The Current Status and Future Direction of the LAVD Scheduler

Changwoo Min
changwoo@igalia.com

December 12, 2025

igalia

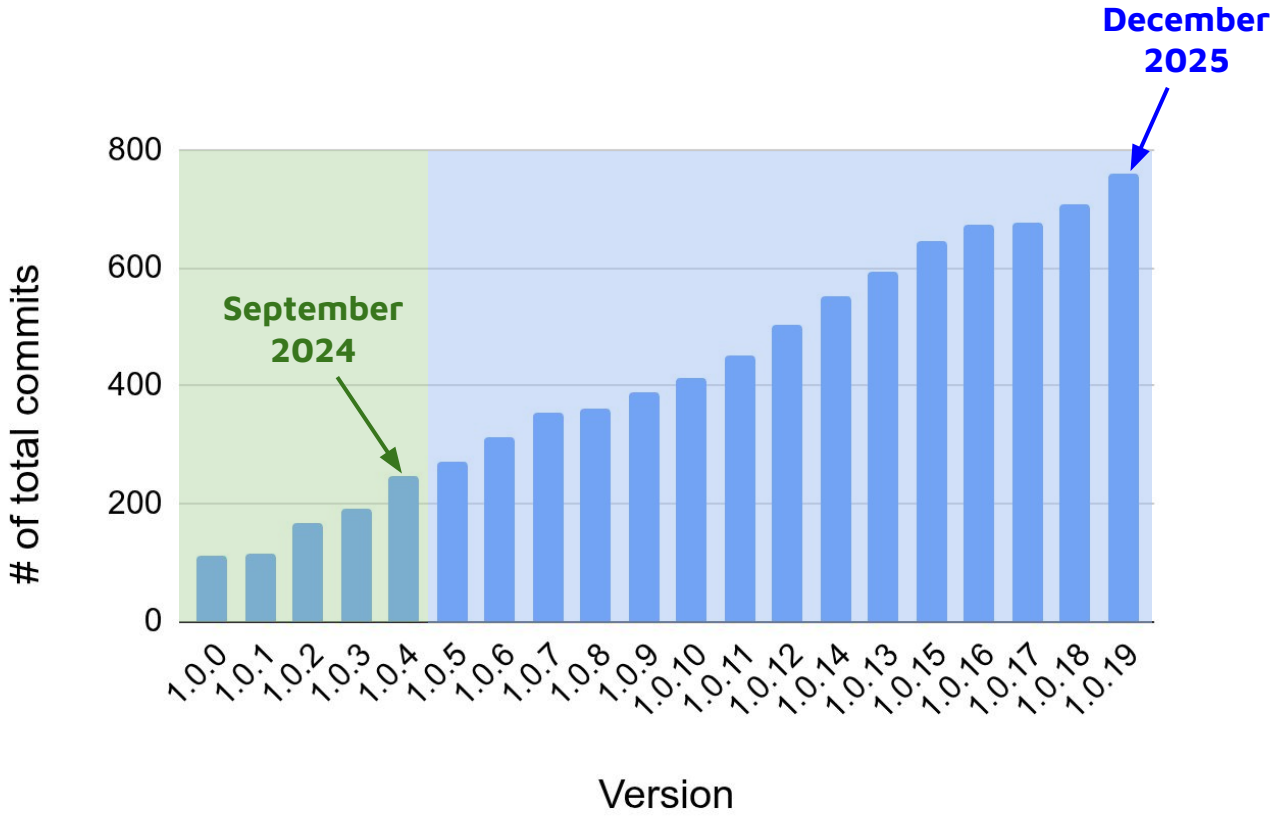
Talk Outline

- **Quick Recap of the LAVD Scheduler**
- **Development Status**
- **Main Focus of Year 2025**
  - Goal: Better support in hybrid processor architectures
  - Solution: Energy model aware scheduling (EMAS)
- **Other (still critical) Improvements**
- **Discussion on EMAS & Future Directions**

