GPU performance monitoring with perf event

Lin Ming, Intel OTC
ming.m.in@intel.com

Agenda

- Perf event introduction
- Intel GPU counters
- Export GPU counters
- Live GPU counters --- “perf gpu top”
- 3D API callgraph --- “perf gpu record”

Perf event

A framework for performance analysis

- PMU register/unregister
- A system call: sys_perf_event_open
- A file descriptor per event
- Lockless ringbuffer

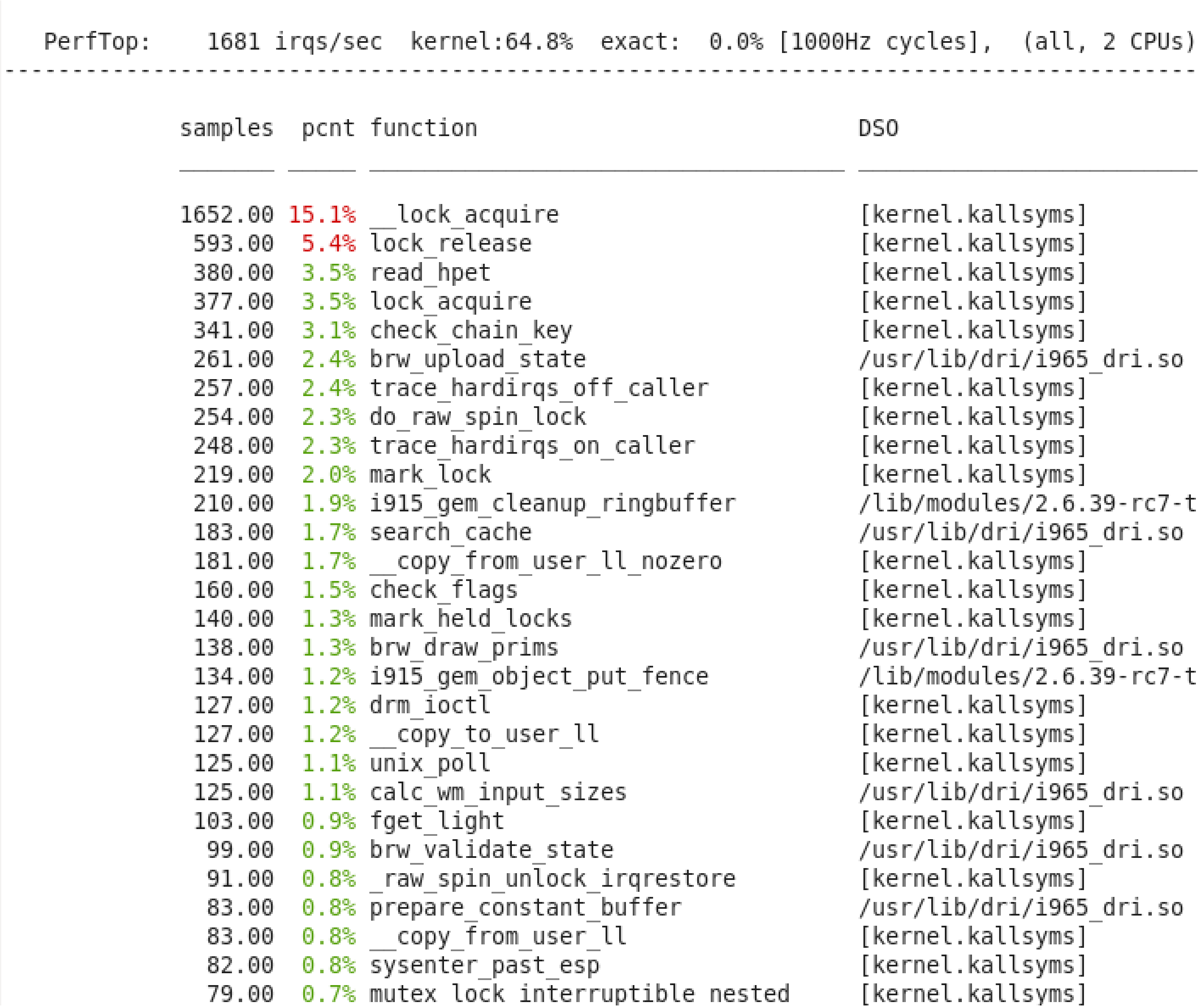
Multiple PMUs support

- CPU counters: cycles, instructions, cache-misses, ...
- Software counters: page faults, context switches, cpu migration, ...
- Tracepoint
- Breakpoint
- GPU counters
- ....

