Evaluación de configuraciones

Práctica 5 – Benchmarking

FDI - UCM

Iván Aguilera Calle – Daniel García Moreno

# UnixBench

Inicialmente, comenzamos ejecutando el comando Run del paquete Byte-UnixBench, con tres repeticiones y con una sola repetición para los programas cuya ejecución sea más lenta:

./Run -i 3 -c 1

Tras la finalización del comando anterior hemos obtenido los siguientes resultados:

También hemos investigado algunos de los programas que componen UnixBench:

* **Whetstone:**
  + Este programa ejecuta 11 bucles, cada uno de los cuales ejecuta un tipo distinto de instrucciones (funciones trigonométricas, raíces cuadradas, logaritmos, matrices …). Al comenzar el programa se establece el número de veces que iterará cada bucle y cuando finalizan todos los bucles se calcula el rendimiento en MWIPS
  + Tipo de benchmark: sintético (mide el rendimiento de un componente individual de un computador).
  + Mide el rendimiento de un sistema.
  + La unidad de los resultados está en MWIPS (millones de instrucciones Whetstone por segundo). Una instrucción Whetstone es una instrucción de punto flotante promedio.
* **Dhrystone:**
  + Este test sirve para medir y comparar el rendimiento de los computadores. Se centra en el manejo de strings (string handling) y no realiza operaciones de punto flotante, tampoco realiza llamadas al sistema, usa pocas variables globales y ejecuta operaciones con punteros formado por 12 procedimientos incluidos en un bucle de medida.
  + Tipo de benchmark: sintético.
  + Intenta medir y comparar el rendimiento de los computadores.
  + La unidad de los resultados está en lps (“Loops per second”).
* **Hanoi:**
  + Este otro programa ejecuta el clásico algoritmo de las torres de Hanoi y mide el número medio de iteraciones por segundo, en las que realiza el test de Hanoi en 20 segundos.
  + Tipo de benchmark: recursivo.
  + Mide el número medio de iteraciones por segundo, en las que realiza el test de Hanoi en 20 segundos
  + lps (“Loops per second”)

**RESULTADOS:**

========================================================================

BYTE UNIX Benchmarks (Version 5.1.3)

System: debian: GNU/Linux

OS: GNU/Linux -- 3.2.0-4-amd64 -- #1 SMP Debian 3.2.63-2

Machine: x86\_64 (unknown)

Language: en\_US.utf8 (charmap="ANSI\_X3.4-1968", collate="ANSI\_X3.4-1968")

CPU 0: Intel(R) Core(TM) i5-4200U CPU @ 1.60GHz (4593.3 bogomips)

x86-64, MMX, Physical Address Ext, SYSENTER/SYSEXIT, SYSCALL/SYSRET

18:41:55 up 28 min, 2 users, load average: 0.05, 0.03, 0.05; runlevel

------------------------------------------------------------------------

Benchmark Run: dom abr 23 2017 18:41:55 - 18:50:55

1 CPU in system; running 1 parallel copy of tests

Dhrystone 2 using register variables 29909339.3 lps (10.0 s, 2 samples)

Double-Precision Whetstone 3531.3 MWIPS (9.9 s, 2 samples)

Execl Throughput 4054.2 lps (29.9 s, 1 samples)

File Copy 1024 bufsize 2000 maxblocks 1048009.0 KBps (30.0 s, 1 samples)

File Copy 256 bufsize 500 maxblocks 319012.0 KBps (30.0 s, 1 samples)

File Copy 4096 bufsize 8000 maxblocks 2491342.0 KBps (30.0 s, 1 samples)

Pipe Throughput 2119730.7 lps (10.0 s, 2 samples)

Pipe-based Context Switching 407332.0 lps (10.0 s, 2 samples)

Process Creation 11807.1 lps (30.0 s, 1 samples)

Shell Scripts (1 concurrent) 5780.4 lpm (60.0 s, 1 samples)

Shell Scripts (8 concurrent) 738.8 lpm (60.0 s, 1 samples)

System Call Overhead 2729796.5 lps (10.0 s, 2 samples)