Quick Recap of the LAVD Scheduler

- **Aimed for better gaming experience on Linux**
  - Target application: unmodified Windows games running on Linux
  - Target hardware: SteamDeck (x86 AMD CPU)
- **Inspired by the characteristics of gaming workloads**
  - Many tasks interact with each other to complete a single high-level job (e.g., moving a game character upon a key press).
  - Tasks in task graphs communicate heavily (i.e., wake-up) with each other.
- **LAVD: Latency-criticality Aware Virtual Deadline**
  - Virtual deadline based scheduler (similar to EEVDF)
  - Schedule latency-critical task first
    - Latency-criticality of a task = *f(*wake-up/blocking frequency, runtime*)*
  - Core compaction for energy saving
    - Limit the number of cores actively used

Development Status



Hybrid Processor Architectures Are Common

Intel

CPU Specifications

Total Cores	12
# of Performance-cores	2
# of Efficient-cores	8
# of Low Power Efficient-cores	2
Total Threads	14
Max Turbo Frequency	4.9 GHz
Performance-core Max Turbo Frequency	4.9 GHz
Efficient-core Max Turbo Frequency	3.8 GHz
Low Power Efficient-core Max Turbo Frequency	2.1 GHz
Performance-core Base Frequency	1.7 GHz
Efficient-core Base Frequency	1.2 GHz
Low Power Efficient-core Base Frequency	700 MHz

AMD

Series	Ryzen 7000 Series
Form Factor	Laptops, Desktops
AMD PRO Technologies	No
Regional Availability	Global , China , NA , EMEA , APJ , LATAM
Former Codename	Phoenix
Processor Architecture	2x Zen 4 , 4x Zen 4c
# of CPU Cores	6
Multithreading (SMT)	Yes
# of Threads	12
Max. Boost Clock	Up to 4.9 GHz
Max Zen4c Clock	Up to 3.5 GHz
Base Clock	3.2 GHz
Zen4 Base Clock	3.7 GHz
Zen4c Base Clock	3 GHz

Qualcomm

CPU

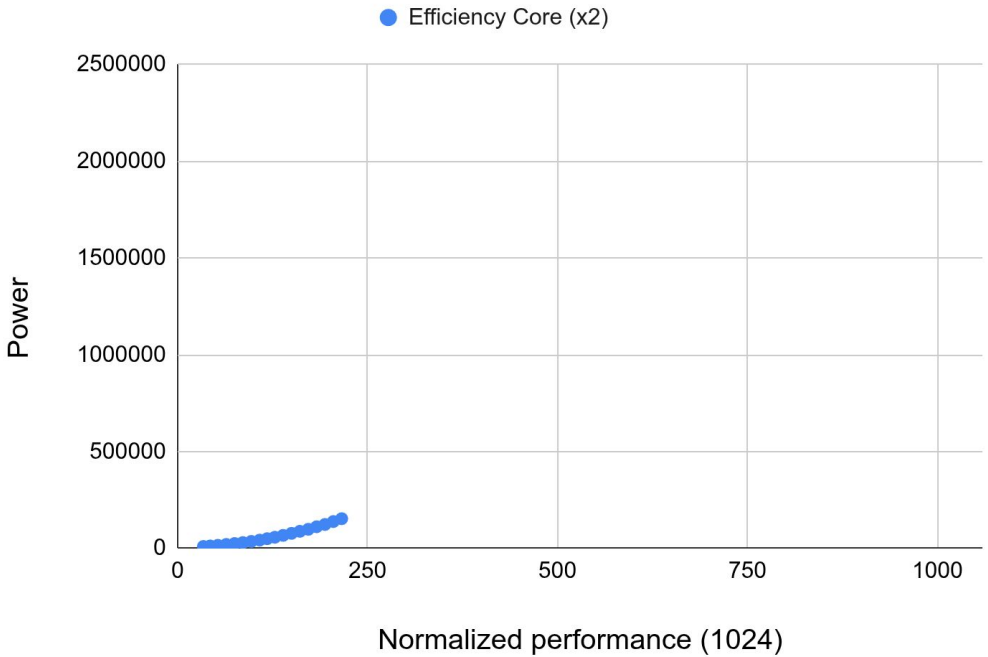
Qualcomm® Kryo™ CPU

- 64-bit Architecture
- 1 Prime core, up to 3.4 GHz[2]
- 5 Performance cores, up to 3.2 GHz
- 2 Efficiency cores, up to 2.3 GHz