Userspace tools

- “perf” tool
- Perfmon
- PAPI

“perf top” - system profiling

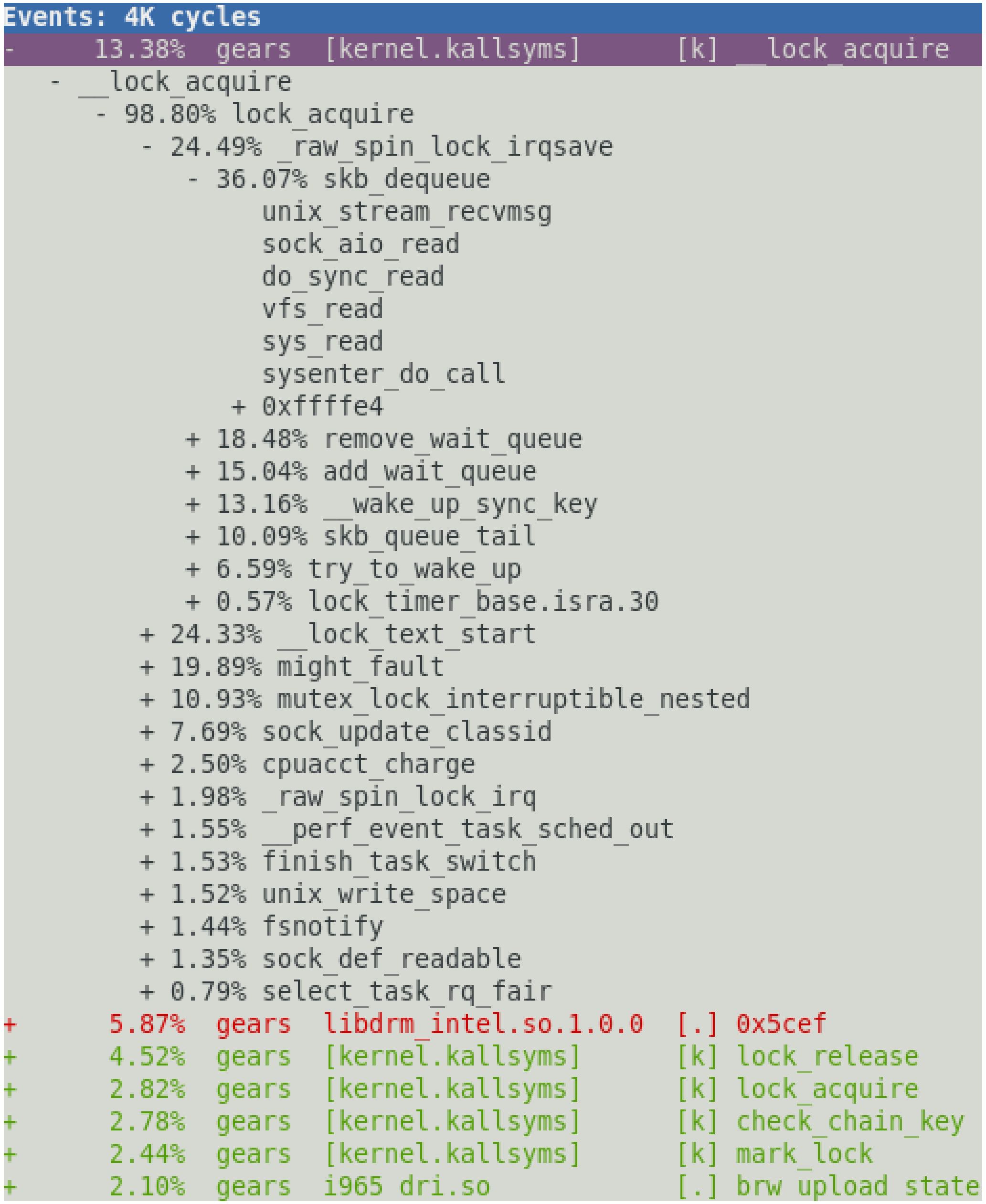


```
   PerfTop:    1681 irqs/sec  kernel:64.8%  exact:  0.0% [1000Hz cycles],  (all, 2 CPUs)
----------------------------------------------------------------------------------------

             samples  pcnt function                            DSO
             _______ _____ ___________________________________ _________________________

             1652.00 15.1% __lock_acquire                      [kernel.kallsyms]
              593.00  5.4% lock_release                        [kernel.kallsyms]
              380.00  3.5% read_hpet                           [kernel.kallsyms]
              377.00  3.5% lock_acquire                        [kernel.kallsyms]
              341.00  3.1% check_chain_key                     [kernel.kallsyms]
              261.00  2.4% brw_upload_state                    /usr/lib/dri/i965_dri.so
              257.00  2.4% trace_hardirqs_off_caller           [kernel.kallsyms]
              254.00  2.3% do_raw_spin_lock                    [kernel.kallsyms]
              248.00  2.3% trace_hardirqs_on_caller            [kernel.kallsyms]
              219.00  2.0% mark_lock                           [kernel.kallsyms]
              210.00  1.9% i915_gem_cleanup_ringbuffer         /lib/modules/2.6.39-rc7-t
              183.00  1.7% search_cache                        /usr/lib/dri/i965_dri.so
              181.00  1.7% __copy_from_user_ll_nozero          [kernel.kallsyms]
              160.00  1.5% check_flags                         [kernel.kallsyms]
              140.00  1.3% mark_held_locks                     [kernel.kallsyms]
              138.00  1.3% brw_draw_prims                      /usr/lib/dri/i965_dri.so
              134.00  1.2% i915_gem_object_put_fence           /lib/modules/2.6.39-rc7-t
              127.00  1.2% drm_ioctl                           [kernel.kallsyms]
              127.00  1.2% __copy_to_user_ll                   [kernel.kallsyms]
              125.00  1.1% unix_poll                           [kernel.kallsyms]
              125.00  1.1% calc_wm_input_sizes                 /usr/lib/dri/i965_dri.so
              103.00  0.9% fget_light                          [kernel.kallsyms]
               99.00  0.9% brw_validate_state                  /usr/lib/dri/i965_dri.so
               91.00  0.8% _raw_spin_unlock_irqrestore         [kernel.kallsyms]
               83.00  0.8% prepare_constant_buffer             /usr/lib/dri/i965_dri.so
               83.00  0.8% __copy_from_user_ll                 [kernel.kallsyms]
               82.00  0.8% sysenter_past_esp                   [kernel.kallsyms]
               79.00  0.7% mutex lock interruptible nested     [kernel.kallsyms]
```

