Performance Modeling of Computer Systems and Networks

Prof. Vittoria de Nitto Personè

Introduction to modeling

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

1

Queueing theory

is an area of mathematics, involving stochastic analysis, which allows one to predict the performance of a computer system and to improve performance

Idea: to analytically model the computer system as consisting of *resources* (like CPU, bandwidth, energy, VM, disk, etc.) and jobs which require these resources. Contention occurs when several jobs simultaneously require a resource, which means some jobs must wait.

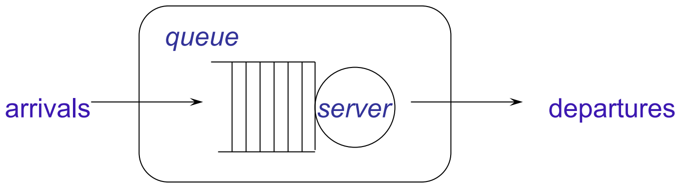
Queueing theory allows you to predict what those “waits” will be and how to reduce it

So improving system performance!

2

conceptual model

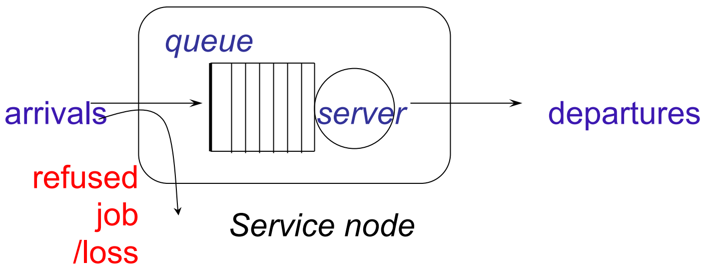
Single server queue



Service node

- *def. 1* a *single server service node* consists of a *server* plus its *queue*

conceptual model



Service node

Assumption:

for the finite *capacity case*,
only the accepted jobs are considered

BUT
the *loss probability* could be of interest!

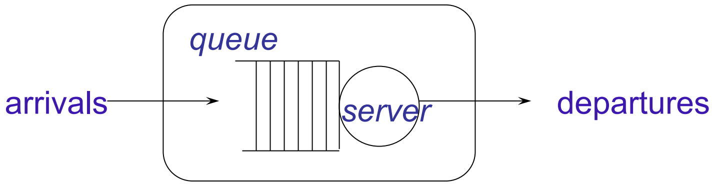
def. 2 *queue discipline (scheduling / service order)*:
the algorithm used when a job is selected
from the queue to enter service

FIFO – first in, first out
LIFO – last in, first out
random – serve in random order
Priority – tipically shortest job first (SJF)
PS – processor sharing

- FIFO (/ FCFS):
  - The order of arrival and departure are the same
  - A job cannot start service if the "previous" job has not left the node; this observation can be used to simplify the simulation
  - Unless otherwise specified, assume FIFO with infinite queue capacity
- service is *non-preemptive*
  - Once initiated, service of a job will continue until completion
- service is *conservative*
  - server will never remain idle if there is one or more jobs in the service node

Discrete-event simulation
Trace-driven simulation

Single server queue



Consider n=10 job and given arrival and service times:

Arrival times:

15 47 71 111 123 152 166 226 310 320

Service times:

43 36 34 30 38 40 31 29 36 30

Prof. Vittoria de Nitto Personè 7

7

Service times:

										mean	variance	resp time
43	36	34	30	38	40	31	29	36	30	34,7	20,21	26,70
63	16	54	10	18	60	51	9	56	10	34,7	504,21	32,70
9	10	10	16	18	51	54	56	60	63	=	=	12,80
63	60	56	54	51	18	16	10	10	9	=	=	77,70

