

Daniel Prieto Remacha
Miguel Zayas Boíza

Práctica 4. Perfilado con ejecutables

En esta práctica veremos cómo obtener un perfil de rendimiento a partir de ejecutables. Utilizaremos una **máquina física** para la parte de `perf`, porque es necesario para acceder a los contadores *hardware*, pero para el resto de apartados se puede usar una máquina virtual.

Contenido:

[perf](#)
[Cuenta de eventos \(~20 min.\)](#)
[Perfilado por eventos \(~40 min.\)](#)
[Callgrind \(~15 min.\)](#)
[strace \(~15 min.\)](#)

perf

En las máquinas del laboratorio, `perf` está ya instalado. Si no se encuentra la versión correspondiente a la del *kernel*, se puede usar directamente cualquiera de los binarios que hay en `/usr/lib/linux-tools/*`. Para ello, añade uno de estos directorios a la ruta de búsqueda, por ejemplo:

```
$ PATH=/usr/lib/linux-tools/5.4.0-65-generic:$PATH
```

El comando `PATH=/usr/lib/linux-tools/5.4.0-65-generic:$PATH` modifica la variable de entorno `PATH` que indica al sistema operativo dónde buscar los programas ejecutables.

Al añadir `/usr/lib/linux-tools/5.4.0-65-generic` al principio de la variable `PATH`, le dices al sistema operativo que busque primero en ese directorio cuando ejecutes un comando.

Esto puede ser útil si quieras usar una versión específica de una herramienta o programa que está instalada en ese directorio

En Debian, se instala con el paquete `linux-tools` o `linux-perf`, en Ubuntu, con el paquete `linux-tools-generic` o `linux-tools-`uname -r`` y en CentOS/Red Hat, con el paquete `perf`.

INSTALACIÓN Y PERMISOS EN DEBIAN

En mi caso he tenido que instalar anteriormente `libc6` con : `apt install -y libc6 debconf`

```
sudo sh -c 'echo kernel.perf_event_paranoid=1 > /etc/sysctl.d/local.conf'
```

Consulta la página de manual de `perf` y las de los subcomandos `perf-list`, `perf-stat`, `perf-record` y `perf-report`.

<code>perf</code>	Herramienta de análisis de rendimiento <code>perf [--version] [--help] [OPCIONES] COMANDO [ARGS]</code>
<code>perf-list</code>	Este comando muestra los tipos de eventos simbólicos que pueden ser seleccionados en los diversos comandos de rendimiento con la opción <code>-e</code> .

perf-stat	Este comando ejecuta un comando y recopila el contador de rendimiento de estadísticas de la misma.
perf-report	Muestra la información del contador obtenida perf-record.

Cuenta de eventos (~20 min.)

Obtén estadísticas de contadores de rendimiento compilando y ejecutando el programa `matrix1.c` (disponible con la práctica) con:

```
$ perf stat ./matrix1
```

Prueba la opción `-r 5`.

```
perf stat -r 5 ./matrix1
```

```
(dani@kali)-[~/Documentos/lab]
$ perf stat -r 5 ./matrix1
Result: 127840000.000000
Result: 127840000.000000
Result: 127840000.000000
Result: 127840000.000000
Result: 127840000.000000

Performance counter stats for './matrix1' (5 runs):

      1.767,87 msec task-clock          #    0,960 CPUs utilized          ( +-  2,11% )
          26 context-switches          #   14,210 /sec             ( +-177,80% )
          2 cpu-migrations           #    1,093 /sec             ( +-122,88% )
        231 page-faults              #  126,249 /sec            ( +- 54,20% )
  6.595.967,220 cycles            #    3,605 GHz             ( +-  1,91% )
 17.965.232,262 instructions       #    2,64  insn per cycle     ( +-  0,01% )
  516.332,024 branches           # 282,193 M/sec            ( +-  0,04% )
     668.679 branch-misses         #    0,13% of all branches    ( +-  0,95% )

 1,8413 +- 0,0425 seconds time elapsed  ( +-  2,31% )
```

`-r 5`: repito la operación 5 veces.

Result: muestra los resultados del contador de rendimiento.

Mide los accesos a la **cache** de datos de primer nivel y los fallos (opción `-e L1-dcache-loads, L1-dcache-load-misses, L1-dcache-stores, L1-dcache-store-misses`) al ejecutar el programa `matrix1`.

La opción `-e`, me permite añadir eventos de `perf-list`, para verlos puedo poner en la terminal: `perf list` (NOTA: Para mostrar los eventos usar la opción `-e`)

Con lo que me mostrará un listado con las posibles opciones, a continuación se muestran las que hemos utilizado.

L1-dcache-load-misses	[Hardware cache event]
L1-dcache-loads	[Hardware cache event]
L1-dcache-stores	[Hardware cache event]
L1-icache-load-misses	[Hardware cache event]
LLC-load-misses	[Hardware cache event]
LLC-loads	[Hardware cache event]
LLC-store-misses	[Hardware cache event]
LLC-stores	[Hardware cache event]

Nos da algo más de info, como que son de tipo hardware, y luego podemos guiarnos por el nombre para saber que está midiendo.

```
perf stat -e L1-dcache-loads ./matrix1
```

```
(dani㉿kali)-[~/Documentos/lab]
└ $ perf stat -e L1-dcache-loads ./matrix1
Result: 127840000.000000

Performance counter stats for './matrix1':
        4.620.557.952      L1-dcache-loads
          1,787452757 seconds time elapsed
          1,782245000 seconds user
          0,004005000 seconds sys
```

Número de eventos que se han producido al hacer carga en la caché. Así como los tiempos que ha necesitado al lanzar ./matrix1

```
perf stat -e L1-dcache-load-misses ./matrix1
```

```
(dani㉿kali)-[~/Documentos/lab]
└ $ perf stat L1-dcache-load-misses ./matrix1
Workload failed: No existe el fichero o el directorio

(dani㉿kali)-[~/Documentos/lab]
└ $ perf stat -e L1-dcache-load-misses ./matrix1
Result: 127840000.000000

Performance counter stats for './matrix1':
        531.213.044      L1-dcache-load-misses
          1,720480589 seconds time elapsed
          1,719978000 seconds user
          0,000000000 seconds sys
```

Comprobamos que sin la opción -e no lo coge, es lógico ya que no le he avisado de que iba a mandarle un evento.

Igual que el anterior pero en este caso son las cargas fallidas en la cache

```
perf stat -e L1-dcache-stores ./matrix1
```

```
(dani㉿kali)-[~/Documentos/lab]
$ perf stat -e L1-dcache-stores ./matrix1
Result: 127840000.000000

Performance counter stats for './matrix1':
      1.030.460.763      L1-dcache-stores

      1,765733371 seconds time elapsed

      1,759753000 seconds user
      0,004008000 seconds sys
```

Almacenamientos que se han hecho en la cache

```
perf stat -e L1-dcache-store-misses ./matrix1
```

```
(dani㉿kali)-[~/Documentos/lab]
$ sudo perf stat -e L1-dcache-store-misses ./matrix1
[sudo] contraseña para dani:
Result: 127840000.000000

Performance counter stats for './matrix1':
      <not supported>      L1-dcache-store-misses

      1,728719294 seconds time elapsed

      1,719310000 seconds user
      0,007978000 seconds sys
```

Almacenamientos fallidos en la cache.

Aparece como <no supported>, creia que era por no tener permisos por lo que lo intente con sudo.

Pero tras buscar en internet y perf, ser un contador Hardware es posible que ciertas maquinas no tengan ese contador/medidor por lo que se entiende que no lo muestre y aparezca ese mensaje de “error”? ¿

Repite el ejercicio anterior con el programa `matrix2.c` (también disponible con la práctica).

