

Ingeniería de Servidores (2014-2015)
GRADO EN INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE GRANADA

Recopilación de preguntas opcionales

Jesús Ángel González Novez

13 de enero de 2015

Índice

1. Opcionales Práctica 1	3
1.1. Muestre (con capturas de pantalla) cómo ha comprobado que el RAID1 funciona.	3
1.2. ¿Qué relación hay entre los atajos de teclado de emacs y los de la consola bash? ¿y entre los de vi y las páginas del manual?	4
2. Opcionales Práctica 2	5
2.1. ¿Qué gestores utiliza OpenSuse (Pista: http://es.opensuse.org/Gestion_de_paquetes)	5
2.2. Fail2Ban. Instale el servicio y pruebe su funcionamiento.	5
2.3. Realice la instalación de MongoDB en alguna de sus máquinas virtuales. Cree una colección de documentos y haga una consulta sobre ellos. (http://docs.mongodb.org/manual/installation/)	5
2.4. Muestre un ejemplo de uso para awk	6
3. Opcionales Práctica 3	7
3.1. Indique qué comandos ha utilizado para realizarlo así como capturas de pantalla del proceso de reconstrucción del RAID.	7
3.2. Escriba un script en python y analice su comportamiento usando el profiler presentado.	7
3.3. Al igual que ha realizado el “profiling” con MySQL, realice lo mismo con MongoDB y compare los resultados (use la misma información y la misma consulta, hay traductores de consultas SQL a Mongo).	8
4. Opcionales Práctica 4	8
4.1. ¿Qué es Scala? Instale Gatling y pruebe los escenarios por defecto.	8
4.2. Ha sido comparado por la empresa Flood.io con Gatling obteniendo la conclusión de que ambos proporcionan tienen un comportamiento y capacidades similares (https://flood.io/blog/11-benchmarking-jmeter-and-gatling). Cuestión opcional 3: Lea el artículo y elabore un breve resumen.	9
5. Opcionales Práctica 5	9
5.1. Cuestión opcional 1: Realice lo mismo que en la cuestión 8 pero para otro servicio.	9

Índice de figuras

1.1. <code>sudo mdadm --manage --set-faulty /dev/md0 /dev/sdb1</code>	3
1.2. <code>sudo mdadm --detail /dev/md0</code>	3
1.3. <code>sudo mdadm /dev/md0 -r /dev/sdb1</code>	4
1.4. <code>sudo mdadm /dev/md0 -a /dev/sdb1</code>	4
1.5. <code>sudo mdadm --detail /dev/md0</code>	4

2.1. Creación de documentos en la terminal	5
2.2. Inserción de documentos en la terminal	6
2.3. Consultando la base de datos en la terminal	6
2.4. Buscar un documento en la base de datos con la terminal	6
5.1. Vista de los motores MySQL configurados.	10
5.2. Resultado del test usando MyISAM.	10
5.3. Resultado del test usando INNODB.	11

1. Opcionales Práctica 1

1.1. Muestre (con capturas de pantalla) cómo ha comprobado que el RAID1 funciona.

Para comprobar esto podemos hacer una simulación de fallo, retirada de disco en caliente, volvemos a poner el disco y comprobamos que efectivamente vuelven a sincronizarse. A continuación pongo los comandos utilizados y alguna captura de pantalla. Para todo esto podemos usar el comando mdadm ¹

Forzaremos un fallo de disco a propósito:

```
jesus@ubuntu:~$ sudo mdadm --manage --set-faulty /dev/md0 /dev/sdb1
[ 617.646889] md/raid1:md0: Disk failure on sdb1, disabling device.
[ 617.646889] md/raid1:md0: Operation continuing on 1 devices.
mdadm: set /dev/sdb1 faulty in /dev/md0
```

