



CPU Power Regression Testing with Wattson

William McVicker (Google)



LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026

Why Power Testing is so difficult?



Hardware Requirements

- Requires expensive custom measurement equipment with device breakout boards per product.
- Limited scalability to the lab
- Delicate setup prone to environmental issues
- Significant lab maintenance.



Reproducibility

- Wide variance across devices due to manufacturing differences.
- Run-to-run variance due to ambient temperature.
- Calibration errors in measurement equipment



Attribution

- Granularity limited to measurement equipment capabilities
 - Best case: per-power rail
 - Typically battery level granularity
- Not very actionable due to limited isolation



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Wattson

Trace-based power estimation tool

LPC'24: Wattson Intro Video:

<https://www.youtube.com/watch?v=EWGUqCAbfhM>

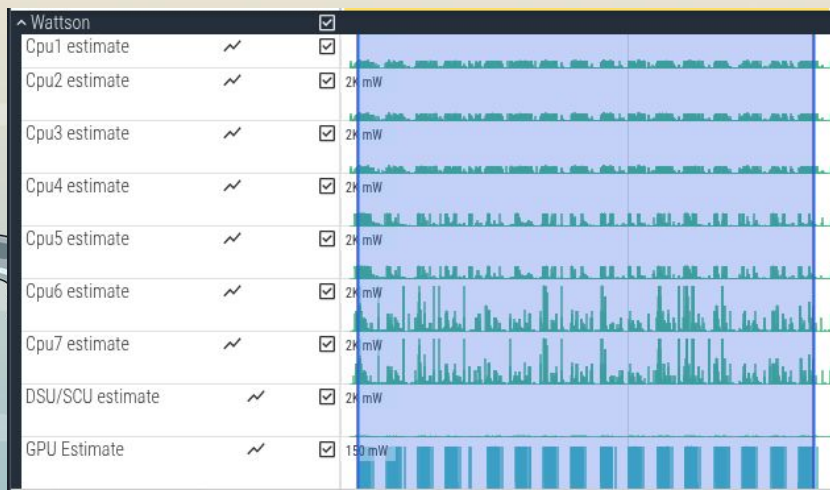
Slides:

<https://lpc.events/event/18/contributions/1842/attachments/1476/3126/LPC%202024%20-%20Wattson.pdf>

No External Hardware Required: Eliminates the need for expensive, clunky breakout boards like Monsoon or custom measurement tools by using ftrace events to estimate power consumption.

High Fidelity & Repeatability: Correlates within 1% of absolute ground-truth measurements on average, representing a significant improvement over on-device physical meters (ODPM) by removing environmental noise (ambient temperature, unit-to-unit hardware variances).

Thread-Level Attribution: Attributes CPU time and energy consumption to individual threads/processes.



Current Selection			
Area Selection Counters Slices CPU by thread CPU by process Wattson estimates Wattson by			
μW mW			
Name ↑	Power (estimated mW)	Energy (estimated mWs)	
	Σ	Σ	
CPU0	86.631	1371.548	
CPU1	64.075	1014.447	
CPU2	66.982	1060.469	
CPU3	64.51	1021.328	
CPU4	62.494	989.406	
CPU5	55.13	872.828	
CPU6	77.263	1223.229	
CPU7	83.948	1329.072	
Dsu_Scu	19.017	301.076	
Gpu	5.156	81.626	

What's New

- GPU and TPU subsystems now supported
- Non-zero energy when all CPUs are in deep idle
- Lots more testing in Android
 - System Health testing
 - System UI testing
 - App launch testing
 - GKI (kernel) testing
 - Google 1P app testing
- General tooling improvements
 - Data collection and calibration improvements
 - Trace collection improvements
 - Detection on Linux distros via DT base compatible string



LINUX

PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Supported Devices

Manufacturer	SoC	Example devices	Upstream support
Google	Tensor G1	Pixel 6, 6 Pro, and 6a	✓
	Tensor G4	Pixel 9, 9 Pro, 9 Pro XL, 9 Pro Fold, 9a, 10a	
	Tensor G5	Pixel 10, 10 Pro, 10 Pro XL, and 10 Pro Fold	
	Tensor G6	Pixel 11, 11 Pro, 11 Pro XL, and 11 Pro Fold	
Qualcomm	SW5100	Pixel Watch 2, 3, and 4	✓
	SXR2230P	Galaxy XR	
	SM8750	Samsung S25, Xiaomi 15 Pro	
MediaTek	MT6897	Transsion Infinix Note 50 Pro+ 5G	
	MT6789	Infinix HOT 70	
	MT6858	Tecno Pova Curve 2 5G	

Catching a Mainline PELT Power Regression

Wattson caught a +30% average CPU power regression in 6.18-rc7 with a simple 10-second BouncyBall animation workload [1] correlating with:

- Increased residency at higher CPU frequencies
- A significant increase in sugov invocations (at least 10x)

	with patch	without patch
sugov invocation rate (Hz)	133.5	3.7

Freq (kHz)	with patch: time spent (ms)	without patch: time spent (ms)
738000	4869	9869
1803000	2936	68
1598000	1072	0
1704000	674	0
...	...	...

