Evaluación de servidores de Bases de Datos MySQL en Linux

Rafael Galán Villén, Antonio Gómez Giménez, Rafael Hormigo Cabello

Número de horas: 12

December 2019

Índice general

1. Definición del Sistema **2**
1.1. Objetivos y definición del sistema 2
1.2. Servicios y sus posibles resultados 2
1.3. Métricas 3
1.4. Parámetros 3

2. Evaluación del Sistema **4**
2.1. Técnicas de evaluación 4
2.2. Carga de trabajo 4
2.3. Diseño de Experimentos 4
2.4. Análisis de los resultados 7

3. Conclusiones y discusión **12**
3.1. Cuestiones 12
3.2. Conclusion 12

Capítulo 1

Definición del Sistema

1.1. Objetivos y definición del sistema

Nuestro objetivo es trabajar sobre una base de datos dada, para ello usaremos una máquina virtual, en nuestro caso trabajaremos en linux, concretamente hemos elegido *Ubuntu 18.04*. Para ello necesitamos realizar distintas inserciones, modificaciones y borrados en distintas tablas de nuestra base de datos.

Para poder hacer lo mencionado anteriormente dedicaremos 6GB de RAM, un CPU y 20GB de almacenamiento a nuestra máquina virtual sobre la cuál instalaremos *LAMPP* y usaremos un programa en *lenguaje c* para la realización de nuestro *benchmark* que será el encargado de hacer la inserciones, modificaciones y borrados, además de *bash*. Con todo esto podremos obtener los resultados de las mediciones, para posteriormente hacer las gráficas y debatir sobre dichos resultados, logrando así la finalidad deseada.

1.2. Servicios y sus posibles resultados

Nuestro sistema nos ofrece los siguientes sevicios, *mysql* que nos permite interaccionar con la base de datos, *top e iotop*, que los usaremos para extraer los datos de las mediciones y *bash*.

Por otro lado a la hora de realizar cada acción nos podemos encontrar con diferentes resultados, los cuales definimos a continuación:
Tanto para el caso de inserción como en el de modificación y como el de borrado podemos concluir que un resultado es válido cuando se realiza cualquiera de las acciones mecionadas en la base de datos, que ese resultado sería invalido cuando no se realice dicha acción en la base de datos y si por algún motivo la base de datos dejase de funcionar nos encontraríamos con que no obtendríamos ningún resultado.

1.3. Métricas

A continuación detallaremos qué datos vamos a medir para la realización de este trabajo, mediremos el % de uso CPU ya que para el uso de una base de datos es un factor de suma relevancia, el % de memoria y la entrada/salida la cuál mediremos en kb/s. Cuando dichas mediciones sean realizadas podremos detallar cual es el factor que nos limita, es decir, dicho de forma mas informal, cuál será el factor que nos haga de *cuello de botella*.

1.4. Parámetros

A la hora de desarrollar nuestro trabajo nos encontramos, principalmente, con dos factores a destacar

Lo primero que tenemos es que, al haber elegido un sistema operativo no dedicado a servidor hay varios procesos que se ejecutan en segundo plano, consumiendo así memoria que se podría destinar a la base de datos.

Otro parámetro a tener en cuenta para nuestro proyecto es que nos encontramos con que al dedicar un solo nucleo no puede haber multiprocesamiento

Capítulo 2

Evaluación del Sistema

2.1. Técnicas de evaluación

Usaremos un script que nos controlará el número de repeticiones mientras que nuestro programa en *lenguaje c* controlará que se hagan el número correcto de acciones para cada vez. Nos centraremos en el estudio del demonio *mysqld* que es el demonio encargado de hacer las operaciones sobre las bases de datos.

2.2. Carga de trabajo

Para medir la carga de trabajo trabajaremos en insertar, modificar y borrar datos en la tabla de *basketball_defensive_Stat*.

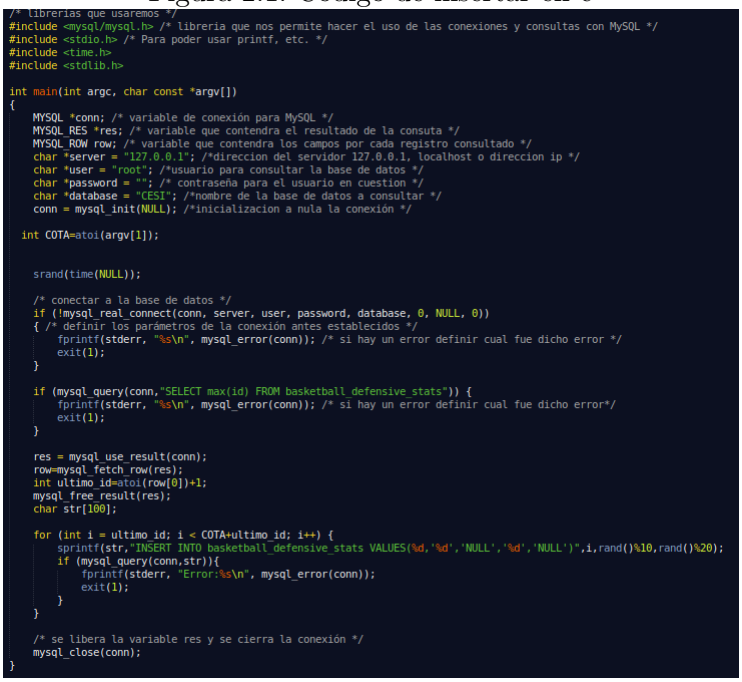
Esta carga consitirá en 20 repeticiones para cada una de las 2 distintas masas de datos que vamos a usar, 100000 y 200000.

