ftrace histograms

A trace-cmd front end interface to ftrace histograms, triggers and synthetic events

Tzvetomir Stoyanov

VMware Open Source Technology Center

The ftrace histogram triggers

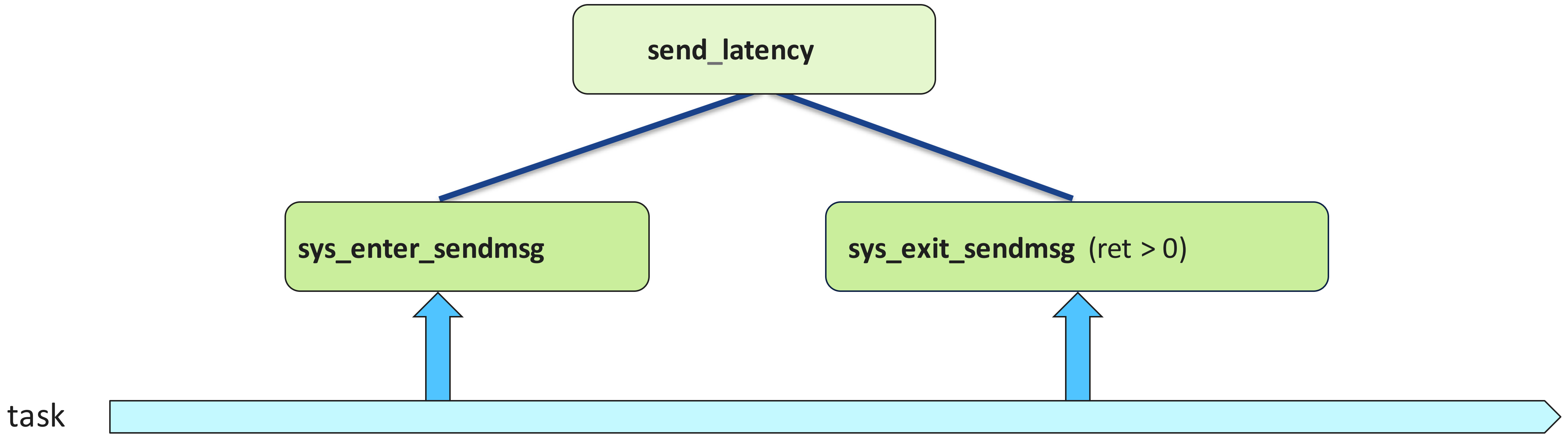
- Implemented by Tom Zanussi, Linux 4.7
- Powerful feature
- Complex interface
- Hard to find configuration errors

Basic histogram config

echo '**hist:key=common_pid.execname**' > events/exceptions/page_fault_user/trigger
cat events/exceptions/page_fault_user/hist

```
{ common_pid: Socket Thread   [      4485] } hitcount:          1
{ common_pid: Web Content     [      4906] } hitcount:          2
{ common_pid: JS Helper       [      4958] } hitcount:          5
{ common_pid: JS Helper       [      4957] } hitcount:          6
{ common_pid: pool-gnome-shel [     12808] } hitcount:          6
{ common_pid: Chrome_~dThread [     18735] } hitcount:          7
{ common_pid: JS Helper       [      4959] } hitcount:          9
{ common_pid: Web Content     [      4935] } hitcount:         10
{ common_pid: gmain           [      3546] } hitcount:         11
{ common_pid: Xorg            [      3437] } hitcount:         14
{ common_pid: JS Helper       [      4915] } hitcount:         15
{ common_pid: Web Content     [     18731] } hitcount:         17
{ common_pid: firefox         [      4441] } hitcount:         21
{ common_pid: WebExtensions   [      4726] } hitcount:         23
{ common_pid: gdbus           [      3548] } hitcount:         24
{ common_pid: bash            [      6124] } hitcount:         28
{ common_pid: JS Helper       [      3571] } hitcount:         70
{ common_pid: bash            [     12810] } hitcount:         82
{ common_pid: gnome-shell     [     12809] } hitcount:        311
{ common_pid: gnome-shell     [      3544] } hitcount:       2695
```

