Simulador d'un sistema interactiu

Pau Rul·lan Ferragut <paurullan@gmail.com>

Pràctica de simulació
Convocatòria setembre
Curs 2010–11

Resum

Com exercici pràctica hem construit un simulador d'esdeveniments discrets que modela un sistema informàtic interactiu. Un cop implementat el simulador l'usarem per determinar el temps de resposta del sistema i estudiar el seu comportament a mida que aumentam els usuaris.

1 Especificació del problema

Cal especificar el nostre simulador tant a nivell numèric com en termes de distribucions estadístiques.

1.1 Paràmetres d'entrada

Paràmetre d'entrada	selecció
Temps mig de reflexió	10 seg
Nombre d'usuaris	variable X
Velocitat de CPU	2500 pps
Velocitat de disc	15.000 rpm
Peticions de disc per transacció	5 peticions
Temps de resposta del sistema	variable Y
Throughput	no
Utilització CPU	no
Utilització disc	no

1.2 Modelització dels components

El nostre simulador està caracteritzat per tres components: la població, la CPU i el disc dur.

1.2.1 La població

Els distints usuaris entren al sistema fent una petició i, un cop resolta, s'esperen un temps de reflexió.

Distribució de la població La població es pot modelar mitjançant una distribució exponencial de mitja 10 segons, $Exp(10)$.

1.2.2 La CPU

S'ha escollit un processador comercial AMD Opteron 6140, de capacitat de treball de 62,4K MIPS[1]. Aquest valor és pel conjunt dels vuit nuclis però podem limitar la simulació a un sol nucli, donant aproximadament uns 7,5K MIPS. Suposam que una visita són 3M instruccions de promig. La capacitat de servei i el temps de servei mig són:

$$\mu = 7,5KMIPS \times \frac{\text{1M instruccions}}{\text{1seg}} \times \frac{\text{1transacció}}{\text{3M instruccions}}$$

$$\mu = 2.500 \, \frac{transaccions}{s}$$

$$s = \frac{1}{\mu} = 0,4\,ms$$

Distribució de la CPU La CPU queda modelada amb una distribució constant $C(0,4)$, en ms.

1.2.3 El disc dur

S'ha usat un disc dur de servidor model Seagate Cheetah 15K[2]. Els discs durs clàssics es composen de tres tasques seqüencials: búsqueda (*seek*), latència i transfarència. Aquestes tasques es modelen típicament mitjançant distribucions exponencials, uniformes i constants respectivament.

Aquest disc dur està especificat com:

- Velocitat de rotació 15.000 rpm
- Temps de seek del fabricant 3,4 ms
- Temps de latència del fabriant 2 ms
- Velocitat de transfarència 600 MB/s

Suposam un conjunt de fitxers grans, com documents PDF, de tamany 1,5MB. Aquest fitxers també els suposarem dispersats aleatòriament al llarg del disc però amb els seus blocs continuus. És a dir, obtenir un fitxer consistirà en:

$$T = seek + lat + trans$$

Càlcul del *seek* Com que al nostre exercici sols feim lectures podem usar el temps de *seek* mig de lectura especificat pel fabricant de 3,4 ms. Aquest estadístic es comporta com una distribució exponencial $Exp(3,4)$ en ms.

Càlcul de la latència El nostre disc té una velocitat de rotació de 15.000 rpm, per la qual cosa presenta un temps de volta de 4 ms. Això implica una distribució uniforme $U(0,4)$ en ms. Cal fixar-se que el fabricant ja havia indicat que la latència mitja era de 2ms.

$$15.000\text{rpm}\,\frac{1\text{ min}}{60\text{ seg}} = 250\text{rps} = 4\text{ ms}$$

Càlcul de la transfarència El temps de transfarència ve pel temps de rotació sobre un sector. El fabricant ens especifica que són 600 MB/s i donat que el nostre fitxer és de 1,5MB tardam 2,5ms de transfarència. Això implica una distribució constant de $C(2,5)$ en ms.

$$600\,\text{MB}\,\frac{1\text{ tasca}}{1,5\text{ MB}} = 400\text{ tasques} = 2,5\text{ ms}$$

Distribució del disc dur D'aquesta manera el disc dur es pot modelar mitjançant una distribució composta per la del *seek*, la latència i la transfarència juntament amb la forma del nostre fitxer (en ms):

$$Exp(3,4) + U(0,4) + C(2,5)$$

2 El simulador

Anam a definir com està organitzat el nostre simulador.

Esdeveniments

1. Tasca d'usuari: un usuari ha acabat de reflexionar i llança una tasca cap a la CPU.
2. Sortida de la CPU: una transacció surt de la CPU i pot anar al disc o tornar a reflexionar. Aquesta decisió està codificada estadísticament segons el nombre de peticions a disc per transacció.
3. Sortida del disc: una transacció del disc sempre passa per la CPU.

Variables d'estat

1. Rellotge global
2. Instant de sortida d'una tasca
3. Instant d'arribada d'una tasca

Variables estadístiques

1. Temps de processament acumulat, necessari per calcular els temps de processament mig.
2. Nombre de repeticions totals.

2.1 Organització interna

El simulador consta de varis mòduls:

Principal coordinador entre els distints mòduls. Està programat al fitxer `main.py` i `sim.py`.

Inicialitzador posa a zero els valors i omple de tasques els servidors dels usuaris. Està adins el `sim.py`.

Temporització mitjançant una cua de prioritats gestiona els distints esdeveniments i calcula el següent a llançar. També està a `sim.py`.

Esdeveniments genera la resposta a un esdeveniment escollit per la temporització. Es troba dins el model de cada servidor, a `model.py`.

Aleatoris ampliació del mòdul escrit a PL1 per cobrir les distribucions contants i exponencials, al `rand.py`.

Informe ajuda a determinar el final de la simulació. També genera els càlculs estadístics i les gràfiques. Són els mòduls `statistic.py`, `suite.py` i `chart_generation.py`.

El diagrama de blocs es troba generalitzat al gràfic 1.

3 Pseudocodi

El pseudocodi del simulador el podem trobar a la figura 2.

4 Desenvolupament

Sobre el desenvolupament real del simulador cal dir:

1. S'ha programat amb el llenguatge de propòsit general Python.
2. No s'ha duit a terme la validació amb QNAP2.
3. Per determinar el tamany del transitori s'han fet distintes proves amb la durada de la simulació. Llavors hem tallat considerant la primera meitat dels esdeveniments transitori.
4. S'han fet un mínim de 50 rèpliques per variable de nombre de clients i llavors se n'han fet tantes com ha calgut per obtenir una desviació de manco el 0,1.

