Simpleperf Introduction

Yabin Cui
android-llvm-dev

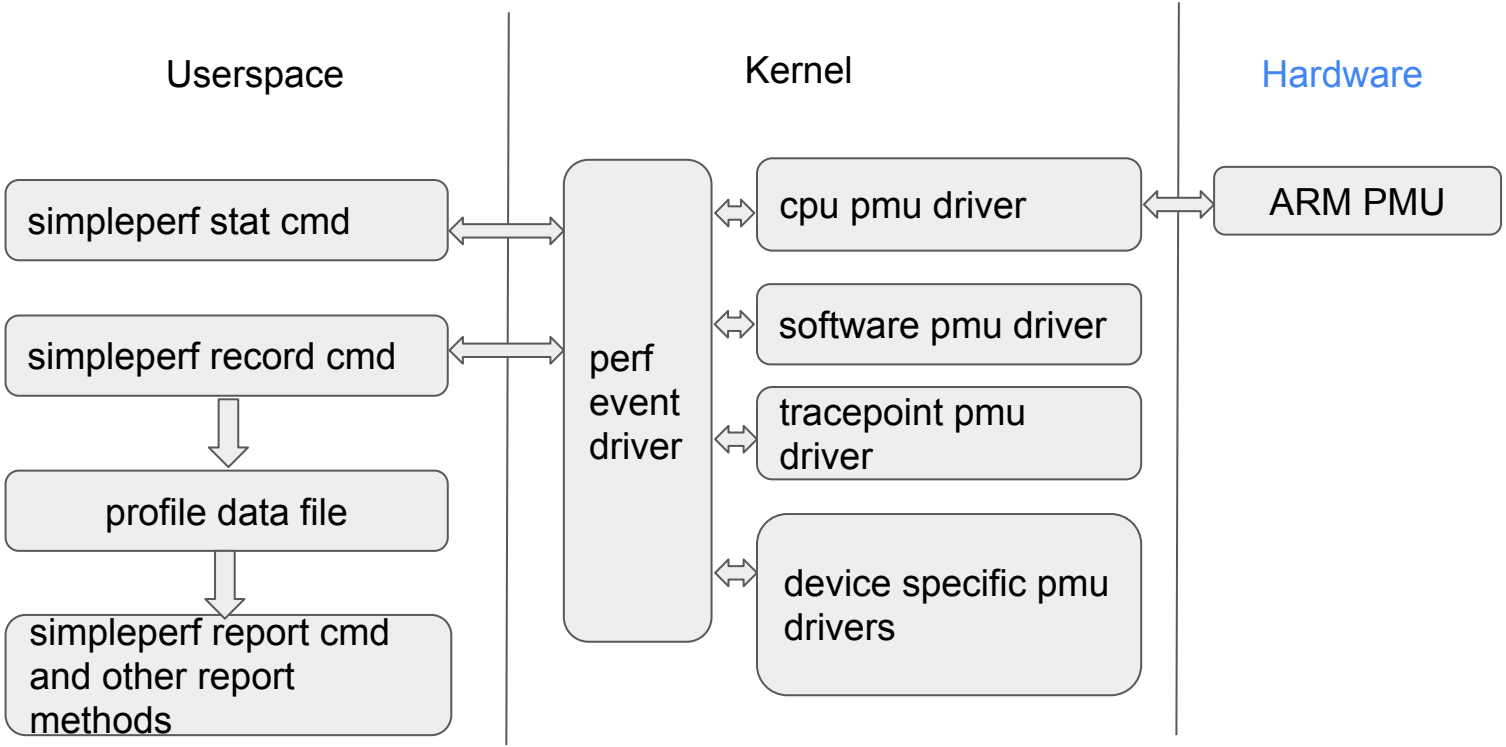
Outline

- What is simpleperf
- How simpleperf works
- Simpleperf commands

What is simpleperf

- A replacement for linux/tools/perf in Android
- A cpu-profiler using linux kernel support and PMU (performance monitor unit) hardware support
- Source code is in https://android.googlesource.com/platform/system/extras/+/master/simpleperf/
- Doc is in https://android.googlesource.com/platform/system/extras/+/master/simpleperf/doc/
- Prebuilt is release in https://android.googlesource.com/platform/prebuilts/simpleperf/