2.3. Diseño de Experimentos

Nuestro diseño de del experimento consiste en que básicamente hemos codificado 3 ficheros en c usando la librería mysql para poder realizar la inserción(2.1), la modificación(2.2) y el borrado(2.3).

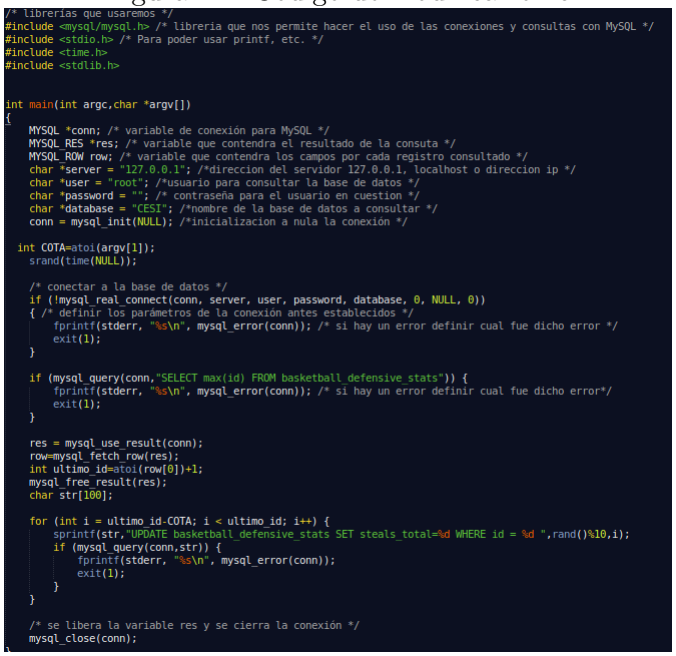
Luego esos ficheros son llamados desde un script(2.4) que los ejecuta y al mismo tiempo consulta con los comandos *top* e *iotop* la cpu la memoria y las e/s.

Figura 2.1: Código de insertar en c



```c
/* librerias que usaremos */
#include <mysql/mysql.h> /* libreria que nos permite hacer el uso de las conexiones y consultas con MySQL */
#include <stdio.h> /* Para poder usar printf, etc. */
#include <time.h>
#include <stdlib.h>

int main(int argc, char const *argv[])
{
    MYSQL *conn; /* variable de conexión para MySQL */
    MYSQL_RES *res; /* variable que contendra el resultado de la consuta */
    MYSQL_ROW row; /* variable que contendra los campos por cada registro consultado */
    char *server = "127.0.0.1"; /*direccion del servidor 127.0.0.1, localhost o direccion ip */
    char *user = "root"; /*usuario para consultar la base de datos */
    char *password = ""; /* contraseña para el usuario en cuestion */
    char *database = "CESI"; /*nombre de la base de datos a consultar */
    conn = mysql_init(NULL); /*inicializacion a nula la conexión */

  int COTA=atoi(argv[1]);


    srand(time(NULL));

    /* conectar a la base de datos */
    if (!mysql_real_connect(conn, server, user, password, database, 0, NULL, 0))
    { /* definir los parámetros de la conexión antes establecidos */
        fprintf(stderr, "%s\n", mysql_error(conn)); /* si hay un error definir cual fue dicho error */
        exit(1);
    }

    if (mysql_query(conn,"SELECT max(id) FROM basketball_defensive_stats")) {
        fprintf(stderr, "%s\n", mysql_error(conn)); /* si hay un error definir cual fue dicho error*/
        exit(1);
    }

    res = mysql_use_result(conn);
    row=mysql_fetch_row(res);
    int ultimo_id=atoi(row[0])+1;
    mysql_free_result(res);
    char str[100];

    for (int i = ultimo_id; i < COTA+ultimo_id; i++) {
        sprintf(str,"INSERT INTO basketball_defensive_stats VALUES(%d,'%d','NULL','%d','NULL')",i,rand()%10,rand()%20);
        if (mysql_query(conn,str)){
            fprintf(stderr, "Error:%s\n", mysql_error(conn));
            exit(1);
        }
    }

    /* se libera la variable res y se cierra la conexión */
    mysql_close(conn);
}
```