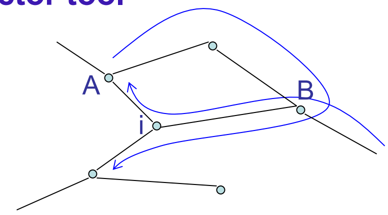
Prof. Vittoria de Nitto Personè 8

8

Modelling power: Predictor tool

consider that we're given

– a network
– some particular packet routes
– the packet arrival rates
– the transmission times
– wire lengths



by queueing theory

- the mean time packets spend waiting at a particular router i,
- the distribution on the queue buildup at router i
- the mean overall time to get from point A to point B in the network

9

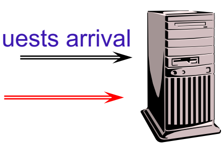
Modelling power: Design tool

system design is often a *counter-intuitive* process

Example 1: doubling arrival rate

evolution of the configuration and workload (upgrade)

Requests arrival



CPU
Serve in
FIFO order

- starting tomorrow the arrival rate will double.
- you should do whatever it takes to ensure that jobs experience the same mean response time. I.e., customers should not notice the effect of the increased arrival rate.

Question: By how much should you increase the CPU speed in order to maintain the same mean response time?

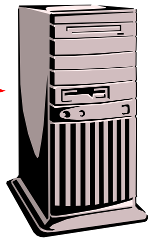
Answer: Less than double!

10

Modelling power: Design tool

Design Example 1: doubling arrival rate

Requests arrival



CPU
Serve in
FIFO order

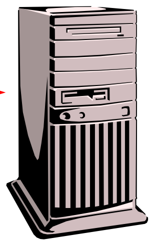
doubling CPU speed together with doubling the arrival rate will result in cutting the mean response time in half!

11

Modelling power: Design tool

Design Example 1: doubling arrival rate

Requests arrival



CPU
Serve in
FIFO order

This is actually identical to the original system, but where time is sped up by a factor of 2
(i.e. a second of service in the original system now requires only half a second. Also, whereas in the old system 3 jobs per second arrive, now 6 jobs per second arrive)

A faster time clock!

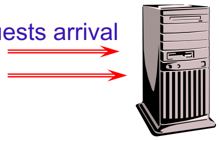
the mean response time becomes half

12

Modelling power: Design tool

Design Example 1: doubling arrival rate

Requests arrival



CPU
Serve in
PS order

Does the answer change?

No!