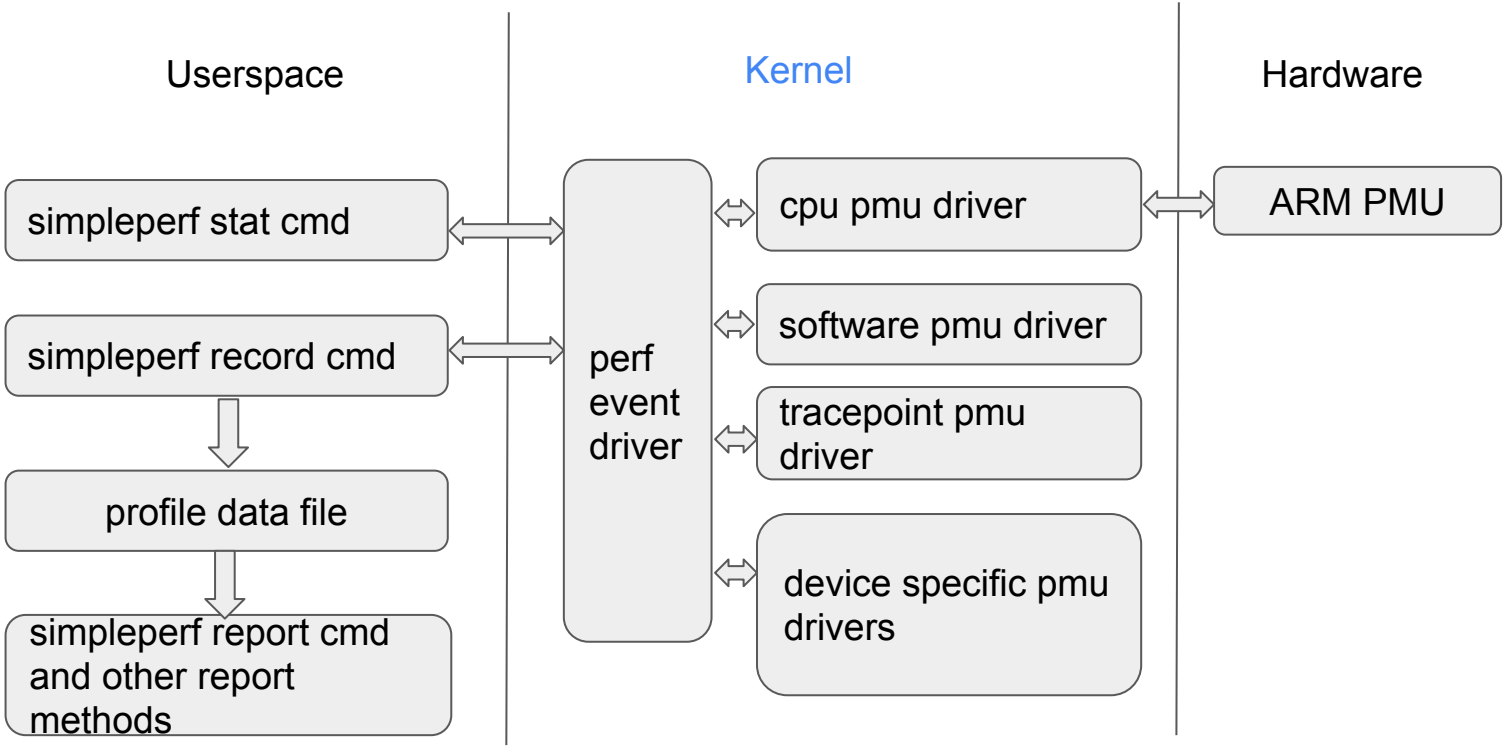
How simpleperf works



ARM PMU

- Described in ARM manual, D7 The Performance Monitors Extension
- PMU counter: Each cpu core has several PMU counters. Each counter is 32-bit, can monitor one PMU event. When the monitored event happens, the counter value increases by one. When a counter overflows, it can trigger an interrupt.
- PMU event: like CPU_CYCLES, BR_PRED (predictable branch), L1D_CACHE (Level 1 data cache access). ARM lists common events and how to interpret them. And the events can be used together to get indirect information, like cache miss rate = cache refill count / cache_access_count.
- The PMU events are growing in newer architectures.

