Wattson

Saravana Kannan (Google)
Samuel Wu (Google)
wattson-external@google.com

LINUX
PLUMBERS
CONFERENCE Vienna, Austria / Sept. 18-20, 2024

Status quo: Performance evaluation

Hardware setup is easy:

- Just the development device
- Can scale to a test lab

Easy to measure:

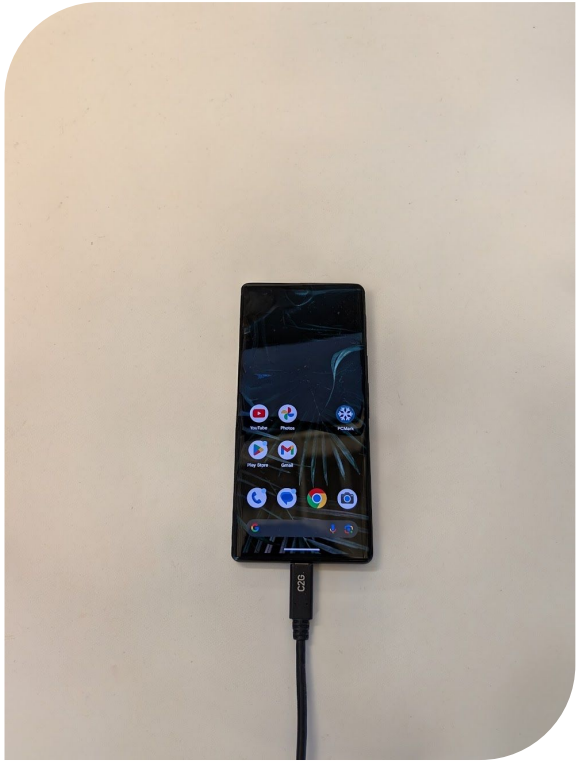
- 100s of benchmarks to choose from

Fairly Repeatable:

- Main challenge is thermal throttling

Attribution & measurement granularity:

- Tracing makes it easy to attribute



Status quo: Power evaluation

Hardware setup is NOT easy:

- Custom solution per board
- Costs $ - $$$
- Can be finicky to set up
- Scaling to a lab has a lot of maintenance/overhead

Hard to measure:

- No common benchmark or tool
- Remote setups might not be possible/hard

Not very repeatable:

- Manufacturing differences/binning
- Measurement hardware calibration errors
- Thermal impact even without throttling

Attribution & measurement granularity:

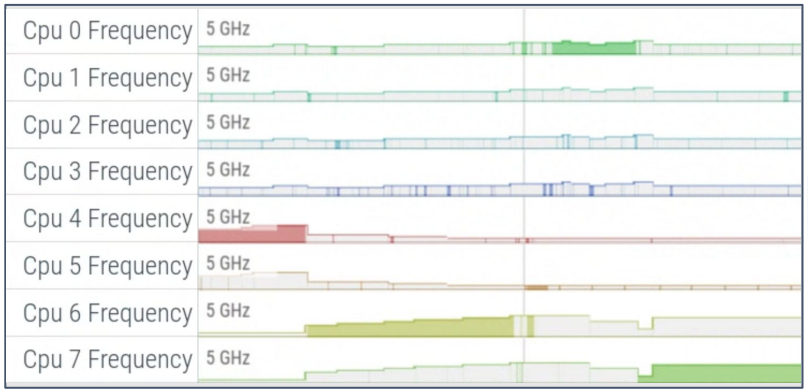
- At best, per power rail attribution over the entire test
- At worst, only at battery level over the entire test

