Ftrace y kernelshark

Diseño y Evaluación de Configuraciones

Curso 2012-13

Miguel Telleria de Esteban

`telleriam AT unican.es`

Computadores y Tiempo Real

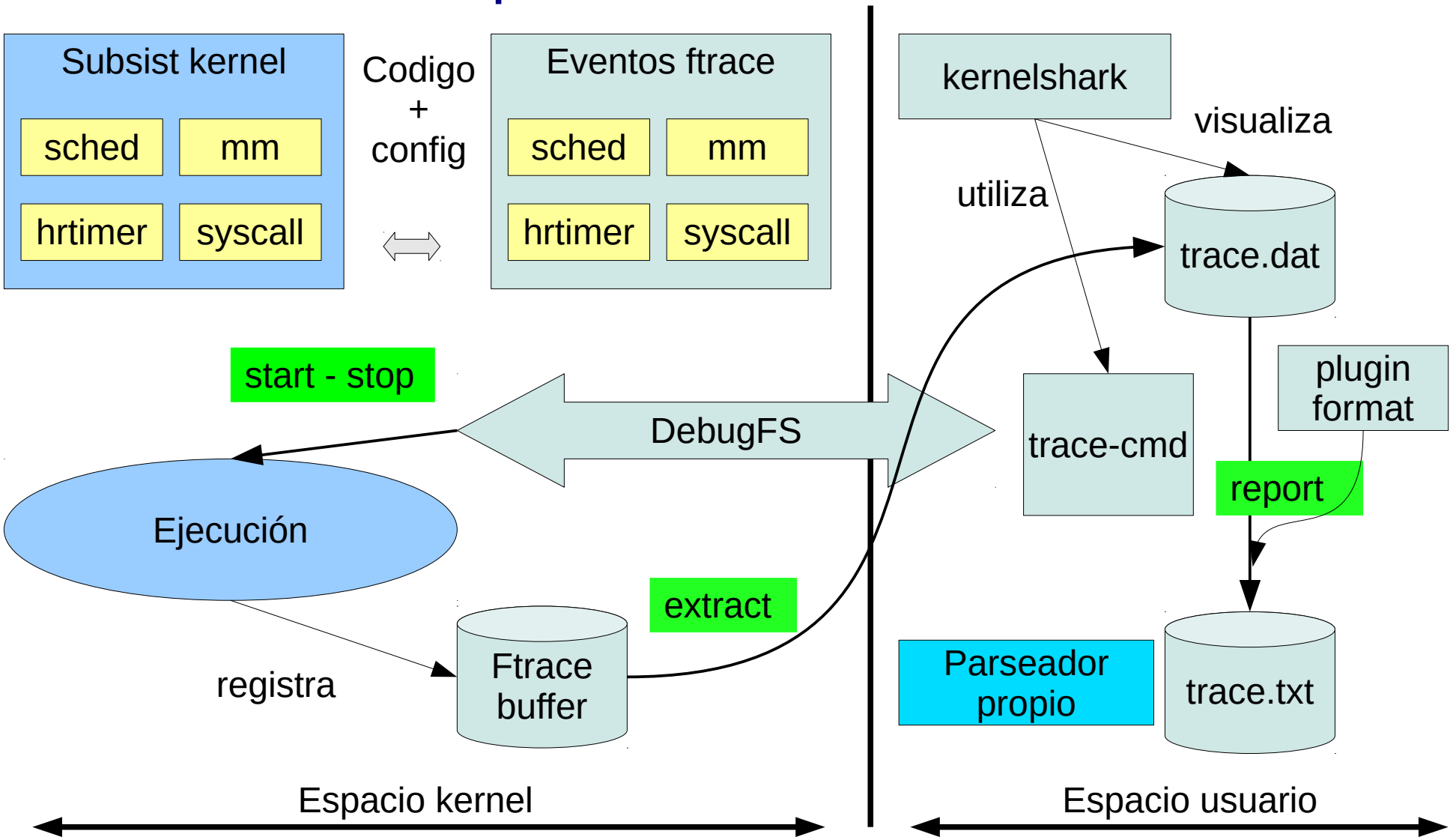
http://www.ctr.unican.es

Ftrace y terminología

Ftrace

- Infraestructura para visualizar lo que hace el kernel de linux a bajo nivel
  - Eventos de planificación
  - Eventos de gestión de memoria
  - Interrupciones
  - …
- Aplicaciones
  - Debugueo del kernel
  - Ver la planificación de procesos
  - Uso de memoria, disco duro
  - System calls
  - …
- Viene de serie con kernel (> 2.6.30)
  - Con cada versión de kernel tenemos más eventos
  - Mantenedor: Steven Rodstedt (contribuidor a RT_PREEMPT)

Arquitectura de Ftrace



Eventos, plugins y tracers

- Los **eventos** se organizan por subsistemas del kernel. En un 2.6.32 (Debian Squeeze).

