Expose NVIDIA's performance counters to the userspace for NV50/Tesla

Nouveau project

Samuel Pitoiset

Supervised by Martin Peres

GSoC student 2013 & 2014

October 8, 2014

Summary

What are performance counters ?

Performance counters

- are blocks in modern processors that monitor their activity;
- count low-level hardware events such as cache hit/misses.

Why performance counters are used ?

- To analyze the bottlenecks of 3D and GPGPU applications;
- To dynamically adjust the performance level of the GPU.

NVIDIA's performance counters

Two kind of counters exposed by NVIDIA

- **compute counters** for GPGPU applications:
  - exposed through CUPTI (CUDA Profiling Tools Interface).
- **graphics counters** for 3D applications:
  - exposed through PerfKit, only on Windows...

Nouveau's performance counters

Current status

- compute counters support for Fermi and Kepler;
- exposed to the userspace through Gallium-HUD;
- Kepler support by Christoph Bumiller (calim);
- Fermi support by myself (GSoC 2013).

but many performance counters left to be exposed...

Proposal

Off-season work

- reverse engineered **graphics counters** using PerfKit on W7.

Google Summer of Code 2014

- expose NVIDIA's graphics counters for Tesla (NV50):
  - **kernel interface** in Nouveau DRM;
  - mesa & GL_AMD_performance_monitor;
  - nouveau-perfkit.

Benefits to the community

- help developers to find bottlenecks in their 3D applications.

Summary

The performance counters engine

PCOUNTER: General overview

- contains most of the performance counters;
- is made of several identical hardware units called domains;
- each domain has 256 input signals;
- input signals are from all over the card (**global counters**);
- performance counters are tied to a clock domain.

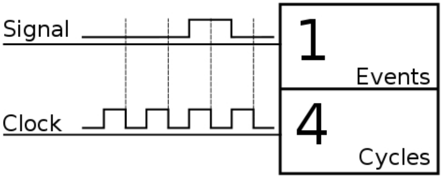


Figure : Example of a simple performance counter

Overview of a domain

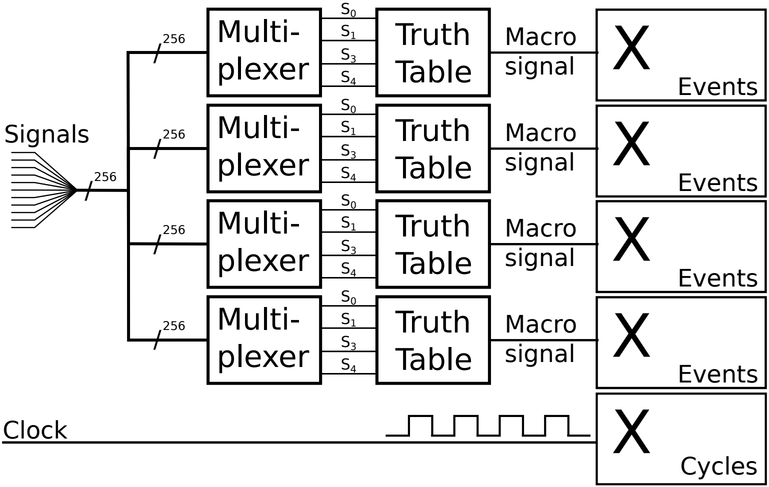


Figure : Schematic view of a domain from PCOUNTER

Other counters ?

Per-context counters (or MP-counters)

- per-channel/process counters in PGRAPH;
- more accurate than global counters;
- same logic as PCOUNTER;
- share some in-engine multiplexers with PCOUNTER;
- currently require running an OpenCL kernel to read them.

Counters - Which signals are known ?

Per-context counters (MP)

- all GPGPU signals for Tesla, Fermi and Kepler reversed;
- reverse engineered by Christoph Bumiller and myself.

Global counters (PCOUNTER)

- very **chipset-dependant**;
- more than 200 signals reverse engineered on NV50/Tesla;
- work done by Marcin Kościelnicki (mwk) and myself.

What about graphics counters ?

- almost-all 3D signals exported by PerfKit on NV50 reversed;
- some per-context counters still need to be reversed.