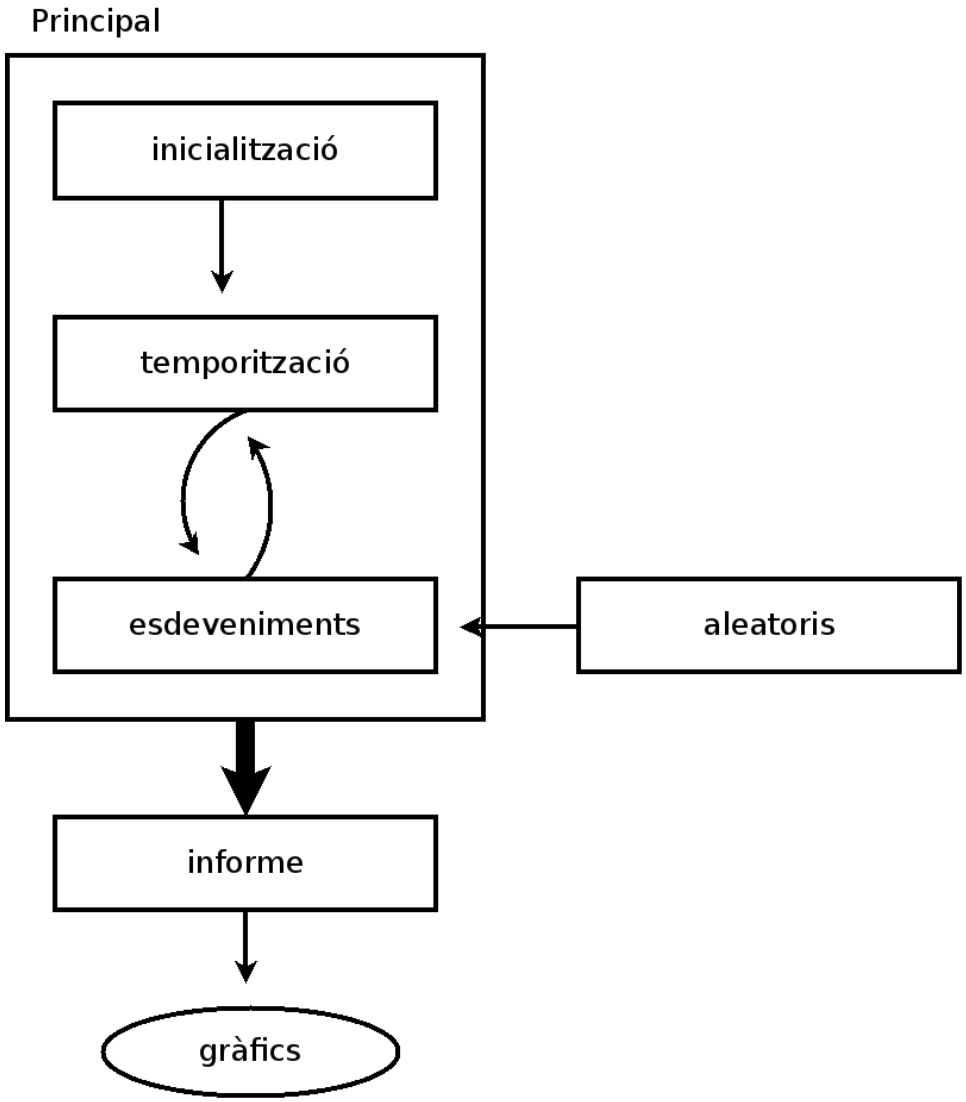

Figura 1: *Diagrama de mòduls del simulador.*

pseudocodi.py

```
def principal:
    n_usuaris = X
    n_peticions_max = N
    tr = simulacio(n_usuaris, n_peticions_max)

def simulacio(n_usuaris, n_peticions_max):
    init
    mentre peticions < n_peticions_max:
        temporitzacio

def init:
    rellotge, temps_resposta, peticions = 0, 0, 0
    pobla(n_usuaris)

def temporitzacio:
    esdeveniment = selecciona(esdeveniments)
    rellotge = temps(esdevenimnet)
    processa(esdeveniment)

def processa(esdeveniment):
    opcio(esdeveniment):
        cpu    -> rutina_cpu
        disc   -> rutina_disc
        usuari -> rutina_usuari

def rutina_disc:
    genera_acces_cpu
    si la cua no es buida:
        decrementa(cua)
        genera_transaccio_disc

def rutina_cpu:
    possibilitat = aleatori( 1 / (n_transaccions_peticio + 1))
    si possibilitat:
        surt_a_reflexionar
    sino:
        genera_transaccio_disc

def rutina_usuari:
    genera_transaccio_cpu
    incrementa_peticions

def surt_a_reflexionar:
    acumulacio_estadistic
    genera_peticio_usuari

def acumulacio_estadistic:
    temps_reposta = rellotge - inici_peticio
```

Figura 2: *Pseudocodi general del simulador.*

5 Resultats

5.1 Determinació del transitori

El procés de càlcul del transitori és un poc bast però efectiu: calculàrem distintes traces per un nombre de clients suficientment gran (100, 200, 250 i 300) i estudiàrem a partir de quin punt el sistema es comportava de manera estable o cíclica. Com a regla general vàrem poder apreciar que obtenint uns vint esdeveniments per client i quedant-nos sols amb la segona meitat de tota la simulació els resultats eren estables.

A les gràfiques es mostra la tendència dels temps de resposta al llarg d'una simulació.

5.2 Càlcul de temps de resposta

Per determinar el punt d'inflexió del nostre sistema calcularem els distints temps de resposta a mida que anam aumentant els clients. Els temps de resposta de les simulació es poden llegir a la figura 7 i es poden veure representats en la gràfica 8.

Després de les simulacions podem observar i concloure que:

1. Sols hem utilitzat un processador pel nostre servidor, de temps de servei $0.4ms$.
2. El disc té un temps de servei de casi $8ms$.
3. Una operació demana una mitja de cinc transaccions de disc.
4. Fins als 200 usuaris el sistema manté un temps de resposta d'un quart de segon ($250ms$), rendiment acceptable per una aplicació interactiva.
5. A partir dels 250 usuaris l'aplicació passa a tenir un temps de resposta d'un segon, fent-la adient sols per aplicacions no purament interactives.
6. Des dels 200 usuaris, afegint un 0.25 de càrrega hem quadruplicat el temps de resposta. Podem afirmar que entre el rang de 200 i 250 usuaris tenim el punt d'inflexió del sistema.
7. Malgrat no tinguem els càlculs de l'utilització de la CPU, la diferència de magnitud de temps de resposta amb el disc (vint vegades) ens indica que probablement el coll de botealla és el disc.

Conclusions Recordant que hem modelat un sistema amb un processador i un disc modern i potent podem veure el gran nombre d'usuaris que s'han de menester per saturar el sistema. El coll de botella és el disc dur doncs és casi vint vegades més lent que el processador. De tota manera cal recordar que si consideràssim un temps de resposta inacceptable a partir del segon ja estam parlant de casi 250 usuaris concurrents per un conjunt *hardware* que no arriba ni als 1000€.

D'aquesta manera i malgrat hem fet grans simplifiacions, com obviar que els computadors actuals estan provistos d'un important conjunt de memòries cau, hem pogut valorar quantitativament el rendiment del sistema i jugar amb el que podria ser la implementació d'un sistema de simulació.

Referències

[1] AMD Opteron 6140 1.2.2

http://www.amd.com/es/products/server/processors/6000-series-platform/pages/6000-series-platform.aspx

http://www.cisco.com/web/DK/assets/docs/presentations/vBootcamp_Performance_Benchmark.pdf

[2] Seagate Cheetah 15K 1.2.3

http://www.seagate.com/www/en-us/products/enterprise-hard-drives/cheetah-15k#tTabContentSpecifications

