Pol Garcia i Oto - pol.garcia.oto@est.fib.upc.edu
Cristian Dozo - cristian.dozo@est.fib.upc.edu
Probabilitat i Estadística - Quadrimestre de primavera 2012

UBUNTU VIRTUALITZAT EN DIFERENTS SO

0. Resum

Objectiu: Examinar el temps d'execució d'un mateix programa en una màquina virtual d'Ubuntu sobre dos sistemes operatius hostes, Mac OS X Lion i Microsoft Windows 7, comparar els resultats i intentar determinar si el rendiment varia segons l'entorn.

Mètode: Usant un mateix ordinador amb arrencada dual i una partició amb *OS X Lion* i una altra amb *Windows 7*, hem instal·lat un software de virtualització *VMWare*[1] per a virtualitzar la mateixa ISO d'*Ubuntu 11.10* i hem mesurat el temps d'execució d'un mateix programa realitzant 100 execucions en cada sistema i amb un *seed*[2] inicial que garantitza que les operacions que realitzaran els programes seran exactament les mateixes per un *seed* igual.

Resultats: Tenim un p-valor d'igualtat que val 1.185e-12 que es quasi 0, la mitjana de temps d'execució de *OS X Lion* és de 0.53165s que és inferior a la de *Windows 7* que és 0.56534s.

Conclusió: Les dues virtualitzacions d'*Ubuntu 11.10* en els dos sistemes operatius no són iguals, és a dir, rebutgem H_0 i per tant el *Mac OS X Lion* com a sistema hoste és millor per a virtualitzar *Ubuntu.*

1. Introducció

Per internet es poden trobar diverses comparatives sobre el rendiment d'aplicacions entre diferents sistemes operatius, però en canvi la cerca es fa difícil al intentar trobar en quin sistema hostejar una màquina virtual. Partim d'un problema real on un usuari té un ordinador *Mac* amb *OS X* i *Windows* però necessita fer servir *Ubuntu* i decideix que no el vol instal·lar nativament sino que el vol virtualitzar sobre un dels dos sistemes operatius que ja té, ha de decidir en quin dels dos sistemes l'hauria d'instal·lar per que un programa que ha d'executar vagi el més ràpid possible. Nosaltres volem avaluar en quin sistema hoste, un programa concret que fa servir moltes operacions i de molt diverses.

Objectiu

L'objectiu de l'estudi és determinar amb les mateixes condicions de hardware, el programa de virtualització i el sistema operatiu virtualitzat si el sistema operatiu hoste influeix en el temps d'un programa en *C++*.

---

1 En *Mac OS X Lion* hem usat *VMWare Fusion 4.0.1* i en *Microsoft Windows 7 VMWare Workstation 8.0*

2 llavor, valor numèric usat per inicialitzar un generador de números pseudoaleatoris, en aquest cas les pseudoaleatorietats del programa que seran influides per aquesta llavor aleatòria, amb llavors iguals els números pseudoaleatoris generats seran els mateixos en les execucions.

2. Material i mètodes

Primerament vam escollir el tipus de programa i el llenguatge que usariem, havia de ser un programa suficientment complex com per fer servir diversos recursos del sistema però també prou ràpid com per poder agafar una quantitat significativa de mostres amb facilitat, també havia de ser un programa on cada execució fes servir uns métodes algorítmics diferents per no basar la prova en un programa lineal, però alhora haviem d'asegurar-nos que els mateixos algoritmes eren utilitzats en els dos sistemes per igual. Així vam decidir que el més adient era fer servir un programa on a partir d'un *seed* (o número) inicial fes una série d'operacions que depenguessin del valor d'aquest i que sempre fossin les mateixes amb un número igual. Tenint en compte tots aquest requisits vam trobar adient fer servir el joc PacMan de l'assignatura d'EDA del quatrimestre de primavera del curs 2012[3] amb un jugador programat per nosaltres.

Per realitzar les excucions i recollir les dades vam programar un *script* en *bash*[4]. que llegís una seqüència de números d'un fitxer, executés el programa amb un seed inicial donat per cada un dels números i guardés en un altre fitxer el temps de cada execució. Els 100 números inicials van ser obtinguts a partir d'una web de generació de números aleatoris[5] per asegurar aleatorietat en les mostres.

Per a l'estudi de les dades obtingudes hem utilitzat el software estadístic R[6].