  block ftrace i915 irq kmem module power

  sched skb (socket buffers) syscall timer workqueue

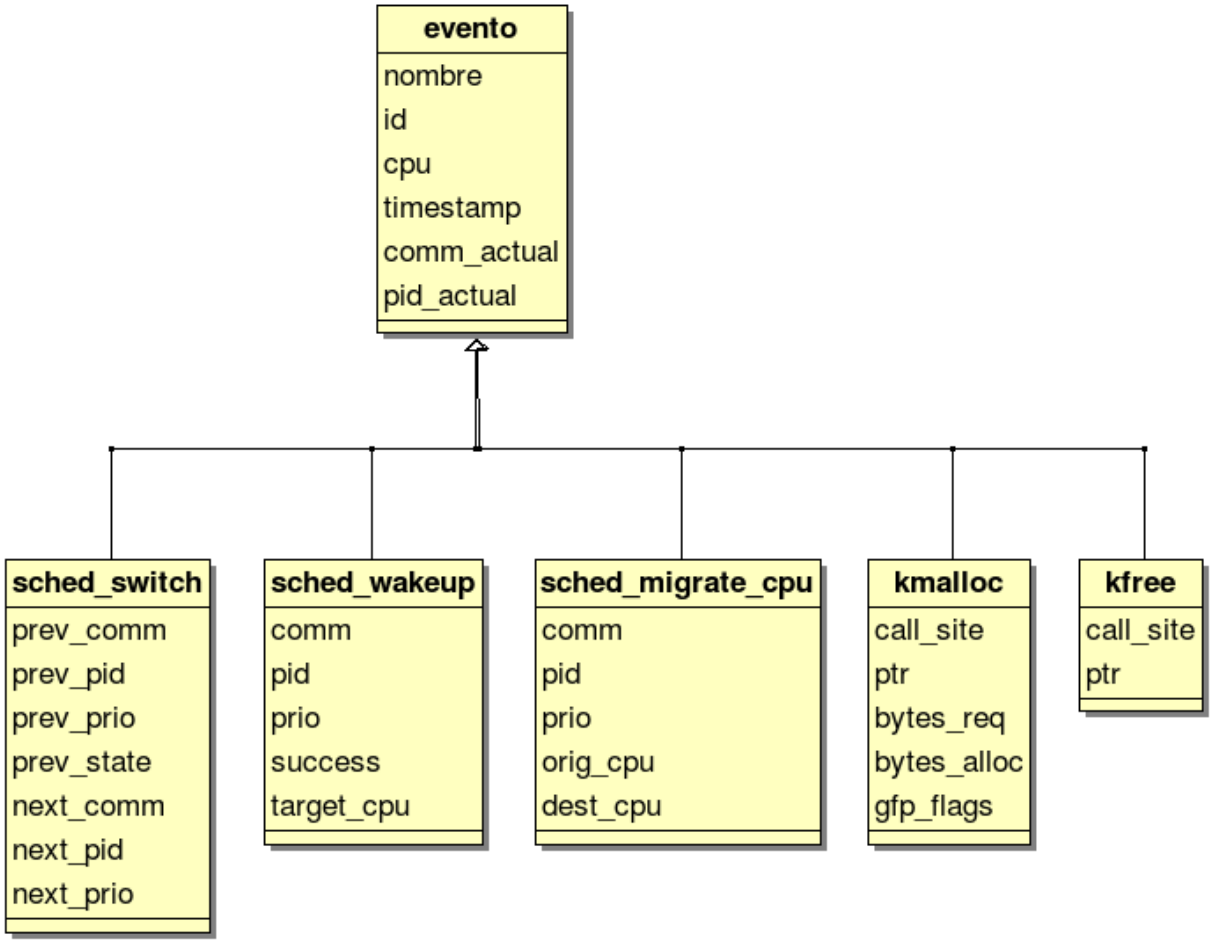
- La lista completa se obtiene con: `# trace-cmd list -e`
- Los **plugins** afectan a la presentación del texto para ciertos eventos. Se obtienen con: `# trace-cmd list -P`
  - Para un parseado homogéneo, recomiendo deshabilitarlos.
- Los **tracers** son programas que engloban a eventos y plugins
  - sched_switch **(obsoleto)** initcall (boot options initcall_debug ftrace=initcall)
  - blk nop (ningun plugin)
  - Otros tracers requieren de la recompilación para activarse:
    - Function function-graph irqsoff wakeuprt

Mas Información: https://wiki.linaro.org/KenWerner/Sandbox/ftrace

¿Quién define qué eventos están disponibles?