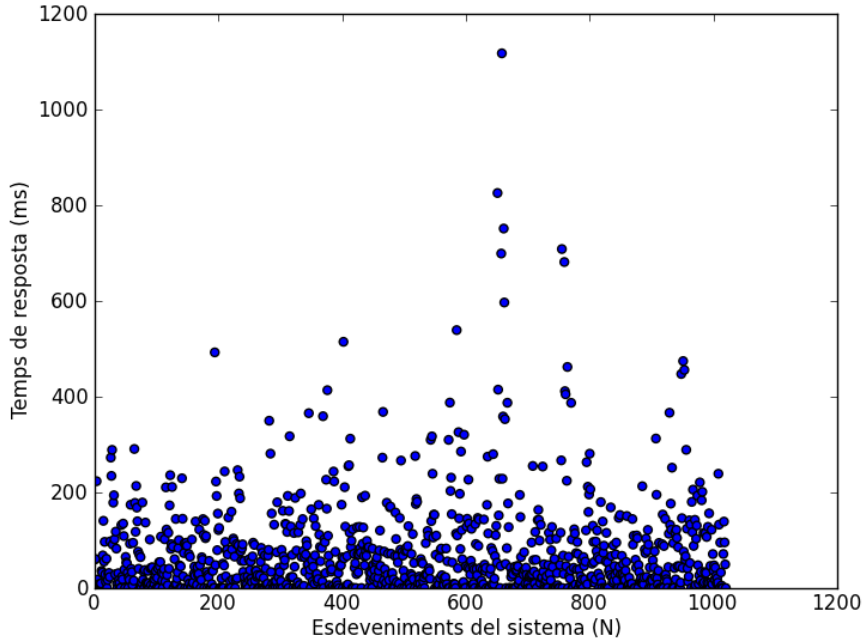

Figura 3: *Execució amb 100 clients per estudiar el transitori*

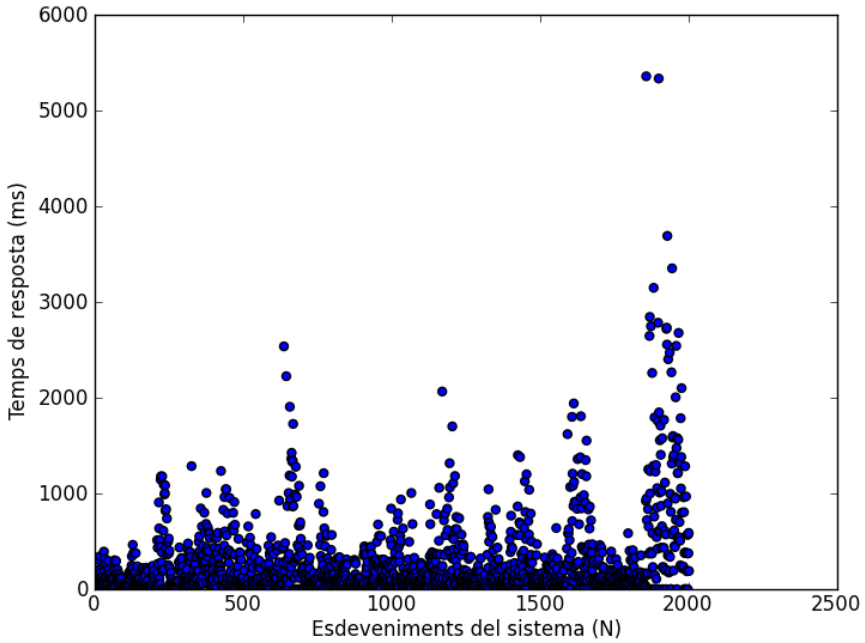

Figura 4: *Execució amb 200 clients per estudiar el transitori*

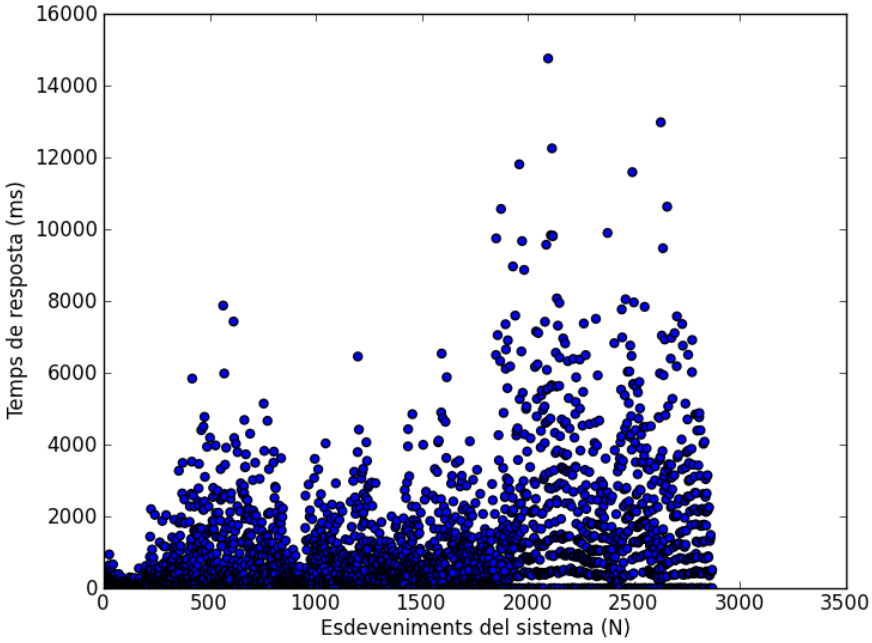

Figura 5: *Execució amb 250 clients per estudiar el transitori*

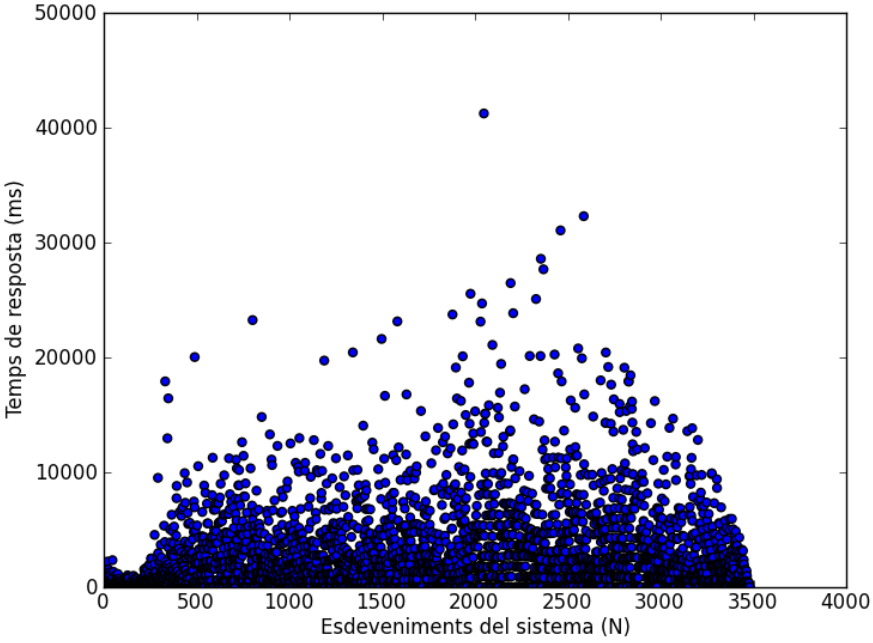

Figura 6: *Execució amb 300 clients per estudiar el transitori*

N usuaris	TR (ms)	N usuaris	TR (ms)
10	42.08	260	1573.98
20	42.26	270	1961.58
30	44.75	280	2334.91
40	49.49	290	2688.70
50	56.00	300	3060.17
60	56.66	310	3421.07
70	63.94	320	3809.61
80	63.48	330	4203.65
90	66.70	340	4591.56
100	75.61	350	4983.98
110	73.92	360	5382.18
120	87.58	370	5762.88
130	84.71	380	6181.55
140	91.32	390	6584.32
150	110.19	400	6949.67
160	110.71	410	7346.90
170	120.57	420	7755.98
180	137.82	430	8086.00
190	190.46	440	8511.05
200	254.08	450	8891.28
210	356.87	460	9328.79
220	484.69	470	9732.96
230	624.87	480	10146.00
240	748.87		
250	1198.98		

Figura 7: *Evolució dels temps de resposta segons es van afegint usuaris. Aquestes simulacions són de rèpliques.*