“perf record/report”



```
Events: 4K cycles
-     13.38%  gears  [kernel.kallsyms]      [k] __lock_acquire
   - __lock_acquire
      - 98.80% lock_acquire
         - 24.49% _raw_spin_lock_irqsave
            - 36.07% skb_dequeue
                 unix_stream_recvmsg
                 sock_aio_read
                 do_sync_read
                 vfs_read
                 sys_read
                 sysenter_do_call
               + 0xffffe4
            + 18.48% remove_wait_queue
            + 15.04% add_wait_queue
            + 13.16% __wake_up_sync_key
            + 10.09% skb_queue_tail
            + 6.59% try_to_wake_up
            + 0.57% lock_timer_base.isra.30
         + 24.33% __lock_text_start
         + 19.89% might_fault
         + 10.93% mutex_lock_interruptible_nested
         + 7.69% sock_update_classid
         + 2.50% cpuacct_charge
         + 1.98% _raw_spin_lock_irq
         + 1.55% __perf_event_task_sched_out
         + 1.53% finish_task_switch
         + 1.52% unix_write_space
         + 1.44% fsnotify
         + 1.35% sock_def_readable
         + 0.79% select_task_rq_fair
+      5.87%  gears  libdrm_intel.so.1.0.0  [.] 0x5cef
+      4.52%  gears  [kernel.kallsyms]      [k] lock_release
+      2.82%  gears  [kernel.kallsyms]      [k] lock_acquire
+      2.78%  gears  [kernel.kallsyms]      [k] check_chain_key
+      2.44%  gears  [kernel.kallsyms]      [k] mark_lock
+      2.10%  gears  i965_dri.so            [.] brw_upload_state
```