Latency between two events

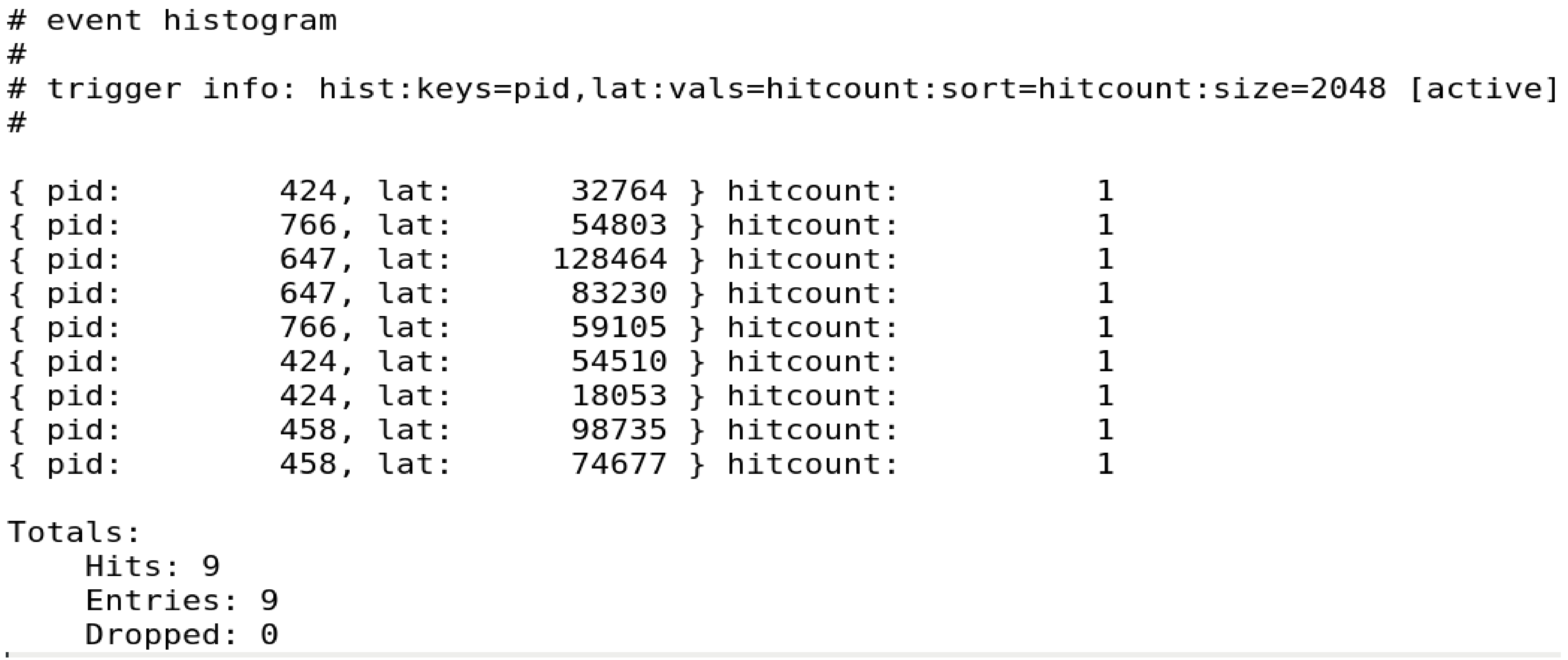


Latency between two events

```
echo 'send_latency u64 lat int pid' > synthetic_events

echo 'hist:key=common_pid:ts0=common_timestamp' >
                                    events/syscalls/sys_enter_sendmsg/trigger

echo 'hist:keys=common_pid:lat=common_timestamp-$ts0:
onmatch(syscalls.sys_enter_sendmsg).send_latency($lat,common_pid) if ret > 0' >
                                    events/syscalls/sys_exit_sendmsg/trigger


echo 'hist:keys=pid,lat' > events/synthetic/send_latency/trigger
```