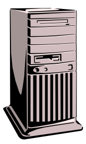
Prof. Vittoria de Nitto Personè 13

13

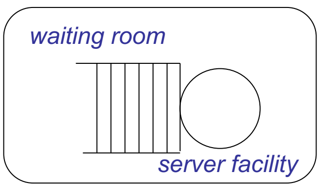
Modelling power: Design tool

Design Example 2

Resource



Server



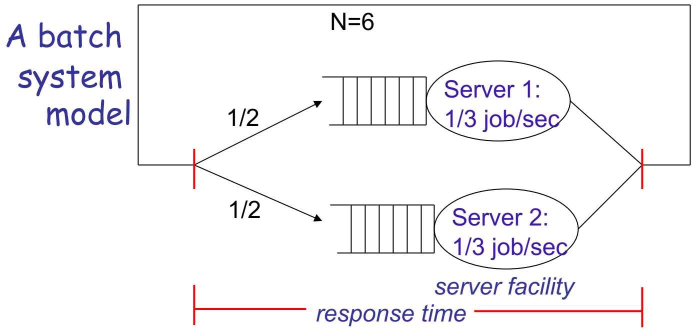
Queue

Prof. Vittoria de Nitto Personè 14

14

Modelling power: Design tool

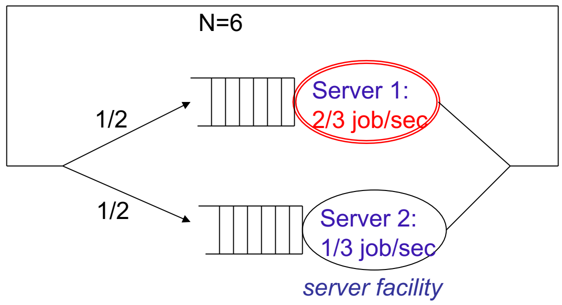
Design Example 2



- The service time distribution is irrelevant

Modelling power: Design tool

Design Example 2



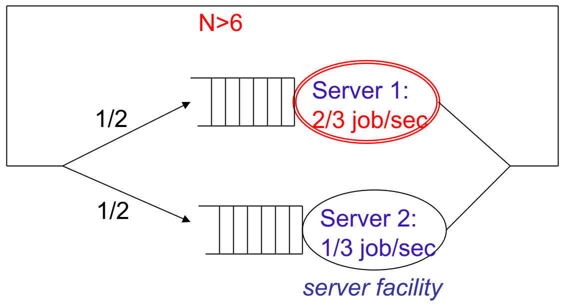
- Is the response time improved?
- Is the throughput improved?

Not really!

sometimes “improvements” do nothing

Modelling power: Design tool

Design Example 2: sometimes “improvements” do nothing



bottleneck identification

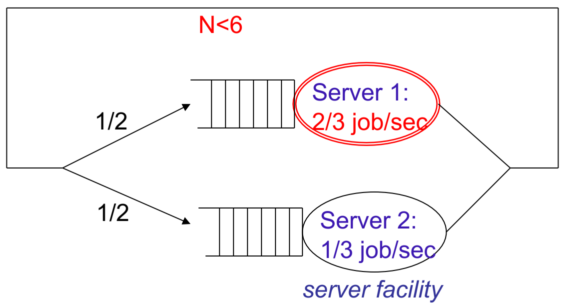
- Is the response time improved?
- Is the throughput improved?

The already negligible effect goes to zero as N increases!

17

Modelling power: Design tool

Design Example 2: sometimes “improvements” do nothing



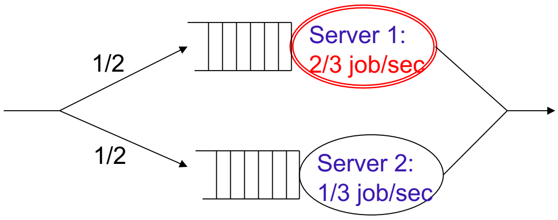
- Is the response time improved?
- Is the throughput improved?

It is possible!
If N is sufficiently low, then the “improvement” helps.
Consider, for example, the case N = 1.

18

Modelling power: Design tool

Design Example 2: sometimes “improvements” do nothing



- Is the response time improved?
- Is the throughput improved?

Absolutely!

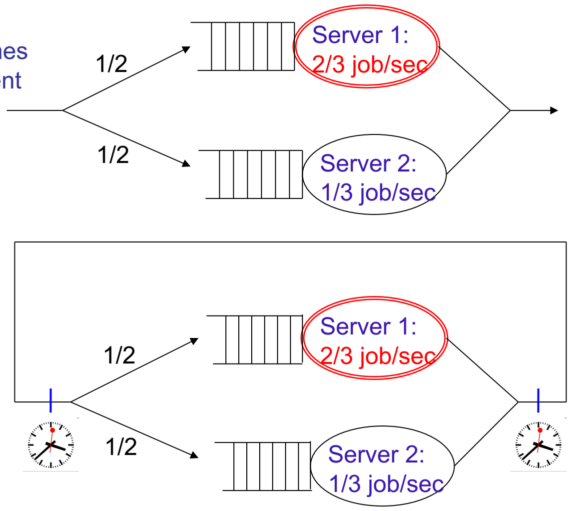
now arrival times are independent of service completions