Figura 1.1: sudo mdadm -manage -set-faulty /dev/md0 /dev/sdb1

Consultamos el estado:

```
jesus@ubuntu:~$ sudo mdadm --detail /dev/md0
/dev/md0:
  Version : 1.2
  Creation Time : Wed Oct  8 16:04:07 2014
  Raid Level : raid1
  Array Size : 8382400 (7.99 GiB 8.58 GB)
  Used Dev Size : 8382400 (7.99 GiB 8.58 GB)
  Raid Devices : 2
  Total Devices : 2
  Persistence : Superblock is persistent

  Update Time : Mon Oct 20 18:20:50 2014
  State : clean, degraded
Active Devices : 1
Working Devices : 1
Failed Devices : 1
Spare Devices : 0

    Name : ubuntu:0 (local to host ubuntu)
    UUID : 168d06f5:900531e4:444e01fd:32e50a3d
    Events : 91

   Number  Major   Minor  RaidDevice State
    -----
     0         8        1         0   active sync  /dev/sda1
     1         0         0         1   removed
     2         8       17         -   faulty spare  /dev/sdb1
```

Figura 1.2: sudo mdadm -detail /dev/md0

¹<http://linux.die.net/man/8/mdadm>

Eliminamos el disco en caliente:

```
jesus@ubuntu:~$ sudo mdadm /dev/md0 -r /dev/sdb1
mdadm: hot removed /dev/sdb1 from /dev/md0
```

Figura 1.3: `sudo mdadm /dev/md0 -r /dev/sdb1`

Volvemos a añadir el disco:

```
jesus@ubuntu:~$ sudo mdadm /dev/md0 -a /dev/sdb1
mdadm: added /dev/sdb1
```

Figura 1.4: `sudo mdadm /dev/md0 -a /dev/sdb1`

Comprobamos como se sincronizan de nuevo:

```
jesus@ubuntu:~$ sudo mdadm --detail /dev/md0
/dev/md0:
  Version : 1.2
  Creation Time : Wed Oct  8 16:04:07 2014
    Raid Level : raid1
    Array Size : 8382400 (7.99 GiB 8.58 GB)
  Used Dev Size : 8382400 (7.99 GiB 8.58 GB)
    Raid Devices : 2
    Total Devices : 2
 Persistence : Superblock is persistent

 Update Time : Mon Oct 20 18:25:20 2014
   State : active, degraded, recovering
Active Devices : 1
Working Devices : 2
Failed Devices : 0
Spare Devices : 1

Rebuild Status : 7% complete

   Name : ubuntu:0 (local to host ubuntu)
  UUID : 168d06f5:900531e4:444e01fd:32e50a3d
  Events : 170

   Number Major Minor RaidDevice State
    0         8       1        0 active sync  /dev/sda1
    2         8      17        1 spare rebuilding /dev/sdb1
```

Figura 1.5: `sudo mdadm -detail /dev/md0`

1.2. ¿Qué relación hay entre los atajos de teclado de emacs y los de la consola bash? ¿y entre los de vi y las páginas del manual?

La "bash shell"² tiene dos modos de edición, "vi"³ y "emacs"⁴. Por defecto esta definido como "emacs". De ahí la relación entre atajos de teclado de ambos. De hecho "emacs" era un editor de texto antiguo y bash tiene los mismos atajos de teclado. Por otro lado "man"⁵ utiliza los mismo atajos para visualización de contenidos que usa "vi"

²<http://www.gnu.org/software/bash/>

³<http://www.cs.colostate.edu/helpdocs/vi.html>

⁴<http://www.gnu.org/software/emacs/>

⁵http://www.gnu.org/prep/standards/html_node/Man-Pages.html

2. Opcionales Práctica 2

2.1. ¿Qué gestores utiliza OpenSuse (Pista: http://es.opensuse.org/Gestion_de_paquetes)

OpenSuse nos ofrece el gestor gráfico de paquetes YaST⁶ y su alternativa para líneas de comandos Zypper⁷.

