Using sched_ext to improve frame rates on the SteamDeck

Ideas behind the LAVD scheduler

Changwoo Min
changwoo@igalia.com

September 18, 2024

igalia

Talk outline

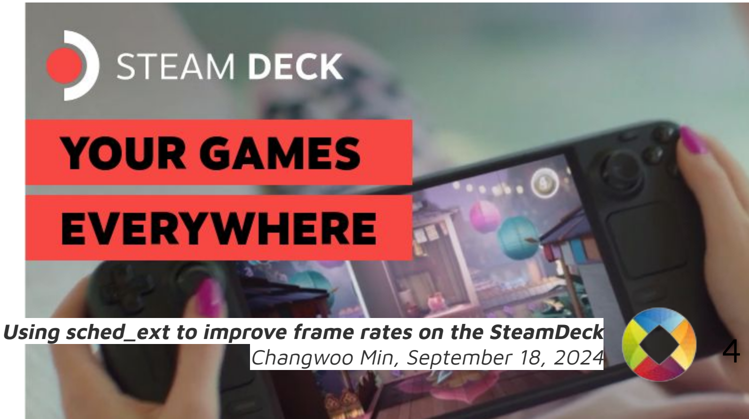
- **Brief introduction to the LAVD scheduler**
  - What is the LAVD (Latency-criticality Aware Virtual Deadline) scheduler specialized for?
- **Gaming workload characterization**
  - Are there any properties in gaming for schedulers?
- **Scheduling tasks in LAVD**
  - When to schedule a task and for how long?
- **All cores are not created equal**
  - Specialization for processor heterogeneity
- **No silver bullet for all usage scenarios**
  - Specialization for usage scenarios
- **Discussion / Q&A**

The LAVD scheduler

- **LAVD: Latency-criticality Aware Virtual Deadline**
- **A sched_ext scheduler inspired by the gaming workload characteristics.**
  - Gaming workload is the main workload it's optimized for.
- **Latency criticality of a task is the key in making scheduling decisions.**
  - It can be derived from gaming workload characteristics.
- **LAVD handles heterogeneous cores differently.**
  - Intel P-/E-cores, ARM big/LITTLE cores.
- **It adapts its scheduling policy according to usage pattern.**
  - Almost idle, medium-load and high-load systems.

Goals and non-goals

- **Goals**
  - Provide **the best gaming experience in Linux platform**
  - Provide **high gaming performance without stuttering**
    - High FPS without sudden FPS dips
    - Maximize Low 1% FPS ⇒ minimize tail latency
    - Maximize average FPS ⇒ maximize throughput
  - Provide **reasonably good performance across various workload and hardware combinations**

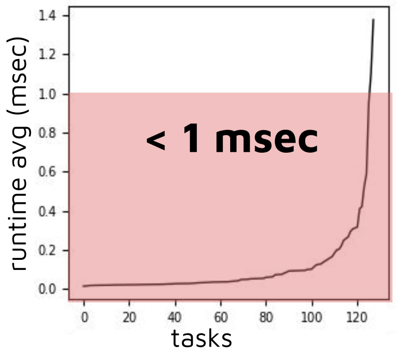


- **Non-goals** (at least for now)
  - Be the best scheduler for server workloads
  - Be the best general-purpose scheduler

Understanding game workloads

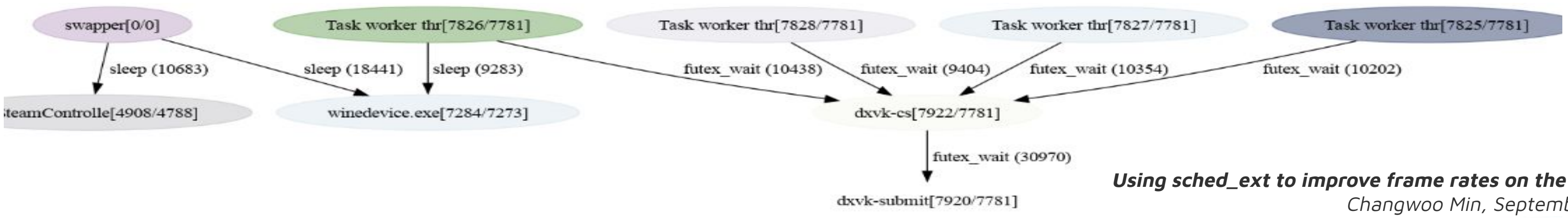
- **Tasks run for very short duration**
  - roughly a few 100s usec on average