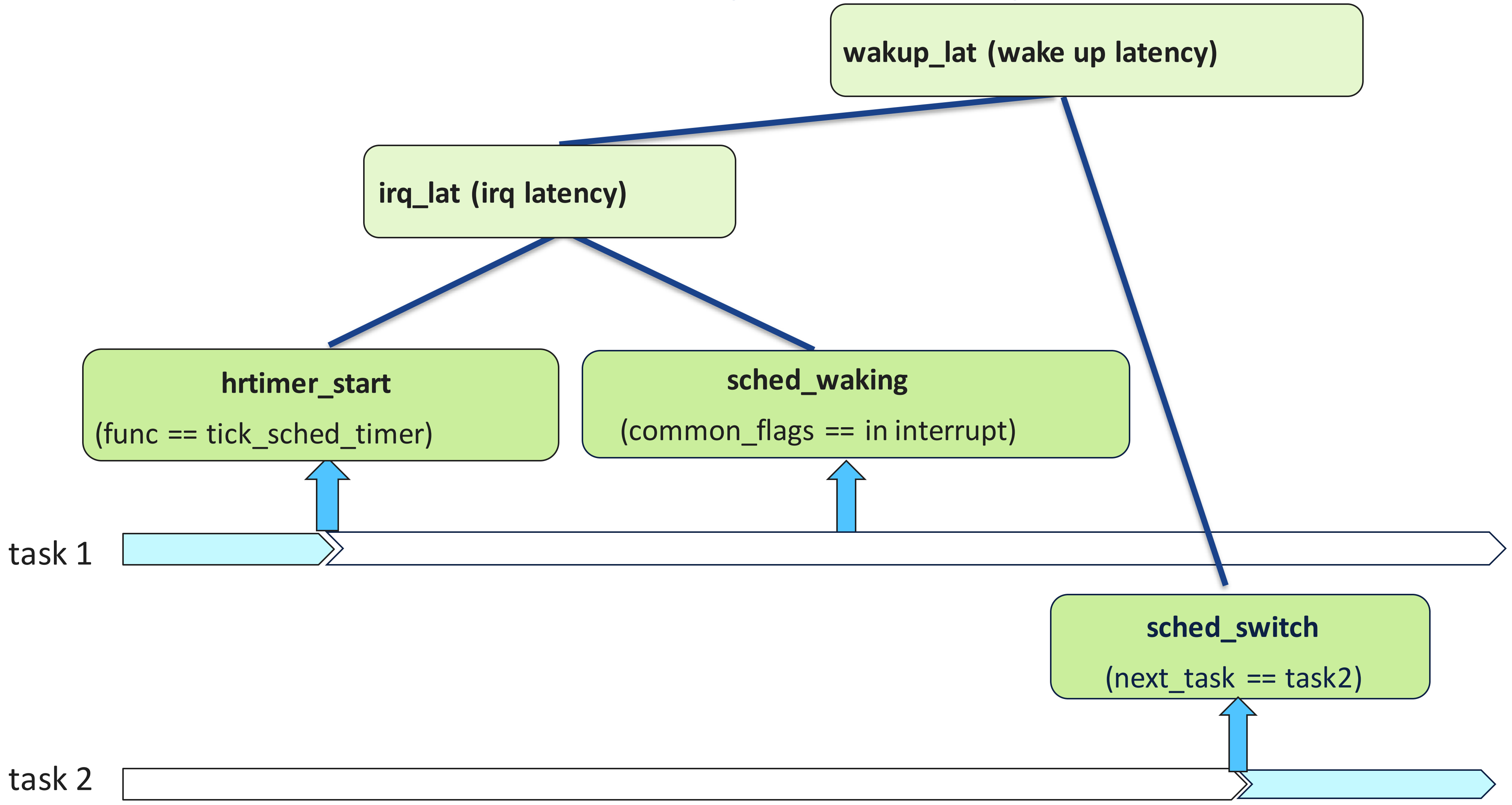
Latency between two events

cat events/synthetic/send_latency/hist



```
# event histogram
#
# trigger info: hist:keys=pid,lat:vals=hitcount:sort=hitcount:size=2048 [active]
#

{ pid:        424, lat:      32764 } hitcount:          1
{ pid:        766, lat:      54803 } hitcount:          1
{ pid:        647, lat:     128464 } hitcount:          1
{ pid:        647, lat:      83230 } hitcount:          1
{ pid:        766, lat:      59105 } hitcount:          1
{ pid:        424, lat:      54510 } hitcount:          1
{ pid:        424, lat:      18053 } hitcount:          1
{ pid:        458, lat:      98735 } hitcount:          1
{ pid:        458, lat:      74677 } hitcount:          1

Totals:
    Hits: 9
    Entries: 9
    Dropped: 0
```