2.2. Fail2Ban. Instale el servicio y pruebe su funcionamiento.

Instalamos de la siguiente forma:

```
sudo apt-get install fail2ban
```

La configuración se guarda en `/etc/fail2ban/jail.conf`, este archivo puede ser modificado en actualizaciones lo que haremos será copiar este archivo a un archivo llamado `jail.local`:

```
sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local
```

Una vez copiada ya podemos editarla con por ejemplo `gedit`:

```
sudo gedit /etc/fail2ban/jail.local
```

2.3. Realice la instalación de MongoDB en alguna de sus máquinas virtuales. Cree una colección de documentos y haga una consulta sobre ellos. (<http://docs.mongodb.org/manual/installation/>)

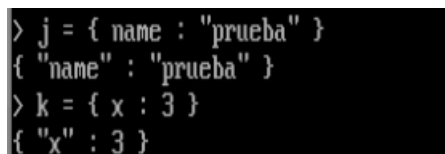
En Ubuntu Server:

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 7F0CEB10
echo 'deb http://downloads-distro.mongodb.org/repo/ubuntu-upstart
dist 10gen' | sudo tee /etc/apt/sources.list.d/mongodb.list
sudo apt-get update\\
sudo apt-get install mongodb-org
```

Ahora ya iniciamos un interprete mongo:

```
mongo
```

Creamos dos documentos:

A terminal window with a black background and white text. It shows the MongoDB shell prompt '>' followed by two commands to create documents. The first command is 'j = { name : "prueba" }' followed by '{ "name" : "prueba" }'. The second command is 'k = { x : 3 }' followed by '{ "x" : 3 }'.

```
> j = { name : "prueba" }
{ "name" : "prueba" }
> k = { x : 3 }
{ "x" : 3 }
```

Figura 2.1: Creación de documentos en la terminal

⁶https://en.opensuse.org/YaST_Software_Management

⁷<http://es.opensuse.org/Zypper>

Los insertamos en la colección testData:

```
> db.testData.insert( j )
WriteResult({ "nInserted" : 1 })
> db.testData.insert( k )
WriteResult({ "nInserted" : 1 })
>
```

Figura 2.2: Inserción de documentos en la terminal

Vemos las colecciones y sus documentos:

```
> show collections
system.indexes
testData
> db.testData.find()
{ "_id" : ObjectId("545a481154aca77e3443d671"), "name" : "prueba" }
{ "_id" : ObjectId("545a481d54aca77e3443d672"), "x" : 3 }
```

Figura 2.3: Consultando la base de datos en la terminal

Buscamos un documento:

```
> db.testData.find( { x : 3 } )
{ "_id" : ObjectId("545a481d54aca77e3443d672"), "x" : 3 }
```

Figura 2.4: Buscar un documento en la base de datos con la terminal

2.4. Muestre un ejemplo de uso para awk

El comando awk nos permite buscar en ficheros determinados patrones que le demos. Su uso es bastante sencillo:

```
awk '{accion/patron}' fichero
```

Por ejemplo supongamos que tenemos el siguiente fichero:

8	1
8	2
8	3
8	4
8	5
8	6
8	7
8	8

8	9
8	10

Si ejecutamos la orden:

```
awk '{print \$1*\$2}' fichero
```

Veremos por pantalla el resultado de ir multiplicando los elementos de la primera columna por la segunda columna, en este caso es la tabla del 8, por lo que la salida sería:

```
8,16,24,32,etc ...
```

Es un ejemplo tonto pero creo que se muestra un ejemplo de uso.

<http://www.es.hscripts.com/tutoriales/linux-commands/awk.html>

3. Opcionales Práctica 3

3.1. Indique qué comandos ha utilizado para realizarlo así como capturas de pantalla del proceso de reconstrucción del RAID.

Dado que en la práctica 2 realicé el mismo proceso, el propio profesor (Alberto G.) me dijo que simplemente lo dijese y que se daba por respondida esta cuestión.

3.2. Escriba un script en python y analice su comportamiento usando el profiler presentado.

El script diseñado es bastante sencillo, consiste en un bucle que incrementa la variable j de uno en uno durante 999999 iteraciones. Para hacer el profile de ese for usaremos cProfile como se muestra en el mismo script.