How simpleperf works



Kernel support

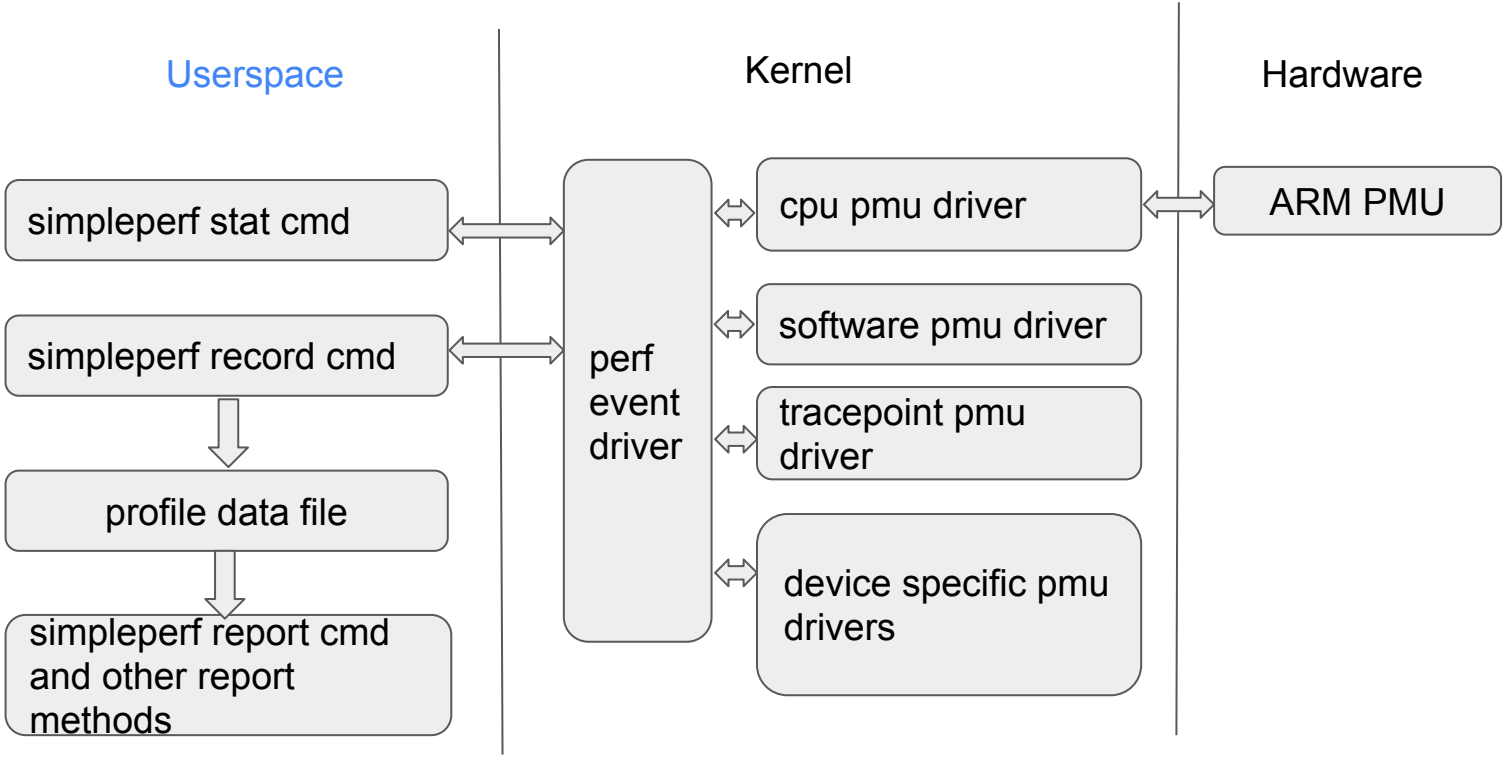
- perf event driver
  - a bridge between userspace and pmu drivers. It lives in kernel/events
  - maps pmu events to perf event types, described in include/uapi/linux/perf_event.h
  - provides a sysfs interface to show supported perf events, in /sys/bus/event_source
  - provides perf_event_open system call to monitor performance of selected threads

```
int perf_event_open(struct perf_event_attr *attr, pid_t pid, int cpu, int group_fd, unsigned
long flags)

    attr - config which perf event to use

    pid - config which thread to monitor, all threads if -1

    cpu - config which cpu to monitor, all cpu is -1

    group_fd, flags - usually not used

    returns a file descriptor, which can be used to read counter values and records
```