Copia los resultados y escribe un breve análisis de los mismos comparándolos con los obtenidos en el ejercicio anterior.

```
(dani㉿kali)-[~/Documentos/lab]
└$ gcc -o matrix2 matrix2.c

(dani㉿kali)-[~/Documentos/lab]
└$ perf stat -r 5 ./matrix2
Result: 127840000.000000
Result: 127840000.000000
Result: 127840000.000000
Result: 127840000.000000
Result: 127840000.000000

Performance counter stats for './matrix2' (5 runs):

      1.306,91 msec task-clock          #    0,978 CPUs utilized      ( +-  0,93% )
          35      context-switches       #   26,243 /sec            ( +- 70,52% )
          2      cpu-migrations        #    1,500 /sec            ( +- 91,38% )
         460      page-faults          # 344,907 /sec            ( +- 27,26% )
  5.256.684.308      cycles           #    3,941 GHz            ( +-  0,52% )
17.985.769.602      instructions      #    3,38  insn per cycle  ( +-  0,00% )
  517.138.230      branches          # 387,750 M/sec          ( +-  0,03% )
     667.004      branch-misses     #    0,13% of all branches  ( +-  0,56% )

      1,3362 +- 0,0120 seconds time elapsed  ( +-  0,90% )

(dani㉿kali)-[~/Documentos/lab]
└$ perf stat -e L1-dcache-loads ./matrix2
Result: 127840000.000000

Performance counter stats for './matrix2':

      4.624.421.452      L1-dcache-loads
      1,318452683 seconds time elapsed

      1,317214000 seconds user
      0,000000000 seconds sys

(dani㉿kali)-[~/Documentos/lab]
└$ perf stat -e L1-dcache-load-misses ./matrix2
Result: 127840000.000000

Performance counter stats for './matrix2':

      66.062.231      L1-dcache-load-misses
      1,314500179 seconds time elapsed

      1,310552000 seconds user
      0,004007000 seconds sys
```

```
└─(dani㉿kali)-[~/Documentos/lab]
$ perf stat -e L1-dcache-stores ./matrix2
Result: 127840000.000000

Performance counter stats for './matrix2':
      1.031.482.841      L1-dcache-stores
      1,363561798 seconds time elapsed
      1,357306000 seconds user
      0,004003000 seconds sys

└─(dani㉿kali)-[~/Documentos/lab]
$ perf stat -e L1-dcache-store-misses ./matrix2
Result: 127840000.000000

Performance counter stats for './matrix2':
      <not supported>      L1-dcache-store-misses
      1,322512597 seconds time elapsed
      1,322513000 seconds user
      0,0000000000 seconds sys
```

matrix2 realiza más accesos a la cache pero tarda mucho menos tiempo a la hora de realizar estas. En relación, matrix2 es más rápido que matrix1.

Perfilado por eventos (~40 min.)

Obtén un perfil de ejecución del programa edges.c al procesar la imagen img.pgm (ambos disponibles con la práctica):

```
$ perf record ./edges img.pgm out.pgm  
$ perf report --stdio
```

Observa la traza de eventos recogidos con `perf script`.

Copia los resultados y escribe un breve análisis de los mismos.

```
[dani@kali] - [~/Documentos/lab]
$ perf record ./edges img.pgm out.pgm
[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 0,249 MB perf.data (5899 samples) ]
```

```
[dani@kali]-[~/Documentos/lab]$ perf report --stdio
# To display the perf.data header info, please use --header/--header-only options.
#
#
# Total Lost Samples: 0
#
# Samples: 5K of event 'cycles'
# Event count (approx.): 5840929244
#
# Overhead Command Shared Object Symbol
# ..... .
65.86% edges edges [.] gaussian
30.78% edges edges [.] laplacian
0.70% edges edges [.] load_image_file
0.59% edges edges [.] save_image_file
0.57% edges libc.so.6 [.] fputc
0.40% edges libc.so.6 [.] _IO_getchar
0.12% edges edges [.] fgetc@plt
0.10% edges edges [.] fputc@plt
0.09% edges [kernel.kallsyms] [k] clear_page_errms
0.07% edges [kernel.kallsyms] [k] pick_next_entity
0.05% edges ld-linux-x86-64.so.2 [.] 0x0000000000001b857
0.02% edges [kernel.kallsyms] [k] get_page_from_freelist
0.02% edges [kernel.kallsyms] [k] __domain_mapping_c
0.02% edges [kernel.kallsyms] [k] filemap_read
0.02% edges [kernel.kallsyms] [k] native_read_msr
0.02% edges [kernel.kallsyms] [k] __update_load_avg_se
0.02% edges [kernel.kallsyms] [k] account_user_time
0.02% edges [kernel.kallsyms] [k] __calc_delta
0.02% edges [iwlwifi] [k] iwl_pcidev_msix_isr
0.02% edges [kernel.kallsyms] [k] trigger_load_balance
0.02% edges [kernel.kallsyms] [k] native_write_msr
0.02% edges [kernel.kallsyms] [k] __raw_spin_lock_irqsave
0.02% edges [kernel.kallsyms] [k] scheduler_tick
0.02% edges [kernel.kallsyms] [k] pick_next_task_stop
0.02% edges [kernel.kallsyms] [k] free_pcp_prepare
0.02% edges [jbd2] [k] add_transaction_credits
0.02% edges [jbd2] [k] start_this_handle -sort comm,dso)
0.02% edges [ext4] [k] ext4_da_reserve_space
0.02% edges [kernel.kallsyms] [k] percpu_counter_add_batch
```

perf script

```
(dani㉿kali)-[~/Documentos/lab]
$ perf script
  What could I be doing wrong?
Question edges 22808 4275.618730: 1 cycles: ffffffa3a78af4 native_write_msr+0x4 ([kernel.kallsyms])
edges 22808 4275.618734: 1 cycles: ffffffa3a78af4 native_write_msr+0x4 ([kernel.kallsyms])
Tags edges 22808 4275.618736: 12 cycles: ffffffa3a78af4 native_write_msr+0x4 ([kernel.kallsyms])
edges 22808 4275.618737: 323 cycles: ffffffa3a78af6 native_write_msr+0x6 ([kernel.kallsyms])
Users edges 22808 4275.618738: 9385 cycles: ffffffa3a3821f native_sched_clock+0xf ([kernel.kallsyms])
edges 22808 4275.618741: 254047 cycles: ffffffa3e3ae76 security_bprm_committing_creds+0x16 ([kernel.kallsyms])
edges 22808 4275.618804: 3054603 cycles: 7ff607c8da0 [unknown] (/usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2)
Companies edges 22808 4275.619552: 4181850 cycles: 5603df37c406 load_image_file+0x22d (/home/dani/Documentos/lab/edges)
edges 22808 4275.620576: 3833784 cycles: 5603df37c42a load_image_file+0x251 (/home/dani/Documentos/lab/edges)
COLLECTIVES edges 22808 4275.621550: 3118538 cycles: 5603df37c3fa load_image_file+0x221 (/home/dani/Documentos/lab/edges)
Explore edges 22808 4275.622313: 2828826 cycles: 7ff607afeb85 _IO_getc+0x5 (/usr/lib/x86_64-linux-gnu/libc.so.6)
edges 22808 4275.623006: 2591028 cycles: 7ff607afeb8d _IO_getc+0xd (/usr/lib/x86_64-linux-gnu/libc.so.6)
edges 22808 4275.623640: 2384088 cycles: 5603df37c060 fgetc@plt+0x0 (/home/dani/Documentos/lab/edges)
TEAMS edges 22808 4275.624224: 2203480 cycles: 7ff607afeb43 _IO_getc+0xc3 (/usr/lib/x86_64-linux-gnu/libc.so.6)
edges 22808 4275.624764: 2046024 cycles: 5603df37c42a load_image_file+0x251 (/home/dani/Documentos/lab/edges)
edges 22808 4275.625265: 1908786 cycles: 5603df37c404 load_image_file+0x22b (/home/dani/Documentos/lab/edges)
Stack Overf... edges 22808 4275.625768: 1691814 cycles: 5603df37c41f load_image_file+0x246 (/home/dani/Documentos/lab/edges)
Teams - Sta... edges 22808 4275.626183: 1585514 cycles: 5603df37c406 load_image_file+0x22d (/home/dani/Documentos/lab/edges)
collaborati... edges 22808 4275.626571: 1506895 cycles: 7ff607afeb8d _IO_getc+0xd (/usr/lib/x86_64-linux-gnu/libc.so.6)
sharing orga... edges 22808 4275.627006: 1439772 cycles: 7ff607afeb3a _IO_getc+0xba (/usr/lib/x86_64-linux-gnu/libc.so.6)
knowledge... edges 22808 4275.627376: 1363061 cycles: 7ff607afeb85 _IO_getc+0x5 (/usr/lib/x86_64-linux-gnu/libc.so.6)
edges 22808 4275.627710: 1308135 cycles: 5603df37c060 fgetc@plt+0x0 (/home/dani/Documentos/lab/edges)
edges 22808 4275.628031: 1267028 cycles: 5603df37c42a load_image_file+0x251 (/home/dani/Documentos/lab/edges)
edges 22808 4275.628341: 1232132 cycles: 5603df37c417 load_image_file+0x23e (/home/dani/Documentos/lab/edges)
edges 22808 4275.628643: 1201958 cycles: 5603df37c41c load_image_file+0x243 (/home/dani/Documentos/lab/edges)
edges 22808 4275.628938: 1176182 cycles: 7ff607afeb42 _IO_getc+0xc2 (/usr/lib/x86_64-linux-gnu/libc.so.6)
edges 22808 4275.629226: 1153886 cycles: 7ff607afeb3a _IO_getc+0xba (/usr/lib/x86_64-linux-gnu/libc.so.6)
edges 22808 4275.629543: 1116533 cycles: 5603df37c41c load_image_file+0x243 (/home/dani/Documentos/lab/edges)
edges 22808 4275.629817: 1087036 cycles: ffffffa43c5a17 clear_page_erms+0x7 ([kernel.kallsyms])
edges 22808 4275.630084: 1075026 cycles: 7ff607afeb43 _IO_getc+0xc3 (/usr/lib/x86_64-linux-gnu/libc.so.6)
edges 22808 4275.630348: 1066602 cycles: 5603df37c3fa load_image_file+0x221 (/home/dani/Documentos/lab/edges)
edges 22808 4275.630610: 1059673 cycles: 5603df37c060 fgetc@plt+0x0 (/home/dani/Documentos/lab/edges)
edges 22808 4275.630870: 1053690 cycles: ffffffa43c5fea copy_user_enhanced_fast_string+0xa ([kernel.kallsyms])
edges 22808 4275.631128: 1048553 cycles: 5603df37c41c load_image_file+0x243 (/home/dani/Documentos/lab/edges)
edges 22808 4275.631386: 1044277 cycles: 5603df37c414 load_image_file+0x23b (/home/dani/Documentos/lab/edges)
edges 22808 4275.631642: 1040663 cycles: 5603df37c414 load_image_file+0x23b (/home/dani/Documentos/lab/edges)
edges 22808 4275.631897: 1037473 cycles: 7ff607afeb85 _IO_getc+0x5 (/usr/lib/x86_64-linux-gnu/libc.so.6)
edges 22808 4275.632152: 1034778 cycles: 5603df37c404 load_image_file+0x22b (/home/dani/Documentos/lab/edges)
edges 22808 4275.632405: 1032538 cycles: 7ff607afeb28 _IO_getc+0xa8 (/usr/lib/x86_64-linux-gnu/libc.so.6)
edges 22808 4275.632659: 1030561 cycles: 7ff607afeb85 _IO_getc+0x5 (/usr/lib/x86_64-linux-gnu/libc.so.6)
edges 22808 4275.632912: 1028770 cycles: 5603df37c404 load_image_file+0x22b (/home/dani/Documentos/lab/edges)
edges 22808 4275.633164: 1027484 cycles: 7ff607afeb28 _IO_getc+0xa8 (/usr/lib/x86_64-linux-gnu/libc.so.6)
edges 22808 4275.633450: 1024336 cycles: 7ff607afeb85 _IO_getc+0x5 (/usr/lib/x86_64-linux-gnu/libc.so.6)
```

