Benchmarks

Memoria de la Practica 5

David Castro Salazar

10/07/2017

Índice

Cuestiones

Figuras

Tablas

Gráficas

Cuestión 1:

Seleccione, instale y ejecute uno, comente los resultados. Atención: no es lo mismo un benchmark que una suite, instale un benchmark.

En primer lugar tenemos que tenemos que descargar e instalar Phorenix Suite. EN mi caso voy a instalar lo en CentOS. Una vez que lo hemos descargado los descomprimimos con la orden "tar –xzvf phoronix-test-suite-"versión".tar.gz. Despues nos metemos dentro de la carpeta, y usamos la orden "./install.sh".

Una vez instalado pasamos a ver los diferentes benchmarks que hay. En mi caso voy a usar el benchmark de Time GNUpg.

Para instalarlo se usa la orden "pforonix-test-suite install system/gnupg", y para ejecutarlo una vez instalado. Se usa la orden "phoronix-test-suite benchmark system/gnupg".

Ahora como podemos comprobar en la figura 1, lo primero que hace es mostrarnos las características del sistema.

```
DavCasSal sáb jul 08:$ phoronix-test-suite benchmark system/gnupg
Phoronix Test Suite v7.2.1
    Installed:    system/gnupg-1.0.1
System Information

  PROCESSOR:
    Core Count:       1
    Extensions:       SSE 4.2 + AVX + RDRAND
    Cache Size:       6144 KB
    Microcode:        0x0

  GRAPHICS:           LLVMpipe
    OpenGL:           2.1 Mesa 10.6.5 Gallium 0.4 (LLVM 3.6 256 bits)
    Display Driver:   modesetting 1.17.2
    Screen:           1024x768

  MOTHERBOARD:        Oracle VirtualBox v1.2
    Memory:           6144MB
    Chipset:          Intel 440FX- 82441FX PMC
    Network:          Intel 82540EM Gigabit

  DISK:               17GB VBOX HDD
    File-System:      xfs
    Mount Options:    attr2 inode64 noquota relatime rw seclabel
    Disk Scheduler:   CFQ

  OPERATING SYSTEM:   CentOS Linux 7
    Kernel:           3.10.0-327.el7.x86_64 (x86_64)
    Desktop:          GNOME Shell 3.14.4
    Display Server:   X Server 1.17.2
    Compiler:         GCC 4.8.5 20150623
    System Layer:     KVM VirtualBox
```

Figura 1 Características del sistema

Una vez mostradas las características nos pedirá si guardar los datos, le daremos a que si para poder observarlos después con más detalle en el navegador. Como se puede ver en el navegador la prueba que se ha hecho quedaría como en la figura 2, que es el resumen, en ella podemos ver el tiempo que tarda en encriptar los paquetes, el erro que puede llegar a tener y la desviación típica que tiene.

Figura 2 Resumen de la prueba

También no muestra una grafica de cómo ha quedado el tes, que se muestra en la figura 3.

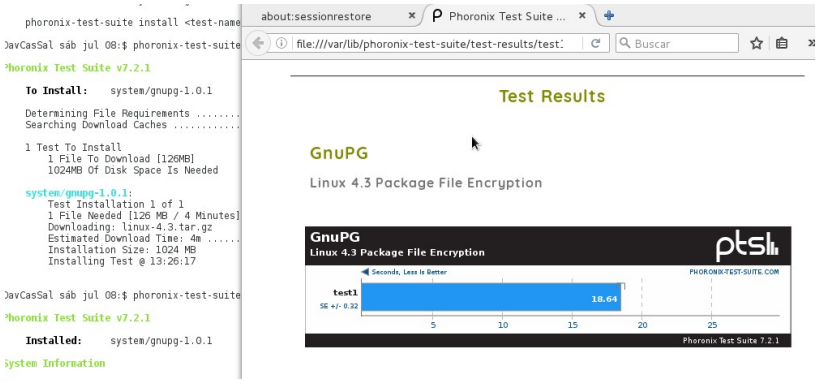

Figura 3 Grafica del test de GNUpg

Cuestión 2:

De los parámetros que le podemos pasar al comando ¿Qué significa -c 5? ¿y -n 100? Monitorice la ejecución de ab contra alguna máquina (cualquiera) ¿cuántas “tareas” crea ab en el cliente?