Kernel support

- pmu drivers
  - register to perf event driver via perf_pmu_register().
  - cpu pmu driver, which operates ARM PMU, lives in drivers/perf.
  - software pmu driver, events like cpu-clock, page-faults, full list is in perf_sw_ids.
  - tracepoint pmu driver, events like sched:sched_switch, full list is in /sys/kernel/tracing/events.
  - device specific pmu drivers.

How simpleperf works



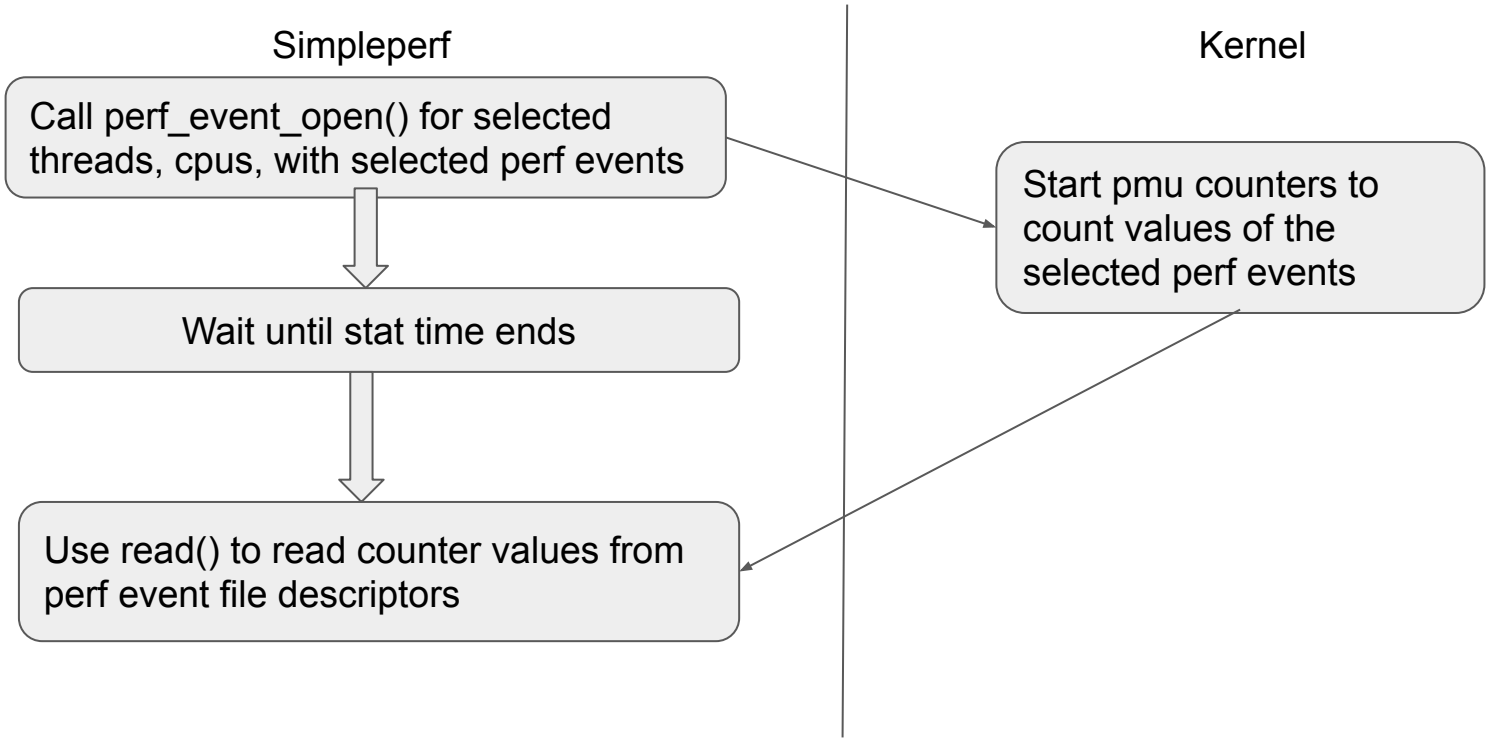
simpleperf commands

- simpleperf is an executable running on device, shipped in /system/bin.
- simpleperf divides its functions into subcommands.
  - list command: list available perf events on device
  - stat command: monitor threads, and print perf event counter values
  - record command: monitor threads, and generate profile data with samples
  - report command: report profile data generated by record command
- simpleperf also provides python scripts running on host
  - to help recording
  - to help reporting

list cmd: list available events

```
$ simpleperf list

List of hardware events:
  branch-misses
  bus-cycles
  cache-misses
  cache-references
  cpu-cycles
  instructions
  stalled-cycles-backend
  stalled-cycles-frontend

List of software events:
  alignment-faults
  context-switches
  cpu-clock
…
```

stat cmd: get perf event counter values



stat cmd: options

$ *simpleperf stat -h*

Usage: simpleperf stat [options] [command [command-args]]
    Gather performance counter information of running [command].

Options:

-p pid1,pid2,...	Stat events on existing processes.
-t tid1,tid2,...	Stat events on existing threads.
-a	Collect system-wide information.
--cpu cpu_item1,cpu_item2,...	Collect information only on the selected cpus.
-e event1[:modifier1],event2[:modifier2],...	Select a list of events to count.
--duration time_in_sec	Monitor for time_in_sec seconds.

