Steps Towards a Gaming-Optimized Scheduler

Changwoo Min
changwoo@igalia.com

December 13, 2025

About Me & LAVD

- **A happy Igalian**
  - An author of the LAVD scheduler
- **What is the LAVD scheduler?**
  - LAVD: Latency-criticality Aware Virtual Deadline scheduler
  - Motivated by gaming workloads
  - Aim to minimize stuttering during game play
  - Primary target: Windows games running on Linux (SteamOS)
  - Implemented based on the sched_ext framework (BPF + Rust)
  - https://github.com/sched-ext/scx/tree/main/scheds/rust/scx_lavd

Goals of the talk

- **Share the lessons learned while developing LAVD**

- **Ignite the discussion about the missing areas where the community needs more attention:**
  - Analysis tool
  - Benchmark
  - Better support for Windows games

What is important in gaming?

- **Performance**
  - Frame Per Second (FPS) is a widely-adopted performance metric.
  - Average FPS: ~= throughput
  - Low 1% FPS, Min FPS: ~= 99 percentile latency
  - "Stuttering" is more relevant to "Low 1%" and "Min FPS".
- **Energy consumption**
  - Many gaming devices are battery-powered. Less energy consumed, more we can play!
    - E.g., Gaming handheld console (SteamDeck), VR glasses (Steam Frame)
  - Hybrid processors are popular (big/medium/little cores).
    - E.g., Intel, AMD, Qualcomm Snapdragon
  - Not all tasks are created equal.
    - Some are okay to run on a slower but energy-efficient core.

What can scheduler do?

- **Scheduler essentially decides three things:**

  (1) Which task should run first? ⇒ Deadline

  - If latency-critical tasks are delayed, it could cause stuttering.

  (2) How long should the task run? ⇒ Time slice

  - A task should run long enough for warm cache, but it should not monopolize a CPU.

  (3) Which CPU should the task run on? ⇒ CPU selection

  - Choose energy-efficient core when appropriate.

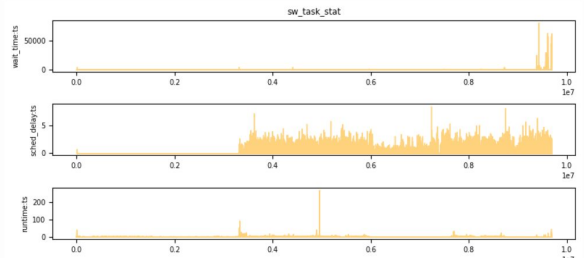
Understanding gaming workload is critical

- **If there are some outstanding properties in gaming, we can leverage those to design gaming-optimized scheduling policies.**
- **We developed an analysis tool, VaporMark**
  - https://github.com/Igalia/vapormark
- VaporMark collects all the scheduling activities during a period using “perf sched record”
- Then, it analyzes the collected trace to understand high-level properties.

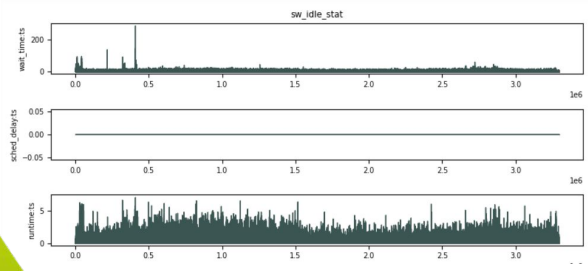
VaporMark analysis report

System-wide task and idle statistics

- System-wide task stats data

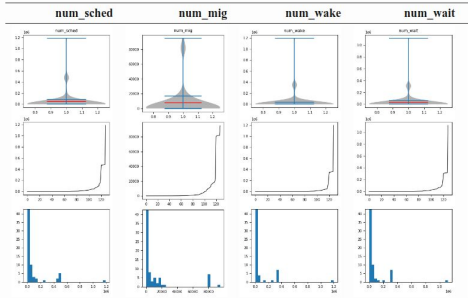


- System-wide idle stats data



Task statistics using per-task average

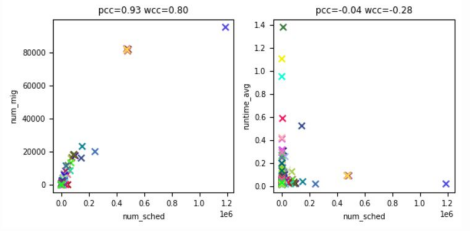
- task stats data



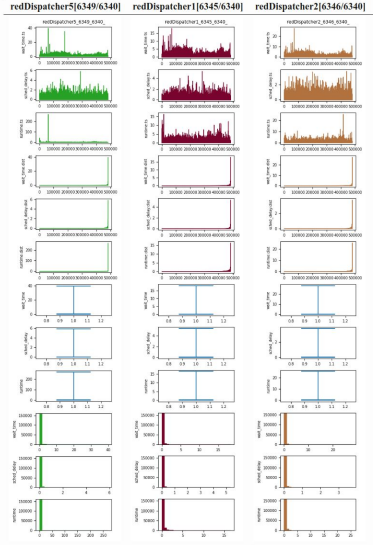
Correlation between task features

- PCC: Pearson correlation coefficient
- WCC: Weighted correlation coefficient using num_sched

- How userful is num_sched in predicting task's behavior?