*** Captured from manufacturers' websites. ***

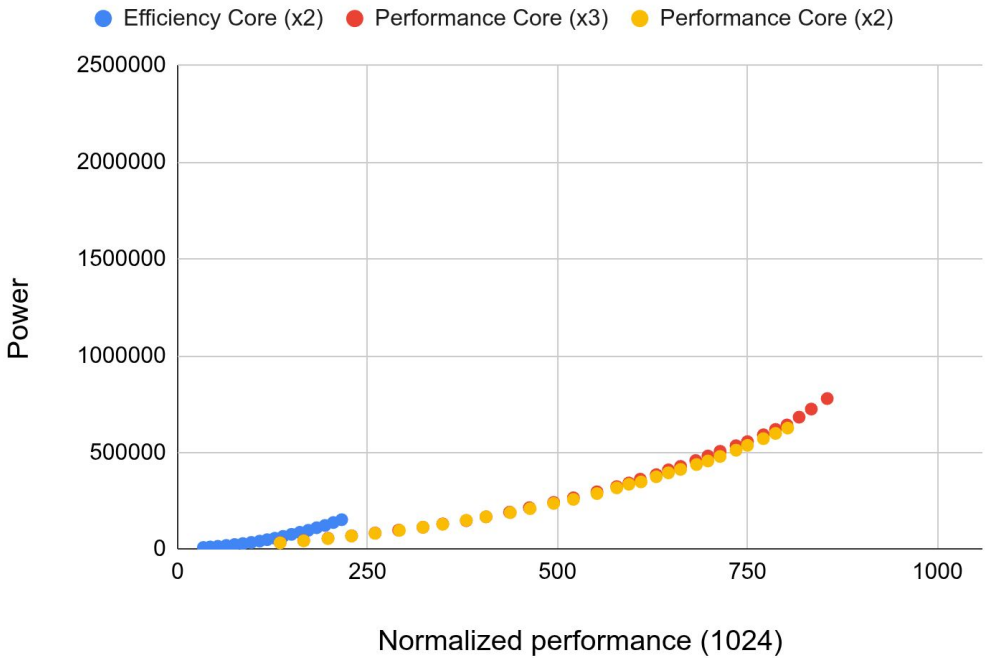
Performance vs. Energy Consumption

- **Different core types have different performance vs. energy consumption tradeoffs.**



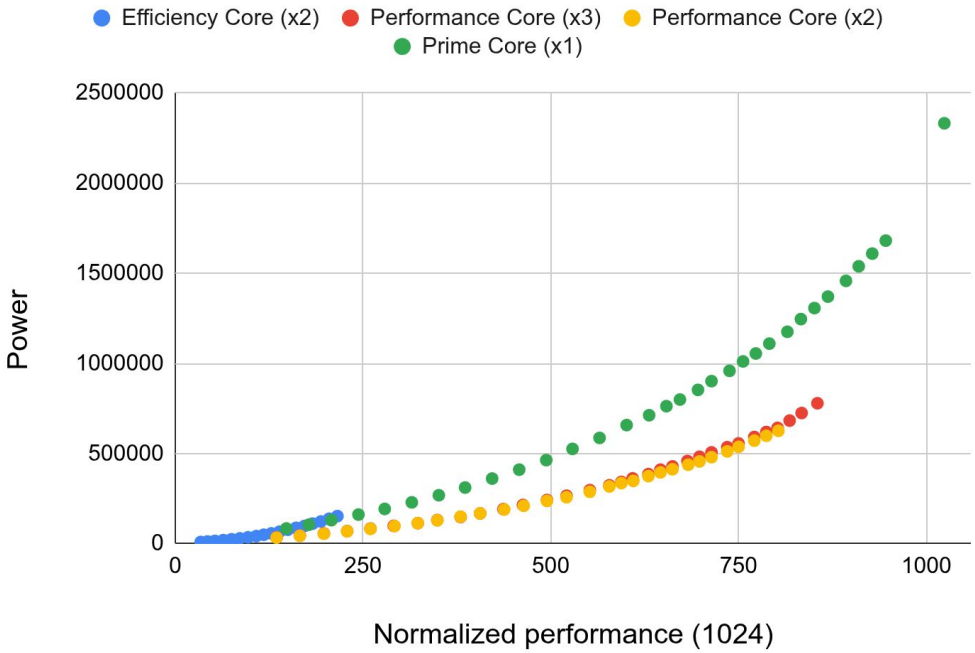
Performance vs. Energy Consumption

- **Different core types have different performance vs. energy consumption tradeoffs.**