NOTA: También se ha crado el fichero perf.data

```
(dani㉿kali)-[~/Documentos/lab]
$ ls
cpu_mem.c  edges  edges.c  img.pgm  matrix1  matrix1.c  matrix2  matrix2.c  out.pgm  perf.data
```

perf record, graba en el archivo perf.data un contador de perfiles, sin mostrar nada por consola que luego podrá ser abierto por perf report.

Al ejecutar perf report, la opción –stdio hace que se muestre por consola como “nano”, sin poder modificarse.

Sin embargo si lo ejecuto sin la opción de –stdio, se ejecuta con un explorador de archivos normal.

No es necesario informar a report de que el archivo es perf.data, lo coge por defecto.

Y perf display, únicamente es otro visor para los datos de perf.data, que lo muestra de forma distinta, en vez de % muestra el numero de ciclos...

```
Samples: 5K of event 'cycles', Event count (approx.): 5906142567
Overhead Command Shared Object Symbol
 66,11% edges edges [.] gaussian
 30,81% edges edges [.] laplacian
 0,60% edges edges [.] load_image_file
 0,49% edges edges [.] save_image_file
 0,49% edges libc.so.6 [.] fputc
 0,47% edges libc.so.6 [.] _IO_getc
 0,15% edges edges [.] fgetc@plt
 0,12% edges [kernel.kallsyms] [k] clear_page_erms
 0,10% edges edges [.] fputc@plt
 0,05% edges ld-linux-x86-64.so.2 [.] 0x000000000013a0a
 0,03% edges [kernel.kallsyms] [k] vfs_write
 0,03% edges [xhci_hcd] [k] xhci_ring_ep_doorbell
 0,03% edges [kernel.kallsyms] [k] check_preemption_disabled
 0,03% edges [kernel.kallsyms] [k] _brelse
 0,03% edges [kernel.kallsyms] [k] __mod_node_page_state
 0,02% edges [kernel.kallsyms] [k] copy_user_enhanced_fast_string
 0,02% edges [kernel.kallsyms] [k] idle_cpu
 0,02% edges [kernel.kallsyms] [k] find_get_pages_range_tag
 0,02% edges [kernel.kallsyms] [k] node_page_state
 0,02% edges [kernel.kallsyms] [k] get_mem_cgroup_from_mm
 0,02% edges [kernel.kallsyms] [k] account_user_time
 0,02% edges [kernel.kallsyms] [k] __fdget_pos
 0,02% edges [kernel.kallsyms] [k] filemap_get_read_batch
 0,02% edges [kernel.kallsyms] [k] in_lock_functions
 0,02% edges [kernel.kallsyms] [k] select_task_rq_fair
 0,02% edges [kernel.kallsyms] [k] __filemap_add_folio
 0,02% edges [kernel.kallsyms] [k] __fsnotify_parent
 0,02% edges [ext4] [k] ext4_get_group_desc
 0,02% edges [kernel.kallsyms] [k] get_page_from_freelist
 0,02% edges [kernel.kallsyms] [k] hrtimer_update_next_event
 0,02% edges [ext4] [k] ext4_da_get_block_prep
 0,02% edges [kernel.kallsyms] [k] __mod_lruvec_page_state
 0,02% edges [kernel.kallsyms] [k] xas_find
 0,02% edges [kernel.kallsyms] [k] workingset_update_node
 0,02% edges [kernel.kallsyms] [k] create_empty_buffers
 0,02% edges [kernel.kallsyms] [k] __raw_spin_lock
 0,02% edges [kernel.kallsyms] [k] reweight_entity
 0,02% edges [kernel.kallsyms] [k] block_invalidate_folio
 0,02% edges [kernel.kallsyms] [k] unlink_anon_vmas
 0,02% edges [kernel.kallsyms] [k] __this_cpu preempt_check
 0,02% edges [kernel.kallsyms] [k] preempt_count_sub
 0,00% edges [kernel.kallsyms] [k] security_bprm_committing_creds
 0,00% edges [kernel.kallsyms] [k] ct_nmi_exit
 0,00% edges [kernel.kallsyms] [k] native_sched_clock
 0,00% edges [kernel.kallsyms] [k] native_write_msr
<command>...
Any command you can specify in a shell.
```

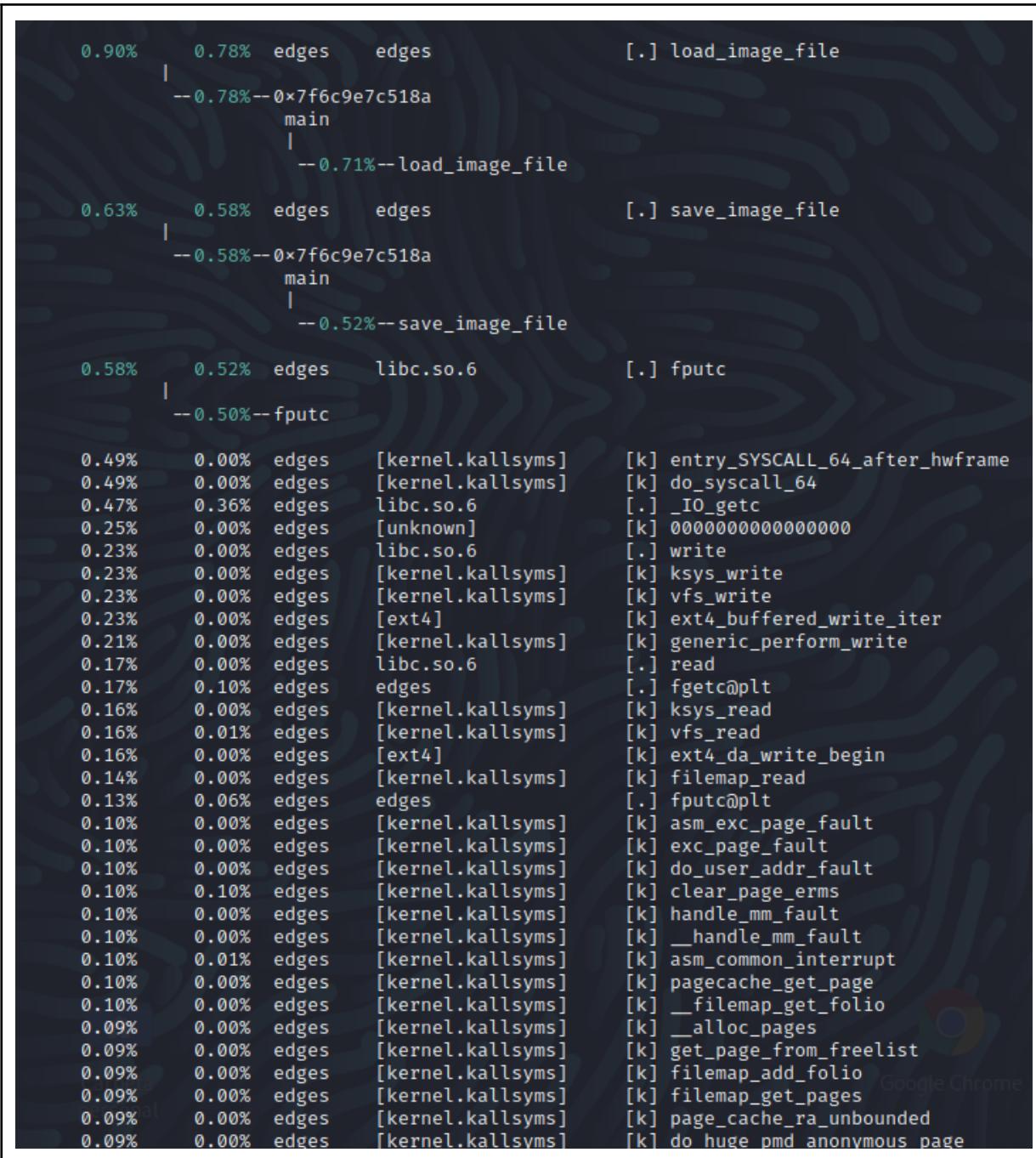
Obtén un grafo de llamadas añadiendo la opción `-g` a los comandos `perf record` y `perf report`. Observa la traza de eventos recogidos con `perf script`.