Figura 2.2: Código de modificar en c



```c
/* librerias que usaremos */
#include <mysql/mysql.h> /* libreria que nos permite hacer el uso de las conexiones y consultas con MySQL */
#include <stdio.h> /* Para poder usar printf, etc. */
#include <time.h>
#include <stdlib.h>


int main(int argc,char *argv[])
{
    MYSQL *conn; /* variable de conexión para MySQL */
    MYSQL_RES *res; /* variable que contendra el resultado de la consuta */
    MYSQL_ROW row; /* variable que contendra los campos por cada registro consultado */
    char *server = "127.0.0.1"; /*direccion del servidor 127.0.0.1, localhost o direccion ip */
    char *user = "root"; /*usuario para consultar la base de datos */
    char *password = ""; /* contraseña para el usuario en cuestion */
    char *database = "CESI"; /*nombre de la base de datos a consultar */
    conn = mysql_init(NULL); /*inicializacion a nula la conexión */

  int COTA=atoi(argv[1]);
    srand(time(NULL));

    /* conectar a la base de datos */
    if (!mysql_real_connect(conn, server, user, password, database, 0, NULL, 0))
    { /* definir los parámetros de la conexión antes establecidos */
        fprintf(stderr, "%s\n", mysql_error(conn)); /* si hay un error definir cual fue dicho error */
        exit(1);
    }

    if (mysql_query(conn,"SELECT max(id) FROM basketball_defensive_stats")) {
        fprintf(stderr, "%s\n", mysql_error(conn)); /* si hay un error definir cual fue dicho error*/
        exit(1);
    }

    res = mysql_use_result(conn);
    row=mysql_fetch_row(res);
    int ultimo_id=atoi(row[0])+1;
    mysql_free_result(res);
    char str[100];

    for (int i = ultimo_id-COTA; i < ultimo_id; i++) {
        sprintf(str,"UPDATE basketball_defensive_stats SET steals_total=%d WHERE id = %d ",rand()%10,i);
        if (mysql_query(conn,str)) {
            fprintf(stderr, "%s\n", mysql_error(conn));
            exit(1);
        }
    }

    /* se libera la variable res y se cierra la conexión */
    mysql_close(conn);
}
```

Figura 2.3: Código de borrar en c

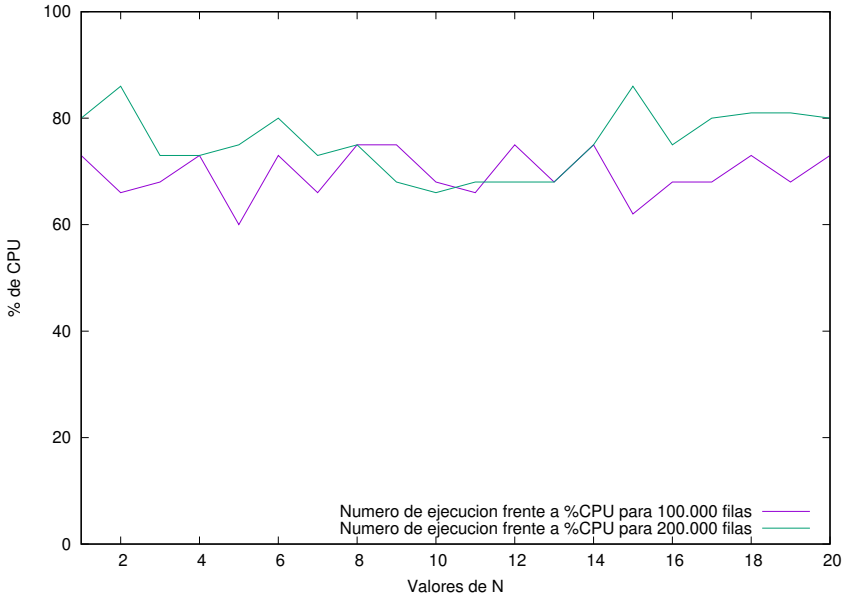
```c
/* librerias que usaremos */
#include <mysql/mysql.h> /* libreria que nos permite hacer el uso de las conexiones y consultas con MySQL */
#include <stdio.h> /* Para poder usar printf, etc. */
#include <time.h>
#include <stdlib.h>


int main(int argc,char *argv[])
{
    MYSQL *conn; /* variable de conexión para MySQL */
    MYSQL_RES *res; /* variable que contendra el resultado de la consuta */
    MYSQL_ROW row; /* variable que contendra los campos por cada registro consultado */
    char *server = "127.0.0.1"; /*direccion del servidor 127.0.0.1, localhost o direccion ip */
    char *user = "root"; /*usuario para consultar la base de datos */
    char *password = ""; /* contraseña para el usuario en cuestion */
    char *database = "CESI"; /*nombre de la base de datos a consultar */
    conn = mysql_init(NULL); /*inicializacion a nula la conexión */

  int COTA=atoi(argv[1]);
    srand(time(NULL));

    /* conectar a la base de datos */
    if (!mysql_real_connect(conn, server, user, password, database, 0, NULL, 0))
    { /* definir los parámetros de la conexión antes establecidos */
        fprintf(stderr, "%s\n", mysql_error(conn)); /* si hay un error definir cual fue dicho error */
        exit(1);
    }

    if (mysql_query(conn,"SELECT max(id) FROM basketball_defensive_stats")) {
        fprintf(stderr, "%s\n", mysql_error(conn)); /* si hay un error definir cual fue dicho error*/
        exit(1);
    }

    res = mysql_use_result(conn);
    row=mysql_fetch_row(res);
    int ultimo_id=atoi(row[0])+1;
    mysql_free_result(res);
    char str[100];

    sprintf(str,"DELETE FROM basketball_defensive_stats WHERE id>%d",ultimo_id-1-COTA);
    if (mysql_query(conn,str)) {
        fprintf(stderr, "%s\n", mysql_error(conn));
        exit(1);
    }

    /* se libera la variable res y se cierra la conexión */
    mysql_close(conn);
```