Distribution of task's runtime average per schedule in a game



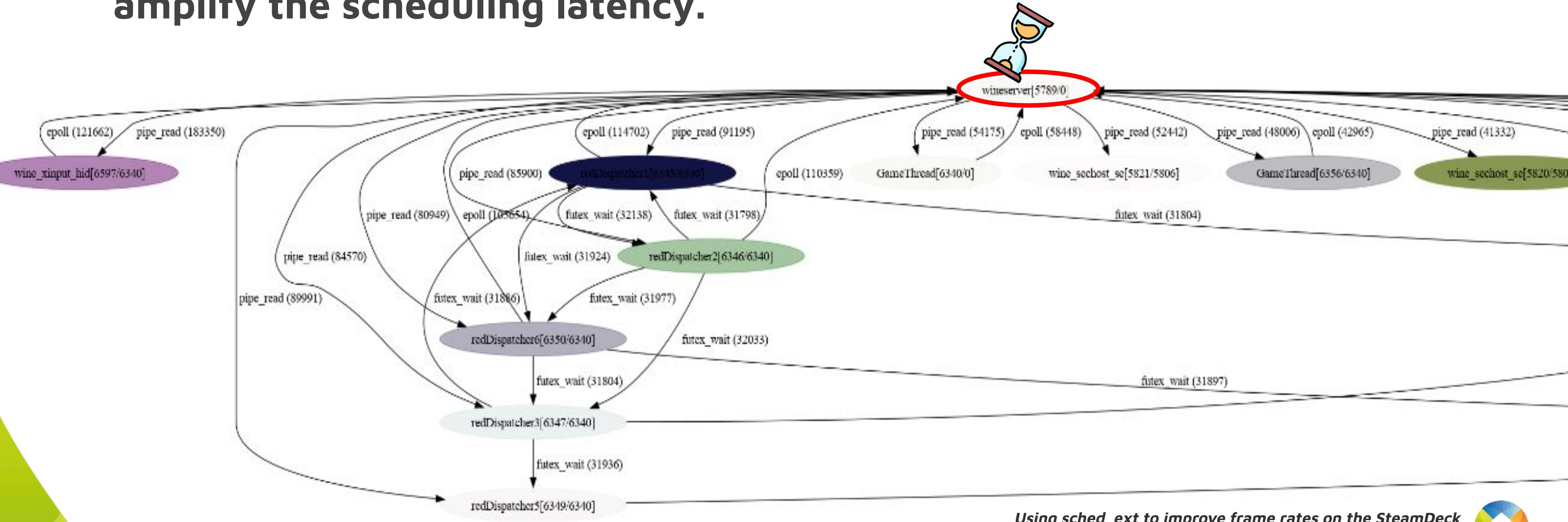
- **Multiple tasks are tightly linked to in a task graph to finish a single job** (e.g., updating a frame).