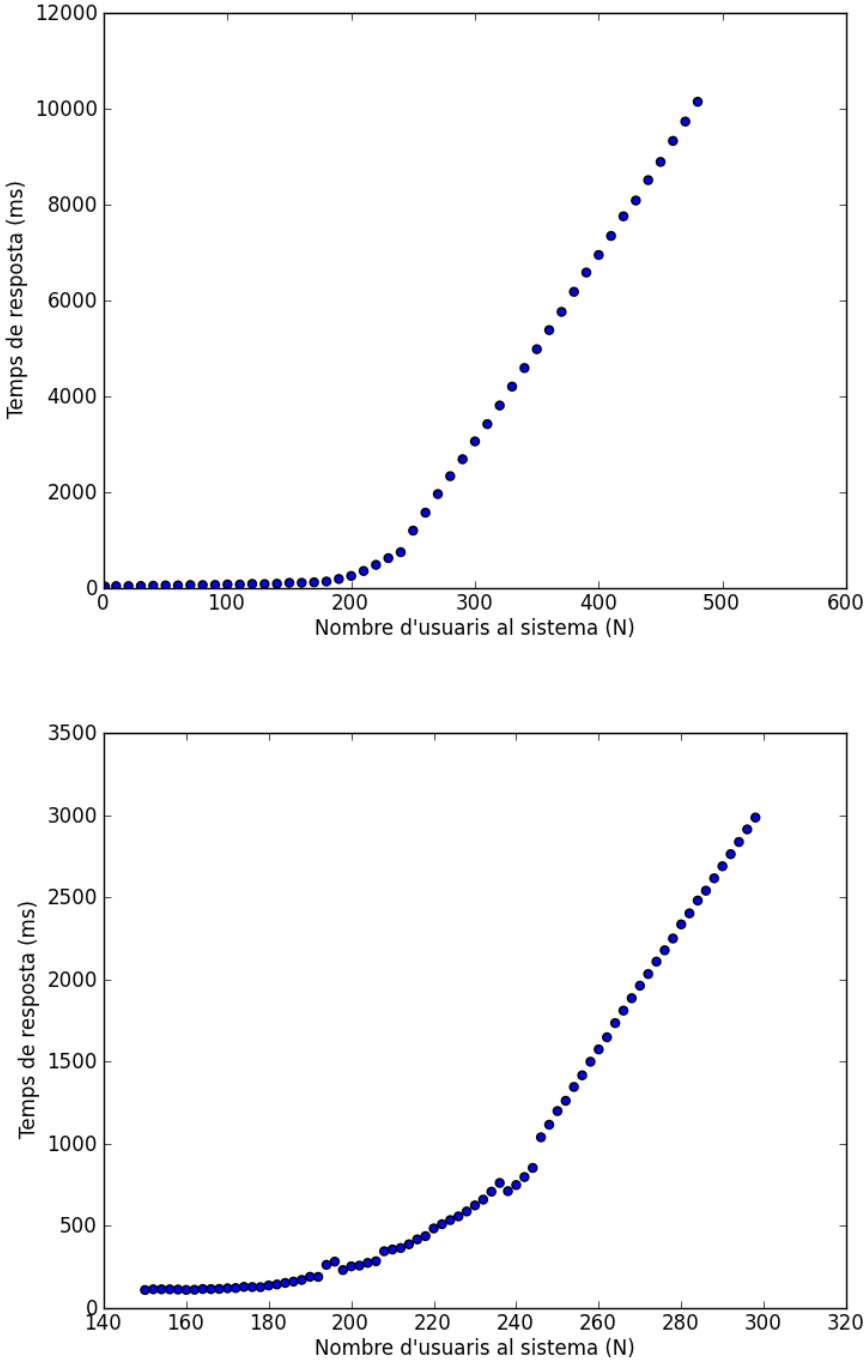

Figura 8: *Evolució dels temps de resposta segons es van afegint usuaris en simulacions mitjançant rèpliques.*

A Codi font

En aquest apèndix trobarem el codi de l'aplicació, el petit programa per obtenir les gràfiques del transitori i els jocs de prova.

A.1 Codi general

Aquí trobarem els distints fitxers del la nostra pràctica:

main.py Programa principal.

suite.py Gestor de tasques i llançament de rèpliques.

sim.py El mòdul principal de la simulació.

model.py Modelització dels usuaris, la cpu i el disc.

rand.py El nostre mòdul de nombres aleatoris.

statistic.py Mòdul del càlcul estadístic.

chart_generator.py Mòdul per pintar les gràfiques.

main.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from __future__ import print_function, division

__version__ = "0.0.1"

import suite

if __name__ == '__main__':
    s = suite.SimSuite()
    # general una llista de clients de deu en deu fins a tres-cents
    clients = range(10, 310)[::10]
    s.client_range = clients
    s.simulation()
    s.chart()
    print(s)
```

suite.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from __future__ import print_function, division

__version__ = "0.0.1"

import sim
```

```
import chart_generator


class SimSuiteBase(object):

    def __init__(self, client_range, sub_class):
        self.client_range = client_range
        self.sub_class = sub_class
        self.values = []

    def simulation(self):
        for clients in self.client_range:
            simulation = self.sub_class(n_clients=clients)
            value = simulation.sim()
            self.values.append(value)

    def chart(self):
        c = chart_generator.Chart()
        c.values = self.values
        c.scale = self.client_range
        c.scatter_rt("sim.png")
        c.scatter_rt_log("sim_log.png")
        c.scatter_rt_y_log("sim_y_log.png")


    def __str__(self):
        s = ""
        for client, value in zip(self.client_range, self.values):
            s += "%d\t%.2f\n" % (client, value, )
        return s


class SimSuite(SimSuiteBase):

    def __init__(self, client_range=[1, 5, 10]):
        super(SimSuite, self).__init__(client_range, sim.SimulationWithReplica)


class SimSuiteWithoutReplica(SimSuiteBase):

    def __init__(self, client_range=[1, 5, 10]):
        super(SimSuiteWithoutReplica, self).__init__(client_range, sim.Simulation)
```

sim.py

```
#!/usr/bin/env python
#-*- coding: utf-8 -*-

__version__ = "0.0.1"
```

```
5
6  import heapq
7  import scipy.stats as stats
8
9  import model
10 import statistic
11 import chart_generator
12
13 class Queue(object):
14
15     def __init__(self):
16         self.queue = []
17
18     @property
19     def first(self):
20         return heapq.heappop(self.queue)
21
22     def add(self, item):
23         heapq.heappush(self.queue, item)
24
25     def __len__(self):
26         return len(self.queue)
27
28 class Simulation(object):
29
30     def __init__(self, n_clients=10, seed=0):
31         """Crea una nova simulació on el nombre de clients és la variable."""
32         # Quantes passes farem a cada intent de simulació
33         self.n_steps = n_clients * 20
34         self.task = 0
35         self.clock = 0
36         self.n_clients = n_clients
37         self.user = model.User(seed)
38         self.cpu = model.CPU(seed)
39         self.disk = model.Disk(seed)
40         self.queue = Queue()
41         self.statistic = statistic.Statistic(n_clients)
42
43     def begin(self):
44         for task in range(self.n_clients):
45             self.add_task(self.user, task)
46         self.task = self.n_clients
47
48     def add_task(self, server, task):
49         time = server.add(task)
50         if time:
51             self.enqueu(server, time)
52
53     def enqueu(self, server, time):
```

```
54              time += self.clock
55              self.queue.add((time, server))
56
57      def sim(self):
58          self.simulation()
59          return self.statistic.calc_rt()
60
61      def simulation(self):
62          self.begin()
63          while True:
64              for step in range(self.n_steps):
65                  self.step()
66              if self.statistic.is_safe_to_stop:
67                  break
68
69      def step(self):
70          """
71          Passa de la simulació.
72          Primer es resol la tasca en marxa.
73          Es concatenen possibles tasques pel mateix servidor.
74          Llavors es resol la continuació d'altres serveis.
75          """
76          time, server = self.queue.first
77          self.clock = time
78          task, time = server.process()
79          if time:
80              self.enqueu(server, time)
81          service = {
82              self.disk: self.service_disk,
83              self.cpu: self.service_cpu,
84              self.user: self.service_user,
85              }
86          service[server](task)
87
88      def service_disk(self, task):
89          """Tasca i connexió d'un disc."""
90          self.add_task(self.cpu, task)
91
92      def service_user(self, task):
93          """Tasca i connexió dels usuaris. Entrada d'esdeveniment"""
94          self.task += 1
95          new_task = self.task
96          self.statistic.add_in(new_task, self.clock)
97          self.add_task(self.cpu, new_task)
98
99      def service_cpu(self, task):
100         """Tasca i connexió de la CPU. Inclou el càlcul de sortida."""
101         choose = self.cpu.choose()
102         if choose == 'disk':
```

```
103             self.add_task(self.disk, task)
104         elif choose == 'user':
105             self.statistic.add_out(task, self.clock)
106             self.add_task(self.user, task)
107 
108 
109 class SimulationWithReplica(object):
110 
111     def __init__(self, n_replica=50, n_clients=10):
112         # replica_pass és per si cal executar vàries vegades
113         self.replica_pass = 1
114         self.error = 0.1
115         # per acceplerar el procés
116         self.error = 0.1
117         self.n_replica = n_replica
118         self.n_clients = n_clients
119         self.chart = chart_generator.Chart()
120         self.values = []
121 
122     def _do_simulation(self):
123         for replica in range(self.n_replica):
124             seed = self.replica_pass * replica
125             sim = Simulation(self.n_clients, seed=seed)
126             sim.simulation()
127             self.values.append(sim.statistic.calc_rt())
128         self.replica_pass += 1
129 
130     def sim(self):
131         while True:
132             self._do_simulation()
133             if self._error_good_enought():
134                 break
135         return self.calc_rt()
136 
137     def calc_rt(self):
138         return stats.tmean(self.values)
139 
140     def _error_good_enought(self):
141         """Determina les configuracions de si cal tornar a replicar. L'enunciat
142         determina un 0.10 de marge d'error. """
143         # http://docs.scipy.org/doc/scipy/reference/stats.html#module-scipy.stats
144         error = stats.tsem(self.values)
145         return error < self.error
```