Indica cómo afecta la opción `-g` al volumen de información recogida.

`-g` Copia los resultados y escribe un breve análisis de los mismos.
Enables call-graph (stack chain/backtrace) recording for both kernel space and user space.

```
(dani㉿kali)-[~/Documentos/lab]
└─$ perf record -g ./edges img.pgm out.pgm
[ perf record: Woken up 2 times to write data ]
[ perf record: Captured and wrote 0,559 MB perf.data (6343 samples) ]
```

```
dani@kali: ~/Documentos/lab
Archivo Acciones Editar Vista Ayuda
# To display the perf.data header info, please use --header/--header-only options.
#
# Total Lost Samples: 0
#
# Samples: 6K of event 'cycles'
# Event count (approx.): 6162325396
#
# Children      Self  Command Shared Object      Symbol
# .....      .....  .....  .....  .....  .....
#
# 99.04%    0.00%  edges   libc.so.6          [.] 0x00007f6c9e7c518a
|   0x7f6c9e7c518a
|   main
|   |   97.03%--edges
|   |   |   67.95%--gaussian
|   |   |   -- 29.08%--laplacian
|   |   |   0.83%--load_image_file
|   |   |   -- 0.57%--save_image_file
|
# 99.04%    0.00%  edges   edges            [.] main
|   main
|   |   97.03%--edges
|   |   |   67.95%--gaussian
|   |   |   -- 29.08%--laplacian
|   |   |   0.83%--load_image_file
|   |   |   -- 0.57%--save_image_file
|
# 97.03%    0.00%  edges   edges            [.] edges
|   edges
|   |   67.95%--gaussian
|   |   -- 29.08%--laplacian
|
# 67.95%    67.78%  edges   edges            [.] gaussian
|   -- 67.78%--0x7f6c9e7c518a
|   |   main
|   |   edges
|   |   gaussian
|
# 29.08%    29.06%  edges   edges            [.] laplacian
|   -- 29.06%--0x7f6c9e7c518a
|   |   main
|   |   edges
```



```
(dani@kali)-[~/Documentos/lab]
└─$ perf script
edges 3115 276.973026:           1 cycles:
    ffffffff6478af6 native_write_msr+0x6 ([kernel.kallsyms])
    ffffffff640dc23 __intel_pmu_enable_all.constprop.0+0x83 ([kernel.kallsyms])
    ffffffff6668a62 perf_event_exec+0x492 ([kernel.kallsyms])
    ffffffff67659e2 begin_new_exec+0x5e2 ([kernel.kallsyms])
    ffffffff67dbe22 load_elf_binary+0x2c2 ([kernel.kallsyms])
    ffffffff6764b3f bprm_execve+0x27f ([kernel.kallsyms])
    ffffffff67650bd do_execveat_common.isra.0+0x1ad ([kernel.kallsyms])
    ffffffff6765f32 __x64_sys_execve+0x32 ([kernel.kallsyms])
    ffffffff6e13158 do_syscall_64+0x58 ([kernel.kallsyms])
    ffffffff6700009b entry_SYSCALL_64_after_hwframe+0x63 ([kernel.kallsyms])
    7f2a7eef3987 [unknown] ([unknown])

edges 3115 276.973048:           1 cycles:
    ffffffff6478af6 native_write_msr+0x6 ([kernel.kallsyms])
    ffffffff640dc23 __intel_pmu_enable_all.constprop.0+0x83 ([kernel.kallsyms])
    ffffffff6668a62 perf_event_exec+0x492 ([kernel.kallsyms])
    ffffffff67659e2 begin_new_exec+0x5e2 ([kernel.kallsyms])
    ffffffff67dbe22 load_elf_binary+0x2c2 ([kernel.kallsyms])
    ffffffff6764b3f bprm_execve+0x27f ([kernel.kallsyms])
    ffffffff67650bd do_execveat_common.isra.0+0x1ad ([kernel.kallsyms])
    ffffffff6765f32 __x64_sys_execve+0x32 ([kernel.kallsyms])
    ffffffff6e13158 do_syscall_64+0x58 ([kernel.kallsyms])
    ffffffff6700009b entry_SYSCALL_64_after_hwframe+0x63 ([kernel.kallsyms])
    7f2a7eef3987 [unknown] ([unknown])

edges 3115 276.973058:           4 cycles:
    ffffffff6478af6 native_write_msr+0x6 ([kernel.kallsyms])
    ffffffff640dc23 __intel_pmu_enable_all.constprop.0+0x83 ([kernel.kallsyms])
    ffffffff6668a62 perf_event_exec+0x492 ([kernel.kallsyms])
    ffffffff67659e2 begin_new_exec+0x5e2 ([kernel.kallsyms])
    ffffffff67dbe22 load_elf_binary+0x2c2 ([kernel.kallsyms])
    ffffffff6764b3f bprm_execve+0x27f ([kernel.kallsyms])
    ffffffff67650bd do_execveat_common.isra.0+0x1ad ([kernel.kallsyms])
    ffffffff6765f32 __x64_sys_execve+0x32 ([kernel.kallsyms])
    ffffffff6e13158 do_syscall_64+0x58 ([kernel.kallsyms])
    ffffffff6700009b entry_SYSCALL_64_after_hwframe+0x63 ([kernel.kallsyms])
    7f2a7eef3987 [unknown] ([unknown])

edges 3115 276.973064:          22 cycles:
    ffffffff6478af6 native_write_msr+0x6 ([kernel.kallsyms])
    ffffffff640dc23 __intel_pmu_enable_all.constprop.0+0x83 ([kernel.kallsyms])
    ffffffff6668a62 perf_event_exec+0x492 ([kernel.kallsyms])
    ffffffff67659e2 begin_new_exec+0x5e2 ([kernel.kallsyms])
    ffffffff67dbe22 load_elf_binary+0x2c2 ([kernel.kallsyms])
    ffffffff6764b3f bprm_execve+0x27f ([kernel.kallsyms])
    ffffffff67650bd do_execveat_common.isra.0+0x1ad ([kernel.kallsyms])
    ffffffff6765f32 __x64_sys_execve+0x32 ([kernel.kallsyms])
    ffffffff6e13158 do_syscall_64+0x58 ([kernel.kallsyms])
    ffffffff6700009b entry_SYSCALL_64_after_hwframe+0x63 ([kernel.kallsyms])
    7f2a7eef3987 [unknown] ([unknown])

edges 3115 276.973071:         120 cycles:
    ffffffff6478af6 native_write_msr+0x6 ([kernel.kallsyms])
    ffffffff640dc23 __intel_pmu_enable_all.constprop.0+0x83 ([kernel.kallsyms])
    ffffffff6668a62 perf_event_exec+0x492 ([kernel.kallsyms])
    ffffffff67659e2 begin_new_exec+0x5e2 ([kernel.kallsyms])
    ffffffff67dbe22 load_elf_binary+0x2c2 ([kernel.kallsyms])
```

