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.

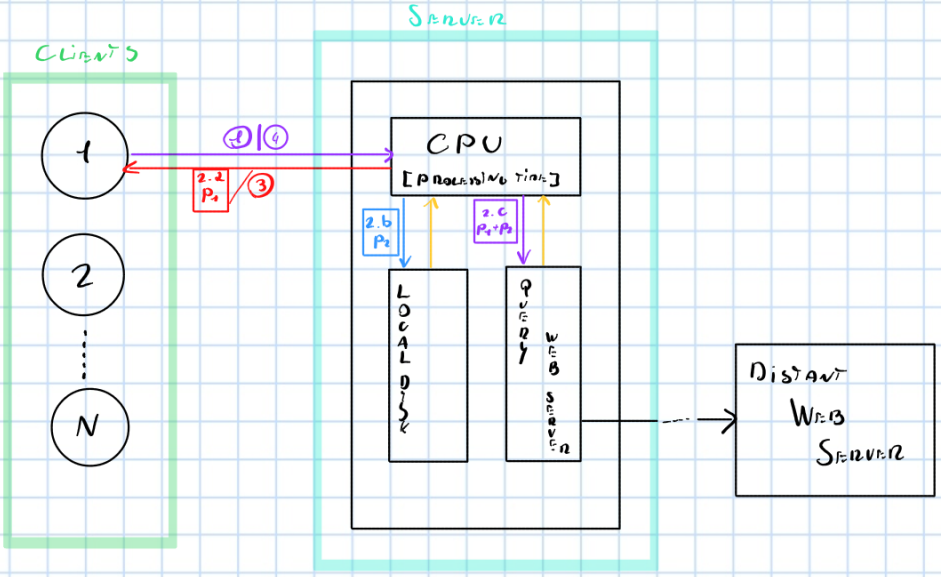
The local CPU, the local disk and the remote web server handle one request at a time in a FIFO order. Local processing, local disk access and remote query service times are exponential IID RVs, and they are different from one iteration to another, even for the same transaction.

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)

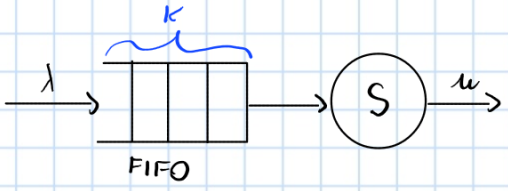


Clienti

Il client richiede immediatamente, qual volta che è possibile farlo una transazione. Quindi non aspetta mai di ricevere il server.

Modellazione CPU, Local Disk e Query Server

- Eseguono una richiesta per volta
- La coda implementata ha dimensione K e politica FIFO
- Tutti e 3 i tempi di servizio sono variabili random IID esponenziali
  - Tutte diverse tra loro, anche per la stessa transazione



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

Parametri

- K: grandezza della coda
- N: numero di client
- Velocità (parametri RV esponenziale) CPU
- // // Lettura Hard-Disk
- Tempo // di risposta Query
- λ: distribuzione degli arrivi dai client

Obbiettivi

- Throughput: transazioni completate per unità di tempo
- Bottle neck: verificare la presenza
- Scalabilità: cosa è meglio potenziare (aumentare parametri) per avere prestazioni migliori (Best Parameter Choice)
- Saturazione Max: il numero massimo di client che può essere usato senza diminuire esponenzialmente le prestazioni

Dubbi:

- È necessario considerare più server?
  - No, solo uno.
- Il codice fa riferimento a un server multithreading, alludendo alla concorrenza, tuttavia si specifica che le operazioni possono essere fatte con priorità FIFO. Il server è quindi concorrente o sequenziale?
- La connessione è TCP o UDP? Cambia il tempo di risposta e la possibile gestione dei pacchetti scartati.
- Il client richiede subito una nuova transazione una volta finita, o la distribuzione delle richieste la gestiamo noi?
  - Sì, con λ
- Il server Web quanto tempo di elaborazione ha?
- Il client deve una nuova transazione immediatamente?
- È possibile usare la libreria Omnet per simulare i protocolli TCP e UDP.