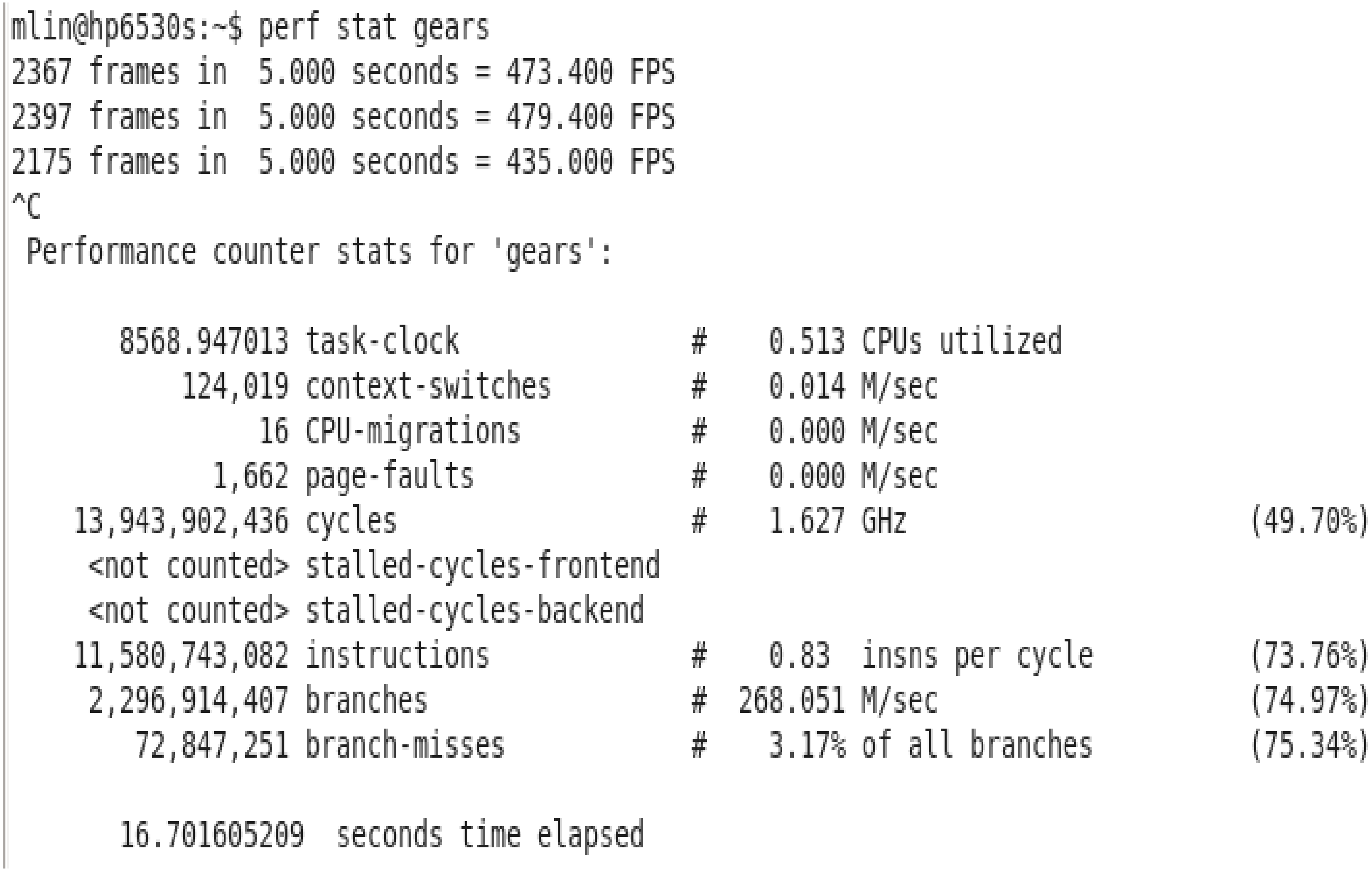
perf record:
run a command
and record its profile into perf.data