Prof. Vittoria de Nitto Personè 19

19

Modelling power: Design tool

now arrival times are independent of service completions

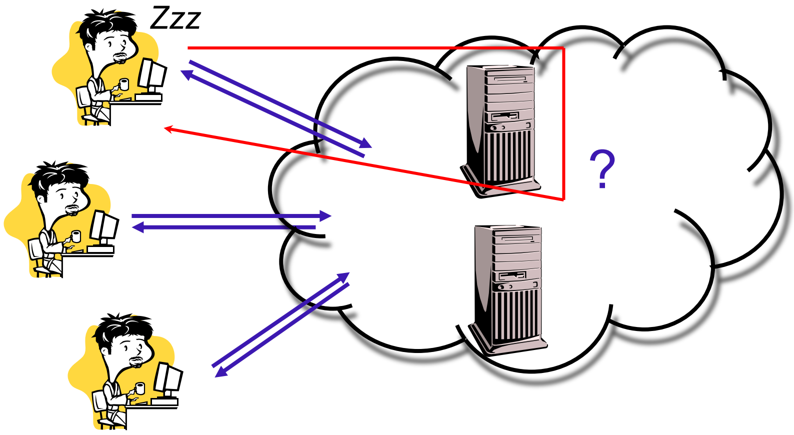


Prof. Vittoria de Nitto Personè 20

20

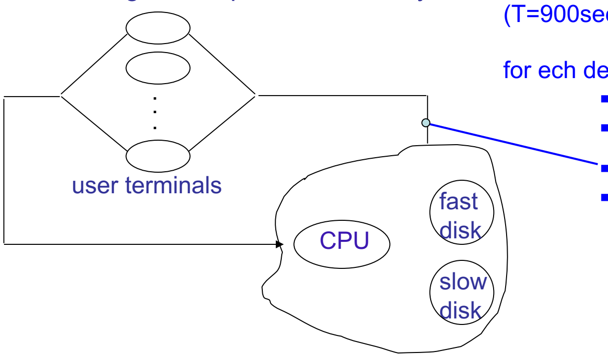
Modelling power: Design tool

Design Example 3: case study



Modelling power: Design tool

Design Example 3: case study



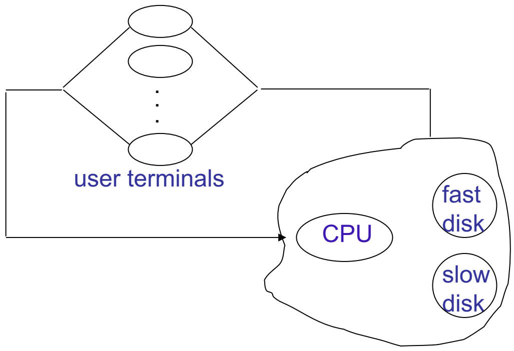
During the observation interval (T=900sec)

for ech device

- busy time
- n° of completions
- n° of system completions
- mean user “think time”

Modelling power: Design tool

Design Example 3: case study



- CPU twice fast
- load balance between disks
- a second fast disk ?

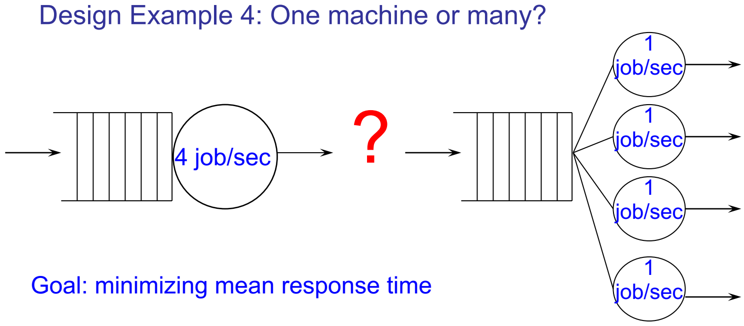
The answer is *counter-intuitive!*

Operational Analysis
- Easy
- Independent on distributional assumptions
- Do not need to know the full network topology

Which of the following changes is most effective in increasing throughput?

Modelling power: Design tool

Design Example 4: One machine or many?