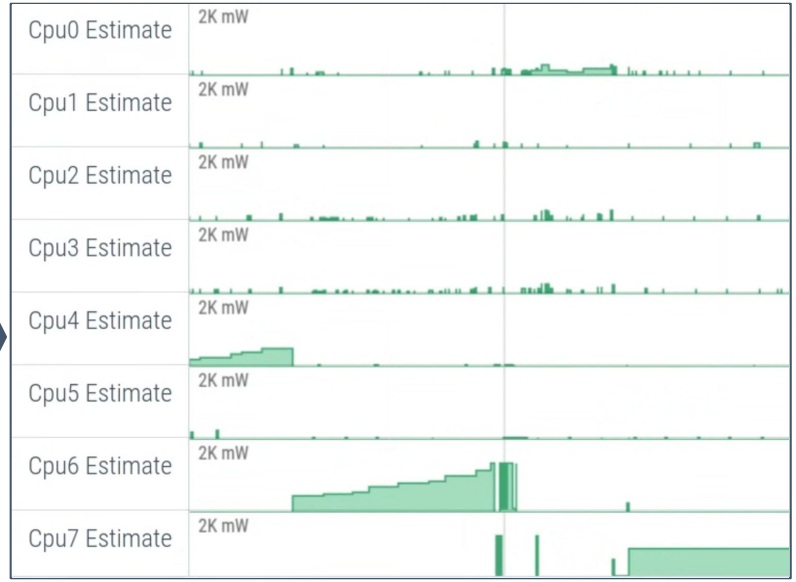
- Leverages kernel's ftrace
- Keeps it low overhead