Obtén un perfil de ejecución del programa anterior usando como evento los fallos de lectura de la *cache* de datos de primer nivel (opción `-e L1-dcache-load-misses`) y muestreando con frecuencia 10.000 (opción `-F 10000`).

```
-F, --freq=
    Profile at this frequency. Use max to use the currently maximum allowed frequency, i.e. the value in
    the kernel.perf_event_max_sample_rate sysctl. Will throttle down to the currently maximum allowed
    frequency. See --strict-freq.
```

```
(dani㉿kali)-[~/Documentos/lab]
$ perf record -e L1-dcache-load-misses -F 10000 ./edges img.pgm out.pgm
[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 0,233 MB perf.data (5512 samples) ]

(dani㉿kali)-[~/Documentos/lab]
$ perf report --stdio
# To display the perf.data header info, please use --header/--header-only options.
#
#
# Total Lost Samples: 0
#
# Samples: 5K of event 'L1-dcache-load-misses'
# Event count (approx.): 3615454
#
# Overhead  Command  Shared Object      Symbol
# .....  .....
#
 36.25% edges      edges            [.] gaussian
 25.63% edges      edges            [.] laplacian
 18.19% edges      [kernel.kallsyms] [k] clear_page_erms
  2.93% edges      [kernel.kallsyms] [k] copy_user_enhanced_fast_string
  2.58% edges      edges            [.] save_image_file
  1.37% edges      libc.so.6       [.] _IO_getc
  0.94% edges      libc.so.6       [.] fputc
  0.88% edges      [kernel.kallsyms] [k] get_mem_cgroup_from_mm
  0.80% edges      [kernel.kallsyms] [k] memcg_check_events
  0.69% edges      [kernel.kallsyms] [k] blk_cgroup_congested
  0.58% edges      [kernel.kallsyms] [k] __fput
  0.47% edges      [kernel.kallsyms] [k] check_preemption_disabled
  0.42% edges      [kernel.kallsyms] [k] get_page_from_freelist
  0.31% edges      [kernel.kallsyms] [k] __filemap_add_folio
  0.27% edges      [kernel.kallsyms] [k] try_charge_memcg
  0.23% edges      [kernel.kallsyms] [k] __x64_sys_write
  0.21% edges      ld-linux-x86-64.so.2 [.] 0x000000000000757f
  0.17% edges      [kernel.kallsyms] [k] vfs_write
  0.17% edges      [kernel.kallsyms] [k] update_curr
  0.17% edges      [kernel.kallsyms] [k] filemap_add_folio
  0.16% edges      [kernel.kallsyms] [k] release_pages
  0.16% edges      [kernel.kallsyms] [k] __rcu_read_unlock
  0.13% edges      [kernel.kallsyms] [k] folio_unlock
  0.13% edges      [kernel.kallsyms] [k] free_pcp_prepare
  0.13% edges      [kernel.kallsyms] [k] preempt_count_sub
  0.12% edges      [kernel.kallsyms] [k] find_get_pages_range_tag
  0.12% edges      [ext4]           [k] mpage_process_page_bufs
  0.12% edges      [ext4]           [k] ext4_fill_raw_inode
  0.11% edges      [kernel.kallsyms] [k] update_cfs_group
  0.11% edges      [kernel.kallsyms] [k] block_invalidate_folio
  0.11% edges      [kernel.kallsyms] [k] __fget_light
  0.11% edges      [kernel.kallsyms] [k] __folio_mark_dirty
  0.11% edges      [kernel.kallsyms] [k] __get_obj_cgroup_from_memcg
  0.11% edges      [jbd2]          [k] jbd2__journal_start
  0.11% edges      [kernel.kallsyms] [k] kmem_cache_alloc
  0.11% edges      [kernel.kallsyms] [k] preempt_count_add
  0.11% edges      [jbd2]          [k] add_transaction_credits
  0.11% edges      [jbd2]          [k] start_this_handle
  0.11% edges      [kernel.kallsyms] [k] __radix_tree_lookup
  0.11% edges      [kernel.kallsyms] [k] __update_load_avg_se
  0.10% edges      [kernel.kallsyms] [k] account_user_time
```

Tiene sentido, si miramos en el apartado anterior las ejecuciones de `./Edge` es muchísimo

mayor a los fallos que ha tenido por lectura en la cache.

La -F que indica la frecuencia, si quitamos este nos aparece un valor menor, es decir que ha tenido menos fallos de carga en la cache porque se ha ejecutado un menor número de veces.

Gaussian es la más afectada al poner la frecuencia a 10mil

A continuación sin la opción -F 10000

```
# To display the perf.data header info, please use --header/--header-only options.
#
# Total Lost Samples: 0
#
# Samples: 1K of event 'L1-dcache-load-misses'
# Event count (approx.): 2232389
#
# Overhead  Command  Shared Object      Symbol
# .....  .....
#
28.21% edges   [kernel.kallsyms]  [k] clear_page_ermss
26.88% edges   edges           [.] gaussian
10.98% edges   edges           [.] laplacian
 4.27% edges   [kernel.kallsyms]  [k] copy_user_enhanced_fast_string
 3.47% edges   [kernel.kallsyms]  [k] check_preemption_disabled
 3.21% edges   edges           [.] save_image_file
 2.28% edges   [kernel.kallsyms]  [k] get_page_from_freelist
 1.83% edges   edges           [.] load_image_file
 1.32% edges   [kernel.kallsyms]  [k] get_mem_cgroup_from_mm
 1.17% edges   [kernel.kallsyms]  [k] copy_page
 1.09% edges   [kernel.kallsyms]  [k] __filemap_add_folio
 0.89% edges   [kernel.kallsyms]  [k] node_dirty_ok
 0.66% edges   libc.so.6        [.] fputc
 0.52% edges   [kernel.kallsyms]  [k] update_vsyscall
 0.44% edges   [kernel.kallsyms]  [k] preempt_count_add
 0.44% edges   [kernel.kallsyms]  [k] _raw_spin_unlock_irqrestore
 0.43% edges   [kernel.kallsyms]  [k] _raw_spin_lock
 0.38% edges   [kernel.kallsyms]  [k] mem_cgroup_css_rstat_flush
 0.27% edges   [kernel.kallsyms]  [k] free_swap_cache
 0.26% edges   [kernel.kallsyms]  [k] cgroup_rstat_updated
 0.26% edges   [kernel.kallsyms]  [k] page_counter_uncharge
 0.25% edges   [ext4]            [k] ext4_writepages
 0.25% edges   [kernel.kallsyms]  [k] find_lock_entries
 0.24% edges   [kernel.kallsyms]  [k] __free_one_page
 0.24% edges   [kernel.kallsyms]  [k] _raw_spin_lock_irqsave
 0.23% edges   [kernel.kallsyms]  [k] unmap_page_range
 0.23% edges   [kernel.kallsyms]  [k] __queue_work
 0.23% edges   [kernel.kallsyms]  [k] balance_dirty_pages_ratelimited_flags
 0.23% edges   [kernel.kallsyms]  [k] create_empty_buffers
 0.22% edges   [ext4]            [k] ext4_dirty_inode
 0.22% edges   [kernel.kallsyms]  [k] __filemap_get_folio
 0.22% edges   [kernel.kallsyms]  [k] vfs_write
 0.22% edges   [kernel.kallsyms]  [k] __count_memcg_events
 0.22% edges   [kernel.kallsyms]  [k] charge_memcg
 0.22% edges   [kernel.kallsyms]  [k] __rcu_read_lock
 0.22% edges   [kernel.kallsyms]  [k] __dquot_alloc_space
 0.22% edges   [kernel.kallsyms]  [k] generic_write_end
 0.22% edges   libc.so.6        [.] write
 0.22% edges   [kernel.kallsyms]  [k] __folio_mark_dirty
 0.22% edges   [kernel.kallsyms]  [k] find_get_pages_range_tag
 0.22% edges   [kernel.kallsyms]  [k] syscall_return_via_sysret
 0.22% edges   [kernel.kallsyms]  [k] __mark_inode_dirty
 0.22% edges   [kernel.kallsyms]  [k] percpu_counter_add_batch
 0.22% edges   [ext4]            [k] ext4_fill_raw_inode
 0.22% edges   [kernel.kallsyms]  [k] memcg_slab_post_alloc_hook
 0.22% edges   [kernel.kallsyms]  [k] __wake_up_common
 0.22% edges   [kernel.kallsyms]  [k] node_page_state
```

Obtén un perfil de ejecución del programa anterior usando como evento los fallos del predictor de saltos (opción `-e branch-misses`) y un periodo de 1 evento (opción `-c 1`).