stat cmd: example

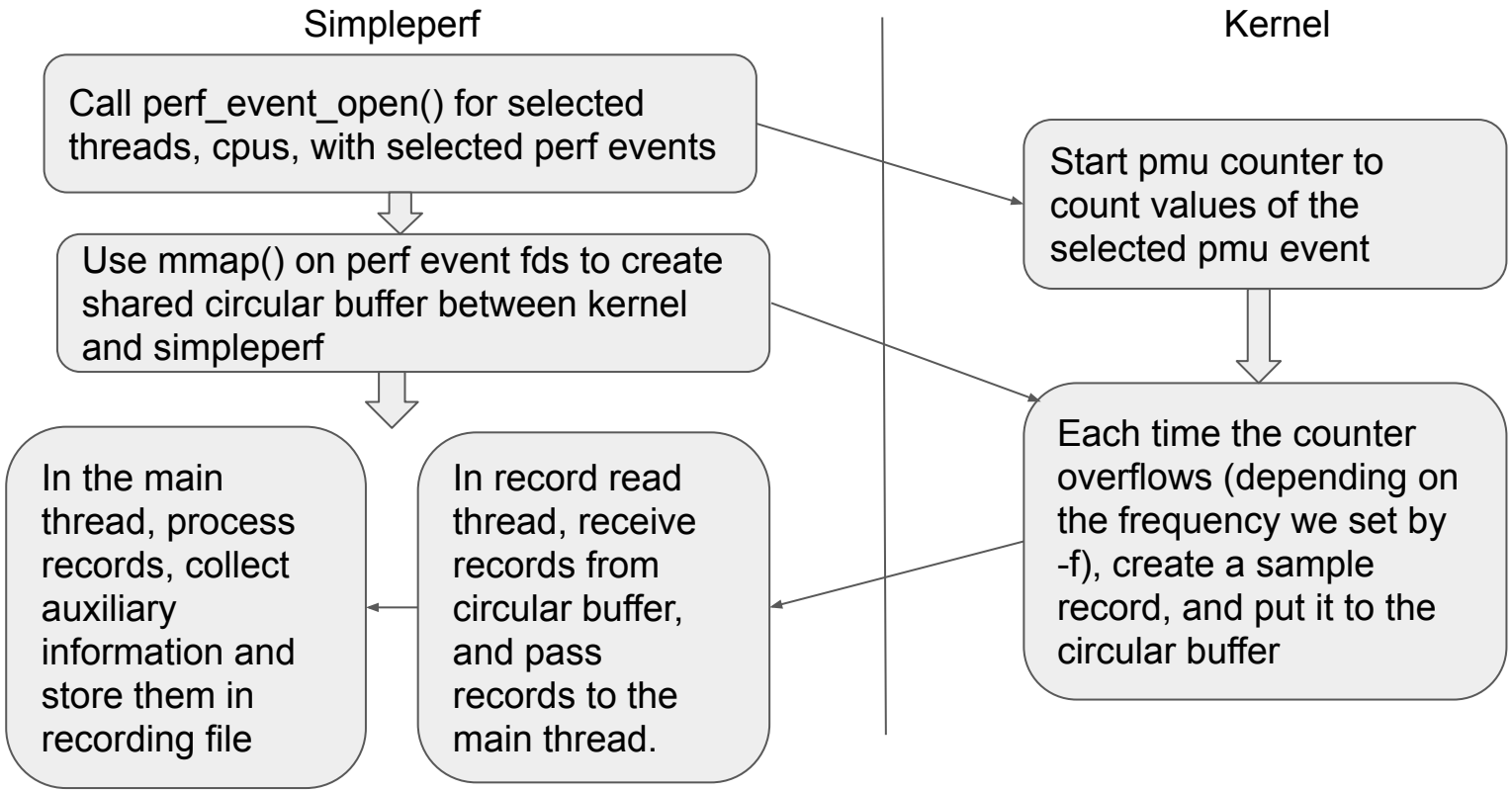
$ *simpleperf stat -e cache-references,cache-misses -a --duration 1*

Performance counter statistics:

```
#           count  event_name          # count / runtime,  runtime / enabled_time
  774,728,087  cache-references   # 96.513 M/sec              (100%)
   31,985,983   cache-misses         # 4.128672% miss rate  (100%)
```

Total test time: 1.001893 seconds.

record cmd: generate profile data with samples



record cmd: options

```
$ simpleperf record -h

Usage: simpleperf record [options] [--] [command [command-args]]
       Gather sampling information of running [command].

Options:
-p pid1,pid2,...        Record events on existing processes.
-t tid1,tid2,...        Record events on existing threads.
-a                      System-wide collection.
--cpu cpu_item1,cpu_item2,...                  Collect information only on the selected cpus.
-e event1[:modifier1],event2[:modifier2],...   Select a list of events to count.
-f freq                                        Set event sample frequency. It means recording at
                                                most [freq] samples every second.
--duration time_in_sec                         Monitor for time_in_sec seconds
-o record_file_name                            Set record file name, default is perf.data.
--call-graph fp | dwarf[,<dump_stack_size>]    Enable call graph recording.
-g                                             Same as '--call-graph dwarf'.
```

record cmd: example

$ *simpleperf record -g sleep 1*

simpleperf I cmd_record.cpp:696] Recorded for 1.01908 seconds. Start post processing.
simpleperf I cmd_record.cpp:771] Samples recorded: 56. Samples lost: 0.

record cmd: sample format

The profile data contains a list of samples.