[3] El joc i el codi font principal es poden trobar a http://pacman.jutge.org/

[4] Adjuntat a l'annex

[5] http://www.random.org/

[6] The R Foundation for Statistical Computing http://www.r-project.org/

2.1 Proves d'hipòtesi

I. Prova de significació (bilateral):

a) $H_0: \mu_1 = \mu_2$ $H_1: \mu_1 \neq \mu_2$ $D = y_1 - y_2$ (Dades Windows - Dades Mac)

II. Premisses:

a) Normalitat (D -> N)

b) Mostra aparellada

III. Estadístic i distribució de referència:

$$\hat{t} = \frac{\overline{D} - \mu_0}{S_D / \sqrt{n}} \qquad \hat{t} \rightarrow t_{n-1}$$

IV. Regió Crítica:

n = 100 Punt crític amb risc 5%: 1.984217

```
R: qt(0.975, 99) = 1.984217
```

V. Construcció de l'interval de confiança per la diferència:

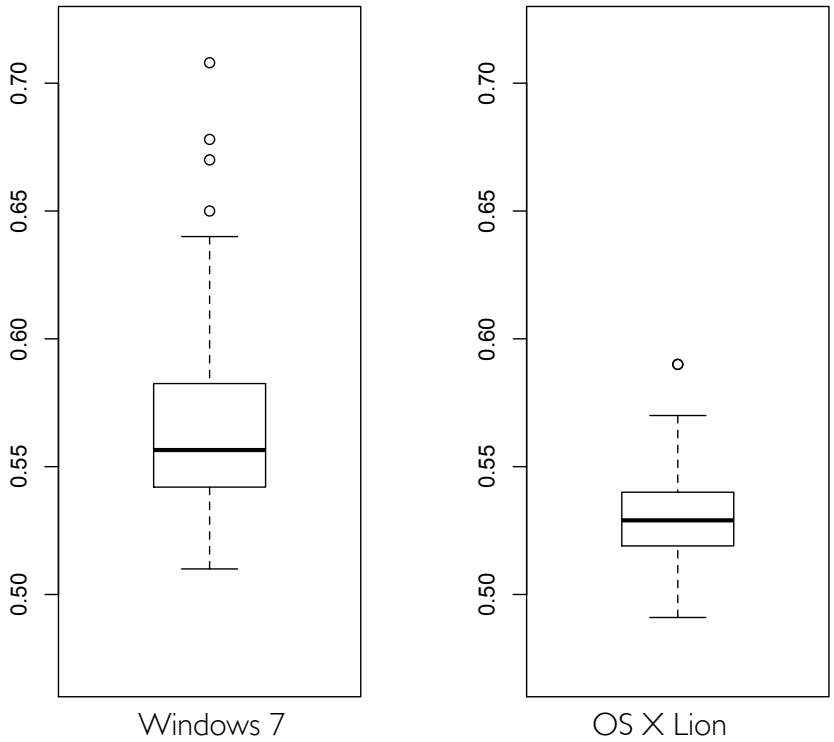
$$IC(\mu_D, 0'95) = \overline{d} \pm t_{n-1,1-\alpha/2} \sqrt{S_D^2 / n}$$

3. Resultats

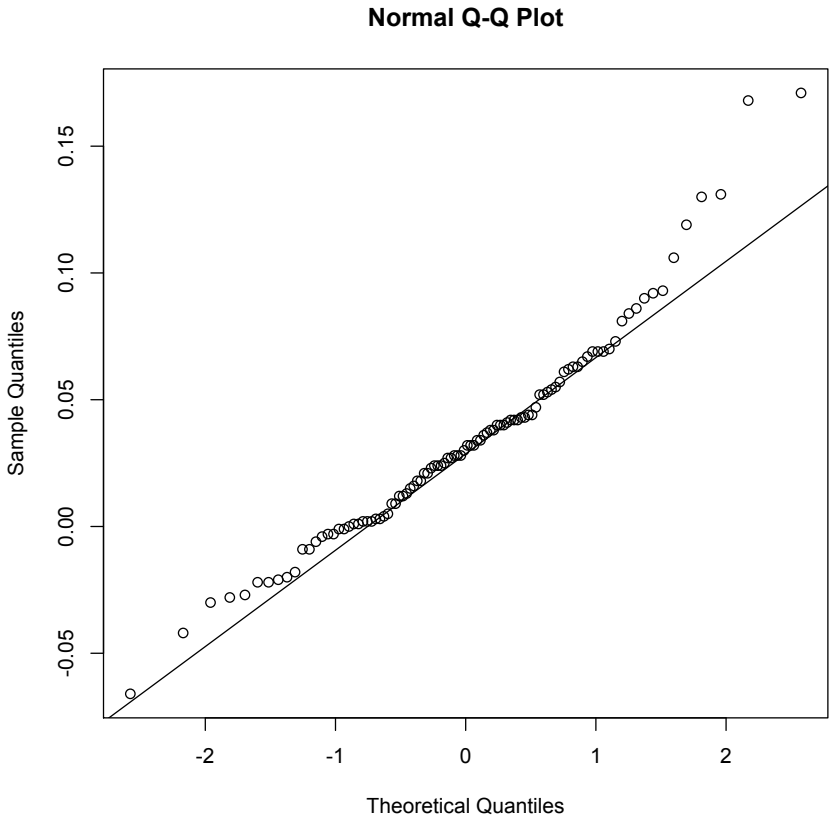
Podem observar que la mitjana de temps d'execució de *OS X Lion* és inferior a la de *Windows 7*, diferència que es pot veure clarament en el Gràfic 1 on es veu que la majoria dels valors de *OS X Lion* estan centrats al voltant de la mitjana amb uns extrems pròxims i en canvi els valors de *Windows 7* arriben a extrems més llunyans i més dispersos però sense arribar a valors tan inferiors com els mínims de *OS X Lion*. Gràcies al Gràfic 2 podem corroborar la nostra premisa de Normalitat en les variables i que per tant els nostres càlculs són vàlids, tenint una certesa del 95% de que els temps dels dos SO no són iguals.