Copia los resultados y escribe un breve análisis de los mismos.

```
(dani㉿kali)-[~/Documentos/lab]
└─$ perf record -e branch-misses -c 1 ./edges img.pgm out.pgm
[ perf record: Woken up 7 times to write data ]
[ perf record: Captured and wrote 1,593 MB perf.data (50966 samples) ]

(dani㉿kali)-[~/Documentos/lab]
└─$ perf report --stdio
# To display the perf.data header info, please use --header/--header-only options.
#
#
# Total Lost Samples: 0
#
# Samples: 50K of event 'branch-misses'
# Event count (approx.): 50966
#
# Overhead  Command  Shared Object      Symbol
# .....  .....
#
  55.30%  edges     edges           [.] laplacian
  14.20%  edges     edges           [.] gaussian
  1.83%  edges     [kernel.kallsyms] [k] read_tsc
  1.30%  edges     [kernel.kallsyms] [k] check_preemption_disabled
  0.97%  edges     [kernel.kallsyms] [k] update_load_avg
  0.76%  edges     [kernel.kallsyms] [k] hrtimer_interrupt
  0.76%  edges     [kernel.kallsyms] [k] preempt_count_add
  0.72%  edges     [kernel.kallsyms] [k] sysvec_apic_timer_interrupt
  0.70%  edges     [kernel.kallsyms] [k] lapic_next_deadline
  0.60%  edges     [kernel.kallsyms] [k] __update_load_avg_cfs_rq
  0.54%  edges     [kernel.kallsyms] [k] __update_load_avg_se
  0.52%  edges     [kernel.kallsyms] [k] task_tick_fair
  0.50%  edges     [kernel.kallsyms] [k] __irqentry_text_end
  0.50%  edges     [kernel.kallsyms] [k] __hrtimer_next_event_base
  0.47%  edges     [kernel.kallsyms] [k] tick_sched_timer
  0.43%  edges     [kernel.kallsyms] [k] __this_cpu preempt_check
  0.40%  edges     [kernel.kallsyms] [k] update_blocked_averages
  0.40%  edges     [kernel.kallsyms] [k] update_curr
  0.39%  edges     [kernel.kallsyms] [k] __hrtimer_run_queues
  0.39%  edges     [kernel.kallsyms] [k] __irq_exit_rcu
  0.38%  edges     [kernel.kallsyms] [k] update_sd_lb_stats.constprop.0
  0.34%  edges     [kernel.kallsyms] [k] exit_to_user_mode_prepare
  0.34%  edges     [kernel.kallsyms] [k] select_task_rq_fair
  0.32%  edges     [kernel.kallsyms] [k] irq_exit_rcu
  0.31%  edges     [i915]   [k] execlists_submission_tasklet
  0.31%  edges     [kernel.kallsyms] [k] psi_group_change
  0.27%  edges     [kernel.kallsyms] [k] rebalance_domains
  0.27%  edges     [kvm]    [k] pvclock_gtod_notify
  0.26%  edges     [kernel.kallsyms] [k] __raw_spin_lock_irq
  0.26%  edges     [kernel.kallsyms] [k] debug_smp_processor_id
  0.25%  edges     [kernel.kallsyms] [k] trigger_load_balance
  0.25%  edges     [kernel.kallsyms] [k] hrtimer_update_next_event
  0.24%  edges     [kernel.kallsyms] [k] native_apic_msr_eoi_write
  0.24%  edges     [kernel.kallsyms] [k] rb_erase
  0.23%  edges     [kernel.kallsyms] [k] preempt_count_sub
  0.22%  edges     [kernel.kallsyms] [k] __softirqentry_text_start
  0.21%  edges     [kernel.kallsyms] [k] load_balance
  0.20%  edges     [kernel.kallsyms] [k] rCU_sched_clock_irq
  0.19%  edges     [kernel.kallsyms] [k] check_preempt_wakeup
  0.18%  edges     [i915]   [k] process_csb
  0.18%  edges     [kernel.kallsyms] [k] __sysvec_apic_timer_interrupt
```

Ahora vemos que el número de saltos predichos fallidos es de 50 mil, algo infimo pero que la mayoría de estos los provoca laplacian, a diferencia de las cargas en la cache que las provocaba gaussian.

Tiene sentido que siga siendo un valor relativamente pequeño.

Obtén el tiempo de ejecución del programa sin instrumentación y con ella:

```
$ time -p ./edges img.pgm out.pgm
$ perf record time -p ./edges img.pgm out.pgm
```

Calcula la sobrecarga producida por la instrumentación como la diferencia entre los tiempos de ejecución con y sin instrumentación, dividida por el tiempo de ejecución con instrumentación y multiplicada por 100, en porcentaje.

Escribe los tiempos de ejecución y la sobrecarga.

Aparecía un error al ejecutar el segundo comando.

ACORDARSE: Instalar time

```
—(dani@kali)-[~/Documentos/lab]
$ time -p ./edges img.pgm out.pgm
-p: no se encontró la orden

real    0,06s
user    0,05s
sys     0,01s
cpu     99%

—(dani@kali)-[~/Documentos/lab]
$ perf record time -p ./edges img.pgm out.pgm
real 1.43
user 1.43
sys 0.00
[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 0,243 MB perf.data (5747 samples) ]
```

He abierto el fichero, para comprobar que se había ejecutado ya que el tiempo era demasiado elevado en comparación y tenía dudas.

```
—(dani@kali)-[~/Documentos/lab]
$ perf script
time 17047 3261.359190:          1 cycles: ffffffff6478af4 native_write_msr+0x4 ([kernel.kallsyms])
time 17047 3261.359195:          1 cycles: ffffffff6478af4 native_write_msr+0x4 ([kernel.kallsyms])
time 17047 3261.359197:         12 cycles: ffffffff6478af4 native_write_msr+0x4 ([kernel.kallsyms])
time 17047 3261.359198:        326 cycles: ffffffff6478af6 native_write_msr+0x6 ([kernel.kallsyms])
time 17047 3261.359199:       9530 cycles: ffffffff6db7152 nmi_cpu_backtrace+0x2 ([kernel.kallsyms])
time 17047 3261.359202:      255486 cycles: ffffffff683ae67 security_bprm_committing_creds+0x7 ([kernel.kallsyms])
time 17047 3261.359265:     2988072 cycles: ffffffff6530e45 __rcu_read_unlock+0x15 ([kernel.kallsyms])
edges 17048 3261.359568:          1 cycles: ffffffff6478af4 native_write_msr+0x4 ([kernel.kallsyms])
edges 17048 3261.359570:          1 cycles: ffffffff6478af4 native_write_msr+0x4 ([kernel.kallsyms])
```

Sobrecarga producida por instrumentación.

Diferencia de los tiempos de ejecución= $>1.43-0.06=1.37$

$1.37/1.43=0.958 \rightarrow 95.8\%$

Callgrind (~15 min.)

Si usas una máquina virtual, instala valgrind con:

```
$ sudo apt-get update
$ sudo apt-get install valgrind
```

Consulta la página de manual de valgrind y callgrind_annotation.

-Valgrind: programa para debugear y crear perfiles de ejecutables. Tiene varias herramientas de debug y es modular por lo que permite crear fácilmente herramientas nuevas.

-Callgrind_annotation: coge el archivo de salida generado por el comando valgrind y pinta la información de una forma fácil de leer.

Obtén un perfil de ejecución con simulación de la memoria **cache** de los programas **matrix1** y **matrix2** con la herramienta **callgrind**:

```
$ valgrind --tool=callgrind --cache-sim=yes ./matrix1
$ callgrind_annotation callgrind.out.<PID>
```

- **Ir**: I cache reads (instructions executed)
- **I1mr**: I1 cache read misses (instruction wasn't in I1 cache but was in L2)
- **I2mr**: L2 cache instruction read misses (instruction wasn't in I1 or L2 cache, had to be fetched from memory)
- **Dr**: D cache reads (memory reads)
- **D1mr**: D1 cache read misses (data location not in D1 cache, but in L2)
- **D2mr**: L2 cache data read misses (location not in D1 or L2)
- **Dw**: D cache writes (memory writes)
- **D1mw**: D1 cache write misses (location not in D1 cache, but in L2)
- **D2mw**: L2 cache data write misses (location not in D1 or L2)