model.py

```
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3 
```

```
4  from __future__ import print_function
5
6  __version__ = "0.0.1"
7
8  import rand
9
10
11 class Server(object):
12
13     def __init__(self, seed=0):
14         self.generator = rand.Generator(seed)
15         self.name = "server"
16         self.working = None
17         self.queue = []
18
19     def add(self, task):
20         if not self.working:
21             self.working = task
22             return self.service()
23         else:
24             self.queue.append(task)
25             return None
26
27     def process(self):
28         """
29         Acabam de processar una petició. Tornam l'element processat i miram si
30         tenim més tasques.
31         """
32         assert (self.working), "No es pot processar si no hi ha tasques"
33         task = self.working
34         if self.queue:
35             self.working = self.queue.pop(0)
36             time = self.service()
37         else:
38             self.working = None
39             time = None
40         return (task, time)
41
42     @property
43     def empty(self):
44         """El servidor és buit"""
45         return self.working == None
46
47     @property
48     def empty_queue(self):
49         """La qua del servidor és buida"""
50         return len(self.queue) == 0
51
52
```

```
53      def service(self):
54          """Determina el temps de processament en ms. Cal reimplementar-lo per
55          cada sub-servidor"""
56          return 1
57
58      def __len__(self):
59          return len(self.queue)
60
61
62  class User(Server):
63      """
64      Segons el nostre enunciat els ususaris tenen un temps de reflexió de
65      Exp(10*1000) en ms.
66      """
67
68      def __init__(self, seed=0):
69          super(User, self).__init__(seed)
70          self.name = "user"
71          self.lambd = 10 * 1000  # 10s en ms
72
73      def add(self, task):
74          return self.service()
75
76      def process(self):
77          return 0, None
78
79      @property
80      def empty_queue(self):
81          """Truc per simular un servidor sense ques: que sempre estigui buit."""
82          return True
83
84      @property
85      def empty(self):
86          return True
87
88      def service(self):
89          """Exp(10*1000) ms"""
90          return self.generator.exp(self.lambd)
91
92
93  class CPU(Server):
94      """Segons el nostre enunciat la cpu té un temps de resposta C(0.4) ms."""
95
96      def __init__(self, seed=0):
97          super(CPU, self).__init__(seed)
98          self.name = "cpu"
99
100     def service(self):
101         """C(0.4) ms"""
```

```
102             return self.generator.constant(0.4)
103
104         def choose(self):
105             """Escull amb un 0.833 de probabilitats anar al disk (5 transaccions)"""
106             prob = self.generator.num
107             if prob < 0.833:
108                 return 'disk'
109             else:
110                 return 'user'
111
112
113 class Disk(Server):
114     """
115     Segons el nostre enunciat el disc té un temps de resposta que és la
116     combinació de les tres distribucions: seek, latència i transferència de la
117     forma (en ms) Exp(3.4) + U(0, 4) + C(2, 5).
118     """
119
120     def __init__(self, seed=0):
121         super(Disk, self).__init__(seed)
122         self.name = "disk"
123
124     def service(self):
125         """ Exp(3.4) + U(0, 4) + C(2, 5)"""
126         exp = self.generator.exp(3.4)
127         u = self.generator.uniform(0, 4)
128         c = self.generator.constant(3.5)
129         return exp + u + c
```

rand.py

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4  __version__ = "0.0.2"
5
6  import math
7
8
9  class Generator():
10     """Genèrador congruencia lineal multiplicatiu mòdul m primer d'ordre 1
11     El llistat de llavors l'ha donat el professor a l'enunciat de la primera
12     pràctica.
13     Nombres bons:
14     m = 2147483647 = 2**31 -1
15     a =  630360016 = 2**4 * 11 * 13 * 137 * 2011
16     """
17     def __init__(self, seed=0):
18         self._seed_list = [
```

```
            1973272912,281629770, 20006270, 1280689831, 2096730329,
            1933576050, 913566091,246780520, 1363774876, 604901985,
            1511192140, 1259851944, 824064364,150493284, 242708531,
            75253171, 1964472944, 1202299975, 233217322, 1911216000,
            726370533,403498145,993232223, 1103205531, 762430696,
            1922803170, 1385516923, 76271663,413682397,726466604,
            336157058, 1432650381, 1120463904, 595778810, 877722890,
            1046574445, 68911991, 2088367019, 748545416, 622401386,
            2122378830,640690903, 1774806513, 2132545692, 2079249579,
            78130110,852776735, 1187867272, 1351423507, 1645973084,
            1997049139,922510994, 2045512870,898585771, 243649545,
            1004818771,773686062,403188473,372279877, 1901633463,
            498067494, 2087759558,493157915,597104727, 1530940798,
            1814496276, 536444882, 1663153658,855503735, 67784357,
            1432404475,619691088, 119025595,880802310,176192644,
            1116780070,277854671, 1366580350, 1142483975, 2026948561,
            1053920743,786262391, 1792203830, 1494667770, 1923011392,
            1433700034, 1244184613, 1147297105,539712780, 1545929719,
            190641742, 1645390429,264907697,620389253, 1502074852,
            927711160, 364849192, 2049576050,638580085, 547070247,
            ]
        self._nseed = seed % len(self._seed_list)
        self._z = self._seed_list[self._nseed]
        self._m = 2**31 - 1
        self._a = 2**4 * 11 * 13 * 137 * 2011

    @property
    def num(self):
        self._z = (self._a * self._z) % self._m
        return self._z/float(self._m)

    def next_generator(self):
        """Crea un generador amb la següent secció de llavor"""
        return Generator((self._nseed +1) % len(self._seed_list))

    def exp(self, lambd):
        """
        Distribució exponencial de mitja 1/lamd
        http://en.wikipedia.org/wiki/Exponential_distribution
        """
        if lambd <= 0:
            return 0
        x = self.num
        return - lambd * math.log(1 - x)

    def constant(self, x):
        """
        Distribució constant. Escrita per motius de completesa.
        """
```

```
68          return x
69
70      def uniform(self, inf, sup):
71          """Distribució uniforme"""
72          assert(inf >= 0)
73          assert(sup > 0)
74          assert(inf < sup)
75          x = self.num
76          return x * (sup - inf) + inf
77
78
79  if __name__ == '__main__':
80      n = 10
81      gen = Generator()
82      for i in range(n):
83          print gen.num
```

statistic.py

```
1   #!/usr/bin/env python
2   # -*- coding: utf-8 -*-
3
4   from __future__ import print_function, division
5
6   __version__ = "0.0.1"
7
8   import scipy.stats as stat
9
10  import chart_generator
11
12  class Statistic(object):
13
14      def __init__(self, n_clients=10):
15          self.in_tasks = {}
16          self.out_tasks = {}
17          self.n_clients = n_clients
18          self.events = 10
19          self.threshold = 0.5
20
21      def add_in(self, task, time):
22          self.in_tasks[task] = time
23
24      def add_out(self, task, time):
25          self.out_tasks[task] = time
26
27      def calc_rt_filtered(self):
28          sub_list_keys = self.out_tasks.keys()
29          sub_list_keys.sort()
30          sub = int(len(sub_list_keys)*self.threshold)
```

```
31          sub_keys = sub_list_keys[sub:]
32          return self._calc_rt_base(sub_keys)
33
34      def calc_rt(self):
35          return self.calc_rt_nonfiltered()
36
37      def calc_rt_nonfiltered(self):
38          return self._calc_rt_base(self.out_tasks.keys())
39
40      def _calc_rt_base(self, keys):
41          values = []
42          for key in keys:
43              in_time = self.in_tasks[key]
44              out_time = self.out_tasks[key]
45              values.append(out_time - in_time)
46          return stat.tmean(values)
47
48      def calc_rt_evolution(self):
49          values = []
50          for key in self.out_tasks.iterkeys():
51              in_time = self.in_tasks[key]
52              out_time = self.out_tasks[key]
53              values.append(out_time - in_time)
54          return values
55
56      def plot_rt_evolution(self, filename):
57          chart = chart_generator.Chart()
58          chart.values = self.calc_rt_evolution()
59          chart.scatter_rt_evolution(filename)
60
61      @property
62      def is_safe_to_stop(self):
63          return len(self) > (self.n_clients * self.events)
64
65      def __len__(self):
66          return len(self.out_tasks)
```

chart-generator.py