Wattson: A trace based CPU power evaluation tool

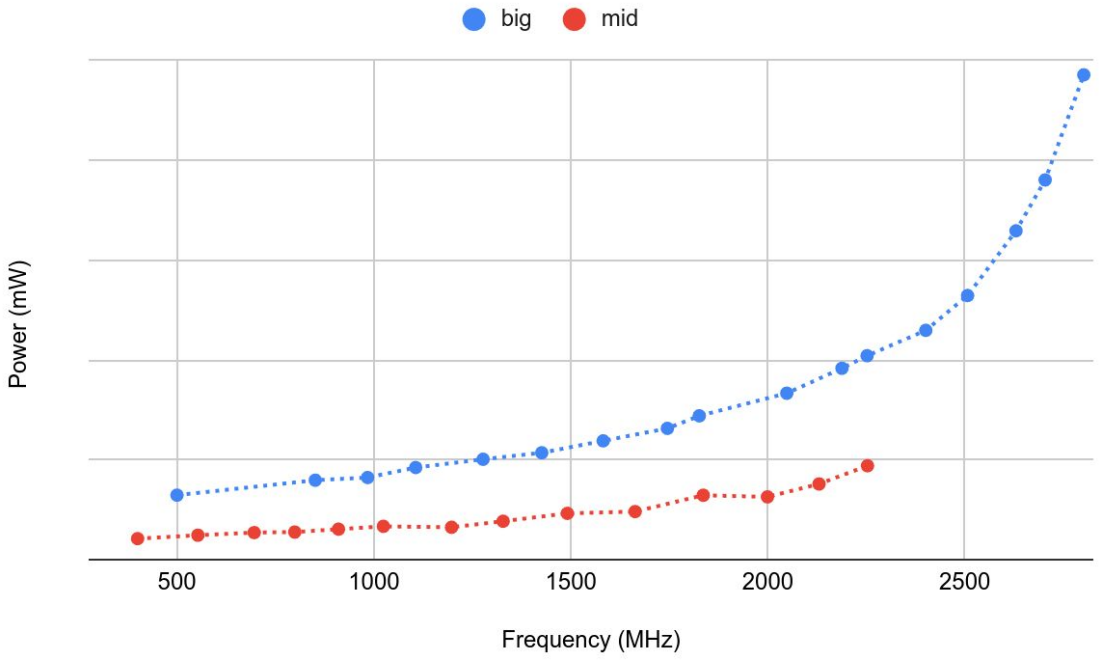
- % change in power
- e.g. A/B testing

Power estimation components

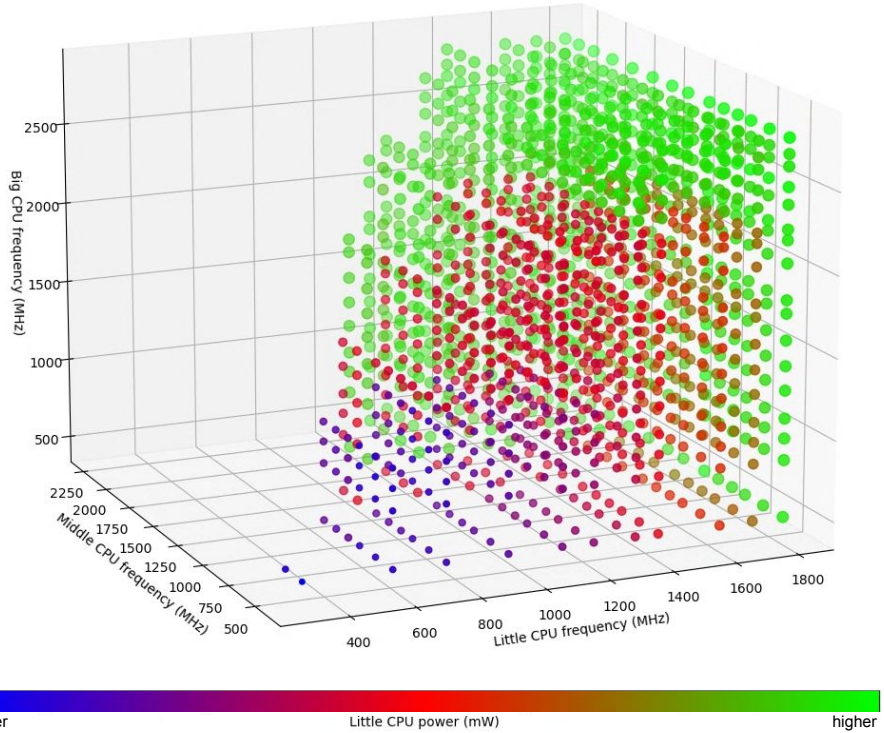




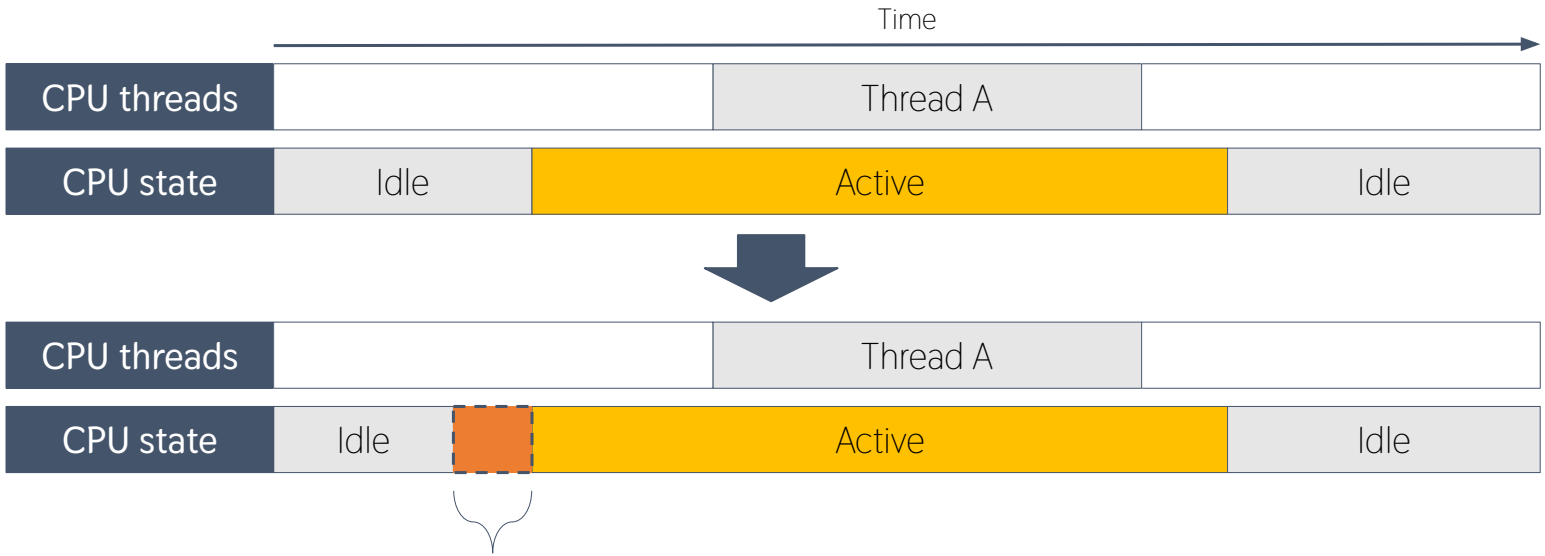
CPU power curves