More complex latency use case



More complex latency use case

echo '**irq_lat** u64 lat pid_t pid' > synthetic_events
echo '**wake_lat** u64 lat u64 irqlat pid_t pid' >> synthetic_events

echo 'hist:keys=common_pid:**irqts**=common_timestamp.usecs if function == 0xffffffff81200580'
> events/timer/**hrtimer_start**/trigger
(0xffffffff81200580 is the address of tick_sched_timer)

echo 'hist:keys=common_pid:**lat**=common_timestamp.usecs-$irqts:**onmatch**(timer.hrtimer_start).
irq_lat($lat,pid) if common_flags & 1' > events/sched/**sched_waking**/trigger

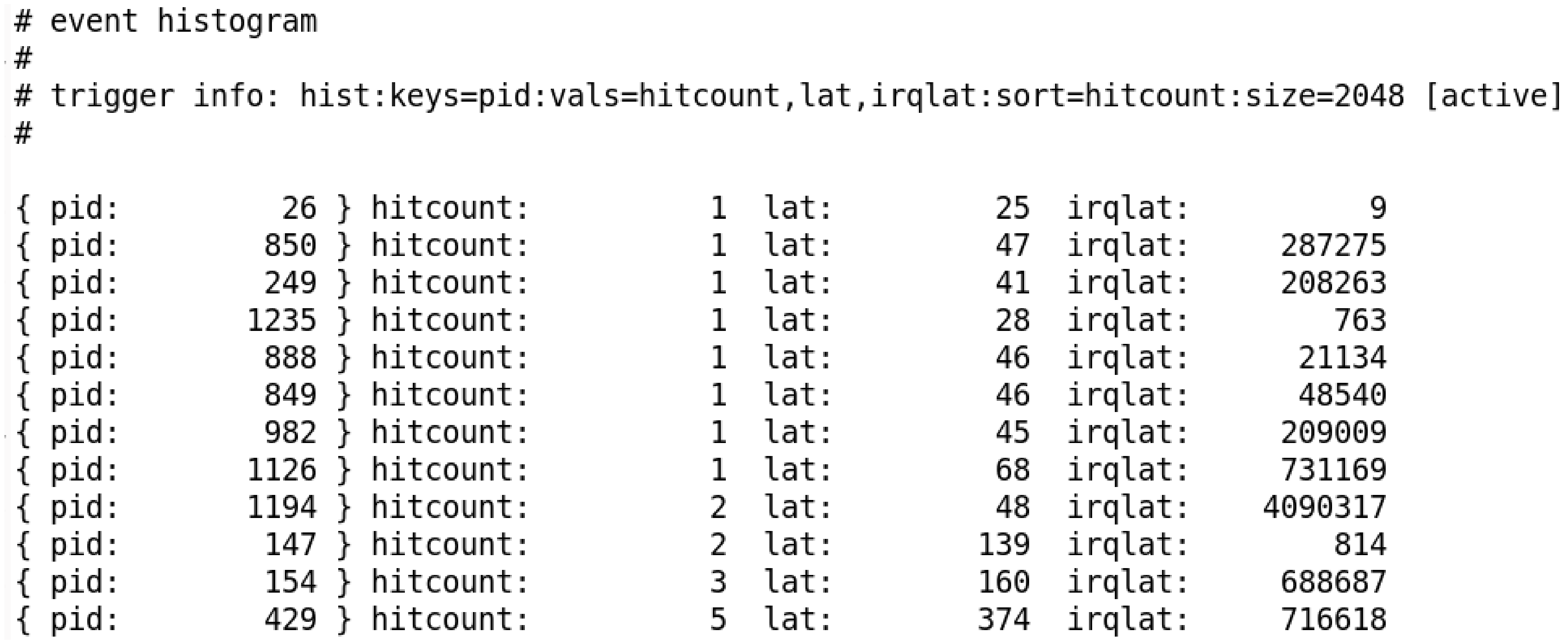
echo 'hist:keys=pid:**wakets**=common_timestamp.usecs,irqlat=lat'
> events/synthetic/**irq_lat**/trigger