perf report:

read perf.data and display the profile

“perf stat”

Run a command and gather performance counter statistics



```
mlin@hp6530s:~$ perf stat gears
2367 frames in  5.000 seconds = 473.400 FPS
2397 frames in  5.000 seconds = 479.400 FPS
2175 frames in  5.000 seconds = 435.000 FPS
^C
 Performance counter stats for 'gears':

       8568.947013 task-clock                #    0.513 CPUs utilized
           124,019 context-switches          #    0.014 M/sec
                16 CPU-migrations            #    0.000 M/sec
             1,662 page-faults               #    0.000 M/sec
    13,943,902,436 cycles                    #    1.627 GHz                     (49.70%)
     <not counted> stalled-cycles-frontend
     <not counted> stalled-cycles-backend
    11,580,743,082 instructions              #    0.83  insns per cycle         (73.76%)
     2,296,914,407 branches                  #  268.051 M/sec                   (74.97%)
        72,847,251 branch-misses             #    3.17% of all branches         (75.34%)

      16.701605209  seconds time elapsed
```

Add a new PMU

- Initialize the event for the PMU

```
int (*event_init)  (struct perf_event *event);
```

- Adds/Removes a counter to/from the PMU

```
int  (*add) (struct perf_event *event, int flags);
void (*del) (struct perf_event *event, int flags);
```

- Starts/Stops a counter present on the PMU

```
void (*start) (struct perf_event *event, int flags);
void (*stop) (struct perf_event *event, int flags);
```

- Updates the counter value of the event

```
void (*read) (struct perf_event *event);
```

- Fully disable/enable this PMU (optional)

```
void (*pmu_enable) (struct pmu *pmu);
void (*pmu_disable) (struct pmu *pmu);
```

- Group events scheduling (optional)

```
void (*start_txn)  (struct pmu *pmu);
int  (*commit_txn) (struct pmu *pmu);
void (*cancel_txn) (struct pmu *pmu);
```

Intel GPU counters

- Core active
- Core stall
- VS active
- VS stall
- VS threads loaded
- VS ready
- GS active
- GS stall
- GS threads loaded
- GS ready
- PS active
- PS stall
- PS threads loaded
- PS ready
- Pixel kill count
- Alpah test Pixels failed
- CS starve
- Vertices
- Vertex Fetch Output Primitives
- Vertex Shader Invocation
- Hull Shader Invocation
- Domain Shader Invocation
- Geometry Shader Thread Invocation
- Clipper Thread Invocation
- Clipper Output Primitives
- Pixels Shaded
- CL active
- ......
- ......

Intel GPU counters II

- Pipelines Statistics Counter Registers
  MMIO, i915_read/i915_read64

- Performance Statistics Registers
  MI_REPORT_PERF_COUNT command

  - Allocate GPU memory
  - Map GPU memory
  - Emit MI_REPORT_PERF_COUNT command
  - Read counters

Export GPU counters

```
$ tree /sys/bus/event_source/devices/gpu

├── events
│   ├── cl_invocation_count
│   ├── cl_primitives_count
│   ├── gs_invocation_count
│   ├── gs_primitives_count
│   ├── ia_primitives_count
│   ├── ia_vertices_count
│   ├── maxq_flip_a
│   ├── maxq_flip_b
│   ├── num_flip_a
│   ├── num_flip_b
│   ├── ps_depth_count
│   ├── so_num_prims_written
│   └── so_prim_storage_needed
├── power
├── subsystem -> ../../bus/event_source
├── type
└── uevent
```

