# Práctica 2.4. Perfilado con ejecutables

En esta práctica usaremos la máquina física.

## perf (~45 min)

Consulta la página de manual de `perf` (paquete `linux-tools`) y de `perf-list`, `perf-stat`, `perf-record` y `perf-report`.

### Cuenta de eventos

Obtén estadísticas de contadores de rendimiento del programa `matrix1.c` (disponible en el Campus Virtual) con `perf stat`. Prueba la opción `-r 5`.

Mide el número accesos y fallos de la *cache* de datos de primer nivel (`-e L1-dcache-loads,L1-dcache-load-misses,L1-dcache-stores,L1-dcache-store-misses`) del programa `matrix1.c` (puede que no todos los eventos *hardware* estén soportados) . Observa también el tiempo de ejecución.

Repite el ejercicio anterior con el programa `matrix2.c` (también disponible en el Campus Virtual), que es muy parecido al anterior, pero intercambia los dos bucles internos.

**Entrega:** Copia los resultados y escribe un breve análisis de los mismos.

### Perfilado basado en eventos

Obtén el tiempo consumido por cada función del programa `edges.c`. Para ello, realiza el muestreo con `perf record` y genera el informe con `perf report --stdio`.

**Entrega:** Copia los resultados y escribe un breve análisis de los mismos.

Repite el ejercicio anterior añadiendo la opción `-g` en el muestreo.

Repite el ejercicio anterior muestreando con otros eventos interesantes (opción `-e`), como fallos de página o fallos de lectura o escritura de la *cache* de datos de primer nivel.

Repite el ejercicio anterior muestreando a distintas frecuencias de muestreo (opción `-F`) y cuentas de eventos (opción `-c`). Hay que considerar que la frecuencia máxima está limitada y que los eventos ocurren a velocidades muy distintas (por ejemplo, tiene sentido muestrear con cada fallo de página, pero no con cada acceso a la *cache*).

**Entrega:** Escribe un breve análisis de los resultados. Compara `perf` con `gprof` y `google-pprof` (sobrecarga, precisión, ámbito...).

## valgrind (~20 min)

Consulta la página de manual de `valgrind`. (paquete `valgrind`).

Obtén medidas del uso de la *cache* de primer nivel de los programas `matrix1.c` y `matrix2.c` con la herramienta `cachegrind`:

```
$ valgrind --tool=cachegrind ./matrix1
$ valgrind --tool=cachegrind ./matrix2
```

**Entrega:** Copia los resultados y escribe un breve análisis de los mismos. Compara `valgrind` con `perf` (sobrecarga, precisión, ámbito...).

## strace (~25 min)

Consulta la página de manual de `strace`.

Observa los ficheros que abre la orden `vmstat` trazando la llamada al sistema `open` (opción `-e open`). ¿De dónde saca toda la información? ¿Y `ps`, `top` y `sar`?

Obtén un resumen estadístico (opción `-c`) de las llamadas al sistema realizadas por la orden:

```
$ find /usr &> /dev/null
```

Usa la opción `-o` de `strace` para escribir la salida en un fichero (si no, se enviará a `/dev/null` debido a la redirección).

¿Para qué sirven las 5 llamadas más usadas?

**Entrega:** Escribe un breve análisis de los resultados y responde a las preguntas.