```
1   #!/usr/bin/env python
2   # -*- coding: utf-8 -*-
3
4   from __future__ import print_function
5
6   import pylab
7
8   __version__ = "0.0.1"
9
10
```

```
11 class Chart(object):
12
13     def __init__(self):
14         self.values = []
15         self.names = []
16         self.scale = []
17
18         self.ylabel = "Valors Y"
19         self.title = u"Títol del gràfic"
20         self.color = 'grey'
21
22     def save_response_time(self, filename):
23         self.title = u"Temps de resposta de les búsquedes"
24         self.ylabel = "Temps (en segons)"
25         self.save_bar(filename)
26
27     def save_visited_nodes(self, filename):
28         self.title = u"Nombre de nodes visitats en les búsquedes"
29         self.ylabel = "Nodes"
30         self.save_bar(filename)
31
32     def save_bar(self, filename):
33         if len(self.values) != len(self.names):
34             raise TypeError
35         fig = pylab.figure()
36         ax = fig.add_subplot(1, 1, 1)
37         ind = range(len(self.values))
38         ax.bar(ind, self.values, facecolor=self.color, align='center')
39         ax.set_ylabel(self.ylabel)
40         ax.set_title(self.title, fontstyle='italic')
41         ax.set_xticks(ind)
42         ax.set_xticklabels(self.names)
43         fig.autofmt_xdate()
44         pylab.savefig(filename)
45         pylab.close()
46
47     def _scatter_rt_base(self):
48         pylab.xlabel("Nombre d'usuaris al sistema (N)")
49         pylab.ylabel("Temps de resposta (ms)")
50         pylab.scatter(self.scale, self.values)
51         pylab.xlim(0)
52         pylab.ylim(0)
53
54     def scatter_rt(self, filename):
55         self._scatter_rt_base()
56         pylab.savefig(filename)
57         pylab.close()
58
59     def scatter_rt_log(self, filename):
```

```
60          self._scatter_rt_base()
61          pylab.xlabel("Nombre d'usuaris al sistema (N) [escala log]")
62          pylab.xscale("log")
63          pylab.savefig(filename)
64          pylab.close()
65
66      def scatter_rt_y_log(self, filename):
67          self._scatter_rt_base()
68          pylab.ylabel("Temps de resposta (ms) [log]")
69          pylab.yscale("log")
70          pylab.savefig(filename)
71          pylab.close()
72
73      def scatter_rt_evolution(self, filename):
74          pylab.xlabel("Esdeveniments del sistema (N)")
75          pylab.ylabel("Temps de resposta (ms)")
76          pylab.scatter(range(len(self.values)), self.values)
77          pylab.xlim(0)
78          pylab.ylim(0)
79          pylab.savefig(filename)
80          pylab.close()
```

A.2 Càlcul del transitori

Hem usat aquest petit programa per obtenir un parell de mostres gràfiques i així tenir pistes sobre com determinar el transitori.

calcul-transitori.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from __future__ import print_function, division

__version__ = "0.0.1"

import sim

if __name__ == '__main__':

    for number in [100, 150, 200, 250, 300]:
        s = sim.Simulation(number)
        s.simulation()
        name = "evo-%d.png" % number
        s.statistic.plot_rt_evolution(name)
```

A.3 Jocs de prova

Per la tasca de programació hem aprofitat per aplicar per primera vegada un model de programació anomenat *programació dirigida per proves* (de l'anglès *test driven development* o TDD). El que es fa és *primer* escriure els jocs de proves per cadasqun dels mètodes que es van necessitant i no s'avança a la següent característica fins que totes les proves anteriors funcionen. Aquesta metodologia pot pareixer més engorrosa però un cop el programa va més enllà d'un petit mòdul i alguna gràfica s'accelera la programació per que és molt senzill detectar els problemes causats pel nou desenvolupament.

Aquest comentari el feim per indicar que el gran nombre de tests de a causa de l'estil de programació poc habitual.

test-chart.py

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4  from __future__ import print_function
5
6  __version__ = "0.0.1"
7
8  import unittest
9  import os
10 import filecmp
11
12 import chart_generator
13
14 def make_filenames(name, filedir="test/test_img"):
15     """Per facilitar la creació dels noms interns"""
16     d = filedir + "/"
17     ext = ".png"
18     filename = d + name + ext
19     master   = d +name + "_master" + ext
20     return filename, master
21
22 class TestMakeFilenames(unittest.TestCase):
23
24     def test_make_filenames(self):
25         """Codi per provar el mètode make_filenames"""
26         f, m = make_filenames("a")
27         self.assertEqual(f, "test/test_img/a.png")
28         self.assertEqual(m, "test/test_img/a_master.png")
29
30
31 class TestChart(unittest.TestCase):
32
33     def setUp(self):
```

```
34          self.chart = chart_generator.Chart()
35          self.filedir = "test/test_img"
36          if not os.access(self.filedir, os.W_OK):
37              os.mkdir(self.filedir)
38
39
40      def test_first(self):
41          self.chart.values = [5]
42          self.chart.names = ["Cuixot"]
43          filename, master = make_filenames("test_first")
44          self.chart.save_bar(filename)
45          self.assertTrue(filecmp.cmp(filename, master))
46
47      def test_second(self):
48          self.chart.values = [5, 7]
49          self.chart.names = ["Cuixot", "Indiot"]
50          filename, master = make_filenames("test_second")
51          self.chart.save_bar(filename)
52          self.assertTrue(filecmp.cmp(filename, master))
53
54      def test_third(self):
55          self.chart.names = list("hola_nena")
56          self.chart.values = range(1, len(self.chart.names)+1)
57          filename, master = make_filenames("test_third")
58          self.chart.save_bar(filename)
59          self.assertTrue(filecmp.cmp(filename, master))
60
61      def test_four_size(self):
62          """En realitat la cota superior ve per si caben enters"""
63          self.chart.values = [5, 7.1, 6.4]
64          self.chart.names = ["Cuixot", "Indiot", "Pollastre"]
65          filename, master = make_filenames("test_four")
66          self.chart.save_bar(filename)
67          self.assertTrue(filecmp.cmp(filename, master))
68
69      def test_temps_resposta_u(self):
70          """
71          Exemple de com s'ha de cridar pels temps de resposta de la pràctica.
72          """
73          self.chart.values = [0.15, 0.004, 0.003, 0.04]
74          self.chart.names = [
75              "Profunditat",
76              u"Veinat més proper",
77              "A*",
78              "A* relaxat",
79              ]
80          filename, master = make_filenames("test_temps_resposta_u")
81          self.chart.save_response_time(filename)
82          self.assertTrue(filecmp.cmp(filename, master))
```

```
83
84      def test_nodes_visitats_u(self):
85          """
86          Exemple de com s'ha de cridar pels nodes visitats en la pràctica.
87          """
88          self.chart.values = [5, 6, 2, 3]
89          self.chart.names = [
90              "Profunditat",
91              u"Veinat més proper",
92              "A*",
93              "A* relaxat",
94              ]
95          filename, master = make_filenames("test_nodes_visitats_u")
96          self.chart.save_response_time(filename)
97          self.assertTrue(filecmp.cmp(filename, master))
98
99      def test_different_values_exception(self):
100         """No podem tenir un gràfic amb manco categories que valors"""
101         self.chart.values = [5]
102         self.chart.names = ["Cuixot", "Indiot"]
103         self.assertRaises(TypeError, self.chart.save_bar, "")
104
105
106 class TestEstatistic(unittest.TestCase):
107
108     def setUp(self):
109         filedir = "img"
110         if not os.access(filedir, os.W_OK):
111             os.mkdir(filedir)
112         self.chart = chart_generator.Chart()
113
114     def test_scatter_log(self):
115         self.chart.values = [1, 1.5, 2, 3, 4]
116         self.chart.scale = [1, 5, 10, 100, 150]
117         filename, master = make_filenames("test_scatter_log")
118         self.chart.scatter_rt_log(filename)
119         self.assertTrue(filecmp.cmp(filename, master))
120
121     def test_scatter_lin(self):
122         self.chart. values = [1, 1.5, 2, 3, 4]
123         self.chart.scale = [1, 5, 10, 100, 150]
124         filename, master = make_filenames("test_scatter_lin")
125         self.chart.scatter_rt(filename)
126         self.assertTrue(filecmp.cmp(filename, master))
127
128
129 if __name__ == '__main__':
130     unittest.main()
```

test-rand.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from __future__ import print_function

__version__ = "0.0.1"

import unittest

import rand

class TestGenerator(unittest.TestCase):

    def setUp(self):
        self.gen = rand.Generator()

    def test_first(self):
        num = self.gen.num
        self.assertAlmostEqual(num, 0.4005279)

    def test_constant(self):
        constant = 5
        num = self.gen.constant(constant)
        self.assertEqual(num, constant)

    def test_exp(self):
        lambd = 10*1000
        num = self.gen.exp(lambd)
        self.assertAlmostEqual(num, 5117.058580378)

    def test_exp_2(self):
        num_list = [self.gen.exp(1) for i in range(4)]
        results = [0.51170585, 0.95084457, 0.56844949, 0.4829024]
        for num, result in zip(num_list, results):
            self.assertAlmostEqual(num, result)

    def test_uniform(self):
        inf, sup = 0, 10
        result = 4.00527908
        num = self.gen.uniform(inf, sup)
        self.assertAlmostEqual(result, num)

    def test_uniform_2(self):
        inf, sup = 0, 10
        num_list = [self.gen.uniform(inf, sup) for i in range(4)]
        results = [4.00527908, 6.135854695986889, 4.335970293886946, 3.8300998]
        for num, result in zip(num_list, results):
```

```
48              self.assertAlmostEqual(num, result)
49
50
51  if __name__ == '__main__':
52      unittest.main()
```

