About me:

• Working on Linux platforms since 2004, with a background on embedded devices.


• Joined Mesa in 2014, working on performance tools and automation.
Investigate system bottlenecks first

- top, gputop, rapl
- 100% GPU utilization with lower CPU utilization indicates a GPU-bound workload
- TDP limited workloads cause GPU clock rate to fall.
- MESA_DEBUG=perf
GPU Performance Analysis Workflow

- CPU Bound workloads have traditional tools
  - perf, callgrind, cachegrind, sysprof
- GPU performance analysis has a sparse landscape of Linux tools
  - AMD GPU PerfStudio, Nvidia Linux Graphics Debugger, QApiTrace
  - Leverage GPU hardware counters to quantify the cost of asynchronous GPU operations.
  - Live experimentation to see the effect on performance.
  - Deeply investigate a graphics workload.
GPU Tools stumbling blocks

- Generally hardware-specific
- Mostly closed source
- Linux support is an afterthought
- Tracing/retracing not reliable
- Low numbers of users
- Mesa support for GPU performance counters
FrameRetrace: frame analysis based on ApiTrace

- Widely used and high quality trace/retrace
- https://github.com/janesma/apitrace
- Cross-platform: Linux and Windows
- Hardware agnostic: Support for Intel, AMD. More to come.
- Upstream GPU Counter support in Mesa and Kernel for Haswell and later.
- Leveraged by Intel Mesa team to identify and fix several performance issues in i965.
FrameRetrace: frame analysis based on ApiTrace

- GPU Metrics for each render
- Render target visualization and experiments
- Api log
- Batch disassembly
- Shader analysis, live editing, and assembly
- Uniform constant display and live editing
- Render experiments
- State display and live editing
Demo
Other features

- Windows support provides important leverage for open source driver teams seeking to find Mesa performance gaps.

- Proposed features:
  - Display texture state, with mip clamp experiment
  - Display geometry mesh
  - Depth buffer visualization
  - Overdraw / hotspot rendertarget visualization
  - UI improvements
  - Support for more hardware
  - Android support
Caveats

- Currently a one-person project, with help
  - Thanks to Laura Ekstrand, Robert Bragg, Lionel Landerwelín, Eero Taminen, Pekka Jylhä-Ollila
- Experiments require intricate state tracking
- Some workloads do not have single-frame run loops
FrameRetrace has been enhanced to support Intel’s on-die AMD GPUs

- GPUS, GPUBusy, TessellatorBusy, HSBusy, DSBusy, GSBusy, PSBus, CSBus, VSVert, HSPatches, HSVALUInstCount, HSSALUInstCount, HSSALUInstBus, DSVertIce, GSTri, GSVertOut, GSVLInstCount, GSSLUInstCount, GSVL Inst Bus, GSSALUInstBus, PrimitivesIns, ClippedPrims, PASTalledOnRasterizer, PSPixelsOut, PSExtInsts, PSSALInstCount, PSVALUInstCount, PSALUInstBus, CSThreadGroups, CWavefronts, CSThreads, CSVALUInsts, CSVALUInstCount, CSSALUInsts, CSVfetchInsts, CSSFetchInsts, CSVWriteInsts, CSFlatVMemInsts, CSVALUInstBus, CSSALUInstBus, CSMemUnitBusy, CSMemUnitStalled, CSFetchSize, CWriteSize, CSCacheHit, CWriteUnitStalled, CSGDSInsts, CSLDSInsts, CSFlatLDSInsts, CSALUInstStalledByLDS, CSLDSBankConflict, TexUnitBusy, TexFilterPct, TexVolFilterPct, DepthStencilTestBusy, HiZTilesAccepted, PreZTilesDetailCulled, HIZQuadsCulled, PostZQuads, PreZSamplesPassing, PreZSamplesFailing, PreZSamplesFailingZ, PostZSamplesPassing, PostZSamplesFailing, PostZSamplesFailingZ, ZUnitStalled, CBMemRead, CBMemWritten, CBSlowPixelPct

AMD does not meaningfully implement their own metrics extension, and requires the GPA library to produce data.

Raspberry Pi and Nouveau support AMD’s extension
GPUTop demo

- New UI built on top of ImGui
  - Simplifies build and deployment
  - System-wide & per-context metrics graphs
  - Text collection of GPU metrics
  - (in progress) timeline of trace events
Questions?