- A la hora de la captura y extracción: **El kernel y su config**
  - Eventos y tracers disponibles
    - Cuanto más avanzado el kernel, más eventos tenemos
  - Salida de esta fase: **trace.dat**
- A la hora del análisis: El fichero **trace.dat**
  - El fichero trace.dat auto-contiene la lista de eventos y parámetros.
  - El análisis se puede hacer desde otro ordenador con diferente:
    - Arquitectura (ej i386, amd64)
    - Kernel (ej 2.6.32, 3.2)
    - Endiannes (big-endian, little-endian)
- Ni kernelshark ni trace-cmd definen los eventos
  - Versiones nuevas de las herramientas ayudan al análisis no al contenido

Eventos y parametros



Para obtener la lista: trace-cmd report --events

Eventos para un diagrama de ejecución

- Eventos de la familia sched:
  - sched_wakeup
    - Un thread entra en la ready queue (aun no ejecuta)
  - sched_switch
    - Un thread comienza o deja de ejecutar
  - sched_migrate_task
    - Un thread pasa a la ready queue de otro core
- Latencia:
  - Diferencia de tiempo entre el sched_wakeup y el correspondiente sched_switch

Workflows

Trace-cmd y kernelshark

- Trace-cmd: Front-end consola de Ftrace via DebugFS
  - Selección de eventos, plugin-tracers y opciones
  - Activación y parada de la captura
  - Traspaso a user-space de los ficheros de traza en binario
  - Traducción del binario para generar salida en texto
- Kernelshark: Interfaz gráfica de visualización y captura Ftrace
  - Realiza todas las operaciones de trace-cmd excepto la traducción de trace.dat a texto.
  - Diagramas de ejecución en el tiempo
  - Filtra CPU, tarea y eventos
  - Hace zooms in y out
  - Mide tiempos dentro del diagrama
  - Aun algo inestable pero bastante usable

Obtención de trace-cmd y kernelshark

- Obtener las fuentes del git de su autor Steven Rodstedt

  git://git.kernel.org/pub/scm/linux/kernel/git/rostedt/trace-cmd.git

  - Compilar e instalar:
    - make gui && make doc && sudo make install_gui && sudo make install_doc
- Como paquete Debian:
  - En Ubuntu tienen la versión 1.03 **obsoleta: no captura desde kernelshark**
  - En la web tenemos empaquetada la versión 2.1.0
    - Descargar: wget
      http://www.ctr.unican.es/asignaturas/dec/Doc/kernelshark_debian_squeeze.tar.gz
    - Descomprimir (tar -zxvf kernelshark_debian_squeeze.tar.gz)
    - Instalar: sudo dpkg -i *.deb

Workflow con KernelShark

1. Preparar el sistema para tracear
    1. Preparar la configuración
    2. Arrancar y ver que funciona
    3. Pararlo
2. Ejecución bajo traceo
    1. Configurar eventos de captura en kernelshark
    2. Comenzar la captura
    3. **Ejecutar sistema**
        1. Anotar los PID's que se quieren observar
    4. Detener la captura
    5. Guardar fichero trace.dat (copiarlo porque ya existe en el directorio)
    6. Exportar a texto con
        1. trace-cmd report -R <fichero trace.dat>

Workflow con trace-cmd

- Como root:
  - Arrancar y parar una captura
    - Trace-cmd start -e sched
    - Trace-cmd stop
  - Extraer el fichero (binario) de trazas
    - Trace-cmd extract
- Como usuario:
  - Pasar el fichero binario a texto
    - Trace-cmd report -R <fichero.dat> (por defecto trace.dat)
  - Visualizar gráficamente
    - Kernelshark <fichero.dat>
- Más información y opciones: Páginas man

Workflow con debugfs (si no tenemos trace-cmd)

- Montar el debug fs en un directorio
  - Mount -t debugfs none /debug (/debug debe existir)
  - Esto hace que aparezca /debug/tracing
- Consultar los traceadores que tenemos compilados en el kernel
  - cat /debug/tracing/available_tracers
- Seleccionar sched_switch
  - echo sched_switch > /debug/tracing/current_tracer
- Activar el traceado
  - Echo 1 > /debug/tracing/tracing_enable
- Esperar
- Desactivar el traceado
  - Echo 0 >/debug/tracing/tracing_enable
- El traceado está en el fichero /debug/tracing/trace