Summary

Reverse engineering of graphics counters

Reverse engineering on Windows...

- 3D signals are exposed through PerfKit, only on Windows;
- can't use envytools (a collection of NVIDIA-related tools);
- ... because libpciaccess doesn't work on Windows!

Bring it on!

- added libpciaccess support for Windows/Cygwin;
- envytools can now be used on Windows;
- no MMIO traces and no valgrind-mmt...;
- let's start the reverse engineering process. :)

How does it work?

Reverse engineering process

1. configure the hardware counters with PerfKit on W7;
2. dump the configuration with some tools of envytools:
   - but some multiplexers are very difficult to find!
3. regenerate the same result by polling the counters on W7;
4. reproduce the configuration on Linux/Nouveau;
5. go to step 1...
   - around 50 graphics counters exposed on Tesla family;
   - and 14 different chipsets (ouch)!

OGL Performance Experiments

- a modified version of OGLPerfHarness (PerfKit);
- to help in the reverse engineering process.

OGL Performance Experiments

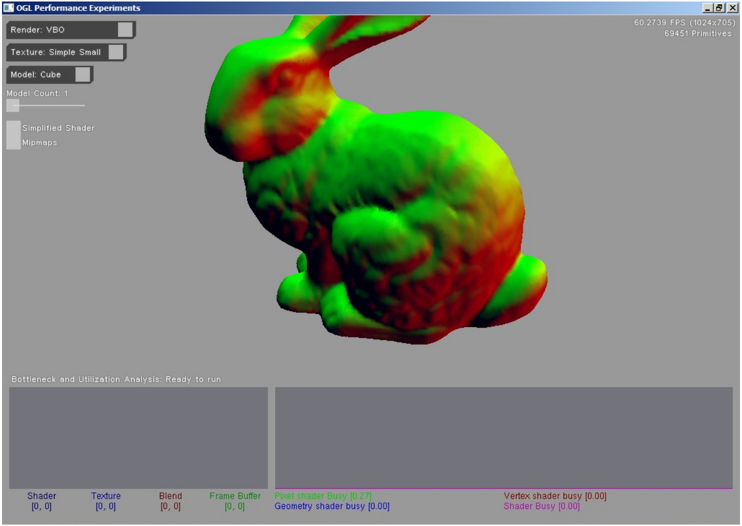


Figure : Screenshot of OGLPerfHarness (based on PerfKit) on W7

Summary

Introduction

Why is a kernel interface needed ?

- because **global counters** have to be programmed via MMIO:
  - only root or the kernel can write to them.

What the interface has to do ?

- set up the configuration of counters;
- poll counters;
- expose counter's data to the userspace (readout).

Synchronization

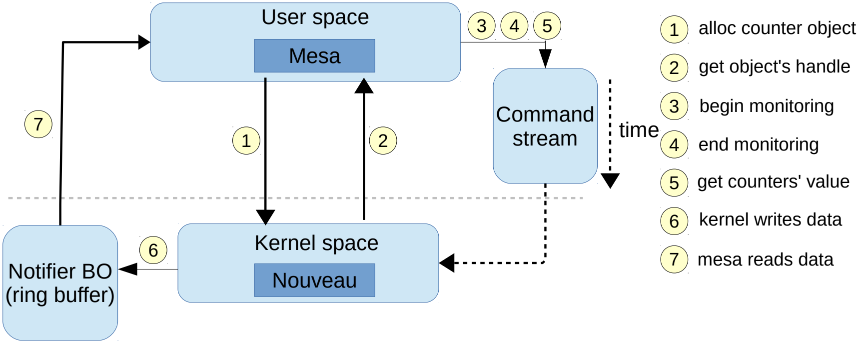
Synchronizing operations

- CPU: ioctls;
- GPU: **software methods**.