Top 50% of waiters and wakers



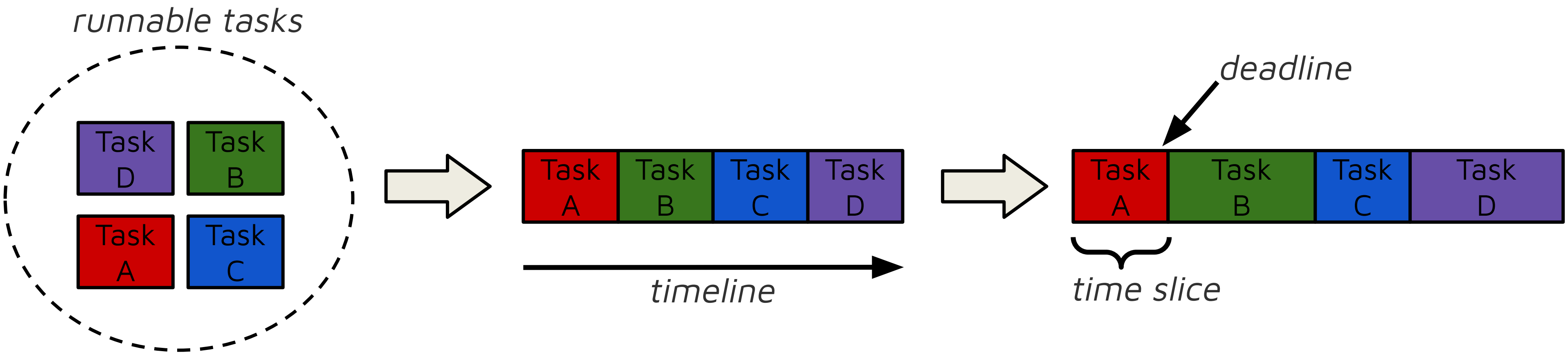
Implication of the characteristics

- **Scheduling delay in a critical path** of a task graph will significantly amplify the scheduling latency.



Scheduling tasks in LAVD

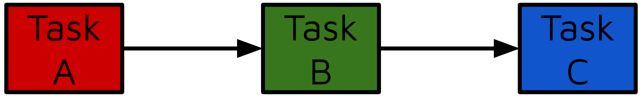
- **Task scheduling determines two things:**
  - Given a set of tasks, which task should run first, second, etc?
  - How long each task should run?



Which task should run first?

- **Schedule a latency-critical task first**
  - Then, how to define a latency criticality of a task?

Let's leverage the task graph!

- 

- **Wake up frequency**
  - Task A wakes up Task B, then Task B wakes up Task C.
- **Wait frequency**
  - Task C waits for Task B, Task B waits for Task A.

Which task should run first?

- **Implications of wake-up/wait frequency**
  - High wake up frequency ⇒ important producer in a task graph
  - High wait frequency ⇒ important consumer in a task graph
  - Both are high ⇒ important task in the middle

- **Virtual deadline of a task**
  - **pipeline factor = f(wake freq, wait freq)**
  - given priority = f(nice priority)
  - fairness factor = f(vruntime)

How long each task should run?

- **We want to make each and every task run in a fixed time interval.**
  - a fixed time interval == targeted latency (e.g., 15 msec)
  - This helps ensuring the forward progress of all tasks without starvation.

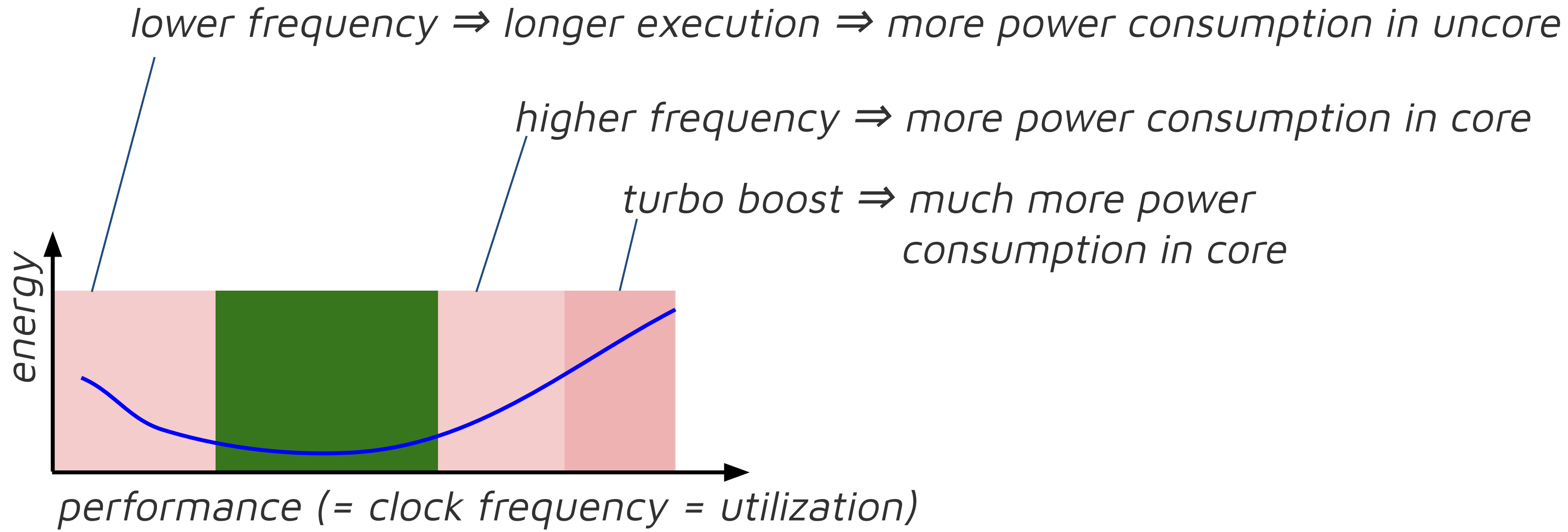
- **Time slice = f(number of runnable tasks)**
  - More tasks ⇒ shorter time slice for each task
  - Less tasks ⇒ longer time slice for each task