matrix1

```
usuario@debian:~/Escritorio/laboratorio$ valgrind --tool=callgrind --cache-sim=yes ./matrix1
==3312== Callgrind, a call-graph generating cache profiler
==3312== Copyright (C) 2002-2011, and GNU GPL'd, by Josef Weidendorfer et al.
==3312== Using Valgrind-3.7.0 and LibVEX; rerun with -h for copyright info
==3312== Command: ./matrix1
==3312==
--3312-- warning: L3 cache found, using its data for the LL simulation.
==3312== For interactive control, run 'callgrind_control -h'.
Result: 127840000.000000
==3312==
==3312== Events      : Ir Dr Dw I1mr D1mr D1mw ILmr DLmr DLmw
==3312== Collected   : 15906684820 4104353499 1029137207 909 529197938 800500 901 81165 240476
==3312==
==3312== I    refs:      15,906,684,820
==3312== I1   misses:          909
==3312== LLi misses:          901
==3312== I1   miss rate:     0.0%
==3312== LLi miss rate:     0.0%
==3312==
==3312== D    refs:      5,133,490,706  (4,104,353,499 rd + 1,029,137,207 wr)
==3312== D1   misses:      529,998,438  ( 529,197,938 rd +      800,500 wr)
==3312== LLd misses:      321,641  (      81,165 rd +      240,476 wr)
==3312== D1   miss rate:    10.3% (      12.8% +      0.0% )
==3312== LLd miss rate:    0.0% (      0.0% +      0.0% )
==3312==
==3312== LL  refs:      529,999,347  ( 529,198,847 rd +      800,500 wr)
==3312== LL  misses:      322,542  (      82,066 rd +      240,476 wr)
==3312== LL  miss rate:    0.0% (      0.0% +      0.0% )
```

```

usuario@debian:~/Escritorio/laboratorio$ callgrind_annotate callgrind.out.3312
-----
Profile data file 'callgrind.out.3312' (creator: callgrind-3.7.0)
-----
I1 cache: 32768 B, 64 B, 8-way associative
D1 cache: 49152 B, 64 B, 12-way associative
LL cache: 8388608 B, 64 B, 8-way associative
Timerange: Basic block 0 - 515228503
Trigger: Program termination
Profiled target: ./matrix1 (PID 3312, part 1)
Events recorded: Ir Dr Dw I1mr D1mr D1mw ILmr DLmr DLmw
Events shown: Ir Dr Dw I1mr D1mr D1mw ILmr DLmr DLmw
Event sort order: Ir Dr Dw I1mr D1mr D1mw ILmr DLmr DLmw
Thresholds: 99 0 0 0 0 0 0 0 0
Include dirs:
User annotated:
Auto-annotation: off

----- Ir Dr Dw I1mr D1mr D1mw ILmr DLmr DLmw -----
15,906,684,823 4,104,353,499 1,029,137,207 909 529,197,938 800,500 901 81,165 240,476 PROGRAM TOTALS

----- Ir Dr Dw I1mr D1mr D1mw ILmr DLmr DLmw file:function -----
15,906,576,829 4,104,324,810 1,029,124,808 6 529,196,652 800,001 6 80,002 240,001 ????:main [/home/usuario/Escritorio/laboratorio/matrix1]

```

Se puede ver que los resultados obtenidos con este programa son muy similares a los que se obtuvieron utilizando perf, aunque en Dr el valor es un poco más pequeño utilizando este programa que usando perf.

matrix2

```

usuario@debian:~/Escritorio/laboratorio$ valgrind --tool=callgrind --cache-sim=yes ./matrix2
==3346== Callgrind, a call-graph generating cache profiler
==3346== Copyright (C) 2002-2011, and GNU GPL'd, by Josef Weidendorfer et al.
==3346== Using Valgrind-3.7.0 and LibVEX; rerun with -h for copyright info
==3346== Command: ./matrix2
==3346==
--3346-- warning: L3 cache found, using its data for the LL simulation.
==3346== For interactive control, run 'callgrind_control -h'.
Result: 127840000.000000
==3346==
==3346== Events : Ir Dr Dw I1mr D1mr D1mw ILmr DLmr DLmw
==3346== Collected : 15923970424 4108195100 1030418808 913 64162092 880501 904 117789 321265
==3346==
==3346== I refs: 15,923,970,424
==3346== I1 misses: 913
==3346== LLi misses: 904
==3346== I1 miss rate: 0.0%
==3346== LLi miss rate: 0.0%
==3346==
==3346== D refs: 5,138,613,908 (4,108,195,100 rd + 1,030,418,808 wr)
==3346== D1 misses: 65,042,593 ( 64,162,092 rd + 880,501 wr)
==3346== LLd misses: 439,054 ( 117,789 rd + 321,265 wr)
==3346== D1 miss rate: 1.2% ( 1.5% + 0.0% )
==3346== LLd miss rate: 0.0% ( 0.0% + 0.0% )
==3346==
==3346== LL refs: 65,043,506 ( 64,163,005 rd + 880,501 wr)
==3346== LL misses: 439,958 ( 118,693 rd + 321,265 wr)
==3346== LL miss rate: 0.0% ( 0.0% + 0.0% )

```

```

usuario@debian:~/Escritorio/laboratorio$ callgrind_annotate callgrind.out.3346
-----
Profile data file 'callgrind.out.3346' (creator: callgrind-3.7.0)
-----
I1 cache: 32768 B, 64 B, 8-way associative
D1 cache: 49152 B, 64 B, 12-way associative
LL cache: 8388608 B, 64 B, 8-way associative
Timerange: Basic block 0 - 515870905
Trigger: Program termination
Profiled target: ./matrix2 (PID 3346, part 1)
Events recorded: Ir Dr Dw I1mr D1mr D1mw ILmr DLmr DLmw
Events shown: Ir Dr Dw I1mr D1mr D1mw ILmr DLmr DLmw
Event sort order: Ir Dr Dw I1mr D1mr D1mw ILmr DLmr DLmw
Thresholds: 99 0 0 0 0 0 0 0 0
Include dirs:
User annotated:
Auto-annotation: off

----- Ir Dr Dw I1mr D1mr D1mw ILmr DLmr DLmw -----
15,923,970,427 4,108,195,100 1,030,418,808 913 64,162,092 880,501 904 117,789 321,265 PROGRAM TOTALS

----- Ir Dr Dw I1mr D1mr D1mw ILmr DLmr DLmw file:function -----
15,923,862,433 4,108,166,411 1,030,406,409 9 64,160,804 880,002 9 116,626 320,790 ????:main [/home/usuario/Escritorio/laboratorio/matrix2]

```

Del mismo modo en todos los casos aparecen valores muy similares entre las dos opciones, siendo la única excepción el valor de Dr, que es un poco menor utilizando callgrind.

Copia los resultados y escribe un breve análisis de los mismos comparándolos con los obtenidos con perf.

Compara google-pprof, perf y callgrind en cuanto a implementación, sobrecarga, precisión, exactitud, ámbito de medida, resolución temporal y facilidad de uso.

- Sobrecarga: google-pprof no cuenta con una gran sobrecarga en comparación con perf o con callgrind que cuentan con una gran sobrecarga.
- Precisión y exactitud: de los tres el que más precisión da es callgrind.
- Ámbito de medida:
- Resolución temporal: callgrind tiene una frecuencia por defecto 10 y google-pprof de 100 mientras que perf tiene una frecuencia por defecto de 1000 o 4000.
- Facilidad de uso: el más difícil de usar puede ser google pprof en el sentido de que necesitas saberte muchas opciones para el comando, aunque todos funcionan de una forma muy similar.

strace (~15 min.)

Si usas una máquina virtual, instala strace con:

```
$ sudo apt-get update
$ sudo apt-get install strace
```

Consulta la página de manual de strace.

-strace: ejecuta un programa y recoge las llamadas y señales de cada proceso. El nombre de cada llamada al sistema, los argumentos y el valor de retorno se pintan por salida de error estándar o por un fichero que se especifique con la opción -o.

Observa los ficheros que abre la herramienta vmstat trazando la llamada open() (opción -e /open*). Haz lo mismo con ps, top y /usr/lib/sysstat/sadc.

Indica de dónde obtienen la información de monitorización.

Con la opción -e /open nos fijamos solo en las llamadas de tipo open que se hagan.

vmstat

```
usuario@debian:~/Escritorio/laboratorio$ strace -e open vmstat
open("/etc/ld.so.cache", O_RDONLY)      = 3
open("/lib/x86_64-linux-gnu/libprocps.so.0", O_RDONLY) = 3
open("/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY) = 3
open("/proc/stat", O_RDONLY|O_CLOEXEC) = 3
open("/usr/lib/locale/locale-archive", O_RDONLY) = 3
open("/usr/share/locale/locale.alias", O_RDONLY) = 3
open("/usr/share/locale/es_ES.UTF-8/LC_MESSAGES/procps-ng.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
open("/usr/share/locale/es_ES.utf8/LC_MESSAGES/procps-ng.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
open("/usr/share/locale/es_ES/LC_MESSAGES/procps-ng.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
open("/usr/share/locale/es.UTF-8/LC_MESSAGES/procps-ng.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
open("/usr/share/locale/es.utf8/LC_MESSAGES/procps-ng.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
open("/usr/share/locale/es/LC_MESSAGES/procps-ng.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
procs -----memory----- swap-- io--- system-- cpu---
   r b swpd free  buff cache si so bi bo in cs us sy id wa
open("/proc/meminfo", O_RDONLY)      = 3
open("/proc/stat", O_RDONLY)        = 4
open("/proc/vmstat", O_RDONLY)      = 5
0 0 0 181692 24108 696784 0 0 41 143 78 7457 13 3 85 0
```