La opción –c sirve para indicar el número de peticiones múltiple que se pueden llevar al mismo tiempo. Y la opción –n sirve para indicar el número de peticiones que se van a usar para hacer la prueba.

```
DavCasSal vie jul 07:$ ab -c 5 -n 100 http://localhost/
This is ApacheBench, Version 2.3 <$Revision: 1430300 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient).....done


Server Software:        Apache/2.4.6
Server Hostname:        localhost
Server Port:            80

Document Path:          /
Document Length:        4897 bytes

Concurrency Level:      5
Time taken for tests:   0.034 seconds
Complete requests:      100
Failed requests:        0
Write errors:           0
Non-2xx responses:      100
Total transferred:      516800 bytes
HTML transferred:       489700 bytes
Requests per second:    2904.70 [#/sec] (mean)
Time per request:       1.721 [ms] (mean)
Time per request:       0.344 [ms] (mean, across all concurrent requests)
Transfer rate:          14659.64 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.1      0       0
Processing:     0    1   2.2      1      10
Waiting:        0    1   2.2      0      10
Total:          1    1   2.2      1      11

Percentage of the requests served within a certain time (ms)
  50%      1
  66%      1
  75%      1
  80%      1
  90%      3
```

Figura 4 Orden ab ejecutada

Cuestión 3:

Ejecute ab contra a las tres máquinas virtuales (desde el SO anfitrión a las máquina virtuales de la red local) una a una (arrancadas por separado).¿Cuál es la que proporciona mejores resultados? Muestre y coméntelos. (Use como máquina de referencia Ubuntu Server para la comparativa).

Lo primero que tenemos que hacer es instalar XAMPP en la maquina anfitriona para podes ejecutar la orden ab contra las otras maquinas virtuales. Primero nos tenemos que dirigir al directorio en el que está situado el Apache Benchmark.

Ahora ejecuto la orden "./ab.exe –c 5 –n 100 http://ip_SO/", en primer lugar lo hago sobre Ubuntu server como se muestra en la figura 5.

```
PS C:\apache\bin> .\ab.exe -c 5 -n 100 http://192.168.188.133/
This is ApacheBench, Version 2.3 <$Revision: 1748469 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.188.133 (be patient).....done


Server Software:        Apache/2.4.18
Server Hostname:        192.168.188.133
Server Port:            80

Document Path:          /
Document Length:        11321 bytes

Concurrency Level:      5
Time taken for tests:   0.074 seconds
Complete requests:      100
Failed requests:        0
Total transferred:      1159500 bytes
HTML transferred:       1132100 bytes
Requests per second:    1347.76 [#/sec] (mean)
Time per request:       3.710 [ms] (mean)
Time per request:       0.742 [ms] (mean, across all concurrent requests)
Transfer rate:          15261.05 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.5      0       1
Processing:     1    3   3.0      2      18
Waiting:        1    2   0.8      2       4
Total:          1    4   3.1      3      18

Percentage of the requests served within a certain time (ms)
  50%      3
  66%      3
  75%      3
  80%      4
  90%      4
  95%      9
  98%     18
  99%     18
 100%     18 (longest request)
```

Figura 5 orden ab Sobre la IP de ubuntu server

Ahora pasamos a hacerlo sobre Windows server, como muestro en la figura 6

```
PS C:\apache\bin> .\ab.exe -c 5 -n 100 http://192.168.188.132/
This is ApacheBench, Version 2.3 <$Revision: 1748469 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.188.132 (be patient).....done


Server Software:        Microsoft-IIS/7.5
Server Hostname:        192.168.188.132
Server Port:            80

Document Path:          /
Document Length:        689 bytes

Concurrency Level:      5
Time taken for tests:   0.052 seconds
Complete requests:      100
Failed requests:        0
Total transferred:      93200 bytes
HTML transferred:       68900 bytes
Requests per second:    1917.80 [#/sec] (mean)
Time per request:       2.607 [ms] (mean)
Time per request:       0.521 [ms] (mean, across all concurrent requests)
Transfer rate:          1745.50 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.5      0       1
Processing:     1    2   0.5      2       3
Waiting:        0    2   0.8      2       3
Total:          1    2   0.6      2       4

Percentage of the requests served within a certain time (ms)
  50%      2
  66%      3
  75%      3
  80%      3
  90%      3
  95%      3
  98%      4
  99%      4
 100%      4 (longest request)
```