All cores are not created equal

- **Processor architecture is heterogeneous.**
  - Intel Alder Lake: P (performance) core vs. E (efficiency) core
  - ARM: big core vs. LITTLE core
- **Even in symmetric multi-processor (SMP), cores are unequal.**
  - Turbo bootsable cores are only a few.
  - SMT hypertwin
- **They have different performance vs. power tradeoffs.**
  - When to use big (or LITTLE) cores?
  - When to use (or not to use) SMT cores?
  - In a lightly-loaded system, should we use all the cores?

Should we use all cores all the time?

- **No, that's because cores have an optimal operating frequency range.**

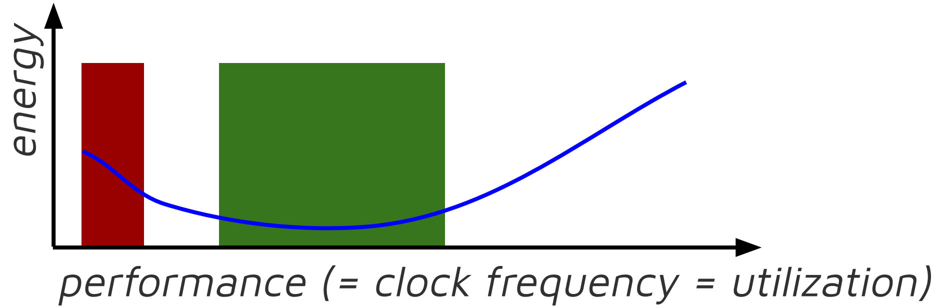


Core compaction

- **Let's run minimal number of core in an optimal frequency range.**
  - Run 16-cores in 5% CPU utilization for each

    vs.

  - ***Run 2-cores in 40% CPU utilization for each***
  - around 50% fits well in Intel & AMD processors



No silver bullet for all usage scenarios

- **When to use big cores? When to use SMT cores?**
  - SMT hypertwin and LITTLE cores are bad for performance but good for energy saving.
  - There is no single correct answer. It mostly depends on user's goal.
    - Music streaming + code editing

      vs.
    - Kernel compilation

Usages can be captured by CPU utilization

- **Light load (around 10%)**
  - Usage: code editing + reading a PDF + music/video streaming
  - LITTLE cores and SMT hypertwins can serve such workloads well, consuming less energy.
- **Medium load (<70%)**
  - Usage:running a casual game, a kernel module compilation
  - Running less performance-critical tasks on LITTLE cores can save energy without hurting the performance.
- **Heavy load (>70%)**
  - Usage: running a AAA game, full kernel compilation
  - Achieving the high performance is the primary goal of scheduling.