Figura 2.4:

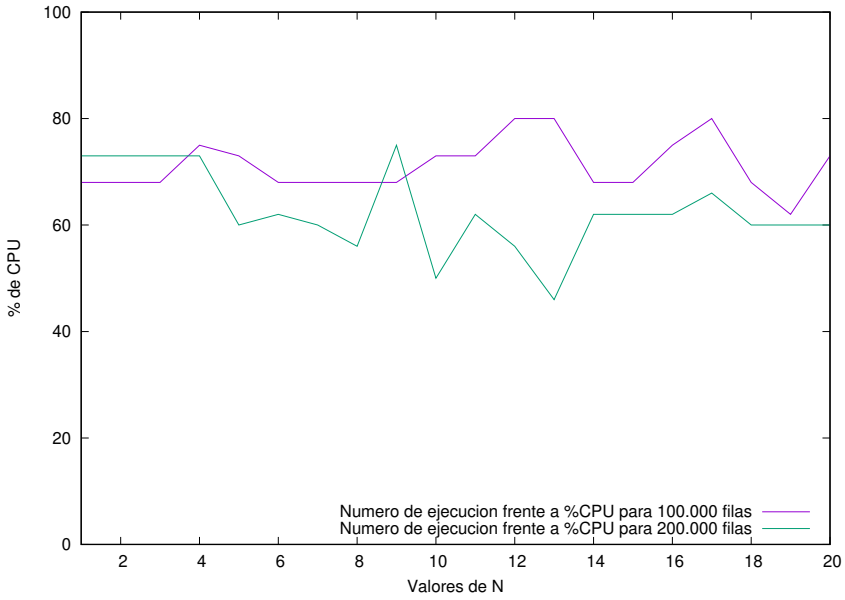
```bash
#!/bin/bash
pruebas=100000;
cd /home/cesi/Escritorio/
for (( pruebas = 100000; pruebas <500000 ; pruebas=pruebas+100000 )); do
  for (( i = 0; i < 20; i++ )); do
    echo "Insertar para $pruebas elementos"
    echo "Insertar para $pruebas elementos" >> insertar.txt
    ./insertar $pruebas &
    pidio=$(pidof mysqld)
    echo $(top -b -n 1 -p $pidio | sed -rn "$ s/.*[SIZR][ ]*([0-9]*\,[0-9]*)[ ]+([0-9]*\,[0-9]*).*$/\1 \2/p" >> insertar.txt)
    echo $(iotop -o -b -n 1 | grep mysqld | sed -rn "1 s/.* ([0-9]*\.[0-9]*\ [BK]\/s)[ ]+([0-9]*\.[0-9]* [BK]\/s).*$/\2/p" >> insertar.txt)
    pid=$(pidof insertar)
    while [[ $(ps | grep -c $pid) -ne 0 ]]; do
      x=0;
    done
    echo "Modificar para $pruebas elementos"
    echo "Modificar para $pruebas elementos" >> modificar.txt
    ./modificar $pruebas &
    pidio=$(pidof mysqld)
    echo $(top -b -n 1 -p $pidio | sed -rn "$ s/.*[SIZR][ ]*([0-9]*\,[0-9]*)[ ]+([0-9]*\,[0-9]*).*$/\1 \2/p" >> modificar.txt)
    echo $(iotop -o -b -n 1 | grep mysqld | sed -rn "1 s/.* ([0-9]*\.[0-9]*\ [BK]\/s)[ ]+([0-9]*\.[0-9]* [BK]\/s).*$/\2/p" >> modificar.txt)
    pid=$(pidof modificar)
    while [[ $(ps | grep -c $pid) -ne 0 ]]; do
      x=0;
    done
    echo "Borrar para $pruebas elementos"
    echo "Borrar para $pruebas elementos" >> borrar.txt
    ./borrar $pruebas &
    pidio=$(pidof mysqld)
    echo $(top -b -n 1 -p $pidio | sed -rn "$ s/.*[SIZR][ ]*([0-9]*\,[0-9]*)[ ]+([0-9]*\,[0-9]*).*$/\1 \2/p" >> borrar.txt)
    echo $(iotop -o -b -n 1 | grep mysqld | sed -rn "1 s/.* ([0-9]*\.[0-9]*\ [BK]\/s)[ ]+([0-9]*\.[0-9]* [BK]\/s).*$/\2/p" >> borrar.txt)
    pid=$(pidof borrar)
    while [[ $(ps | grep -c $pid) -ne 0 ]]; do
      x=0;
    done
  done
done
```