echo 'hist:keys=next_pid:lat=common_timestamp.usecs-$wakets,irqlat=$irqlat:**onmatch**(synthetic.
irq_lat).wake_lat($lat,$irqlat,next_pid)'
> events/sched/**sched_switch**/trigger

echo 1 > events/synthetic/**wake_lat**/enable

More complex latency use case

cat events/synthetic/wake_lat/hist



```
# event histogram
#
# trigger info: hist:keys=pid:vals=hitcount,lat,irqlat:sort=hitcount:size=2048 [active]
#

{ pid:         26 } hitcount:          1  lat:          25  irqlat:          9
{ pid:        850 } hitcount:          1  lat:          47  irqlat:     287275
{ pid:        249 } hitcount:          1  lat:          41  irqlat:     208263
{ pid:       1235 } hitcount:          1  lat:          28  irqlat:        763
{ pid:        888 } hitcount:          1  lat:          46  irqlat:      21134
{ pid:        849 } hitcount:          1  lat:          46  irqlat:      48540
{ pid:        982 } hitcount:          1  lat:          45  irqlat:     209009
{ pid:       1126 } hitcount:          1  lat:          68  irqlat:     731169
{ pid:       1194 } hitcount:          2  lat:          48  irqlat:    4090317
{ pid:        147 } hitcount:          2  lat:         139  irqlat:        814
{ pid:        154 } hitcount:          3  lat:         160  irqlat:     688687
{ pid:        429 } hitcount:          5  lat:         374  irqlat:     716618
```

The goal

Design a simplified CLI interface for configuring histograms, using trace-cmd frontend:

- Cover all histogram operations, or at least the most commonly used cases.
- Simple interface, fit into a single line.
- Avoid configuration errors

A proposal

Using a SQL-like syntax to describe relations between events:

- Events -> tables
- Event fields -> columns

--select 'var: name=value, name=value from **start_event** join **end_event**
on **start_event.field** == **end_event.field**
where **conditions**'

Latency between two events

```
 echo 'send_latency u64 lat pid_t pid' > synthetic_events
 echo 'hist:key=common_pid:ts0=common_timestamp' >
                                             events/syscalls/sys_enter_sendmsg/trigger
echo 'hist:keys=common_pid:lat=common_timestamp-$ts0:
onmatch(syscalls.sys_enter_sendmsg).send_latency($lat,common_pid) if ret > 0' >
                                             events/syscalls/sys_exit_sendmsg/trigger
echo 'hist:keys=pid,lat' > events/synthetic/send_latency/trigger
```

```
 --select 'send_latency: lat=sys_exit_sendmsg.common_timestamp -   sys_enter_sendmsg.common_timestamp,
pid=sys_exit_sendmsg.common_pid
from syscalls.sys_enter_sendmsg join syscalls.sys_exit_sendmsg
          on sys_enter_sendmsg.common_pid == sys_exit_sendmsg.common_pid
        where sys_exit_sendmsg.ret > 0'
```

More complex latency use case

```
--select 'irq_lat: lat=sched_waking.common_timestamp.usecs -   hrtimer_start.common_timestamp.usecs,
pid=sched_waking.pid
                from timer.hrtimer_start join sched.sched_waking
                  on hrtimer_start.common_pid == sched_waking.common_pid
                where hrtimer_start.function == 0xffffffff81200580 and
                     sched_waking.common_flags & 1'


--select 'wake_lat: lat=sched_switch.common_timestamp.usecs - irq_lat.common_timestamp.usecs, irqlat=irq_lat.lat,
pid=sched_switch.next_pid
           from sched.sched_switch join irq_lat on sched_switch.next_pid == irq_lat.pid'
```