Figura 6 Ab sobre Windows server

Y por ultimo para centOS seria la figura 7.

```
PS C:\apache\bin> .\ab.exe -c 5 -n 100 http://192.168.188.134/
This is ApacheBench, Version 2.3 <$Revision: 1748469 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.188.134 (be patient).....done


Server Software:        Apache/2.4.6
Server Hostname:        192.168.188.134
Server Port:            80

Document Path:          /
Document Length:        4897 bytes

Concurrency Level:      5
Time taken for tests:   0.070 seconds
Complete requests:      100
Failed requests:        0
Non-2xx responses:      100
Total transferred:      517900 bytes
HTML transferred:       489700 bytes
Requests per second:    1424.79 [#/sec] (mean)
Time per request:       3.509 [ms] (mean)
Time per request:       0.702 [ms] (mean, across all concurrent requests)
Transfer rate:          7206.02 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.5      0       1
Processing:     2    3   0.5      3       4
Waiting:        2    3   0.5      3       4
Total:          2    3   0.5      3       4

Percentage of the requests served within a certain time (ms)
  50%      3
  66%      4
  75%      4
  80%      4
  90%      4
  95%      4
  98%      4
  99%      4
 100%      4 (longest request)
PS C:\apache\bin>
```

Figura 7 Orden ab Sobre centos

Como se puede comprobar Ubuntu serve es la que más tiempo ha tardado en con respecto a los demás. Ubuntu server a tardad

Cuestión 4:

Instale y siga el tutorial en http://jmeter.apache.org/usermanual/build-web-test-plan.html realizando capturas de pantalla y comentándolas. En vez de usar la web de jmeter, haga el experimento usando sus máquinas virtuales ¿coincide con los resultados de ab?

Lo primero que tenemos que hacer es dirigirnos a la página y descargarnos el archivo. Una vez descargado abrimos el documento "README", en el que nos explica cómo iniciar Jmeter. Una vez iniciado pasamos a seguir el tutorial.

Para comenzar en el tutorial nos pide es añadir un numero de hilos. Tenemos que pulsar con el botón derecho sobre Plan de pruebas > añadir hilo > grupo de hilos. Una vez dentro cambiamos el nombre, el numero de hilos y las veces que se van a ejecutar, como se muestra en la figura 5.

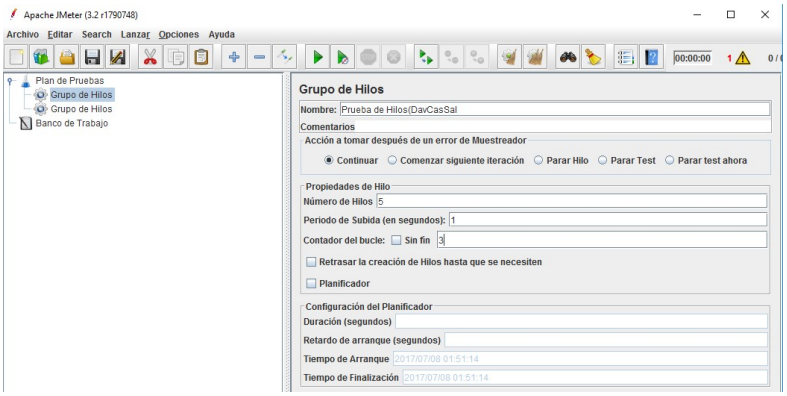

Figura 8 Configuración de Grupo de Hilos

Ahora pasamos a decir la dirección IP en la que vamos a hacer el test para ello le pulsamos el botón derecho sobre Grupos de hilos y nos dirigimos a añadir>elemento de configuración>valores por defecto para peticiones http. En la figura 6 se muestra que ponerla. Allí tenemos que poner la ip del servidor.

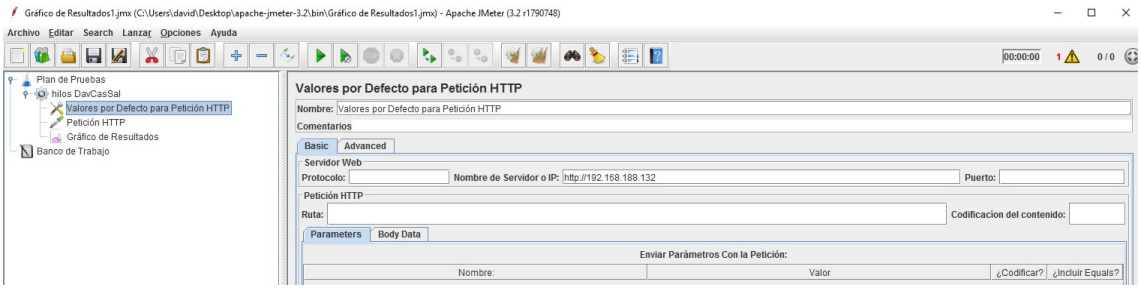

Figura 9 IP del servidor para hacer el test

Ahora pasamos a crear un apaticion HTTP, para ello pulsamos con el botón derecho sobre le grupo de hilos, añadir>muestreador>petición HTTP. Como se muestra en la figura 6 lo único que hay que cambiar es añadir en la ruta /.

