ETH} = \frac{(240000 \text{ Byte} + 400 \text{ Byte} + \text{TCP Overhead} + \text{TCP Overhead} + \text{IP Overhead} + \text{IP Overhead} + \text{ETH Overhead}) \times 8}{0,5 \text{ Gbps}} = 6 \text{ ms}$$

Una volta trovati tutti i Service Demand  $D_i$  andiamo a lavorare i vari fattori di utilizzazione.

$$U_{CPU,i} = \frac{\lambda}{q} P_i D_{CPU,i} \quad i=1,2,3$$

→ Ricordiamo che abbiamo 3 classi, chi richiede i file  $P_1, P_2, P_3$ , chi richiede i file  $P_4, \dots, P_8$  e chi richiede gli altri.

Lion l'arrivo rate di una classe  $\lambda P_i$  ma ho q CPU quindi  $\lambda/q \times P_i$ .

$$U_{CPU}^{TOT} = \sum_{i=1}^3 U_{CPU,i}$$

$$R_{CPU,i} = \frac{D_{CPU,i}}{1 - U_{CPU}^{TOT}}$$

→ Ricorda che con il multiclass il denominatore del Response Time ci va il fattore di utilizzazione totale delle classi.

$$U_{DISK} = \frac{\lambda}{q} P_2 D_{DISK} \quad R_{DISK} = \frac{D_{DISK}}{1 - U_{DISK}}$$

Sempre per individuare il bottleneck devo andare a guardare i fattori di utilizzazione dei singoli componenti oppure i Response Time.

Se andrei a guardare il Service Demand farei un errore perché: prendiamo in considerazione un componente con un Service Demand alto rispetto agli altri, se puo' tale componente viene eseguita con una bassa frequenza allora non e piu' poi cosi' rilevante.

In genere guardo il fattore di utilizzazione del singolo componente per le singole classi