Little CPU's power volume

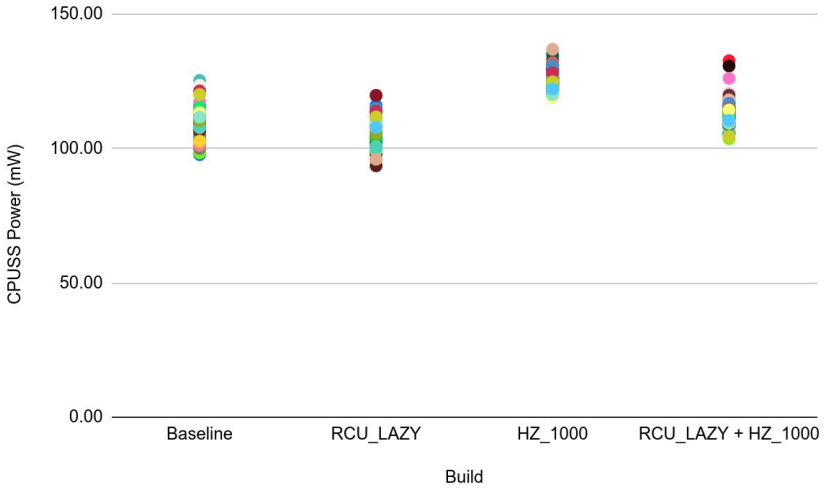


Kernel vs hardware idle exit times



Additional idle exit time added by Wattson

Estimated CPUSS power over 100 BouncyBall runs



Build	Baseline	RCU_LAZY	HZ_1000	RCU_LAZY + HZ_1000
Average	109 mW	102 mW	125 mW	113 mW