Each sample can contain below information (full list is here):

time - timestamp in CLOCK_MONOTONIC

pid, tid - process id, thread id

cpu - cpu

period - how many events have happened since last sample

ips[] - callstack (frame-pointer based call stack)

regs[] - userspace register values

stack[] - user stack data up to 64k

dwarf based call stack generated by stack unwinding

report cmd: report profile data

$ *simpleperf report*

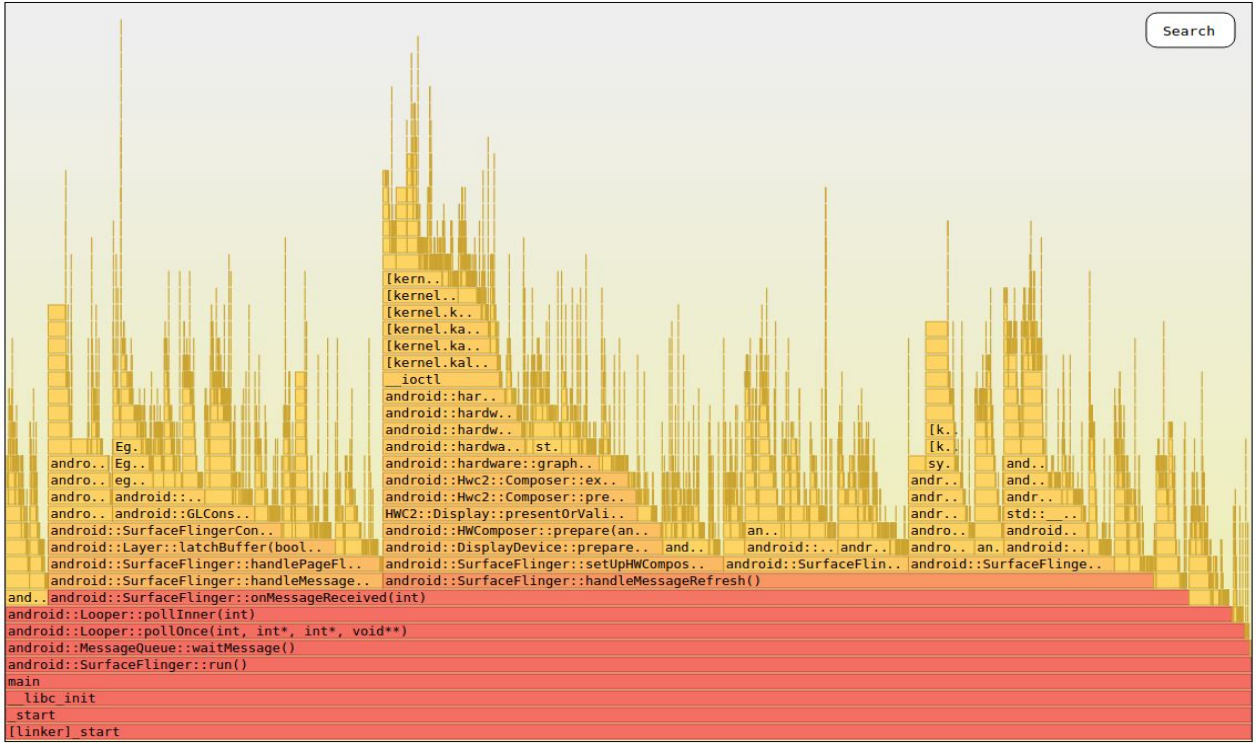
Cmdline: /system/bin/simpleperf record -g sleep 1
Arch: arm64
Event: cpu-cycles (type 0, config 0)
Samples: 56
Event count: 13885436

Overhead	Command	Pid	Tid	Shared Object	Symbol
9.61%	sleep	14852	14852	[kernel.kallsyms]	vma_link
8.97%	sleep	14852	14852	linker64	soinfo_do_lookup_impl
6.42%	sleep	14852	14852	linker64	BionicAllocator::alloc_impl
6.11%	sleep	14852	14852	[kernel.kallsyms]	__follow_mount_rcu
5.83%	sleep	14852	14852	[kernel.kallsyms]	clear_page

…

report profile data on host

Pull record file on host and use multiple report methods (scripts are listed here).



Q&A