2.4. Análisis de los resultados

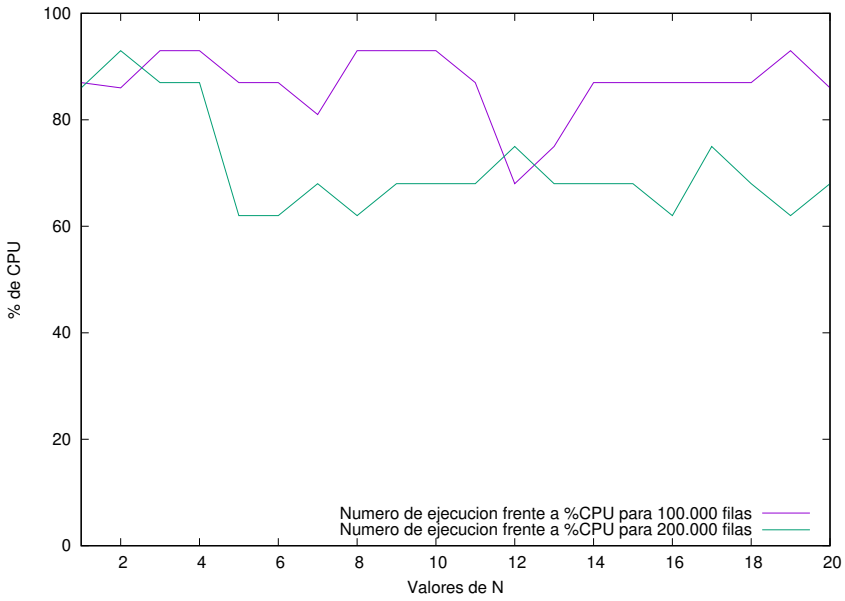
A continuación veremos los resultados obtenidos con nuestro experimento, empezaremos viendo las gráficas obtenidas para la medición del % de CPU
Primero veremos los resultados obtenidos para la inserción:



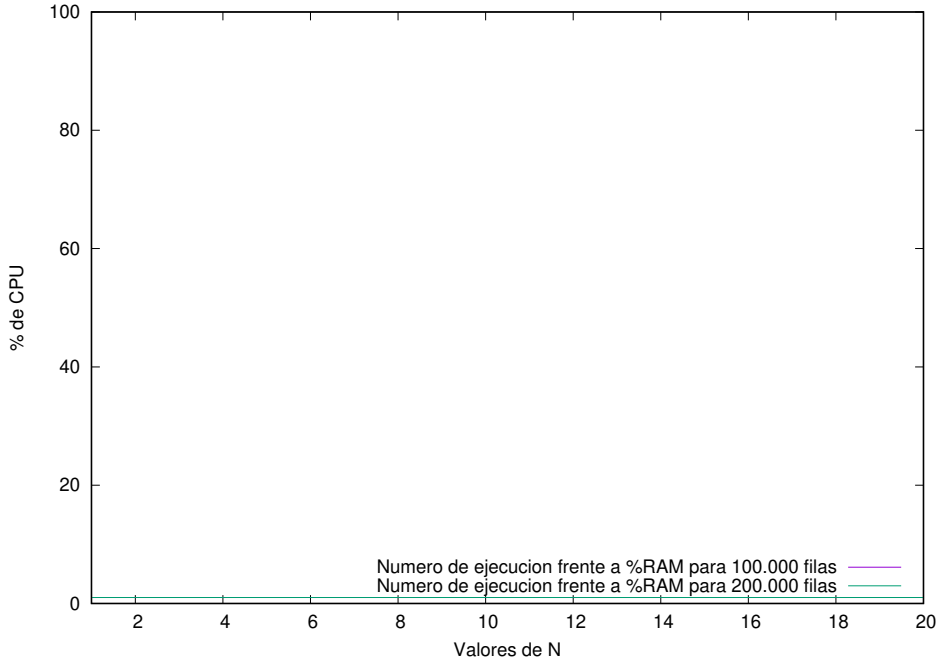
Luego veremos los resultados para la modificación:



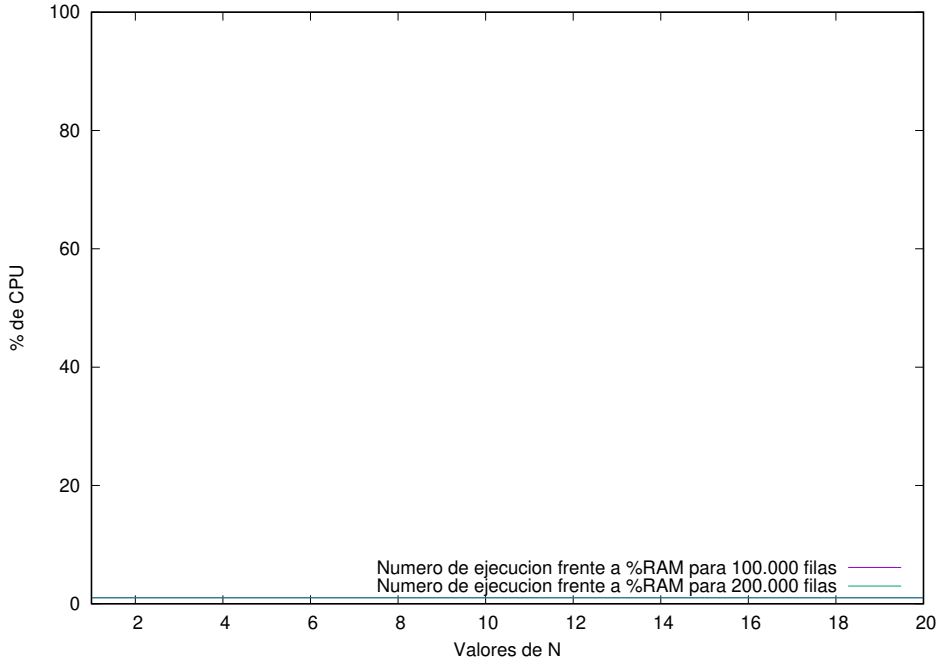
Y por último para el borrado:



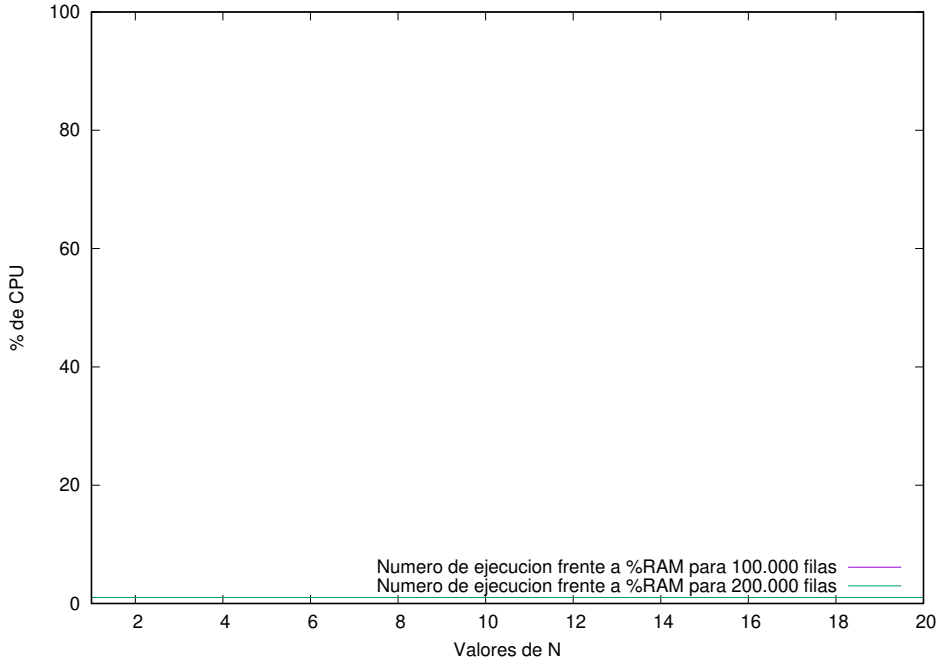
Por lo tanto podemos concluir que la acción que más % de CPU consume es el borrado, mientras que la que menos es la de la modificación, esto se debe a que es más fácil cambiar solo algunos datos.
Posteriormente tenemos las gráficas donde medimos el % de RAM en uso
Primero de la inserción:



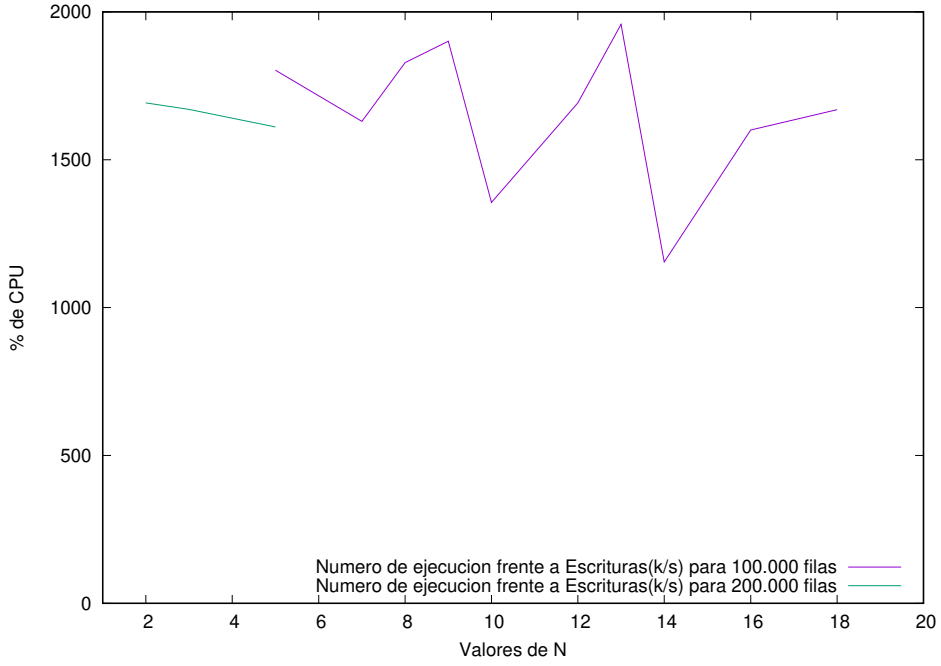
Luego de la modificación:



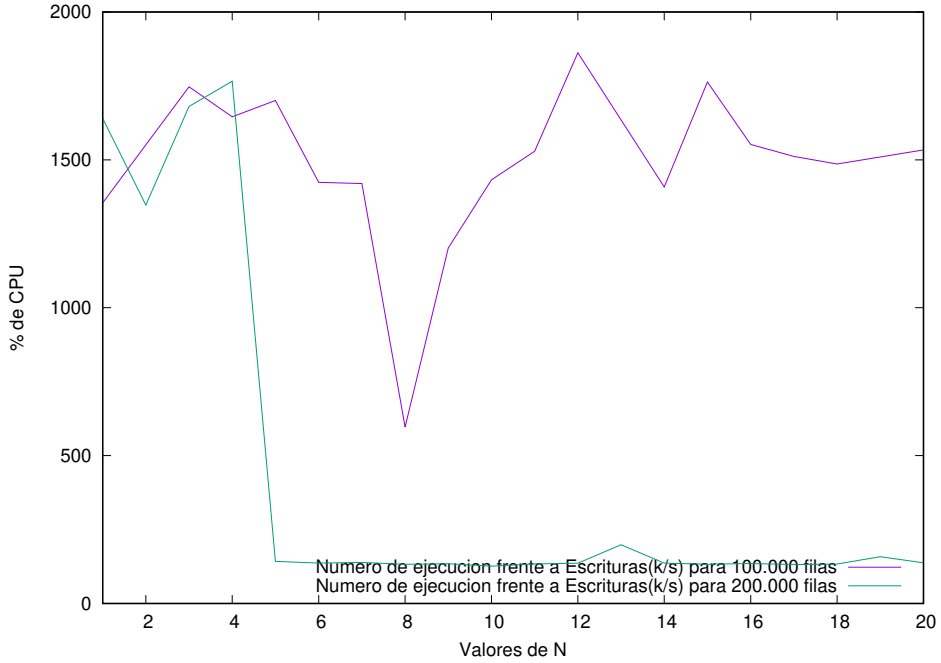
Y por último del borrado:



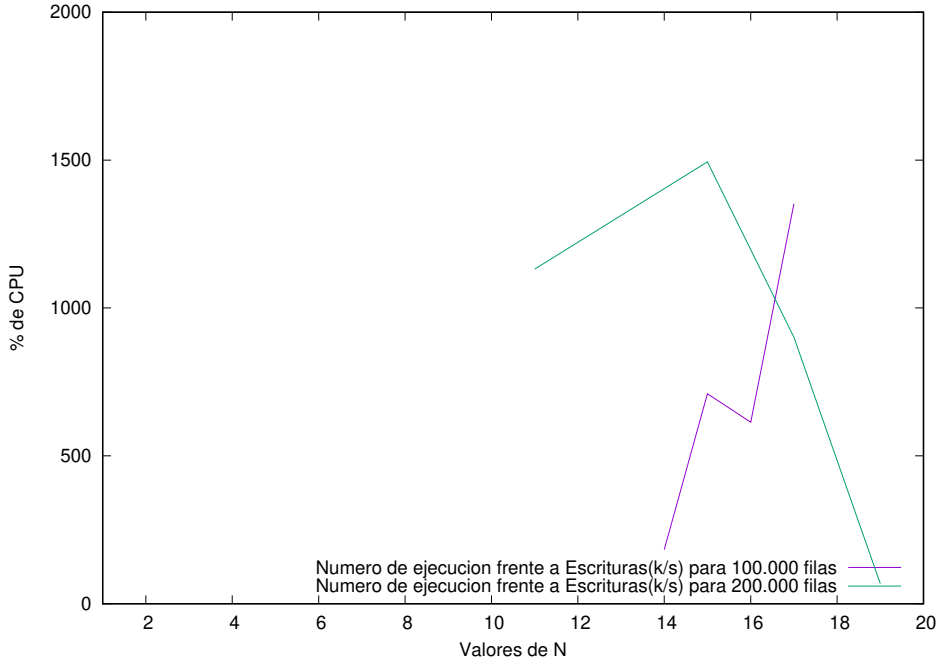
Y finalmente tenemos lás gráficas para e/s, que aquí ha habido varios resultados los cuáles nuestra máquina ha sido incapaz de medir
Primero la inserción:



Luego la modificación



Y por último el borrado:



Capítulo 3

Conclusiones y discusión

3.1. Cuestiones

Como conclusión, ¿qué cuestiones más relevantes te plantea el trabajo? Al menos 3 cuestiones junto con su respuesta.

3.2. Conclusion

ras realizar nuestro script con su respectivo Berchmark y haber extraído los datos de prueba, hemos realizado las gráficas con el servicio gnuplot, de tal forma que hemos llegado a las siguientes conclusiones globales.
-Lenguaje Usado. Tras realizar nuestro Berchmack con el lenguaje C y acompañandolo con un SCRIPT de bash hemos llegado a la conclusión de que que tanto C como bash no consumían recursos del sistema en comparación con el proceso de MySQL que se encontraba en esos momentos en ejecución.
-Lectura del Disco. El Berchmack realiza operaciones sobre la Base de Datos insertando, modificando y borrando filas, de esta forma, nos hemos dado cuenta que nuestro Berchmack no realiza operaciones de Lectura, es decir, solo realiza operaciones de Escrituras que vienen dadas por las operaciones anteriormente mencionadas. Si hubieramos realizado una operación de consulta si se hubiera podido medir la lectura sobre el disco, pero hemos considerado que con la escritura era suficiente.
-Número de Pruebas. Hemos realizado pruebas para los valores de 100.000 filas y 200.000 filas para las operaciones de inserción, modificación y borrado, siendo independientes entres sí. 100.000 filas es un número relativamente alto y tras realizar las pruebas hemos detectado que hay una diferencia poco notable al comparar estos valores entre sí ya que en los dos caso ocurren los siguientes casos: 1.La CPU es el cuello de botella, es decir, ya sea con 100.000 filas o con 200.000 filas llega a un punto donde se enlentece todo el proceso por culpa de la CPU. El resto de valores, como

por ejemplo la RAM, permanecían constantes mientras que la CPU tenía un valor cercano a 1002.El borrado es absurdamente rápido, hasta el punto de que casi era imperceptible el hecho de extraer la información del proceso ya que incluso para 1.000.000 de elementos tardaba escasos segundo, dando a entender que la escritura sobre el disco no es un problema. 3.Tras comparar las tres pruebas con sus respectivas 20 ejecuciones para cada una podemos encontrarnos que insección y modificar son similares a la hora de uso de CPU, siendo ligeramente superior en inserción, mientras que borrado es prácticamente imperceptible. A la hora de comparar las métricas nos damos cuenta que la que toma un valor considerable es CPU y es por ello donde nos vamos a centrar.

En resumen, por todo lo explicado anteriormente, debemos centrarnos en la CPU ya que es el cuello de botella de nuestro sistema y tiene sentido, ya que nuestra máquina virtual solo dispone de un núcleo por consiguiente no puede realizar computación en paralelo, teniendo así que realizar el berchmark mientras realiza el resto de procesos del propio sistema. Por lo tanto si deseamos realizar una mejora de nuestro sistema, ya sea porque se vayan a realizar de forma constante un flujo de operaciones del nivel de la carga que hemos administrado a la hora de realizar el Berchmark, tendremos que mejorar la CPU de nuestro sistema comprando una CPU con mayor frecuencia o con más núcleos para que pueda realizar operaciones en paralelo y mejorar la eficiencia. Si se mejorara enormemente la CPU, el cuello de botella probablemente pasaría a ser las escrituras en disco ya que el uso de la RAM era constante para todas las operaciones, casi imperceptible. En el caso de ocurrir el caso anterior sería necesario comprar un disco que permita escrituras más rápidas.

Por último dejar claro que por simplemente comprar mejor equipo tu ordenador no va a ser el mejor, siempre es necesario observar el objetivo del mismo(la carga que va a soportar), y en el caso de que no satisfaga el objetivo siempre solucionar los cuellos de botella por que sino seguirá funcionando igual aunque mejores el resto de elementos.

Bibliografía