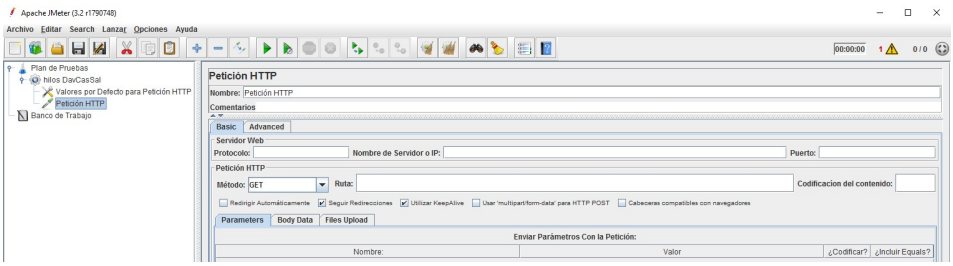

Figura 10 Cambio de la ruta a /

Por último, ponemos un receptor del tipo grafico para almacenar los datos que salga en el test, quedara como se muestra en la figura 8.

Figura 11 Gráfico de resultados

Ahora paso a hacer una prueba con cada servidor que teníamos, en primer lugar probamos con Ubuntu Server en la figura 12.

Figura 12 Grafico de resultados para Ubuntun Server

Ahora lo hacemos para centOS como se muestre en la figura 13

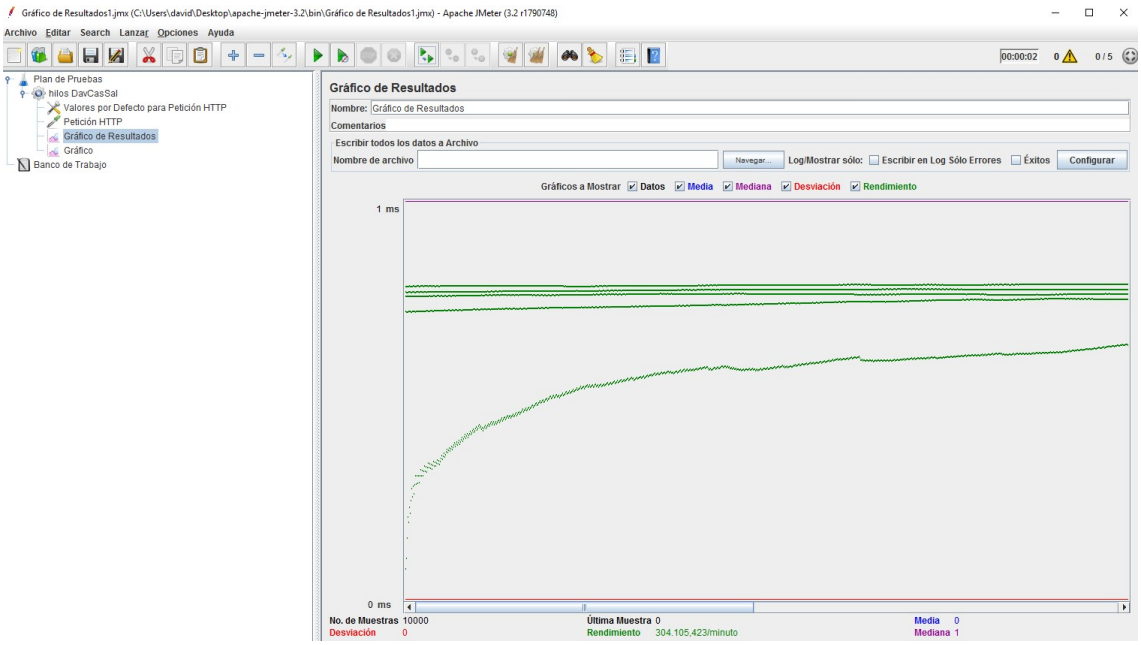

Figura 13 Grafico de resultados para centOS

Y por ultimo para Windows server.