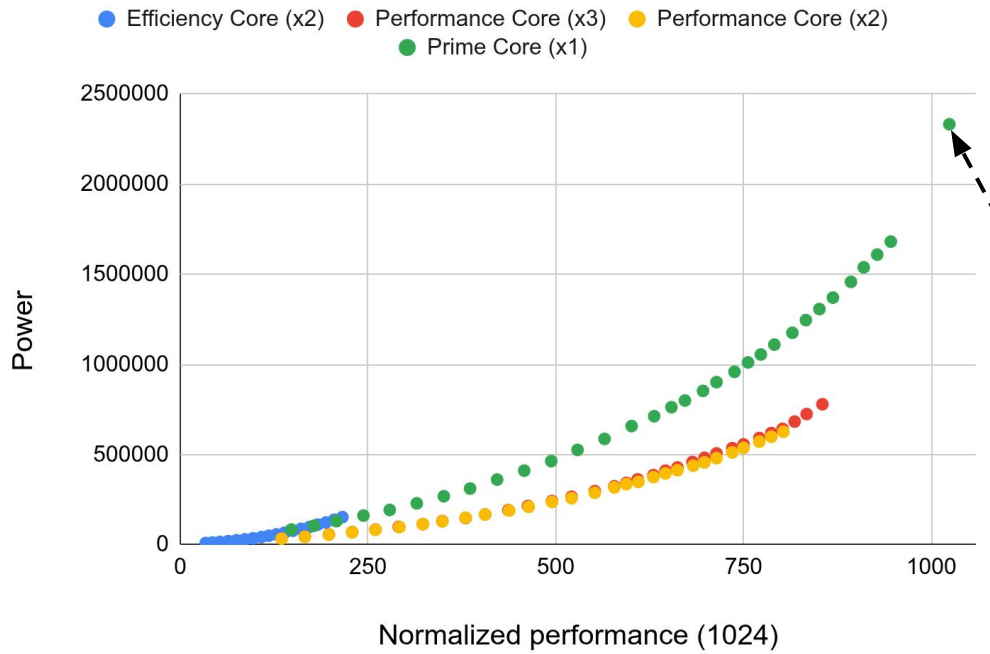
Performance vs. Energy Consumption

- **Different core types have different performance vs. energy consumption tradeoffs.**



Performance vs. Energy Consumption

- **Different core types have different performance vs. energy consumption tradeoffs.**