test-statistic.py

```
1   #!/usr/bin/env python
2   # -*- coding: utf-8 -*-
3
4   from __future__ import print_function, division
5
6   __version__ = "0.0.1"
7
8   import unittest
9
10  import statistic
11
12  class TestStatistic(unittest.TestCase):
13
14      def setUp(self):
15          self.s = statistic.Statistic()
16
17      def test_calc_rt(self):
18          """Calcula un temps de resposta senzill"""
19          self.s.add_in(1, 2)
20          self.s.add_out(1, 5)
21          rt = self.s.calc_rt()
22          self.assertEqual(rt, 3)
23
24      def test_calc_rt_two_filtered(self):
25          """Calcula un temps de resposta senzill.
26          Malgrat és filtrat es comporta igual.
27          """
28          tasks = [(2, 3), (3, 4), (5, 6), (7, 8)]
29          for num, task in enumerate(tasks):
30              in_t, out_t = task
31              self.s.add_in(num, in_t)
32              self.s.add_out(num, out_t)
33          rt = self.s.calc_rt()
34          self.assertEqual(rt, 1)
35
36      def test_calc_rt_two_nonfiltered(self):
37          """Calcula un temps de resposta senzill"""
38          tasks = [(2, 3), (3, 4), (5, 6), (7, 8)]
39          for num, task in enumerate(tasks):
40              in_t, out_t = task
41              self.s.add_in(num, in_t)
```

```
42              self.s.add_out(num, out_t)
43          rt = self.s.calc_rt()
44          self.assertEqual(rt, 1)
45
46
47      def test_calc_rt_three_filtered(self):
48          """Calcula un temps de resposta senzill.
49          Malgrat és filtrat es comporta igual.
50          """
51          tasks = [(2, 2.5), (3, 4), (5, 5.5), (7, 7.5), (8, 8.5)]
52          for num, task in enumerate(tasks):
53              in_t, out_t = task
54              self.s.add_in(num, in_t)
55              self.s.add_out(num, out_t)
56          rt = self.s.calc_rt_filtered()
57          self.assertEqual(rt, 0.5)
58
59      def test_calc_rt_three_nonfiltered(self):
60          """Calcula un temps de resposta senzill"""
61          tasks = [(2, 2.5), (3, 4), (5, 5.5), (7, 7.5), (8, 8.5)]
62          for num, task in enumerate(tasks):
63              in_t, out_t = task
64              self.s.add_in(num, in_t)
65              self.s.add_out(num, out_t)
66          rt = self.s.calc_rt()
67          self.assertEqual(rt, 3/5)
68
69
70  if __name__ == '__main__':
71      unittest.main()
```

test-model.py

```
1   #!/usr/bin/env python
2   #-*- coding: utf-8 -*-
3
4   from __future__ import print_function, division
5
6   __version__ = "0.0.1"
7
8   import unittest
9
10  import model
11
12
13  class TestServer(unittest.TestCase):
14
15      def setUp(self):
16          self.server = model.Server()
```

```
17
18      def test_add_queue(self):
19          self.server.add(1)
20          self.assertFalse(self.server.empty)
21          self.assertTrue(self.server.empty_queue)
22          self.server.process()
23          self.assertTrue(self.server.empty)
24          self.assertTrue(self.server.empty_queue)
25
26      def test_empty_queue(self):
27          self.server.add(1)
28          self.assertTrue(self.server.empty_queue)
29          self.server.add(2)
30          self.assertFalse(self.server.empty_queue)
31          self.server.process()
32          self.assertTrue(self.server.empty_queue)
33          self.server.process()
34          self.assertTrue(self.server.empty_queue)
35
36      def test_process(self):
37          self.server.add(1)
38          self.server.add(2)
39          self.server.add(3)
40          task, time = self.server.process()
41          self.assertEqual(task, 1)
42          task, time = self.server.process()
43          self.assertEqual(task, 2)
44          task, time = self.server.process()
45          self.assertEqual(task, 3)
46
47      def test_process_time(self):
48          self.server.add(1)
49          self.server.add(2)
50          self.server.add(3)
51          task, time = self.server.process()
52          self.assertEqual(time, self.server.service())
53          task, time = self.server.process()
54          self.assertEqual(time, self.server.service())
55          task, time = self.server.process()
56          self.assertEqual(time, None)
57
58      def test_service_example(self):
59          first = 1
60          self.server.add(first)
61          item, service = self.server.process()
62          self.assertEqual(first, item)
63          self.assertEqual(service, None)
64
65
```

```
66  class TestUser(unittest.TestCase):
67
68      def setUp(self):
69          self.user = model.User()
70
71      def test_service(self):
72          time = self.user.service()
73          result = 5117.05858038
74          self.assertAlmostEqual(time, result)
75
76      def test_service_2(self):
77          num_list = [self.user.service() for i in range(5)]
78          results = [
79              5117.05858038,
80              9508.44572792,
81              5684.49491893,
82              4829.02431238,
83              7054.95497321,
84              ]
85          for num, result in zip(num_list, results):
86              self.assertAlmostEqual(num, result)
87
88  class TestCPU(unittest.TestCase):
89
90      def setUp(self):
91          self.cpu = model.CPU()
92
93      def test_service(self):
94          """El temps de la CPU és constant a 0.4"""
95          const = 0.4
96          time = self.cpu.service()
97          self.assertAlmostEqual(time, const)
98
99      def test_choose(self):
100         """Escollim un 0.833 de vegades el disk"""
101         steps = 10*1000
102         choices = [self.cpu.choose() for i in range(steps)]
103         n_disk = filter(lambda x: x == 'disk', choices)
104         appereances = len(n_disk)/steps
105         self.assertLess(appereances, 0.85)
106         self.assertGreater(appereances, 0.8)
107         self.assertAlmostEqual(appereances, 0.8318)
108
109
110 class TestDisk(unittest.TestCase):
111
112     def setUp(self):
113         self.disk = model.Disk()
114
```

```
115     def test_service(self):
116         time = self.disk.service()
117         self.assertAlmostEqual(time, 7.694141795723532)
118 
119     def test_service_seed_one(self):
120         self.disk = model.Disk(seed=1)
121         time = self.disk.service()
122         self.assertAlmostEqual(time, 5.248957469228879)
123 
124     def test_four_services(self):
125         time = sum([self.disk.service() for i in range(4)])
126         self.assertAlmostEqual(time, 30.92443576341943)
127 
128     def test_five_services(self):
129         time = sum([self.disk.service() for i in range(5)])
130         self.assertAlmostEqual(time, 39.459474646190515)
131 
132 if __name__ == '__main__':
133     unittest.main()
```

test-sim.py

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  
4  from __future__ import print_function, division
5  
6  __version__ = "0.0.1"
7  
8  import unittest
9  import filecmp
10 
11 import sim
12 
13 def make_filenames(name, filedir="test/test_img"):
14     """Per facilitar la creació dels noms interns"""
15     d = filedir + "/"
16     ext = ".png"
17     filename = d + name + ext
18     master   = d +name + "_master" + ext
19     return filename, master
20 
21 
22 class TestMakeFilenames(unittest.TestCase):
23 
24     def test_make_filenames(self):
25         """Codi per provar el mètode make_filenames"""
26         f, m = make_filenames("a")
27         self.assertEqual(f, "test/test_img/a.png")
```