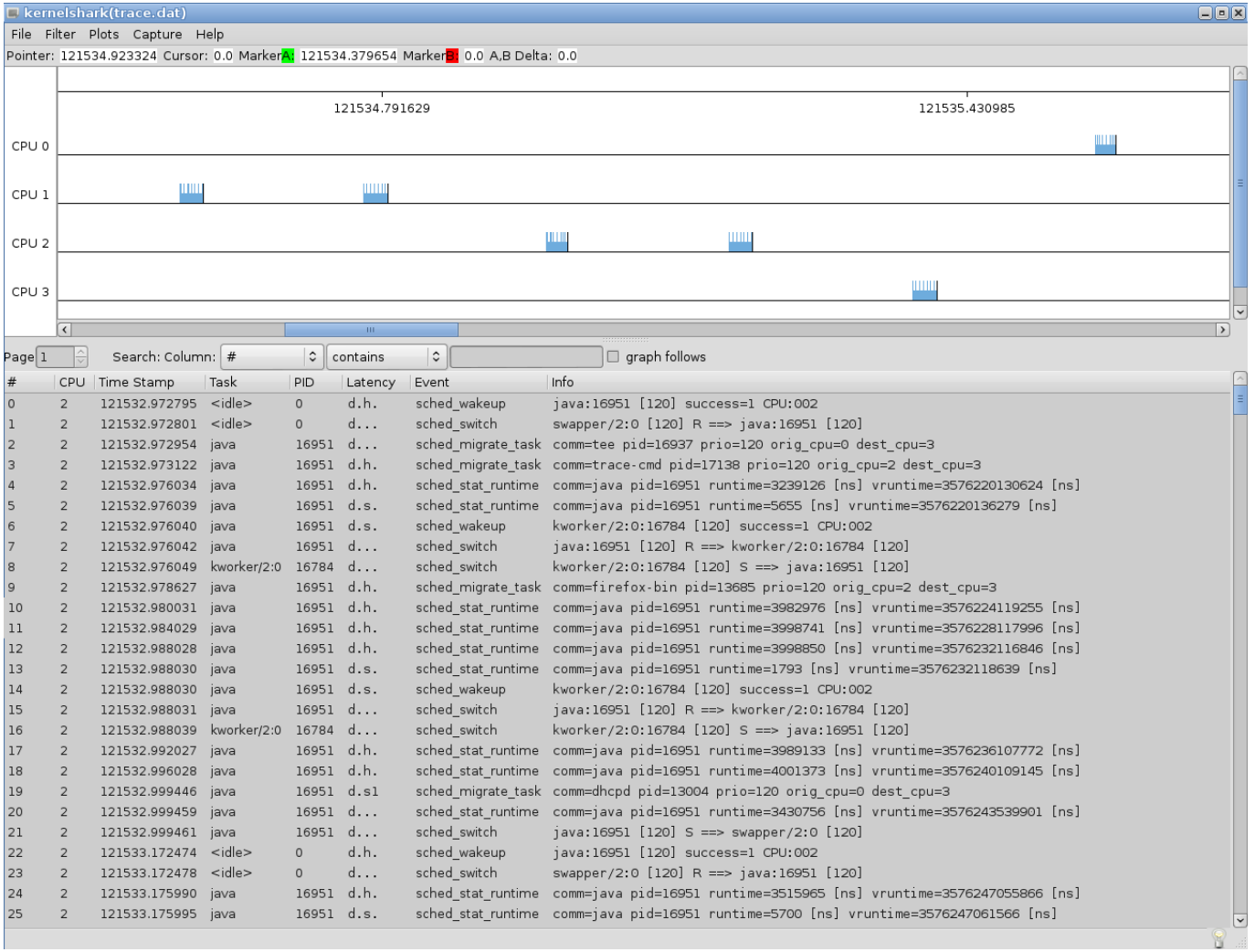
Detalles de DebugFS

- Se monta (como root) por:
  - mount -t debugfs nonedev /sys/kernel/debug
- Interioridades de /sys/kernel/debug/tracing

```
available_events:    Eventos habilitados (RO)
available_tracers    lugin-tracers habilitados (RO)
buffer_size_kb       Tamaño del buffer circular del kernel (RW)
current_tracer       Plugin-tracer usado (RW)
events               Directorio con los eventos disponibles
options              Directorio con las opciones disponibles
per_cpu              Acceso a los datos de cada core
set_event            Eventos extras que se capturan (RW)
trace                Traza resultado (RO)
trace_clock          Relojes (local core, global core o medio) (RO)
trace_marker         Entrada para datos de usuario entrelazados (WO)
trace_options        Opciones activas de la traza (RW)
trace_pipe           Acceso a la traza en modo consumidor (RO)
tracing_cpumask      Máscara de captura de trazas de CPU (RW)
tracing_on           Traza activa o parada (RW)
tracing_enabled      Versión obsoleta de tracing_on
saved_cmdlines       Buffer interno de para asociar PIDs y cmdlines
printk_formats       Formatos de cada evento (RO)
```