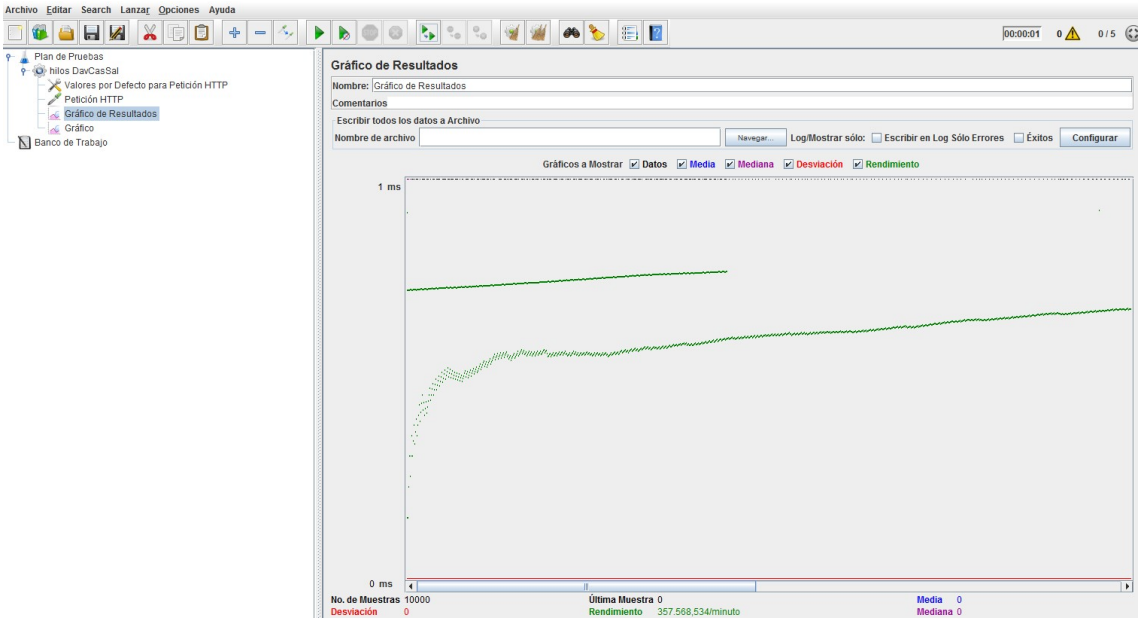

Figura 14 Grafico de resultados para Windows Server

Cuestión 5: Programe un benchmark usando el lenguaje que desee. El benchmark debe incluir:

1) Objetivo del benchmark.

   El objetivo del benchmark consiste en comparar el tiempo que tardan en leer ficheros entre los lenguajes de programación C++, Perl, Python, Ruby y PHP. Para que la medición sea efectiva, el documento que leen los distintos lenguajes será el mismo, será un fichero de números generados aleatoriamente.

2) Métricas (unidades, variables, puntuaciones, etc.)

El único parámetro que vamos a usar es el tiempo, ya que solo queremos ver que lenguaje es más rápido a la hora de leer un fichero.

3) Instrucciones para su uso.

A parte de los programas he creado un script, que va dentro de la carpeta, que crea el fichero random con el Nº de números que el usuario desee y ejecuta los programas para sacar el tiempo de cada uno de ellos.

La figura 15 muestra el contenido del script.

```
#!/bin/bash
#DavCasSal

./random
ruby ruby.rb
php php
perl perl
python python.py
./prueba_c
```

Figura 15 Script para la ejecución de cada fichero

4) Ejemplo de uso analizando los resultados.
Para probar los diferentes códigos en mi prueba he realizado las mediciones con 3 tamaños distintos. El primer tamaño es de un fichero con 50000 elementos, el segundo de 200000 elementos, y el más grande de 750000 elementos.

Una vez que hemos ejecutado el script el resultado obtenido ha sido. El que muestra la tabla 1.

	50000	200000	750000
Perl	13,6630000	38,18300	119,99400
PHP	3,6900043487549	14,25600	59,55696
ruby	8,4324220	31,07756	97,94885
c++	2,716064	10,53405	36,24701
python	3,556967	12,75301	43,38193

Tabla 1 Tabla de datos tras la ejecución del script

Para facilitar la forma de verlo también he sacado un grafico de barras como se muestra en el grafico 1, y un grafico de líneas como se muestra en el grafico 2.