Export GPU counters II

```
$ perf list
……

 ia_vertices_count               [GPU event]
 ia_primitives_count             [GPU event]
 gs_invocation_count             [GPU event]
 gs_primitives_count             [GPU event]
 cl_invocation_count             [GPU event]
 cl_primitives_count             [GPU event]
 ps_depth_count                  [GPU event]
 so_num_prims_written            [GPU event]
 so_prim_storage_needed          [GPU event]
 maxq_flip_a                     [GPU event]
 maxq_flip_b                     [GPU event]
 num_flip_a                      [GPU event]
 num_flip_b                      [GPU event]
```

```
$ perf stat -e ia_vertices_count gears

 Performance counter stats for 'gears':

          406,904 ia_vertices_count

    1.797759968  seconds time elapsed
```

An example
i915 pipeline statistics pmu

- Initialize the event for the PMU

i915_pmu_event_init,

- Adds/Removes a counter to/from the PMU

i915_pmu_add,
i915_pmu_del,

- Starts/Stops a counter present on the PMU

i915_pmu_start,
i915_pmu_stop,

- Updates the counter value of the event

i915_pmu_read,

- Export events via sysfs

i915_pmu_sysfs_add_events,

i915_pmu_event_init

```
int i915_pmu_event_init(struct perf_event *event)
{
        u64 counter = event->attr.config;

        if (event->attr.type != PERF_TYPE_GPU)
                return -ENOENT;

        if (counter >= I915_COUNTER_MAX)
                return -ENOENT;

        event->hw.counter_base = i915_event_map[counter].addr;
        event->hw.counter_size = i915_event_map[counter].size;

        return 0;
}
```

i915_pmu_{start,stop,add,del}

```
void i915_pmu_start(struct perf_event *event, int flags)
{
	u64 now = i915_counter_read(event);

	local64_set(&event->hw.prev_count, now);
}

void i915_pmu_stop(struct perf_event *event, int flags)
{
	i915_perf_event_update(event);
}

int i915_pmu_add(struct perf_event *event, int flags)
{
	if (flags & PERF_EF_START)
		i915_pmu_start(event, flags);

	return 0;
}

void i915_pmu_del(struct perf_event *event, int flags)
{
	i915_pmu_stop(event, flags);
}
```

i915_pmu_read

```
static u64 i915_counter_read(struct perf_event *event)
{
	struct drm_device *dev = i915_pmu_drm_device(event->pmu);
	drm_i915_private_t *dev_priv = dev->dev_private;
	u64 now;

	if (event->hw.counter_size == 64)
		now = I915_READ64(event->hw.counter_base);
	else
		now = I915_READ(event->hw.counter_base);

	return now;
}

void i915_perf_event_update(struct perf_event *event)
{
	s64 prev;
	u64 now;

	now = i915_counter_read(event);
	prev = local64_xchg(&event->hw.prev_count, now);
	local64_add(now - prev, &event->count);
}

void i915_pmu_read(struct perf_event *event)
{
	i915_perf_event_update(event);
}
```

“perf gpu top” - live GPU counters

```
                          GPU counters(2 sec)
========================================================================

         ia_vertices_count: 1555008
       ia_primitives_count: 701792
      gs_invocation_count: 582400
       gs_primitives_count: 0
       cl_invocation_count: 0
       cl_primitives_count: 1284192
            ps_depth_count: 168583756
      so_num_prims_written: 0
    so_prim_storage_needed: 0
               maxq_flip_a: 0
               maxq_flip_b: 0
                num_flip_a: 0
                num_flip_b: 0
```