Recorrido rápido por KernelShark

Análisis desde kernelshark



#	CPU	Time Stamp	Task	PID	Latency	Event	Info
0	2	121532.972795	<idle>	0	d.h.	sched_wakeup	java:16951 [120] success=1 CPU:002
1	2	121532.972801	<idle>	0	d...	sched_switch	swapper/2:0 [120] R ==> java:16951 [120]
2	2	121532.972954	java	16951	d...	sched_migrate_task	comm=tee pid=16937 prio=120 orig_cpu=0 dest_cpu=3
3	2	121532.973122	java	16951	d.h.	sched_migrate_task	comm=trace-cmd pid=17138 prio=120 orig_cpu=2 dest_cpu=3
4	2	121532.976034	java	16951	d.h.	sched_stat_runtime	comm=java pid=16951 runtime=3239126 [ns] vruntime=3576220130624 [ns]
5	2	121532.976039	java	16951	d.s.	sched_stat_runtime	comm=java pid=16951 runtime=5655 [ns] vruntime=3576220136279 [ns]
6	2	121532.976040	java	16951	d.s.	sched_wakeup	kworker/2:0:16784 [120] success=1 CPU:002
7	2	121532.976042	java	16951	d...	sched_switch	java:16951 [120] R ==> kworker/2:0:16784 [120]
8	2	121532.976049	kworker/2:0	16784	d...	sched_switch	kworker/2:0:16784 [120] S ==> java:16951 [120]
9	2	121532.978627	java	16951	d.h.	sched_migrate_task	comm=firefox-bin pid=13685 prio=120 orig_cpu=2 dest_cpu=3
10	2	121532.980031	java	16951	d.h.	sched_stat_runtime	comm=java pid=16951 runtime=3982976 [ns] vruntime=3576224119255 [ns]
11	2	121532.984029	java	16951	d.h.	sched_stat_runtime	comm=java pid=16951 runtime=3998741 [ns] vruntime=3576228117996 [ns]
12	2	121532.988028	java	16951	d.h.	sched_stat_runtime	comm=java pid=16951 runtime=3998850 [ns] vruntime=3576232116846 [ns]
13	2	121532.988030	java	16951	d.s.	sched_stat_runtime	comm=java pid=16951 runtime=1793 [ns] vruntime=3576232118639 [ns]
14	2	121532.988030	java	16951	d.s.	sched_wakeup	kworker/2:0:16784 [120] success=1 CPU:002
15	2	121532.988031	java	16951	d...	sched_switch	java:16951 [120] R ==> kworker/2:0:16784 [120]
16	2	121532.988039	kworker/2:0	16784	d...	sched_switch	kworker/2:0:16784 [120] S ==> java:16951 [120]
17	2	121532.992027	java	16951	d.h.	sched_stat_runtime	comm=java pid=16951 runtime=3989133 [ns] vruntime=3576236107772 [ns]
18	2	121532.996028	java	16951	d.h.	sched_stat_runtime	comm=java pid=16951 runtime=4001373 [ns] vruntime=3576240109145 [ns]
19	2	121532.999446	java	16951	d.s1	sched_migrate_task	comm=dhcpd pid=13004 prio=120 orig_cpu=0 dest_cpu=3
20	2	121532.999459	java	16951	d...	sched_stat_runtime	comm=java pid=16951 runtime=3430756 [ns] vruntime=3576243539901 [ns]
21	2	121532.999461	java	16951	d...	sched_switch	java:16951 [120] S ==> swapper/2:0 [120]
22	2	121533.172474	<idle>	0	d.h.	sched_wakeup	java:16951 [120] success=1 CPU:002
23	2	121533.172478	<idle>	0	d...	sched_switch	swapper/2:0 [120] R ==> java:16951 [120]
24	2	121533.175990	java	16951	d.h.	sched_stat_runtime	comm=java pid=16951 runtime=3515965 [ns] vruntime=3576247055866 [ns]
25	2	121533.175995	java	16951	d.s.	sched_stat_runtime	comm=java pid=16951 runtime=5700 [ns] vruntime=3576247061566 [ns]

diagrama

lista

KernelShark: lo que se muestra en la pantalla

- Vista de diagrama:
  - Diagramas de CPU: Cada task tiene un color distinto.
  - Diagramas de task: Cada CPU que usa la task tiene un color distinto
- Vista de lista (lo no evidente):
  - Timestamp en sg.us
  - Latency: Flags
    - d: disabled interrupts
    - N: Needs schedule
    - h, s: In hard or soft irq
    - Preemption counter (si != '.', el kernel no expulsa otras tareas)
    - (kernel < 2.6.39) Lock depth: Si se tiene le BKL cogido

Operaciones con kernelshark

- Zoom IN:
  - Arrastrar ratón a la DERECHA
- Zoom OUT (al nivel anterior):
  - Arrastrar ratón a la izquierda
- Medida de tiempo:
  - From: left_click To: SHIFT-left_click (El delta está en sg)
- Correspondencia diagrama – Lista
  - Doble click en diagrama (o en lista): Cursor azul y se muestra en la lista
- Si se deja el ratón quieto en el diagrama:
  - Aparace el “hint” con el evento más próximo