System Benchmarks Index Values BASELINE RESULT INDEX

Dhrystone 2 using register variables 116700.0 29909339.3 2562.9

Double-Precision Whetstone 55.0 3531.3 642.1

Execl Throughput 43.0 4054.2 942.8

File Copy 1024 bufsize 2000 maxblocks 3960.0 1048009.0 2646.5

File Copy 256 bufsize 500 maxblocks 1655.0 319012.0 1927.6

File Copy 4096 bufsize 8000 maxblocks 5800.0 2491342.0 4295.4

Pipe Throughput 12440.0 2119730.7 1704.0

Pipe-based Context Switching 4000.0 407332.0 1018.3

Process Creation 126.0 11807.1 937.1

Shell Scripts (1 concurrent) 42.4 5780.4 1363.3

Shell Scripts (8 concurrent) 6.0 738.8 1231.3

System Call Overhead 15000.0 2729796.5 1819.9

========

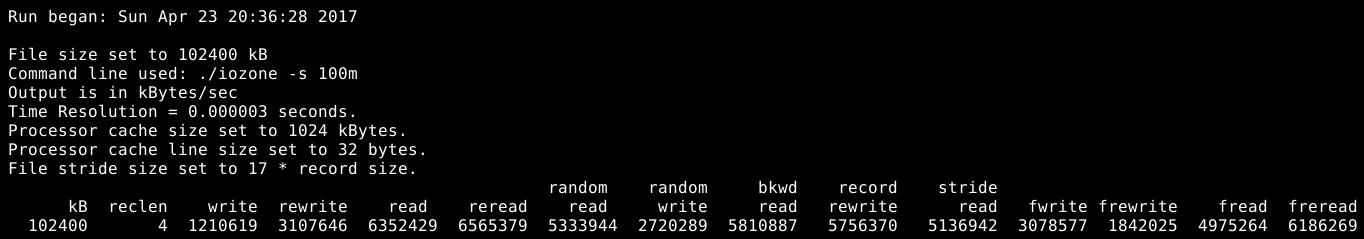
System Benchmarks Index Score 1533.3

# IOzone

IOzone es un benchmark específico para medir el rendimiento de las operaciones de Entrada/Salida de los sistemas de ficheros, en concreto sobre la funciones read(), write(), rewrite(), fread(), fwrite()…

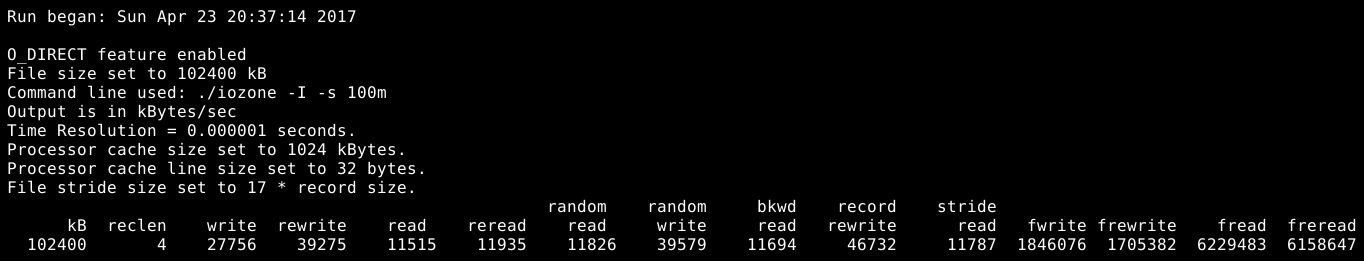
Comenzamos ejecutando el siguiente comando, el cual ejecuta el benchmark con un fichero de pruebas de 100MB:

./iozone -s 100m



Seguimos ejecutando la siguiente orden añadiendo la opción -I, que sirve para decirle al benchmark que las operaciones de entrada/salida que realice vayan directamente al disco sin pasar por el buffer cache:

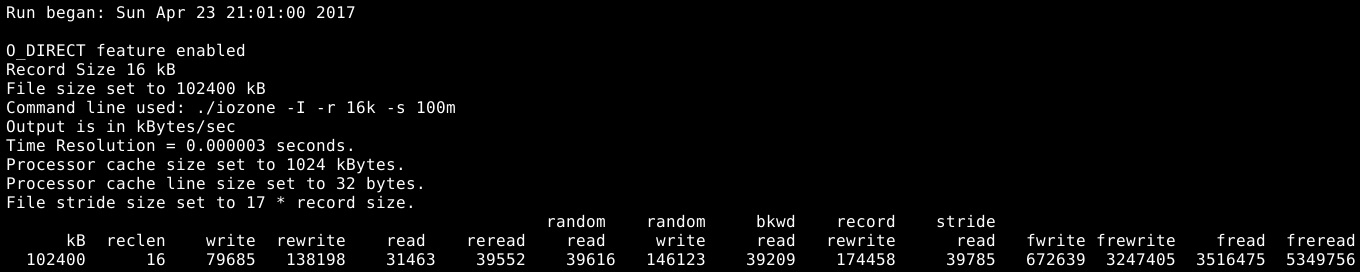
./iozone -I -s 100m



A diferencia del resultado del comando anterior, observamos que O\_DIRECT está activado (por el -I). También se observa que la mayoría de los valores de las operaciones se ven decrementados (velocidades de transmisión más lentas), ya que al no enviar los datos a un buffer intermedio, los datos se están enviando directamente a disco (el cual es un dispositivo más lento que el buffer intermedio) y por lo tanto estamos obteniendo valores más realistas del rendimiento de disco.

Por último, ejecutamos la siguiente orden añadiendo la opción -r 16k, que sirve para especificar el “record size” o tamaño de grabación (se realizan las operaciones sobre bloques de 16 Kilobytes):

./iozone -I -s 100m -r 16k



En este último comando, los valores de rendimiento conseguidos son superiores a los de la ejecución anterior, pero sin superar a los obtenidos en la primera ejecución (sin usar la opción -I). Esto se debe a que al aumentar el tamaño de los bloques que enviamos a disco de 4KB a 16KB, estamos disminuyendo la latencia de envío y aumenta el ancho de banda utilizado, por lo tanto, los valores obtenidos (kBytes/sec) son mayores que en la ejecución anterior.

# Iperf3

En este último apartado utilizaremos iperf3, que es una herramienta que sirve para medir el ancho de banda de redes IP (soporta, entre otros protocolos UDP, TCP, IPv4, IPv6…).

En cada prueba que realiza nos proporciona información sobre el ancho de banda, pérdida de datos y otros parámetros.

Primero arrancamos en una terminal un servidor TCP con la siguiente orden:

src/iperf3 -s

En segundo lugar, iniciamos un cliente en otra terminal diferente con la dirección IP de la máquina (en este caso hemos usado localhost, ya que nos encontrábamos en la misma máquina).

src/iperf3 -c localhost

Y en último lugar, obtenemos los resultados del comando anterior:

usuario@debian:~/iperf-3.1.2$ src/iperf3 -c localhost

Connecting to host localhost, port 5201

[ 4] local ::1 port 34202 connected to ::1 port 5201

[ ID] Interval Transfer Bandwidth Retr Cwnd

[ 4] 0.00-1.00 sec 4.86 GBytes 41.7 Gbits/sec 0 591 KBytes

[ 4] 1.00-2.00 sec 5.18 GBytes 44.5 Gbits/sec 0 591 KBytes

[ 4] 2.00-3.00 sec 5.09 GBytes 43.7 Gbits/sec 0 591 KBytes

[ 4] 3.00-4.00 sec 5.10 GBytes 43.8 Gbits/sec 0 591 KBytes

[ 4] 4.00-5.00 sec 4.95 GBytes 42.5 Gbits/sec 0 591 KBytes

[ 4] 5.00-6.00 sec 5.17 GBytes 44.4 Gbits/sec 0 591 KBytes

[ 4] 6.00-7.00 sec 5.19 GBytes 44.6 Gbits/sec 0 591 KBytes

[ 4] 7.00-8.00 sec 4.95 GBytes 42.5 Gbits/sec 0 591 KBytes

[ 4] 8.00-9.00 sec 5.04 GBytes 43.3 Gbits/sec 0 591 KBytes

[ 4] 9.00-10.00 sec 5.13 GBytes 44.0 Gbits/sec 0 591 KBytes

- - - - - - - - - - - - - - - - - - - - - - - - -

[ ID] Interval Transfer Bandwidth Retr

[ 4] 0.00-10.00 sec 50.7 GBytes 43.5 Gbits/sec 0 sender

[ 4] 0.00-10.00 sec 50.7 GBytes 43.5 Gbits/sec receiver

iperf Done.