3D API trace

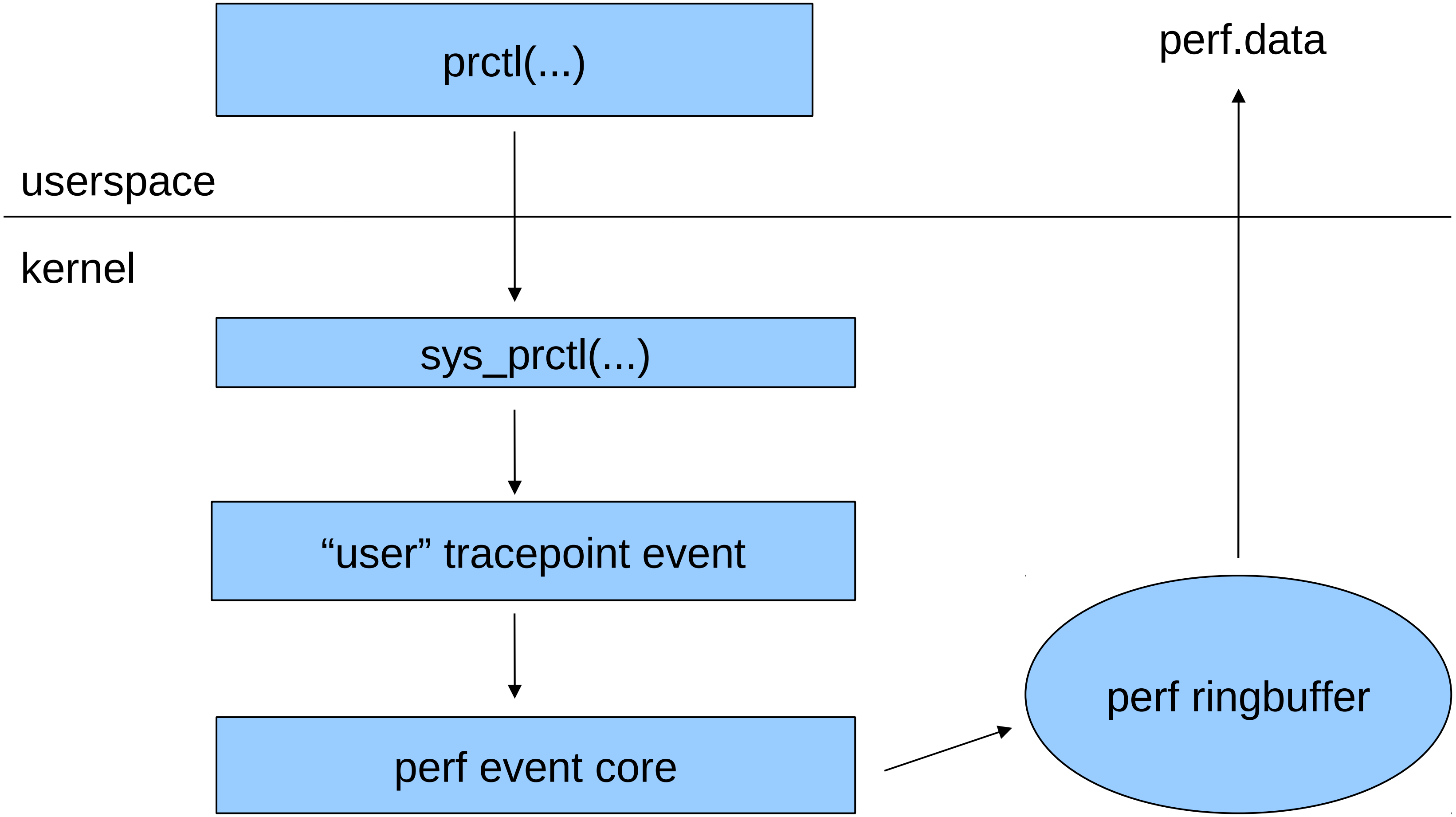
3D library interposers: apitrace
https://github.com/apitrace/apitrace

```
extern "C" PUBLIC
void APIENTRY glCullFace(GLenum mode) {
    unsigned __call = Trace::BeginEnter(__glCullFace_sig);
    Trace::BeginArg(0);
    __traceEnum70(mode);
    Trace::EndArg();
    Trace::EndEnter();
    __glCullFace(mode);
    Trace::BeginLeave(__call);
    Trace::EndLeave();
}
```

```
LD_PRELOAD=/path/glxtrace.so
```

Perf userspace trace

Ingo Molnar: [patch] trace: Add user-space event tracing/injection
https://lkml.org/lkml/2010/11/17/171



prctl(...)

perf.data

userspace

kernel

sys_prctl(...)