KernelShark: filtrados

- Filtrar Eventos (antes y después de captura):
  - Antes de la captura: Se puede elegir que eventos capturar
  - Después de la captura: Filter->events
- Filtrar tareas (después de la captura):
  - Menú: filter->tasks
  - Vista diagrama:
    - Botón derecho: Add <tarea> to filter (las tareas que hagan falta)
    - Enable graph o list filter
  - Vista lista:
    - Botón derecho: Add <tarea> to filter (las tareas que hagan falta)
    - Enable graph o list filter
- Los filtros se quitan con boton-derecho (disable filter)
- Hacer plots de tareas: (Se añade una gráfica +)
  - Vista diagrama: Boton derecho: Plot <tarea>

Parseado de trace-cmd report

Salida de trace-cmd report -R

```
version = 6
Cpus=2

 <idle>-0     [000] 121532.901840: sched_wakeup:
       comm=firefox-bin pid=13685 prio=120 success=1 target_cpu=0


 trace-cmd-17141 [003] 121532.901758: sched_switch:
     prev_comm=trace-cmd prev_pid=17141 prev_prio=120 prev_state=0x1
     next_comm=swapper/3 next_pid=0 next_prio=120
```

- Los 2 bloques corresponden a un única línea
  - Primero los parámetros comunes:
    - <comm_actual>-<pid_actual> <CPU> <timestamp> : <nombre_evento>:
  - Luego los parámetros del evento (-R del report elimina los plugins de formato)
    - <nom_par1>=<par1> <nom_par2>=<par2> ...

Tarea <idle> PID 0

- Esta tarea no existe.
  - No existe nunca un PID 0
  - ftrace necesita conderarla cuando “la CPU no hace nada”.
- La “tarea idle” se manifiesta:
  - <idle>-0: Cuando se está “ejecutando”
  - Swapper/<CPU>: Cuando aparece como argumento de sched_switch

Contenido de cada línea de eventos

- Por cada línea tenemos:
  - Proceso que corre cuando sucede el evento.
    - **NOTA**: Para sched_* nos interesan los procesos de los params (ver más abajo).
  - CPU del evento
  - Timestamp
  - Nombre del evento
  - Parámetros del evento
- Los parámetros de cada evento se obtienen con:
  - trace-cmd --events

Parámetros de eventos

```
sched-switch

REC->prev_comm,
REC->prev_pid
REC->prev_prio,
REC->prev_state "S" "D" "T" "t" "Z" "X" "x" "W"  "R"
REC->next_comm
REC->next_pid
REC->next_prio

sched-wakeup

REC->comm,
REC->pid,
REC->prio,
REC->success,
REC->cpu
```

Dificultades al parsear trace-cmd report

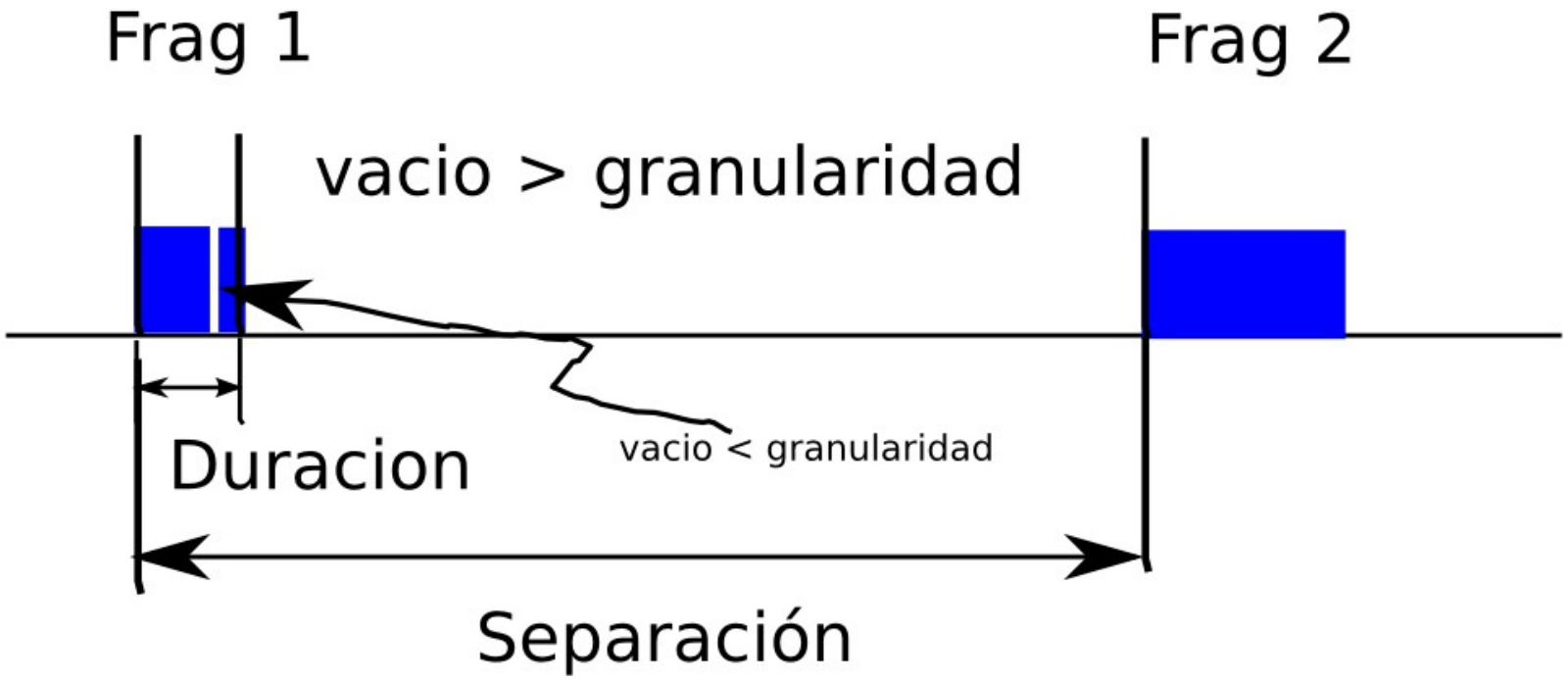
- Atención a:
  - Primeras líneas no son eventos
  - cmdlines con '-' ya que el mismo '-' separa el PID
    - Ej: “trace-cmd-2341”
  - Algunas cmdline tienen espacios (ej “dconf worker”)
  - Idle task:
    - Tiene el mismo PID y diferentes cmdlines (idle-0, swapper/0...)
    - Se ejecuta en varias CPU's a la vez
- Posibles alternativas a trace-cmd report
  - Se puede compilar la librería libtracecmd que lee directamente el fichero.dat
  - También se exporta la librería a python