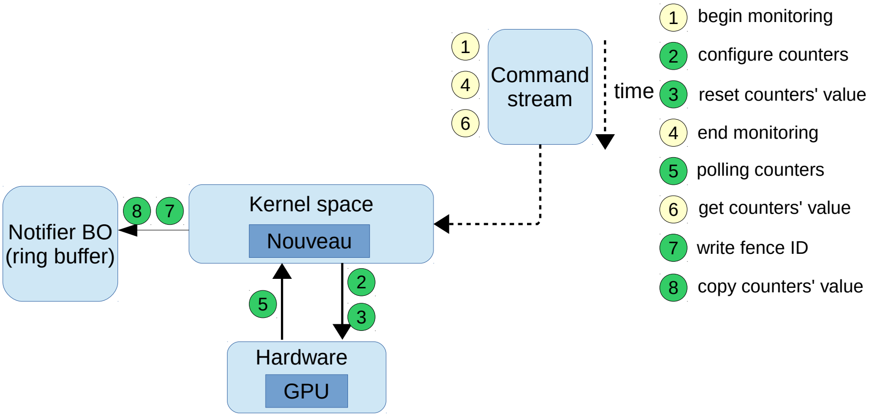
Software method

- command added to the command stream of the GPU context;
- upon reaching the command, the GPU is paused;
- the CPU gets an IRQ and handles the command.

Overview from Mesa's PoV



Overview from the GPU's PoV



How to synchronize different queries ?

A detailed look at the ring buffer

- mesa sends a query ID to read out results;
- this sequence number is written at the offset 0:
    - easy to check if the result is in the ring buffer.
- the ring buffer queues up 8 queries/frames (like the HUD):
    - avoid stalling the command submission.

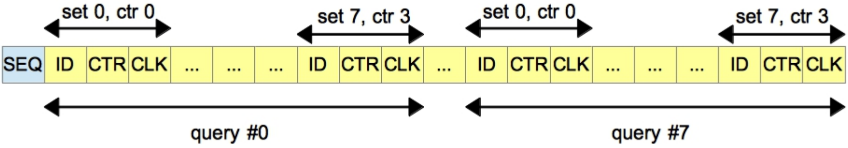


Figure : Schematic view of the ring buffer

Summary

Perfmon APIs

Performance counters APIs

- Proprietary: Perfkit, CUPTI, GL_AMD_perfmon;
- OSS: Gallium HUD only.

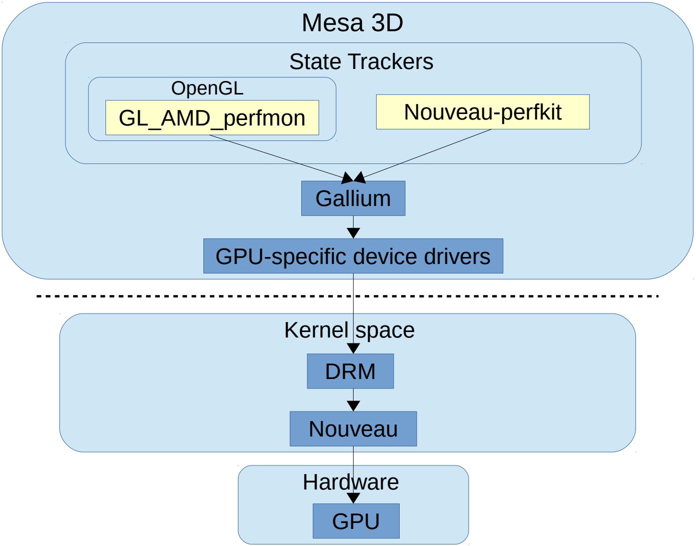
GL_AMD_performance_monitor

- patches available for nvc0, svga, freedreno and radeon drivers;
- my patch set (v4) is pending on mesa-dev:
  - initial work by Christoph Bumiller.

nouveau-perfkit

- a Linux/Nouveau version of NVIDIA PerfKit;
- built on top of mesa (Gallium state tracker like vdpau);
- work in progress.

General overview



Summary

Conclusion

Current status

- all 3D global counters on Tesla (NV50) reversed;
- kernel interface & mesa implementation is on the way:
  - hope to see the code in Linux 3.20.
- GL_AMD_performance_monitor's patches are pending.

TODO list

- implement nouveau-perfkit as a Gallium state tracker;
- reverse engineer more performance counter signals:
  - graphics counters support for Fermi and Kepler.
- all the work which can be done around performance counters.

Questions & Discussions

Questions & Discussions

And for more information you can take a look at my blog
http://hakzsam.wordpress.com