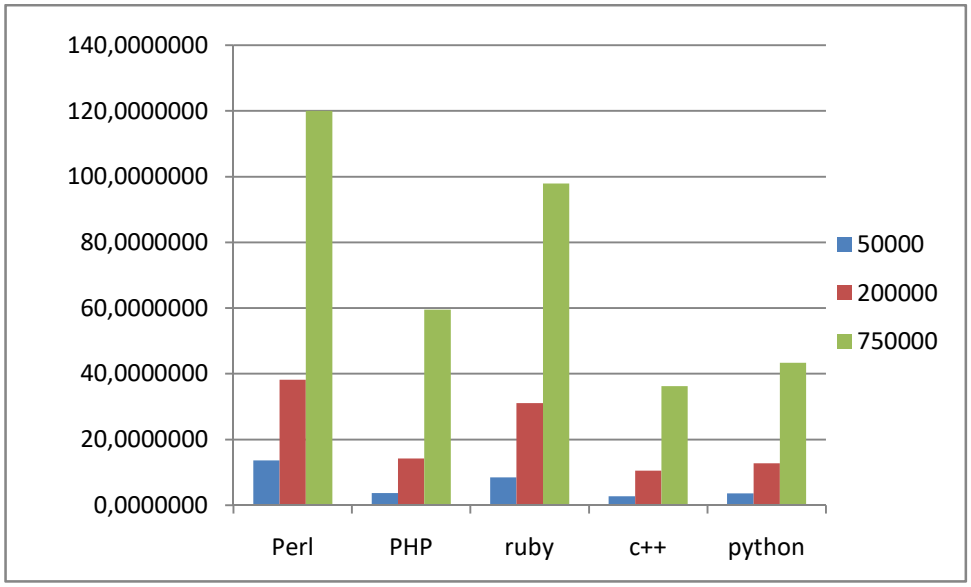

Gráfico 1 Datos de la ejecución del script en forma de barras

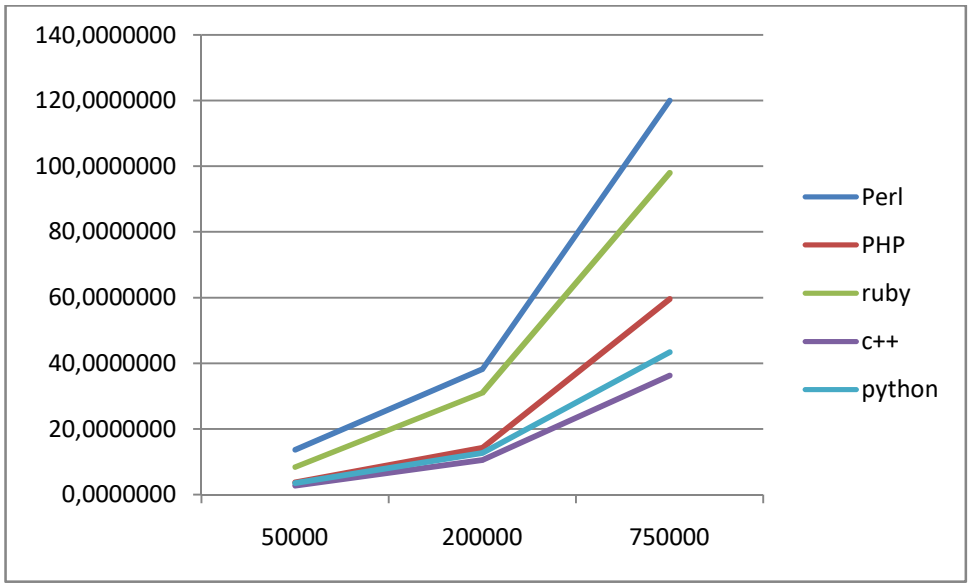

Gráfico 2 Datos de la ejecución del script en forma de líneas

Como podemos comprobar por las graficas 1 y 2, Python, PHP y C++ hasta los 200000 elementos por fichero son los más rápidos, pero PHP a partir de los 200000 elementos comienza a separarse. Por otro lado C++ es más rápido que Python, aunque no por mucho tiempo.

Ya en la parte que más tardan en ejecutarse son, Perl y Ruby. Aunque entre los dos haya diferencia ya que Perl es mucho más lento que Ruby, la grand diferencia está entre ellos y el más rápido ya que por ejemplo C++ llega a ser 3 veces más rápido que para los 20000 elementos.

Para comprobar cómo he hecho las mediciones, dejo dentro de la carpeta de la entrega un .zip que contiene el benchmark que he realizado.