Goal: minimizing mean response time

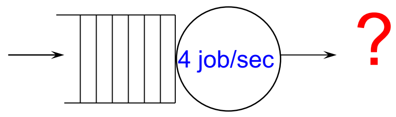
Assumption: jobs *non-preemptible*
each job must be run to completion

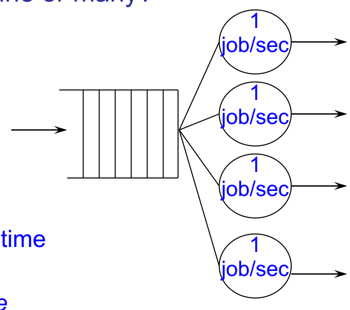
hint: "it depends on the workload."

depends on the variability of the job size distribution

Modelling power: Design tool

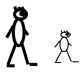
Design Example 4: One machine or many?





Goal: minimizing mean response time

Assumption: jobs *non-preemptible*
each job must be run to completion

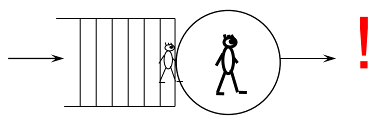
high variability 

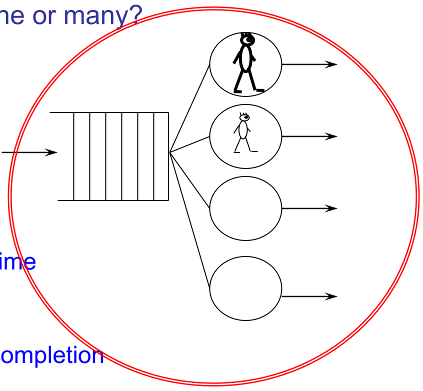
Prof. Vittoria de Nitto Personè 25

25

Modelling power: Design tool

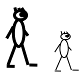
Design Example 4: One machine or many?





Goal: minimizing mean response time

Assumption: jobs *non-preemptible*
each job must be run to completion

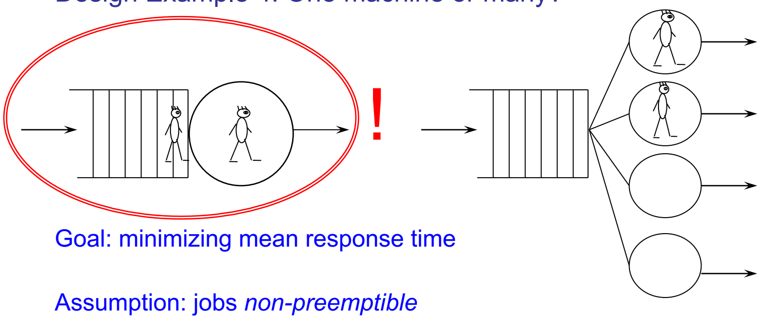
high variability 

Prof. Vittoria de Nitto Personè 26

26

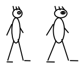
Modelling power: Design tool

Design Example 4: One machine or many?



Goal: minimizing mean response time

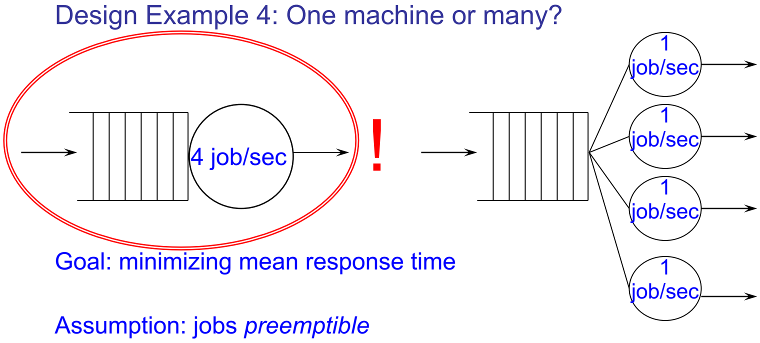
Assumption: jobs *non-preemptible*
each job must be run to completion

low variability 

27

Modelling power: Design tool

Design Example 4: One machine or many?