```
28          self.assertEqual(m, "test/test_img/a_master.png")
29
30
31  class TestSimulation(unittest.TestCase):
32
33      def setUp(self):
34          self.sim = sim.Simulation()
35
36      def test_cpu_one(self):
37          """Confirma el comportament de la CPU"""
38          self.sim.add_task(self.sim.cpu, 1)
39          self.assertEqual(len(self.sim.cpu), 0)
40          self.sim.add_task(self.sim.cpu, 2)
41          self.sim.add_task(self.sim.cpu, 3)
42          self.sim.add_task(self.sim.cpu, 4)
43          self.assertEqual(len(self.sim.cpu), 3)
44          task, time = self.sim.cpu.process()
45          self.assertEqual(task, 1)
46          task, time = self.sim.cpu.process()
47          self.assertEqual(task, 2)
48          task, time = self.sim.cpu.process()
49          self.assertEqual(task, 3)
50          task, time = self.sim.cpu.process()
51          self.assertEqual(task, 4)
52          self.assertEqual(time, None)
53          self.assertEqual(len(self.sim.cpu), 0)
54
55      def test_simple_init(self):
56          self.sim.begin()
57          queue_len = len(self.sim.queue.queue)
58          self.assertEqual(self.sim.n_clients, queue_len)
59
60      def test_multiple_init(self):
61          initial_clients = 100
62          self.sim = sim.Simulation(initial_clients)
63          self.sim.begin()
64          self.assertEqual(initial_clients, self.sim.n_clients)
65          queue_len = len(self.sim.queue.queue)
66          self.assertEqual(queue_len, self.sim.n_clients)
67
68      def test_simple_simulation(self):
69          self.sim.simulation()
70          in_events = len(self.sim.statistic.in_tasks.keys())
71          out_events = len(self.sim.statistic.out_tasks.keys())
72          rt = self.sim.statistic.calc_rt()
73          self.assertEqual(in_events, 115)
74          self.assertEqual(out_events, 114)
75          self.assertEqual(rt, 42.081392820958506)
76
```

```
77      def test_simple_simulation_plot_rt_evolution(self):
78          filename, master = make_filenames("test_simple_simulation_plot")
79          self.sim.simulation()
80          self.sim.statistic.plot_rt_evolution(filename)
81          self.assertTrue(filecmp.cmp(filename, master))
82
83      def test_big_simulation(self):
84          """Simulació amb un nombre considerable de clients"""
85          initial_clients = 100
86          self.sim = sim.Simulation(initial_clients)
87          self.sim.simulation()
88          rt = self.sim.statistic.calc_rt()
89          self.assertEqual(rt, 75.609456257117614)
90
91      def test_big_sim_shortcut(self):
92          """Simulació amb un nombre considerable de clients"""
93          initial_clients = 100
94          self.sim = sim.Simulation(initial_clients)
95          self.sim.simulation()
96          rt = self.sim.statistic.calc_rt()
97          new_sim = sim.Simulation(initial_clients)
98          new_rt = new_sim.sim()
99          self.assertEqual(rt, new_rt)
100
101     def test_very_big_sim_shortcut(self):
102         """Simulació amb un nombre enooooorme de clients"""
103         initial_clients = 1000
104         self.sim = sim.Simulation(initial_clients)
105         self.sim.simulation()
106         rt = self.sim.statistic.calc_rt()
107         new_sim = sim.Simulation(initial_clients)
108         new_rt = new_sim.sim()
109         self.assertEqual(rt, new_rt)
110
111     def test_small_simulation(self):
112         """Simulació amb un sol client"""
113         initial_clients = 1
114         self.sim = sim.Simulation(initial_clients)
115         self.sim.simulation()
116         rt = self.sim.statistic.calc_rt()
117         self.assertEqual(len(self.sim.statistic), 12)
118         self.assertEqual(rt, 34.375171385648777)
119
120     def test_small_two(self):
121         """Simulació amb dos clients"""
122         initial_clients = 2
123         self.sim = sim.Simulation(initial_clients)
124         self.sim.simulation()
125         rt = self.sim.statistic.calc_rt()
```

```
126         self.assertEqual(len(self.sim.statistic), 23)
127         self.assertEqual(rt, 37.480450372637989)
128 
129     def test_small_two_seed_two(self):
130         """Simulació amb dos clients"""
131         initial_clients = 2
132         seed = 1
133         self.sim = sim.Simulation(initial_clients, seed)
134         self.sim.simulation()
135         rt = self.sim.statistic.calc_rt()
136         self.assertEqual(len(self.sim.statistic), 25)
137         self.assertEqual(rt, 28.93994509522874)
138 
139 
140 class TestSimulationWithReplica(unittest.TestCase):
141 
142     def test_one(self):
143         """Comparar que una sola rèplica és igual a una execució"""
144         initial_clients = 1
145         sim_repli = sim.SimulationWithReplica(n_replica=1, \
146                                               n_clients=initial_clients)
147         rt_repli = sim_repli.sim()
148         # I la comparació
149         self.sim = sim.Simulation(initial_clients)
150         self.sim.simulation()
151         rt = self.sim.statistic.calc_rt()
152         self.assertEqual(rt_repli, rt)
153 
154     def test_two(self):
155         repli = sim.SimulationWithReplica()
156         rt = repli.sim()
157         self.assertEqual(rt, 47.092064244653564)
158 
159 if __name__ == '__main__':
160     unittest.main()
```

test-sim-queue.py

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3 
4  from __future__ import print_function, division
5 
6  __version__ = "0.0.1"
7 
8  import unittest
9 
10 import sim
11 
```

```
12
13 class TestQueue(unittest.TestCase):
14
15     def setUp(self):
16         self.q = sim.Queue()
17
18     def test_simple(self):
19         self.q.add(1)
20         item = self.q.first
21         self.assertEqual(item, 1)
22
23
24     def test_second(self):
25         self.q.add(2)
26         self.q.add(1)
27         self.q.add(3)
28         self.q.add(5)
29         self.assertEqual(self.q.first, 1)
30         self.assertEqual(self.q.first, 2)
31         self.assertEqual(self.q.first, 3)
32         self.assertEqual(self.q.first, 5)
33
34     def test_third(self):
35         self.q.add(2)
36         self.q.add(1)
37         self.assertEqual(self.q.first, 1)
38         self.q.add(3)
39         self.assertEqual(self.q.first, 2)
40         self.q.add(5)
41         self.assertEqual(self.q.first, 3)
42         self.assertEqual(self.q.first, 5)
43
44 if __name__ == '__main__':
45     unittest.main()
```

test-suite.py

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4  from __future__ import print_function, division
5
6  __version__ = "0.0.1"
7
8  import unittest
9
10 import suite
11
12 @unittest.skip("Massa temps")
```

```
13 class TestSimSuite(unittest.TestCase):
14
15     def test_one(self):
16         s = suite.SimSuite([1])
17         s.simulation()
18         cadena = str(s)
19         self.assertEqual(cadena, '1\t44.95\n')
20
21     def test_two(self):
22         s = suite.SimSuite([1, 3])
23         s.simulation()
24         cadena = str(s)
25         self.assertEqual(cadena, '1\t44.95\n3\t45.73\n')
26
27 class TestSimSuiteWithoutReplica(unittest.TestCase):
28
29     def test_one(self):
30         s = suite.SimSuiteWithoutReplica([1])
31         s.simulation()
32         cadena = str(s)
33         self.assertEqual(cadena, '1\t34.38\n')
34
35     def test_two(self):
36         s = suite.SimSuiteWithoutReplica([1, 3])
37         s.simulation()
38         cadena = str(s)
39         self.assertEqual(cadena, '1\t34.38\n3\t42.25\n')
40
41
42
43 if __name__ == '__main__':
44     unittest.main()
```