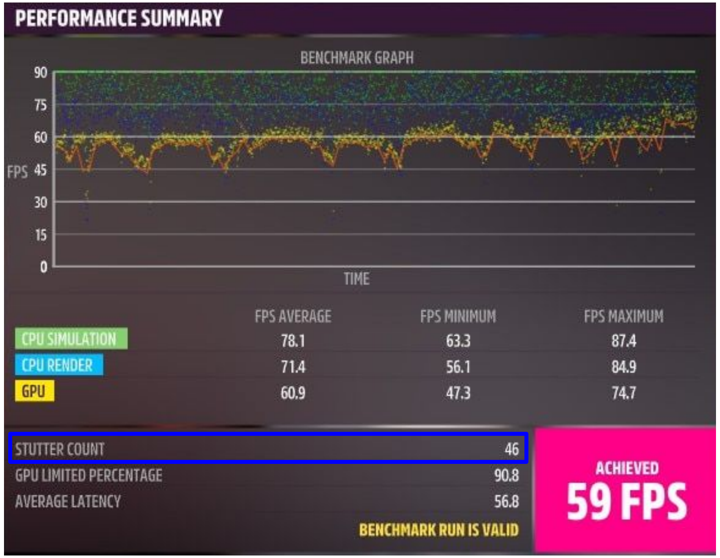
Autopilot mode in LAVD

- **Dynamically adjust core selection policy according to system-wide CPU utilization**
- **Light load ⇒ powersave mode**
  - Utilize LITTLE cores and SMT cores first
    until two such cores can run on their optimal running frequency (around 50%).
- **Medium load ⇒ balanced mode**
  - Allocate the minimal number of cores (core compaction) which servers the load.
  - Classify tasks into performance-critical tasks or not using the pipeline factor.
  - Execute performance-critical tasks on big cores and the others on LITTLE cores.
- **Heavy load ⇒ performance mode**
  - Fully leverage all the cores to race to idle state.

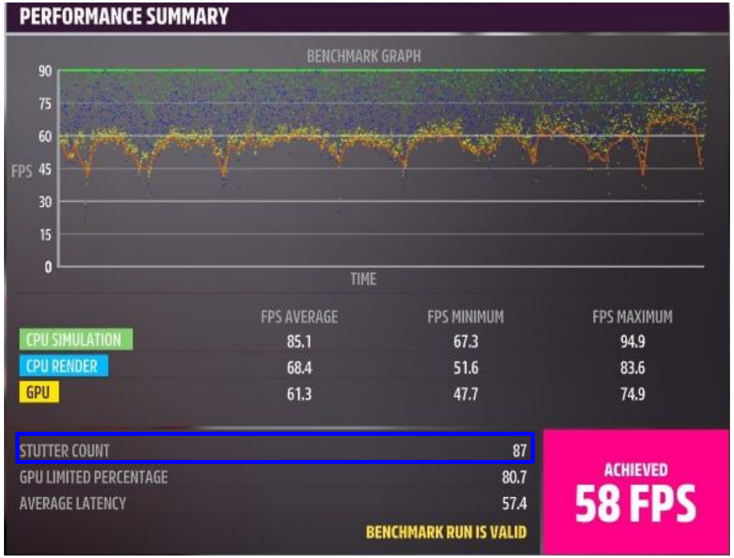
Performance Comparison with EEVDF

- **FH5 in-game benchmark with background recording on SteamDeck OLED**

LAVD	**EEVDF**



1419.49 J (15.4 W)



1423.75 J (15.5 W)

Performance Comparison with EEVDF

- **Tomb Raider in-game benchmark without background task on SteamDeck OLED**

LAVD	EEVDF
752.83 J (10.7 W)	*896.07 J (12.7 W)*

Discussion / Q&A

- What other factors should be considered in making scheduling decisions for latency-critical, interactive workloads, like games?
- Any specialization for Linux graphics stack, WINE, and game engines?
- Any other scheduler features needed?
- Would LAVD be good for (tail) latency-critical server workloads?

Join us!

https://www.igalia.com/jobs

Image credits
https://store.steampowered.com/steamdeck/
https://www.flaticon.com/