Goal: minimizing mean response time

Assumption: jobs *preemptible*
each job can be stopped and restarted where they left off

28

Modelling power: Design tool

Design Example 4: One machine or many?

huge applicability

resource → CPU
power
bandwidth
…

Resource allocation

- power management in data centers[1]
- bandwidth partition

1.5% of the total electricity in the U.S. at a cost of nearly $4.5 billion

Cost vs performance

- it is often cheaper (financially) to purchase many slow servers than a few fast servers
- many slow servers can in some cases consume more total power than a few fast ones

[1] A. Gandhi, M. Harchol-Balter, R. Das, and C. Lefurgy. Optimal power allocation in server farms. In *ACM Sigmetrics 2009 Conference on Measurement and Modeling of Computer Systems*, 2009.

Dispositivi connessi: 1,2 miliardi 2018
4,4 miliardi 2023

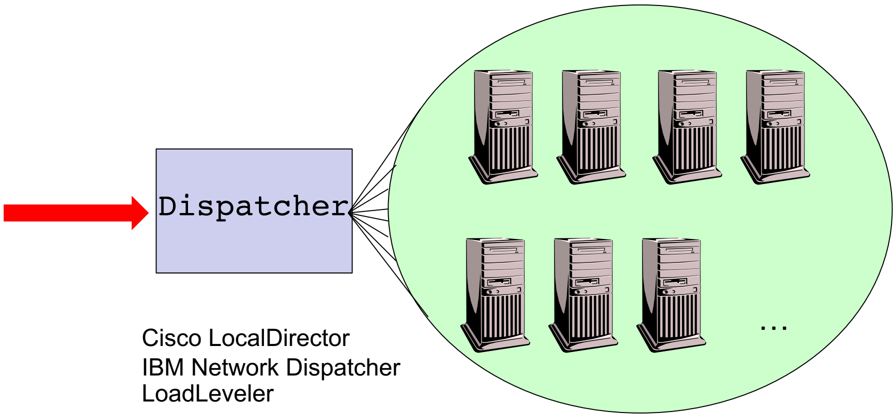
Consumo di energia dell'universo digitale, emissioni di CO_2:
2% 2008→3,7% 2020→8,5% 2025 →14% 2040

https://www.corriere.it/dataroom-milena-gabanelli/emissioni-co2-ambiente-internet-quanto-inquina-nostra-vita-digitale-effetto-serra-consumi-invisibili-streaming-app-video/eb680526-5363-11eb-b612-933264f5acaf-va.shtml

https://www.facebook.com/watch/live/?v=148546343701326&ref=watch_permalink

Modelling power: Design tool

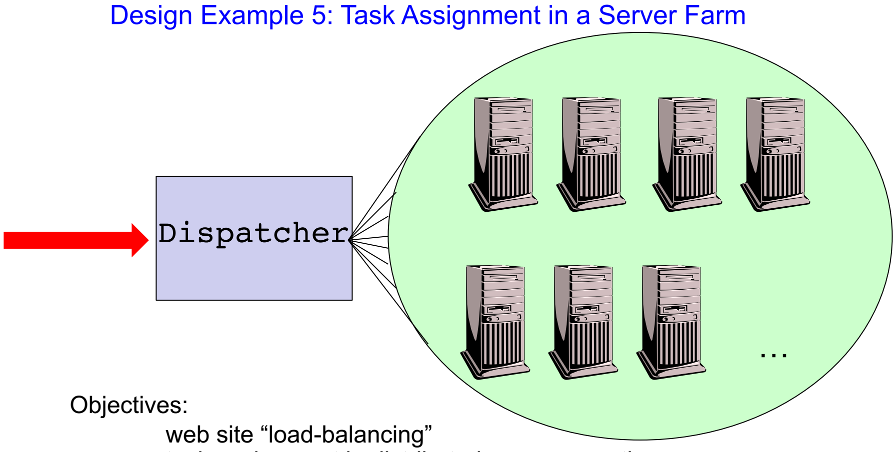
Design Example 5: Task Assignment in a Server Farm