LINUX

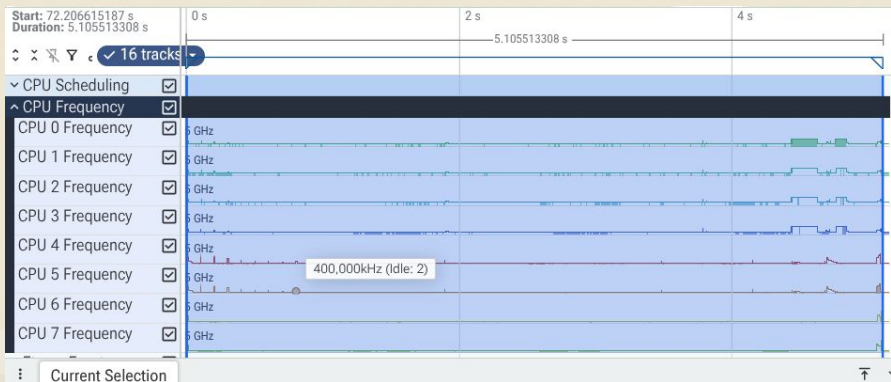
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026

[1] https://lore.kernel.org/all/CAG2KctpO6VKS6GN4QWDji0t92_gNBj7HjjXrE+6H+RwRXt=iLg@mail.gmail.com/

A/B Comparison of Traces

Baseline Build



Area Selection CPU Slices Wattson by thread Wattson by process Export

μ W mW

Process Name	PID	Active power (estimated mW) SUM	Active energy (e... SUM	Idle transi...
	0	32.816	167.54	
/system/bin/surfaceflinger	533	19.885	101.523	
com.prefabulated.touchlatency	6533	18.481	94.356	
system_server	1524	16.093	82.164	
/vendor/bin/hw/android.hardware....	536	8.136	41.536	
	null	3.423	17.478	
/system/bin/hw/android.hardware....	1000	3.165	16.150	

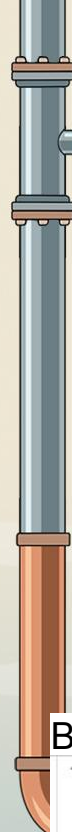
Regressed Build



Area Selection CPU Slices Wattson by thread Wattson by process Export

μ W mW

Process Name	PID	Active power (estimated mW) SUM	Active energy (e... SUM	Idle transitions o...
	0	38.699	197.511	
/system/bin/surfaceflinger	531	29.608	151.115	
com.prefabulated.touchlatency	6684	17.593	89.792	
/vendor/bin/hw/android.hardware.gr...	534	13.265	67.702	
system_server	1607	7.877	40.205	



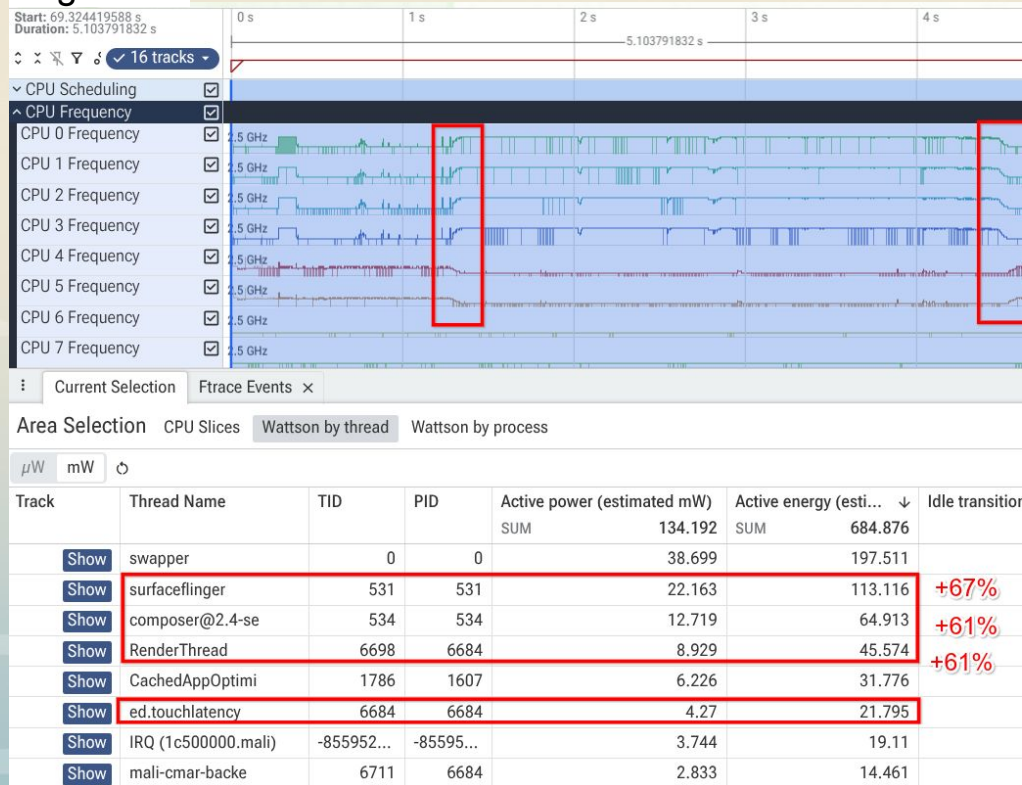
Actionable Findings

- **Thread-level attribution**
 - Rendering threads significantly increased power consumption.
- **Subsystem attribution**
 - Isolated regression to CPU rails
- **High signal-to-noise ratio**
 - Reverting offending change allows the CI testing to recover back to baseline to address the regression.