+12.4%

actual change in power

+13.4%

estimated change in power
for: **HZ_1000**

-6.8%

actual change in power

-6.8%

estimated change in power
for: **RCU_LAZY**

+4.0%

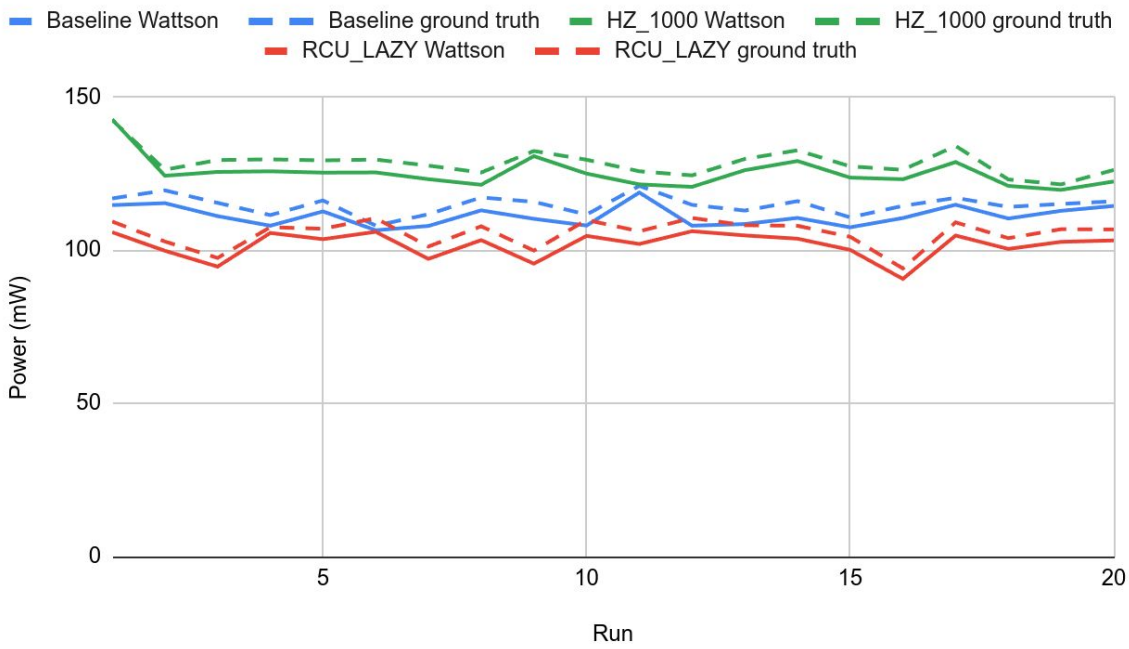
actual change in power

+3.3%

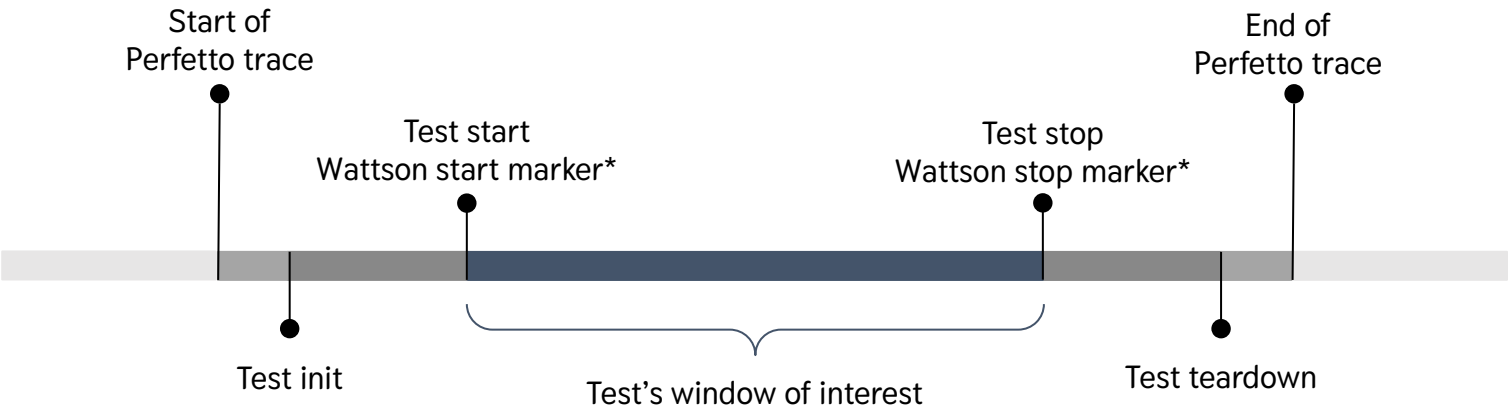
estimated change in power
for: **RCU_LAZY + HZ_1000**

*n=20

Wattson vs ground truth: BouncyBall 10s run



Collecting a Perfetto trace for Wattson



Inserting markers:

```
echo 'I|0|wattson_start' >/sys/kernel/tracing/trace_marker
echo 'I|0|wattson_stop' >/sys/kernel/tracing/trace_marker
```