“user” tracepoint event

perf ringbuffer

perf event core

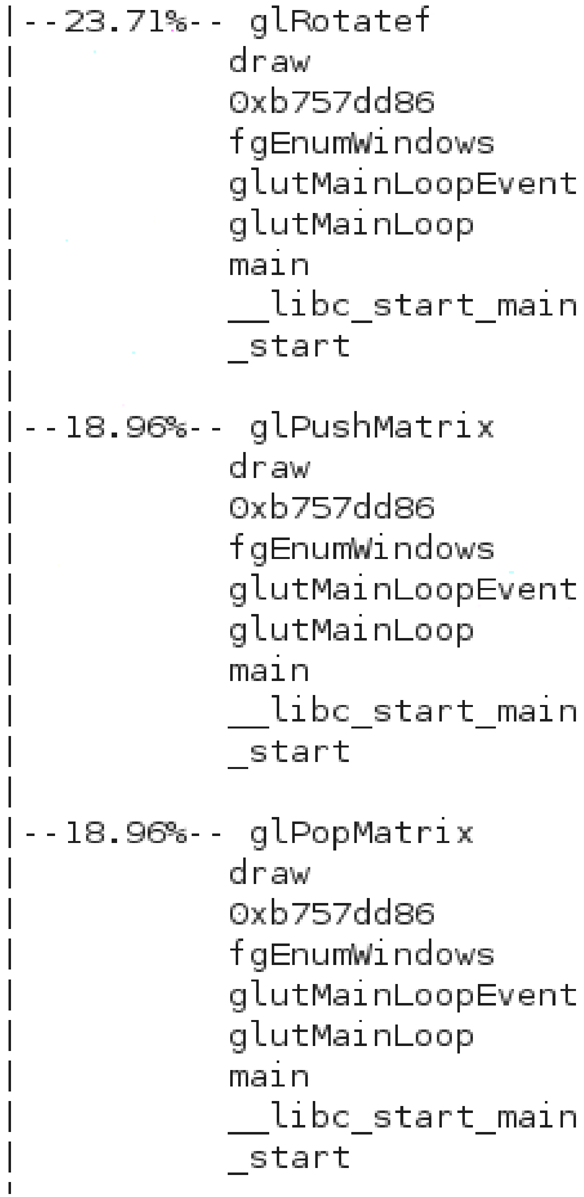
Hack apitrace

```
extern "C" PUBLIC
void APIENTRY glCullFace(GLenum mode) {
    unsigned __call = Trace::BeginEnter(__glCullFace_sig);
    Trace::BeginArg(0);
    __traceEnum70(mode);
    Trace::EndArg();
    Trace::EndEnter();
    __glCullFace(mode);
    Trace::BeginLeave(__call);
    Trace::EndLeave();

    Trace::PerfEvent();
}

void PerfEvent(void) {
        prctl(PR_TASK_PERF_USER_TRACE, "gpu api trace");
}
```

“perf gpu record” - 3D API callgraph



```
|--23.71%-- glRotatef
|           draw
|           0xb757dd86
|           fgEnumWindows
|           glutMainLoopEvent
|           glutMainLoop
|           main
|           __libc_start_main
|           _start
|
|--18.96%-- glPushMatrix
|           draw
|           0xb757dd86
|           fgEnumWindows
|           glutMainLoopEvent
|           glutMainLoop
|           main
|           __libc_start_main
|           _start
|
|--18.96%-- glPopMatrix
|           draw
|           0xb757dd86
|           fgEnumWindows
|           glutMainLoopEvent
|           glutMainLoop
|           main
|           __libc_start_main
|           _start
|
```

```
$ sudo perf gpu record gears

perf record: Woken up 28 times to write data ]
[ perf record: Captured and wrote
6.982 MB perf.data (~305055 samples) ]

$ sudo perf report
```

Questions?

Thanks!