

Project 5 – Multiprogrammed server

A multiprogrammed server provides service to N concurrent clients that request the server to perform transactions. Transactions may involve local CPU processing, access to the local disk, and remote queries to a distant web server. More specifically, the following steps are performed:

1. A new transaction always requires some processing time as a first step;
2. Then:
 - a. with probability p_1 the transaction is terminated;
 - b. with probability p_2 an access to the local disk is required, and then a new CPU processing is required;
 - c. with probability $1-p_1-p_2$ a remote query is required, and then a new CPU processing is required.
3. A reply is sent to the client that originated the request.
4. A user that receives a reply immediately issues another request.

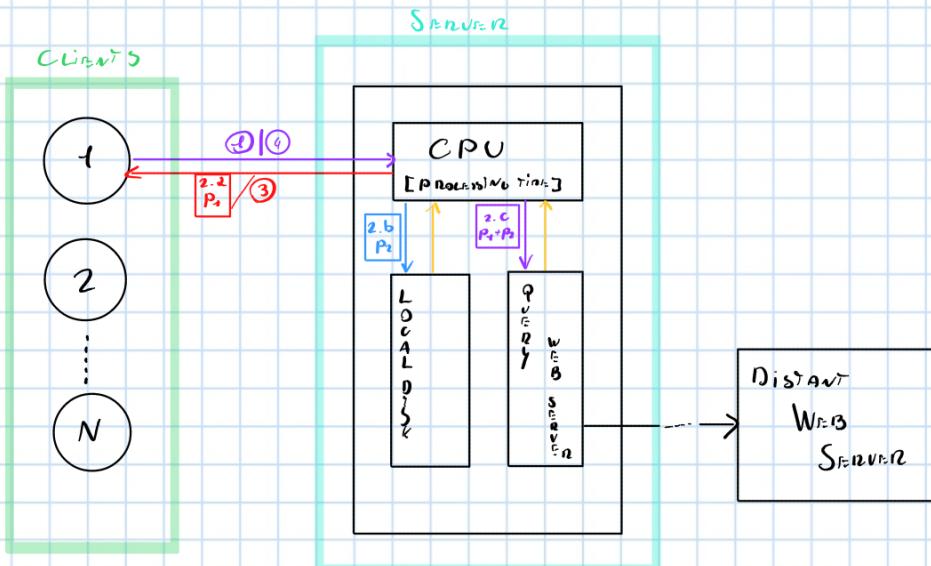
The local CPU, the local disk and the remote web server handle one request at a time in a FIFO order. Local processing, local disk access and remote query service times are exponential IID RVs, and they are different from one iteration to another, even for the same transaction.

Evaluate at least the throughput of the system (i.e., the number of completed transactions per unit of time) under a varying level of multiprogramming. Simulate at least one scenario where the service demands at the three service centers have a considerably different mean (e.g., one order of magnitude).

In all cases, it is up to the team to calibrate the scenarios so that meaningful results are obtained.

Project deliverables:

- a) Documentation (according to the standards set during the lectures and up to 10 pages)
- b) Simulator code
- c) Presentation (up to 10 slides)



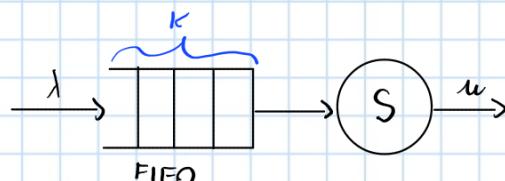
Clieni

Il client richiede immediatamente, ogni volta che è possibile farlo una transazione. Quasi non esiste la distinzione tra servizi.

Modellazione CPU, Local Disk e Query Server

- ESEGUONO UNA RICHIESTA PER VOLTA
- LA CODA INTRUSSORIA HA DIMENSIONE K E POLITICA FIFO
- TUTTI I 3 TIPI DI SERVIZIO SONO VARIABILI NORDI 110 RISPOSTAZIONI

\hookrightarrow Tutte diverse fra loro, anche per la stessa transazione



$$S = \{CPU, Local\ Disk, Query\}$$

Parametri

- K_p, K_d : GRANDEZZA DELLA CODA
- N : NUMERO DI CLIENT
- Velocità (parametri RV esponenziale) CPU
- // // LECTURE Hard-Disk
- Tempo // di risposta Query
- λ : DISTRIBUZIONE DEGLI ARRIVI DEI CLIENT [STUDIO PER OTTIMIZZARE K]