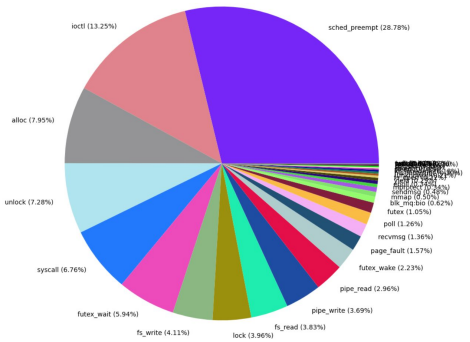


Task's wait_time, sched_delay, and runtimes

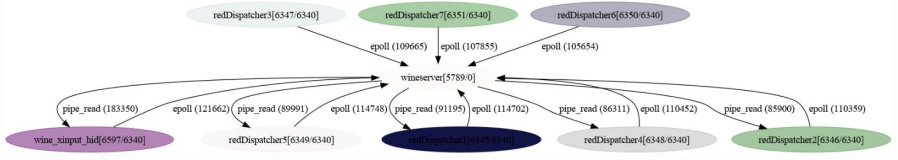


Callstacks tiggered scheduling

- callstacks
- callstack_types

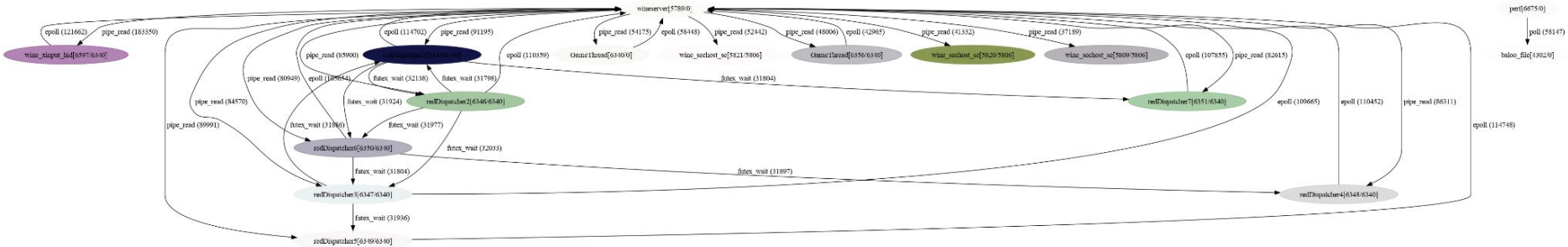


- 30.00 percentile of waker-waiter



Key finding: task graph, waker-wakee

- **To accomplish a single high-level job (e.g., moving a character upon a keypress event), many tasks should tightly collaborate.**
- **Task graphs of top 50 percentile of waker-wakees**



- **Let’s prioritize a task with high waker-wakee frequency!**
  - Those tasks are in the middle of the task chain.

We need more analysis tools

- **VaporMark is good to understand the high-level properties quickly.**
  - Perfetto is better for microscopic analysis in figuring out pathological scheduling behaviors.
- **Two possible directions**:

  (1) A problematic behavior (e.g., FPS drop) happens rarely.
    - Can we trigger monitoring only when the problem is detected?

  (2) Extend the scope of resource monitoring beyond CPU time
    - Other resources (e.g., memory, GPU, disk, etc) are actually used only when a task is running.
    - A scheduler is in a unique position in resource monitoring.
    - A scheduler would be a centralized entity for resource usage control.

Benchmarking games is hard

- **We need reliably reproducible benchmarks to compare two scheduling policies.**
- **Many problem domains define standard set of benchmarks:**
  - E.g., SPEC CPU, SPEC JBB, TPC-C, TPC-H, MLPerf
- **Thankfully, some games provide in-game benchmarks!**
  - E.g., Cyber Punk 2077, Far Cry, Forza Horizon, etc.
- **Or, some games allow to replay the recorded game sequences.**
  - E.g., Counter Strike

Benchmarking games is hard

- **However, it is unclear how representative they are.**
- **Moreover, games are often updated too frequently.**
  - It would be meaningless comparing a result of 2-month ago.
- **Some games rely on external resources (e.g., their game server) while running the benchmark.**
  - So, results vary a lot sometime. :-(

We need a proper benchmark

- **Microbenchmark is useful since it is easy to run and compare results.**
- **However, most scheduler benchmarks focus on stressing schedulers.**
  - E.g., stress-ng, perf bench, hackbench
  - Improving the score of those benchmark may not improve the actual game performance.
  - We may end up shooting a wrong target. :-(

We need a proper benchmark

- **There are some microbenchmarks that mimic the behavior of a certain workloads:**
  - schbench: reproduce the scheduler characteristics of a production web workload
- **We need something similar to schbench for gaming, say *gamebench*?**
- **In addition to that, we need a benchmark suite, which is a collection of games, say *gamesuite*?**
  - E.g., CloudSuite is a benchmark suite for cloud services.
- **Either microbenchmarks and benchmark suites, they should provide rich information:**
  - Not only average FPS, but also Low 1% or min FPS

We need better support for Windows games

- **Lock holder preemption is a well-known problem.**
  - Task A that holds lock X got scheduled out.
  - Task B is scheduled, and tries to acquire the same lock X.
  - However, lock X is already acquired by Task A, so Task B cannot progress.
- **Proxy execution partially addresses the problem.**
  - Handles only kernel locks.
- **LAVD also partially addresses the problem.**
  - Handles only futex, which is a building block of user-space locks.
- **What about NTsync?**
  - NTsync is a support driver for emulation of NT synchronization primitives by user-space NT emulators.
  - A gaming-optimized scheduler needs to know the semantics of NTsync for better handling of lock-holder preemption problem.

Join us!

https://www.igalia.com/jobs