Baseline

Thread Name	Active power...		Active energy ... ↓	
	SUM	115.927	SUM	591.891
swapper		32.816		167.54
CachedAppOptimi		15.051		76.843
surfaceflinger		13.288		67.84
RenderThread		12.586		64.258
composer@2.4-se		7.879		40.228

Regressed



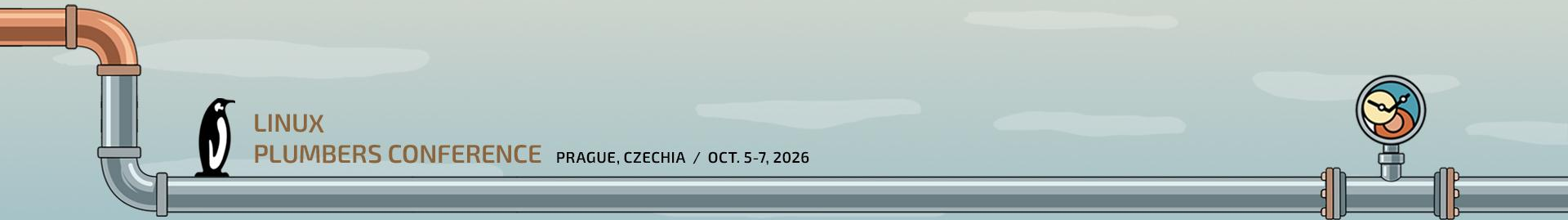
The Fix

Clue from Wattson: Attribution pinpointed increase in CPU power consumption from render threads (surfaceflinger / composer), followed by idle transitions.

Root Cause: When transitioning from RT/DL to idle, `rq_clock_pe1t` was synced for idle entry before prev RT/DL classes finished updating PELT utilization—artificially inflating utilization and causing sugov to spike CPU frequency.

The Fix [1]: Update prev scheduling classes (RT/DL) with the scaled `rq_clock_pe1t` before syncing the clock for idle entry.

[1] <https://lore.kernel.org/all/20260121163317.505635-1-vincent.guittot@linaro.org/>



Use Wattson upstream *today* with Qualcomm SM8750 and/or Pixel 6 to test [your favorite subsystem]!

Collect a trace:

```
$ tracebox --txt -c wattson.cfg --background-wait -o trace.pb
$ echo "I|0|wattson_start" > /sys/kernel/tracing/trace_marker
  <do work here>
$ echo "I|0|wattson_stop" > /sys/kernel/tracing/trace_marker
$ kill -TERM ${TRACEBOX_PID}
```

Upload trace.pb to <http://ui.perfetto.dev> to analyze metrics or extract the metrics on the command line:

```
$ trace_processor_shell --run-metric \
  wattson_markers_rails,wattson_markers_threads trace.pb
```

Guide at <https://source.android.com/docs/core/power/wattson/wattson>



LINUX

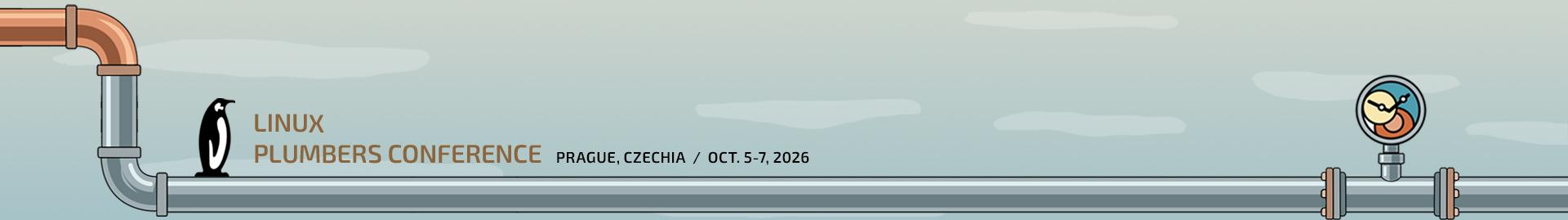
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



What's Next

- Catch power regressions in real-time with **On-device Wattson**
- Expanding into newer subsystems, e.g. DDR and display
- Expanding GPU and NPU power models across more SoCs



Discussion Points

1. Is this something you're interested in?
 - a. What devices would you like to see supported?
2. What testing workflows would you like to see Wattson added to?
 - a. Testing scheduler changes in linux-next to prevent CPU regressions?
 - b. Report on changes in CPU power consumption per-Linux release?

Contact us at wattson-external@google.com

Guide at <https://source.android.com/docs/core/power/wattson/wattson>



LINUX

PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026