Workload

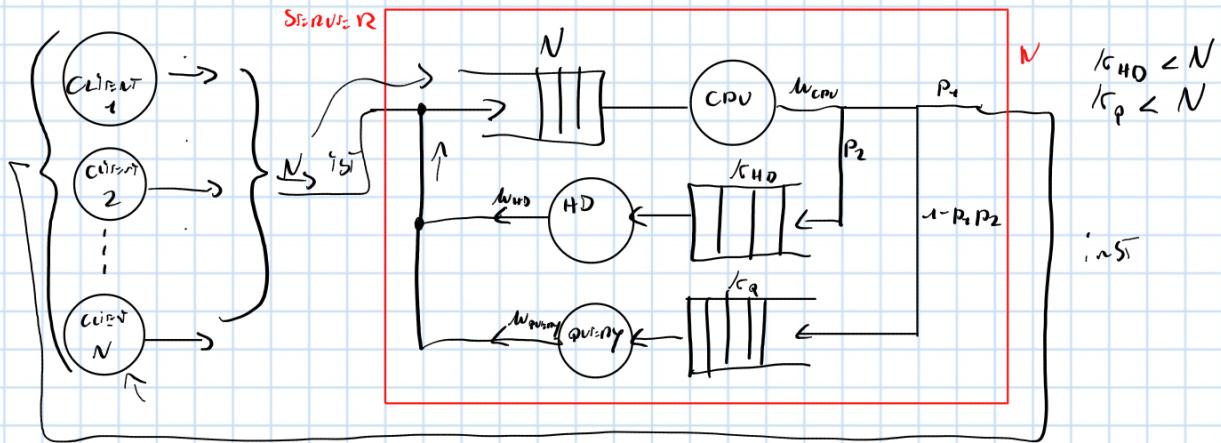
- Packets Intervall Time
- Service Demands: time that they keep the server busy

OBIETTIVI

- THROUGHPUT: TRANSAZIONI CONSUMATE PER UNITÀ DI TEMPO
- BOTTLE NECK: IDENTIFICARE LA PRESSIONE
- SCALABILITÀ: COSA È POSSIBILE MIGRARE (AGGIORNARE PARAMETRI) PER AVERE PRESTAZIONI MINIMI (BEST PRACTICE CHOICE)
- SATURAZIONE MAX: IL NUMERO MASSIMO DI CLIENT CHE NON PUÒ SERVIRE SENZA DIVIDERE RESSORSI CON LE RISORSE

Dubbi:

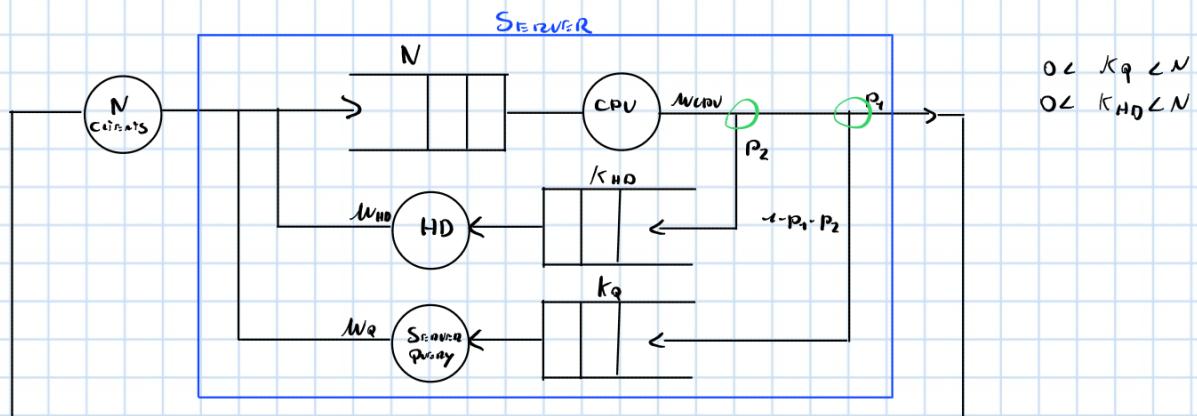
- È NECESSARIO CONSIDERARE PIÙ SERVIZI?
- No, solo uno.
- IL CONCETTO DI RISERVAMENTO + UN SERVIZIO MULTIPROCESSATO, ATTENDONO ALLA CONDIVISIONE, TUTTAVIA SI SPECIFICA CHE LE OPERAZIONI ESISTENTI SONO FATTE CON PRINCIPIO FIFO. IL SERVIZIO DI QUALE CONDIVISIONE O SEQUENZIALE?
- La CONNESSIONE È TCP o UDP? Cambia il tempo di risposta e la possibilità gestione dei pacchetti scambiati.
- Il client richiede subito una nuova transazione una volta finita, da la distribuzione delle richieste la ottimo poi?
- Si, con λ
- Il servizio Web quando tempo di connessione ha?
- Il client avvia una nuova transazione immediatamente?
- È possibile usare la libreria Oracle per sincronizzare i protocolli TCP e UDP.



$$k_{HD} < N$$

$$k_q < N$$

$\vdots \quad \vdots$



$$0 < k_q < N$$

$$0 < k_{HD} < N$$