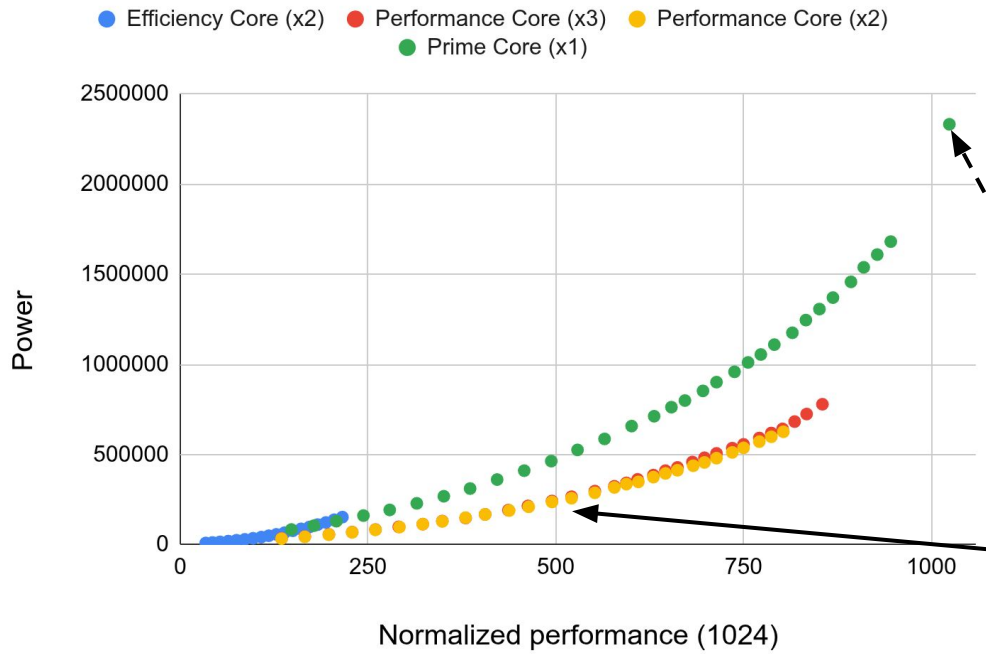


Suppose we need the performance of 1024.
What is the best use of hybrid cores?

Answer 1: Let's use a single prime core!

Performance vs. Energy Consumption

- **Different core types have different performance vs. energy consumption tradeoffs.**



Suppose we need the performance of 1024.
What is the best use of hybrid cores?

Answer 1: Let's use a single prime core!

Answer 2: Let's use two performance cores!

Energy Model Aware Scheduling (EMAS)

- **Key idea**
  - Given the required computing power, suggest a list of CPUs to meet the computing demand with minimum energy consumption.
- **For example,**
  - Q: I need computing power of 1024.
  - A: Use CPU 2 and 3 (performance core).
- **Key challenges**
  - How to get an energy model?
  - How to derive the recommended CPUs for a required computing demand?
  - How to assign a task to a proper core type?

Getting an energy model?

- **The kernel already provides the energy model:**

```
/sys/kernel/debug/energy_model
├── cpu0              # Efficiency core (x2)
│   ├── ps:1017600
│   ├── ...
│   └── ps:902400
├── cpu2              # Performance core (x3)
│   ├── ps:1075200
│   ├── ...
│   └── ps:960000
├── cpu5              # Performance core (x2)
│   ├── ps:1075200
│   ├── ...
│   └── ps:960000
└── cpu7              # Prime core (x1)
    ├── ps:1017600
    ├── ...
    └── ps:902400
```

- **See the details at:**
  - /scx/rust/scx_utils/src/energy_model.rs
- **In the future, the netlink interface will be used:**
  - PM: EM: Add netlink support for the energy model

Deriving suggested CPUs for a demand

- **This is a combinatorial optimization problem:**
  - For all possible sets of {CPU, utilization} pairs,
    - Computing power and energy consumption are determined.
  - For each computing power generated,
    - Choose the set of {CPU, utilization} pairs with minimum energy consumption.
- **Energy Model Optimizer (EMO) constructs this when loading LAVD.**
  - Heuristics based optimization for efficiency.
  - See the details at:
    - /scx/scheds/rust/scx_lavd/src/cpu_order.rs

Example output of EMO

CPU utilization range	Capacity range	CPUs to be used
~ 5.3%	~ 300	0, 1
~ 20.2%	~ 1138	2, 3, 0, 1
~ 60.1%	~ 3386	2, 3, 4, 5, 6
~ 70.6%	~ 3977	2, 3, 4, 5, 6, 0
~ 80.1%	~ 4508	2, 3, 4, 5, 6, 0, 1
~ 100%	~ 5627	7, 2, 3, 4, 5, 6, 0, 1

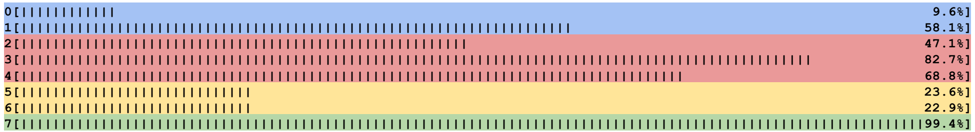
Efficiency core (x2)
Performance core (x3)
Performance core (x2)
Prime core (x1)