```
import cProfile
class prueba:
    def bucle(self):
        j = 1
        print("Valor inicial de j:"+str(j))
        for i in range(999999):
            j = j + 1
            print("Valor final de j:"+str(j))
p = prueba()
cProfile.run('p.bucle()')
```

El resultado que nos da por consola, aparte de la salida de los print, es la siguiente:

```
Valor inicial de j:1
Valor final de j:1000000
4 function calls in 0.158 seconds
```


Ordered by: standard name

ncalls	tottime	percall	cumtime	percall	filename:lineno(function)
1	0.000	0.000	0.158	0.158	<string>:1(<module>)
1	0.112	0.112	0.158	0.158	script1.py:3(bucle)
1	0.000	0.000	0.000	0.000	{method 'disable' of '_lsprof.Profiler' objects}
1	0.046	0.046	0.046	0.046	{range}

3.3. Al igual que ha realizado el “profiling” con MySQL, realice lo mismo con MongoDB y compare los resultados (use la misma información y la misma consulta, hay traductores de consultas SQL a Mongo).

Bueno primero de todo crearemos una base datos llamada 'datos_usuario' y una colección llamada 'usuarios'.

```
>use datos_usuario
switched to db datos_usuario
>db.usuarios.save({name:'Juan',mail:'jl@gmail.com'})
```

Visitando <http://www.querymongo.com> traducimos una sentencia 'select * from usuarios'.

```
>db.usuarios.find();
{ "_id" : ObjectId("5475f06151ce882e732e6d4f"), "name" : "Juan",
  "mail" : "jl@gmail.com" }
```

En mongodb debemos indicar el nivel de profiling con valor 1 para consultas lentas y valor 2 para todas, como la nuestra será una consulta rápida usaremos el 2.

```
>db.setProfilingLevel(2)
```

Ahora realizaremos la consulta usando profiling.

```
>db.system.profile.find()
```

Todo lo referente a mongodb se puede encontrar en su documentación oficial⁸ aunque para desarrollar este ejercicio me he bastado de lo aprendido en una asignatura de 4º curso llamada DAI.

4. Opcionales Práctica 4

4.1. ¿Qué es Scala? Instale Gatling y pruebe los escenarios por defecto.

Scala es un acrónimo de 'Scalable Language'. Esto viene a ser un lenguaje que se adapta a nuestras necesidades de escalabilidad. Entre sus características principales podríamos

⁸<http://www.mongodb.org/>

destacar:

Orientado a objetos, todo en Scala es un objeto, al estilo de Java (de hecho correo en máquina Java y es compatible al 100 % con aplicaciones Java). Funcional, era de esperar que podamos tener funciones de todo tipo en un lenguaje orientado a objetos. Tipado estrictamente, al más puro estilo C. En general es similar a Java salvando algunos detalles. Para instalar Gatling en Ubuntu basta con tirar de repositorios oficiales, es decir:

```
sudo apt-get install gatling
```

Gatling sirve básicamente para dar servicio HTTP y FTP, para más información sobre su uso podemos consultar 'man gatling'

- 4.2. Ha sido comparado por la empresa Flood.io con Gatling obteniendo la conclusión de que ambos proporcionan tienen un comportamiento y capacidades similares (<https://flood.io/blog/11-benchmarking-jmeter-and-gatling>).**
Cuestión opcional 3: Lea el artículo y elabore un breve resumen.

Se van a medir dos entornos distintos, ambos usando nginx. Usará flood para generar carga. Dicha carga vemos que se compone de 20 % instrucciones fetch, 40 % saltos condicionales, 30 % fetch sin caché y el 10 % restante de instrucciones posting. Simulamos que hubiese 10000 usuarios simultáneos con 30000 pet/min, tras pasarle el gatling vemos que nos da 1788s, con jmeter 1625s y con jmeter 2.10 nos da 1698s.

Podemos ver resultados de red bastante similares, sin embargo uno de los entornos requirió bastante más CPU que el otro para obtener dichos resultados. Por tanto, y personalmente, yo elegiría el que menos CPU ha utilizado, claro que habría que ver otros aspectos como si me van a cobrar mucho más por éste y cosas así.

5. Opcionales Práctica 5

- 5.1. Cuestión opcional 1: Realice lo mismo que en la cuestión 8 pero para otro servicio.**

En este caso vamos a suponer que tenemos un servidor humilde en memoria(como es el caso) y queremos optimizar el uso de RAM de todos los servicios posibles, en este caso en concreto vamos a intentar mejorar MySQL.