Una vez “entendido” el fichero...

- Necesitamos tener bloques de ejecución
- Para ellor reconciliamos arranques y comienzos de un thread
  - El thread se **despierta** con **sched_wakeup** (parámetro pid)
  - **Comienza** a ejecutar con sched_switch siendo el **next_pid**
  - **Termina** de ejecutar con sched_switch siendo el **prev_pid**
  - Todo ello en una CPU
- Atención a:
  - La <idle-0> es el único PID que se ejecuta en varias CPU's a la vez.
  - Además la <idle-0> no se despierta.

Granularidad

- Debido al scheduler CFS con SCHED_OTHER existen muchos pequeños tiempo vacíos en la ejecución de un programa.



- Granularidad: Umbral por debajo del cual integramos 2 fragmentos en uno sólo.

Parseador propio

Parseador de trace-cmd report

- Hecho en python localmente por nosotros:
  - Su realización fue propuesta como práctica el curso 2011-12
  - Disponible en la web de la asignatura
  - Correcciones bienvenidas
    - https://github.com/mtelleria/ftrace-sched-python-analyzer

```
usage: analizador_fragmentos.py [-h] [--gran GRAN] [--keep-text] [--pids PIDS]
                                 [--process-idle] [--show-lost-wakeups]
                                 [--with-sanity-checks] [--cpus CPUS]
                                 [--from FROM_REL] [--to TO_REL]
                                 [--from-line FROM_LINE] [--to-line TO_LINE]
                                 [--from-abs FROM_ABS] [--to-abs TO_ABS]
                                 [--info] [--info-pids] [--info-cpus]
                                 [--info-eventos]
                                 file
```

