Evaluación de configuraciones

Práctica 3 – Perfilado con ejecutables

FDI - UCM

Iván Aguilera Calle – Daniel García Moreno

# Perf

En este apartado utilizaremos la herramienta perf para realizar análisis y obtener distintos datos.

Comenzamos consultando algunas páginas del manual:

* **Perf:**
* **Perf-list:** si lo ejecutamos, nos muestra un listado de los eventos disponibles.
* **Perf-stat** nos muestra estadísticas de:
  + Un proceso : perf stat -p <PID> time (como se comporta pid durante 10 s)
  + Todos los procesos del sistema: perf -a sleep 5 (lo vemos solo 5 segundos).
* **Perf-record** guarda una muestra de una aplicación determinada en el archivo perf.data:
  + Perf record -e cycles:k -a –sleep 5
* **Perf-report** sirve para visualizar el contenido del archivo generado con el anterior comando.

## Cuenta de eventos

En este apartado obtendremos estadísticas de rendimiento para el programa “matrix1.c”, usando para ello la orden perf stat -r 5 (explicar lo que hace).

También observaremos el número de accesos y fallos de la cache de datos de primer nivel del programa “matrix1.c”, ejecutando para ello los siguientes comandos:

* perf -e L1-dcache-loads
* perf -e L1-dcache-load-misses
* perf -e L1-dcache-stores
* perf -e L1-dcache-store-misses

Ejecutando los comandos anteriores para el programa “matrix1.c”, obtenemos los siguientes resultados:

//PONER RESULTADOS

//ANALISIS RESULTADOS

//FIJARSE TIEMPO DE EJECUCIÓN

Repetimos los comandos anteriores, pero esta vez para el programa “matrix2.c” (similar al anterior pero que intercambia los dos bucles internos) y obtenemos los siguientes resultados:

//PONER RESULTADOS

//ANALISIS RESULTADOS

//FIJARSE TIEMPO DE EJECUCIÓN

## Perfilado basado en eventos – 1

En este apartado obtendremos el tiempo consumido por cada función del programa “edges.c”. Para ello, hemos realizado el muestreo utilizando la orden perf record y hemos generado el informe ejecutando perf report --stdio.

//PONER RESULTADOS

//ANALISIS DE LOS RESULTADOS

## Perfilado basado en eventos – 2

En este segundo apartado, repetiremos lo que hemos hecho en el apartado anterior añadiendo probando distintas opciones:

* perf record -g:
* perf record -e (muestrea con otros eventos interesantes como fallos de página, fallos de lectura o escritura de la cache de datos de primer nivel).
* perf record -F (distintas frecuencias de muestreo) -c (cuentas de eventos).

//PONER RESULTADOS

//ANÁLISIS DE RESULTADOS

//COMPARAR PERF CON GPROF Y GOOGLE-PPFROF (SOBRECARGA, PRECISIÓN, AMBITO…).

# Valgrind

En esta segunda parte, utilizaremos la herramienta cachegrind para obtener medidas del uso de la cache de primer nivel de los programas “matrix1.c” y “matrix2.c”, ejecutando para ello los siguientes comandos:

* Valgrind --tool=cachegrind ./matrix1
* Valgrind --tool=cachegrind ./matrix2

//PONER RESULTADOS

//ANÁLISIS DE RESULTADOS

//COMPARAR VALGRIND CON PERF (SOBRECARGA, PRECISIÓN, ÁMBITO)…

# Strace

En esta tercera y última parte utilizaremos la orden strace (explicar para qué es).

Observamos los ficheros que abre la orden vmstat, trazando la llamada al sistema open (utilizando la opción -e open de strace).

//RESULTADOS

* ¿De dónde saca toda la información?

//RESPONDER

Repetimos el comando anterior para ps, para top y para sar.

//PONER RESULTADOS

//ANÁLISIS DE RESULTADOS

Obtener un resumen estadístico (opción -c) de las llamadas al sistema realizadas por la orden “find > /usr &> /dev/null”. Usar la opción -o de strace para escribir la salida en un fichero

* ¿Para qué sirven las 5 llamadas más usadas?

//RESPONDER

//PONER RESULTADOS

//ANÁLISIS DE RESULTADOS