Assigning a task to a proper core type

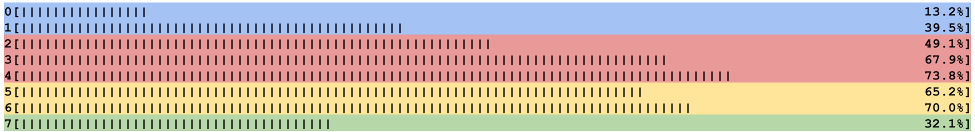
- **First, measure computing demand of a task**
  - At the high-level, same as PELT (Per-Entity Load Tracking) in kernel
  - Computing demand = *f(*task runtime, CPU capacity, CPU frequency*)*
- **Determine if a task is performance-critical or not**
  - Performance criticality = *f(*task's computing demand, wake-up/blocking frequency*)*
- **Top N% tasks in performance criticality are performance-critical tasks**
  - where N% is computing capacity that big cores provides.
- **When selecting a CPU of a task at ops.enqueue() and ops.select_cpu(),**
  - Try to match task's type (performance-critical or not) and CPU's type (big vs. little)
- **There are lots of details:**
  - /scx/scheds/rust/scx_lavd/src/bpf/{main.bpf.c, lat_cri.bpf.c, power.bpf.c, idle.bpf.c}

Experimental Results

- **On a Snapdragon development board.**
- **EEVDF**
  - FPS: around 50
  - CPU utilization



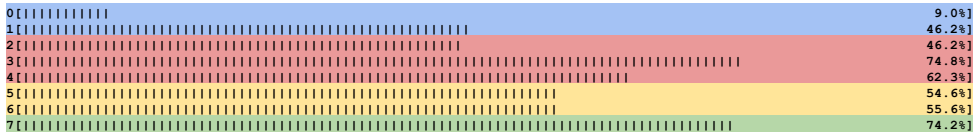
- **LAVD**
  - FPS: around 57, energy-rate: smaller than EEVDF
  - CPU utilization



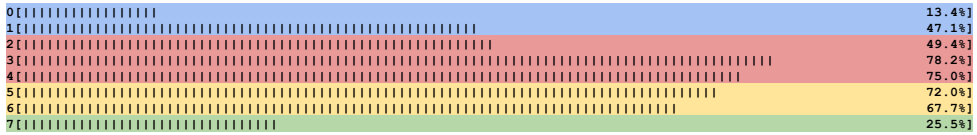
Efficiency core (x2)
Performance core (x3)
Performance core (x2)
Prime core (x1)

Experimental Results

- **On a Snapdragon development board.**
- **EEVDF**
  - FPS: around 50
  - CPU utilization



- **LAVD**
  - FPS: around 50, energy-rate: smaller than EEVDF
  - CPU utilization



Efficiency core (x2)
Performance core (x3)
Performance core (x2)
Prime core (x1)

Many, many bug fixes and optimizations

- Hardening virtual deadline algorithm for extremely overloaded cases.
- Tracing futex to mitigate the lock holder preemption problem.
- Inheriting latency criticality of a waker to be more contextual.
- Better handling of pinned/affinitized tasks
- Improving time slice calculation
- And, many more

Discussion on EMAS

- **Would it be worth to be a common library for other SCX schedulers?**

- **Would more fine-grained core classification be necessary?**
  - Currently, we classify core types into two: big or LITTLE cores.
  - Many hybrid processors provide big, **medium**, and LITTLE cores.

- **Would more sophisticated energy model be necessary?**
  - The actual energy model would be more complicated than what kernel provides.
  - E.g., Wattson: trace based power/energy estimation, LPC 2024

Discussion on Future Directions

- **Better support on Windows games**
  - Tracing Windows locks (NTsync) to mitigate lock holder preemption problem
    - Current futex tracing for lock holder preemption is imprecise
  - Or extend proxy execution for futex and NTsync?
- **Extend the coverage of LAVD to large servers**
  - 8 cores ⇒ 100+ cores
  - A single LLC domain ⇒ 10+ LLC domain
  - Cgroup support for resource isolation (e.g., containers) ⇒ cpu.max support (v2)
- **Anything else?**

Join us!

https://www.igalia.com/jobs

Image credits

https://store.steampowered.com/app/253230/A_Hat_in_Time/
https://store.steampowered.com/app/814380/Sekiro_Shadows_Die_Twice__GOTY_Edition/