```
mysql> use information_schema ;  
mysql> select * from ENGINES;
```

ENGINE	SUPPORT	COMMENT
CSV	YES	CSV storage engine
MRG_MYISAM	YES	Collection of identical MyISAM ta
MyISAM	YES	MyISAM storage engine
BLACKHOLE	YES	/dev/null storage engine (anythin
MEMORY	YES	Hash based, stored in memory, use
InnoDB	DEFAULT	Supports transactions, row-level
ARCHIVE	YES	Archive storage engine
PERFORMANCE_SCHEMA	YES	Performance Schema
FEDERATED	NO	Federated MySQL storage engine

9 rows in set (0.00 sec)

Figura 5.1: Vista de los motores MySQL configurados.

Voy a comparar el rendimiento usando un benchmark sencillo para los motores de almacenamiento INNODB y MyISAM. Para ello usaré el comando **mysqlslap**⁹:

```
mysqlslap --user=root --auto-generate-sql -vv --concurrency=100
--number-of-queries=10000 --engine=innodb -p
```

```
mysqlslap --user=root --auto-generate-sql -vv --concurrency=100
--number-of-queries=10000 --engine=myisam -p
```

Este test lo que hace es algo similar al que vimos con el comando **ab** para Apache, realiza peticiones concurrentemente al servidor MySQL. Para este estudio realizaremos 10000 peticiones con una concurrencia de 100 conexiones.

```
jesus@jesuspc:~$ mysqlslap --user=root --auto-generate-sql -vv --concurr
y=100 --number-of-queries=10000 --engine=myisam -p
Building Create Statements for Auto
Building Query Statements for Auto
Parsing engines to use.
Enter password:
Starting Concurrency Test
Loading Pre-data
Generating primary key list
Generating stats
Benchmark
Running for engine myisam
Average number of seconds to run all queries: 15.209 seconds
Minimum number of seconds to run all queries: 15.209 seconds
Maximum number of seconds to run all queries: 15.209 seconds
Number of clients running queries: 100
Average number of queries per client: 100
```

Figura 5.2: Resultado del test usando MyISAM.

⁹man mysqlslap

```

jesus@jesuspc:~$ mysqlslap --user=root --auto-generate-sql -vv --concurrency=100 --number-of-queries=10000 --engine=innodb -p
Building Create Statements for Auto
Building Query Statements for Auto
Parsing engines to use.
Enter password:
Starting Concurrency Test
Loading Pre-data
Generating primary key list
Generating stats
Benchmark
    Running for engine innodb
    Average number of seconds to run all queries: 21.005 seconds
    Minimum number of seconds to run all queries: 21.005 seconds
    Maximum number of seconds to run all queries: 21.005 seconds
    Number of clients running queries: 100
    Average number of queries per client: 100

```

Figura 5.3: Resultado del test usando INNODB.

Como vemos la diferencia es de unos 4.8s, un tiempo considerable. Por tanto he decidido cambiar de motor de almacenamiento, ya que el que viene por defecto con MySQL es INNODB, voy a cambiarlo a MyISAM ya que mi servidor no requiere de las funcionalidades adicionales de INNODB y puede funcionar con MyISAM.

Para ello en el fichero que tenemos en `/etc/mysql/my.cnf` añadiremos las siguientes líneas:

```

default-storage-engine=myisam
skip_innodb

```

Es también destacable que el motor MyISAM consume bastante menos memoria RAM que el motor INNODB dado que tiene menos funcionalidades.