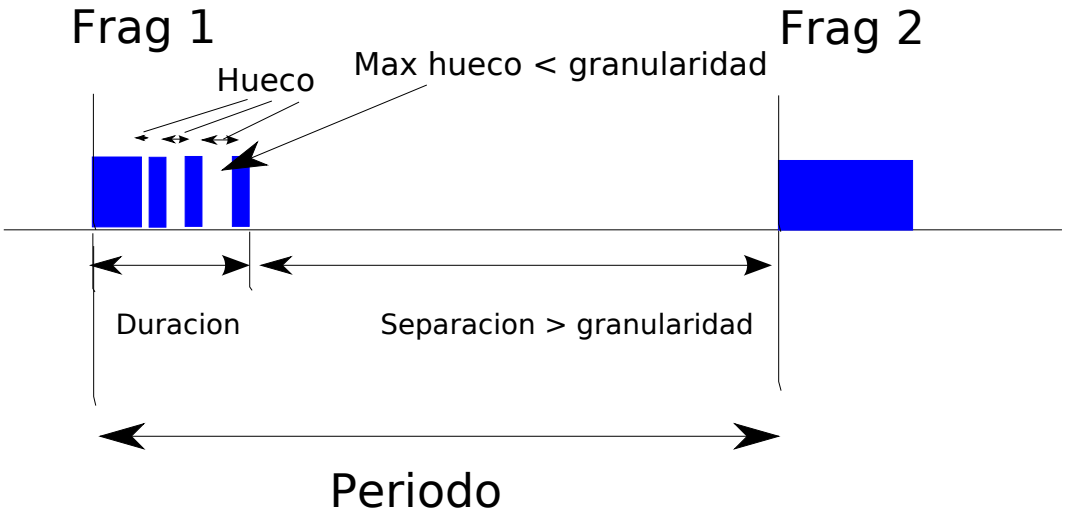
```
Analiza trazas de trace-cmd/kernelshark.

positional arguments:
  file                  Fichero a examinar. Puede ser binario o texto, por
                        defecto trace.dat

optional arguments:
  -h, --help            show this help message and exit
  --gran GRAN           Granularidad en usec para unir bloques de ejecucion
  --keep-text           Mantener el fichero de texto autogenerado
  --pids PIDS           pid1,pid2,...,pidN: Procesar unicamente los PIDs de la lista
  --process-idle        Procesa ademas las tareas swapper (idle) de cada CPU
  --show-lost-wakeups   Muestra wakeups perdidos en huecos debido a alta granularidad
  --with-sanity-checks  Testea la coherencia de eventos y datos desde el punto
                        de vista de este script
  --cpus CPUS           cpuid0,cpuid1,cpuid2...: Procesar unicamente los
                        eventos que ocurren en las CPUs de la lista
  --from FROM_REL       Tiempo de comienzo relativo en milisegundos
  --to TO_REL           Tiempo de final relativo en milisegundos
  --from-line FROM_LINE
                        Linea de fichero a partir de la cual se empieza a procesar
  --to-line TO_LINE     Linea de fichero hasta la cual se sigue procesando
  --from-abs FROM_ABS   Tiempo de comienzo absoluto <segundos>.<usec>
  --to-abs TO_ABS       Tiempo de final absoluto <segundos>.<usec>
  --info                Da datos globales del fichero: inicio, final,
                        duracion, nr_pids, nr_cpus, nr_lineas
  --info-pids           Lista los PIDs con sus basecmd de los threads que tiene la traza
  --info-cpus           Lista las CPUs que intervienen en la traza
  --info-eventos        Lista los eventos y los subsistemas presentes en el
                           fichero de traza
```

Funcionalidad del parseador

- Trabaja con ficheros de texto generados con trace-cmd **sin plugins de formato** (opción -R en trace-cmd report)
- Lanza el trace-cmd report si el fichero es de formato .dat
- Info:
  - PIDs, tiempo de traza, eventos y CPU
- Filtra:
  - Por PID's, eventos y CPU
- Warnings si detecta un evento no esperado o una línea incorrecta.

Resultado del parseador

- Para cada PID calcula bloques de ejecución:
  - Start_ms: Tiempo de inicio Durac_ms: Duración del bloque (+ posibles huecos)
  - Hueco_ms: Porción del tiempo correspondiente a huecos
    - Debe ser pequeño respecto a la duración.
  - Max_Hueco_ms Separacion_ms



Resultado del parseador

N Frag	Start_ms	Durac_ms	CPUs	Hueco_ms	Periodo_ms	Separ_ms	Max_Hueco_ms	Laten_ms
1	99.237	26.661	2	0.013	0.000	0.000	0.007	0.005
2	298.915	22.497	2	0.014	199.678	173.017	0.006	0.004
3	498.402	26.946	3	0.018	199.487	176.990	0.009	0.012
4	698.363	26.980	3	0.013	199.961	173.015	0.007	0.003
5	898.358	22.899	3	0.014	199.995	173.015	0.007	0.003
6	1098.291	24.896	1	0.014	199.933	177.034	0.007	0.004
7	1298.206	26.906	1	0.013	199.915	175.019	0.007	0.007
8	1498.127	23.060	1	0.011	199.921	173.015	0.006	0.004
9	1698.198	24.898	1	0.009	200.071	177.011	0.008	0.002
10	1898.131	27.008	1	0.048	199.933	175.035	0.037	0.006
11	2098.186	22.864	2	0.013	200.055	173.047	0.006	0.006
12	2298.090	24.889	2	0.045	199.904	177.040	0.032	0.008
13	2497.985	26.810	3	0.014	199.895	175.006	0.007	0.007
14	2697.844	22.842	0	0.017	199.859	173.049	0.008	0.007
15	2897.708	25.052	0	0.015	199.864	177.022	0.007	0.006
16	3097.773	20.980	0	0.016	200.065	175.013	0.008	0.002
17	3297.765	22.971	0	0.015	199.992	179.012	0.007	0.003
18	3497.757	23.118	0	0.012	199.992	177.021	0.006	0.007
19	3697.844	20.844	1	0.009	200.087	176.969	0.008	0.002
20	3897.714	22.736	1	0.015	199.870	179.026	0.007	0.007
21	4097.463	25.033	1	0.014	199.749	177.013	0.007	0.002
22	4297.513	27.020	1	0.014	200.050	175.017	0.007	0.003
23	4497.514	29.087	1	0.019	200.001	172.981	0.008	0.006
24	4697.614	23.022	1	0.010	200.100	171.013	0.005	0.002