	Mitjana (s)	**Desviació estàndard**
Windows 7	0.56534	0.03553039
OS X Lion	0.53165	0.01802264
Diferència	0.03369	0.04137381

Taula 1. Estadístics mostrals de les dades



Gràfic 1. Diferència de Boxplot
R: Boxplot(y1) vs Boxplot(y2)



Gráfic 2. Normalitat

```
R: qqnorm(y1-y2)
```

```
R: t.test(y1,y2, paired=TRUE)
```

t = 8.1428 p-valor = 1.185e-12 mitjana de les diferències: 0.03369

Interval del 95% de confiança [0.02548054, 0.04189946]

4. Discussió

4.1 Conclusió Principal:

Tal i com hem pogut comprobar el sistema operatiu hoste influeix en el rendiment del nostre programam en *C++*, i sembla que la millor opció és *Mac OS X Lion*. Per tant, si necesitesim fer servir aquest programa al nostre ordinador ho feriem utilizant *OS X Lion* com a sistema hoste.

4.2 Limitacions:

Degut a les limitacions típiques de les proves experimentals sumades als pocs recursos disponibles per fer un experiment que tingués en compte totes les variables que podrien afectar el rendiment, no sols d'una aplicació concreta sino de la majoria, només podem afirmar que en les condicions concretes de l'experiment i amb aplicacions, SO i hardware semblants el sistema operatiu hoste si afecta al rendiment del nostre programa. Del mateix mode només podem afirmar que el rendiment ha variat en aquest programa i no podem afirmar que això succeiria en tots els programes escrits en *C++*.

Tot i que em fet servir el software de virtualització de la mateixa empresa, la seva implementació ha de ser diferent per Windows com per Mac, per tant aquestes diferencies tambe podrien ser degudes a que un és mes eficient que l'altre, i no tingués res a veure amb el sistema hoste en si, sino amb la màquina virtual.

4.3 Extrapolació:

Tal com hem afirmat a l'apartat anterior hi ha una serie de limitacions que no permeten fer una extrapolació realment interesant, per tant nomes podem afirmar que el rendiment es diferent amb programes i en ordinadors semblants al de la prova.

4.4 Ampliació:

Per intentar fer un experiment menys limitat en quant a conclusions i tal com hem explicat al seu apartat hauriem de fer la prova amb un rang de programes, i ordinadors ben diferents i en una quantitat d'ells molt més gran.

També hauriem d'esbrinar si el software de virtualització es un factor determinant per el rendiment dels programes de la màquina virtual creada.

5. Annex

Script en Bash per executar automàticament les 100 execucions i mesurar el seu temps:

```
#!/bin/bash
MAXCOUNT=100
count=1
while [ "$count" -le $MAXCOUNT ]
    do
    read s // Llegim el seed
    export TIMEFORMAT="%R"
     time (./PacMan PL1 PL2 PL3 PL4 -s $s -i demo.gam -o partida.pac  2>/dev/
null)
    let "count += 1"
done
exit 0
```

I per executar-l'ho: ./script < seeds.txt 2> fitxer sortida.txt

On *script* es el nom de l'script, *seeds.txt* el fitxer amb els 100 seeds (nombres aleatoris) i el *fitxer sortida.txt* on vols que et guardi els resultats.