Cisco LocalDirector
IBM Network Dispatcher
LoadLeveler
…

Modelling power: Design tool

Design Example 5: Task Assignment in a Server Farm

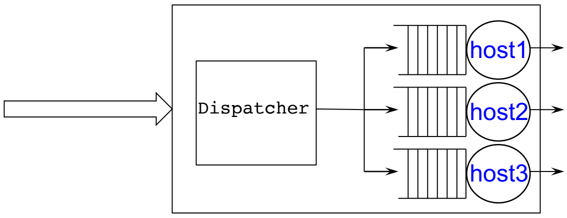


Objectives:

- web site “load-balancing”
- task assignment in distributed supercomputing servers

Modelling power: Design tool

Design Example 5: Task Assignment in a Server Farm



Assumption: homogeneous
single resource for each job
FIFO non-preemptible

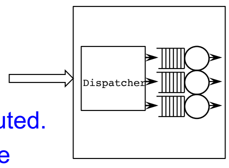
Prof. Vittoria de Nitto Personè 33

33

Modelling power: Design tool

Design Example 5: Task Assignment in a Server Farm

task assignment policies



Random Each job flips a fair coin to determine where it is routed.

Round-Robin The *i*th job goes to host *i* mod *n*, where *n* is the number of hosts, hosts are numbered 0, 1, . . . , *n*-1.

Shortest-Queue Each job goes to the host with the fewest number of jobs.

Central-Queue Rather than have a queue at each host, jobs are pooled at one central queue. When a host is done working on a job, it grabs the first job in the central queue to work on.

Size-Interval-Task-Assignment (SITA) hort” jobs go to the first host,“medium” jobs go to the second host, “lon jobs go to the third host, etc., for some definition of “short,” “medi n.” a

job- size based

Least-Work-Left (LWL) Each job goes t he host with the least total remaining work, where the “work” at a hos the sum of the sizes of jobs there.

Prof. Vittoria de Nitto Personè 34

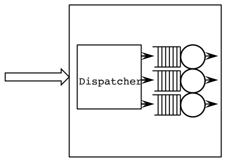
34

Modelling power: Design tool

Design Example 5: Task Assignment in a Server Farm

task assignment policies

Which policies yields the lowest mean response time?



low variability → **LWL**

high variability → **SITA**

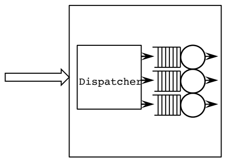
how important was it that we know the size of jobs?

Actually, most task assignment policies do not require knowing the size of jobs.
(It can be proven by induction that LWL is equivalent to Central- Queue.)

Modelling power: Design tool

Design Example 5: Task Assignment in a Server Farm

task assignment policies

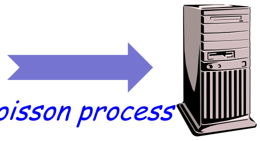


non-preemptible (supercomputer) →
No-load balancing
Different TAPs → different performance
Well analyzed

preemptible (web server farm) →
load balancing to minimize response time
no-load balancing to minimize mean slowdown
Open issue
- variable size distribution
- minimizing the variance of response time

Modelling power: Design tool

Design Example 6: Scheduling policies



Poisson process

- no-assumption on job size distribution
- non-preemptive jobs

Which of these will result in the lowest mean response time?

First-In-First-Out (FIFO) When the server completes a job, it starts working on the job which arrived earliest.

Non-preemptive Last-In-First-Out (LIFO) When the server completes a job, it starts working on job which arrived last.

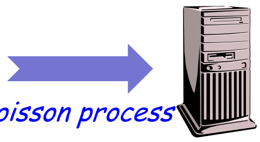
Random When the server completes a job, it starts working on a random job.

These all have the same mean response time !!

37

Modelling power: Design tool

Design Example 6: Scheduling policies



Poisson process

~~Non~~-preemptive Last-In-First-Out (LIFO) ~~When the server completes a job, it starts working on job which arrived last~~.

Whenever a new arrival enters the system, it immediately preempts the job in service

at least moderately variable → A huge performance improvement

hardly variable → Up to a factor of 2 worsening

38