ps

```
usuario@debian:~/Escritorio/laboratorio$ strace -e open ps
open("/etc/ld.so.cache", O_RDONLY)      = 3
open("/lib/x86_64-linux-gnu/libprocps.so.0", O_RDONLY) = 3
open("/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY) = 3
open("/proc/stat", O_RDONLY|O_CLOEXEC)   = 3
open("/usr/lib/locale/locale-archive", O_RDONLY) = 3
open("/proc/self/stat", O_RDONLY)        = 3
open("/proc/uptime", O_RDONLY)           = 3
open("/usr/share/locale/locale.alias", O_RDONLY) = 4
open("/usr/share/locale/es_ES.UTF-8/LC_MESSAGES/procps-ng.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
open("/usr/share/locale/es_ES.utf8/LC_MESSAGES/procps-ng.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
open("/usr/share/locale/es_ES/LC_MESSAGES/procps-ng.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
open("/usr/share/locale/es.UTF-8/LC_MESSAGES/procps-ng.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
open("/usr/share/locale/es.utf8/LC_MESSAGES/procps-ng.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
open("/usr/share/locale/es/LC_MESSAGES/procps-ng.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
open("/proc/sys/kernel/pid_max", O_RDONLY) = 4
open("/proc/meminfo", O_RDONLY)          = 4
open("/proc", O_RDONLY|O_NONBLOCK|O_DIRECTORY|O_CLOEXEC) = 5
open("/proc/1/stat", O_RDONLY)           = 6
open("/proc/1/status", O_RDONLY)         = 6
open("/proc/2/stat", O_RDONLY)           = 6
open("/proc/2/status", O_RDONLY)         = 6
open("/proc/3/stat", O_RDONLY)           = 6
open("/proc/3/status", O_RDONLY)         = 6
open("/proc/6/stat", O_RDONLY)           = 6
open("/proc/6/status", O_RDONLY)         = 6
open("/proc/7/stat", O_RDONLY)           = 6
open("/proc/7/status", O_RDONLY)         = 6
open("/proc/8/stat", O_RDONLY)           = 6
open("/proc/8/status", O_RDONLY)         = 6
open("/proc/9/stat", O_RDONLY)           = 6
open("/proc/9/status", O_RDONLY)         = 6
open("/proc/10/stat", O_RDONLY)          = 6
open("/proc/10/status", O_RDONLY)         = 6
open("/proc/11/stat", O_RDONLY)          = 6
```

top

```
usuario@debian:~$ strace -e open top
open("/etc/ld.so.cache", O_RDONLY)      = 3
open("/lib/x86_64-linux-gnu/libprocps.so.0", O_RDONLY) = 3
open("/lib/x86_64-linux-gnu/libncurses.so.5", O_RDONLY) = 3
open("/lib/x86_64-linux-gnu/libtinfo.so.5", O_RDONLY) = 3
open("/lib/x86_64-linux-gnu/libdl.so.2", O_RDONLY) = 3
open("/proc/stat", O_RDONLY|O_CLOEXEC)   = 3
open("/proc/self/stat", O_RDONLY)        = 3
open("/usr/lib/locale/locale-archive", O_RDONLY) = 3
open("/usr/share/locale/locale.alias", O_RDONLY) = 3
open("/usr/share/locale/es_ES.UTF-8/LC_MESSAGES/procps.ng.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
open("/usr/share/locale/es_ES.utf8/LC_MESSAGES/procps.ng.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
open("/usr/share/locale/es_ES/LC_MESSAGES/procps.ng.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
open("/usr/share/locale/es.UTF-8/LC_MESSAGES/procps.ng.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
open("/usr/share/locale/es.utf8/LC_MESSAGES/procps.ng.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
open("/usr/share/locale/es/LC_MESSAGES/procps.ng.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
open("/etc/toprc", O_RDONLY)             = -1 ENOENT (No such file or directory)
open("/home/usuario/.toprc", O_RDONLY)   = -1 ENOENT (No such file or directory)
open("/usr/lib/x86_64-linux-gnu/gconv/gconv-modules.cache", O_RDONLY) = 3
open("/lib/terminfo/x/xterm", O_RDONLY)  = 3
open("/proc/sys/kernel/pid_max", O_RDONLY) = 3
open("/proc", O_RDONLY|O_NONBLOCK|O_DIRECTORY|O_CLOEXEC) = 3
open("/proc/1/stat", O_RDONLY)           = 4
open("/proc/1/statm", O_RDONLY)          = 4
open("/etc/nsswitch.conf", O_RDONLY)    = 4
open("/etc/ld.so.cache", O_RDONLY)       = 4
open("/lib/x86_64-linux-gnu/libnss_compat.so.2", O_RDONLY) = 4
open("/lib/x86_64-linux-gnu/libnsl.so.1", O_RDONLY) = 4
open("/etc/ld.so.cache", O_RDONLY)       = 4
open("/lib/x86_64-linux-gnu/libnss_nis.so.2", O_RDONLY) = 4
open("/lib/x86_64-linux-gnu/libnss_files.so.2", O_RDONLY) = 4
open("/etc/passwd", O_RDONLY|O_CLOEXEC) = 4
open("/proc/2/stat", O_RDONLY)           = 4
```

```
/usr/lib/sysstat/sadc
```

```
usuario@debian:~$ strace -e open /usr/lib/sysstat/sadc
open("/etc/ld.so.cache", O_RDONLY)      = 3
open("/usr/lib/x86_64-linux-gnu/libsensors.so.4", O_RDONLY) = 3
open("/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY) = 3
open("/lib/x86_64-linux-gnu/libm.so.6", O_RDONLY) = 3
open("/sys/class/i2c-adapter", O_RDONLY|O_NONBLOCK|O_DIRECTORY|O_CLOEXEC) = 3
open("/sys/class/i2c-adapter/i2c-0/name", O_RDONLY) = 4
open("/sys/class/hwmon", O_RDONLY|O_NONBLOCK|O_DIRECTORY|O_CLOEXEC) = 3
open("/etc/sensors3.conf", O_RDONLY)     = 3
open("/etc/sensors.d", O_RDONLY|O_NONBLOCK|O_DIRECTORY|O_CLOEXEC) = 3
open("/usr/lib/locale/locale-archive", O_RDONLY) = 3
open("/sys/devices/system/cpu", O_RDONLY|O_NONBLOCK|O_DIRECTORY|O_CLOEXEC) = 3
open("/proc/stat", O_RDONLY)            = 3
open("/proc/tty/driver/serial", O_RDONLY) = -1 EACCES (Permission denied)
open("/proc/diskstats", O_RDONLY)        = 3
open("/proc/net/dev", O_RDONLY)          = 3
open("/proc/net/dev", O_RDONLY)          = 3
open("/sys/devices/system/cpu", O_RDONLY|O_NONBLOCK|O_DIRECTORY|O_CLOEXEC) = 3
open("/sys/devices/system/cpu", O_RDONLY|O_NONBLOCK|O_DIRECTORY|O_CLOEXEC) = 3
open("/sys/devices/system/cpu/cpu0/cpufreq/stats/time_in_state", O_RDONLY) = -1 ENOENT (No such file or directory)
open("/sys/bus/usb/devices", O_RDONLY|O_NONBLOCK|O_DIRECTORY|O_CLOEXEC) = 3
q!
open("/etc/localtime", O_RDONLY)         = 4
Linuxdebian3.2.0-4-amd64x86_64
usuario@debian:~$
```

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

```
$ find /etc > /dev/null
```

Copia los resultados y escribe un breve análisis de los mismos.

```
strace -c find /etc > /dev/null
```

```

usuario@debian:~$ strace -c find /etc > /dev/null
find: «/etc/ssl/private»: Permiso denegado
find: «/etc/polkit-1/localauthority»: Permiso denegado
% time      seconds  usecs/call   calls  errors syscall
----- -----
98.58    0.001667     10       175          1 newfstatat
  0.89    0.000015      0       370          12 open
  0.53    0.000009      0       346          1 getdents
  0.00    0.000000      0        7          1 read
  0.00    0.000000      0       20          1 write
  0.00    0.000000      0       359          1 close
  0.00    0.000000      0       184          1 fstat
  0.00    0.000000      0       24          1 mmap
  0.00    0.000000      0       12          1 mprotect
  0.00    0.000000      0        3          1 munmap
  0.00    0.000000      0        5          1 brk
  0.00    0.000000      0        2          1 rt_sigaction
  0.00    0.000000      0        1          1 rt_sigprocmask
  0.00    0.000000      0        3          2 ioctl
  0.00    0.000000      0        6          6 access
  0.00    0.000000      0        1          1 execve
  0.00    0.000000      0        1          1 uname
  0.00    0.000000      0       349          1 fchdir
  0.00    0.000000      0        1          1 getrlimit
  0.00    0.000000      0        1          1 arch_prctl
  0.00    0.000000      0        2          1 futex
  0.00    0.000000      0        1          1 set_tid_address
  0.00    0.000000      0        1          1 set_robust_list
----- -----
100.00   0.001691     1874          21 total

```

Se puede ver que hay muchas llamadas a open y close, lo que tiene sentido porque se está recorriendo un directorio, motivo por el que también hay muchas llamadas a fstat y newfstatat, que sirven para ver información sobre directorios y archivos. También se puede ver que hay muchas llamadas a fchdir, supongo que para ir recorriendo los distintos subdirectorios. Por último se puede ver que no hay muchas llamadas a write porque como la información se está enviando a /dev/null se está descartando directamente.