Using Wattson

Get Wattson estimates

- GUI

  https://ui.perfetto.dev/

- Command line

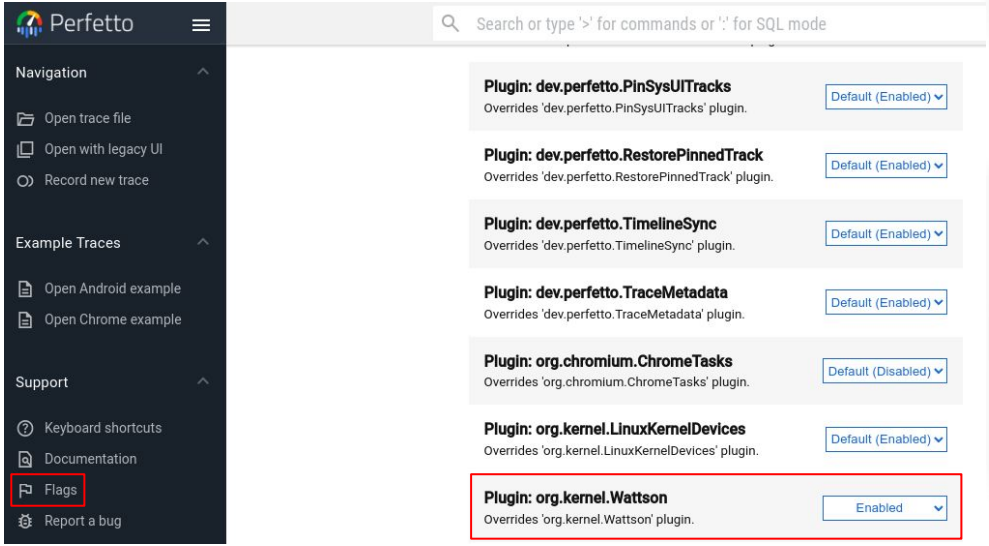
```
# For getting thread level power attribution in JSON format
trace_processor --run-metrics wattson_markers_threads output_trace.pb

# For getting CPUSS estimates in JSON format
trace_processor --run-metrics wattson_markers_rails output_trace.pb
```

Enable Wattson in the perfetto UI

Enable Wattson in UI

a. Navigate to https://ui.perfetto.dev/
b. Click on **Flags** tab
c. **Enable** the Wattson plugin



Power estimate tracks

"Wattson tracks" in the Perfetto UI displays power per CPU over time

RCU_LAZY: shows idle power improvements

Baseline build

Area Selection CPU by thread CPU by process Wattson by thread Wattson by process Wattson by package

Thread Name	TID	PID	Average power (estimated mW)	Total energy (estimated mWs)	Idle transitions overhead (estimated mWs)
			83.39	841.36	74.54
swapper	0	0	28.02	282.39	0
surfaceflinger	497	497	12.12	122.12	3.74
RenderThread	9724	9712	9.77	98.5	3.46
binder:507_2	507	507	6.89	69.43	0.04
ed.touchlatency	9712	9712	3.77	37.95	2.54
mali-cmar-backe	9735	9712	1.83	18.41	1.54
binder:9712_1	9720	9712	1.59	16.03	1.31

RCU_LAZY build

Area Selection CPU by thread CPU by process Wattson by thread Wattson by process Wattson by package

Thread Name	TID	PID	Average power (estimated mW)	Total energy (estimated mWs)	Idle transitions overhead (estimated mWs)
			72.28	730.74	59.64
swapper	0	0	23.56	238.13	0
RenderThread	6644	6631	9.87	99.7	3.24
surfaceflinger	483	483	8.65	87.41	2.1
binder:492_2	492	492	5.26	53.14	0.07
ed.touchlatency	6631	6631	3.78	38.23	2.76
mali-cmar-backe	6655	6631	1.84	18.58	1.8
traced_probes	985	985	1.31	13.25	0.15

RCU_LAZY: Even validate through rcu_preempt

Baseline build

Area Selection CPU by thread CPU by process Wattson by thread Wattson by process Wattson by package

Thread Name	TID	PID	Average power (estimated mW)	Total energy (estimated mWs)	Idle transitions overhead (estimated mWs)
			83.39	841.36	74.54
sugov:6	286	286	0.12	1.2	8.27
rcu_preempt	16	16	0.28	2.79	6.65
simpleinteracti	173	173	0.16	1.57	4.69
surfaceflinger	497	497	12.12	122.12	3.74
app	583	497	1.12	11.31	3.57
RenderThread	9724	9712	9.77	98.5	3.46

RCU_LAZY build

Area Selection CPU by thread CPU by process Wattson by thread Wattson by process Wattson by package

Thread Name	TID	PID	Average power (estimated mW)	Total energy (estimated mWs)	Idle transitions overhead (estimated mWs)
			72.28	730.74	59.64
kworker/1:53	1423	1423	0.03	0.3	0.19
rcu_preempt	16	16	0.01	0.07	0.17
mali-mem-purge	548	483	0.03	0.31	0.17
traced_probes	985	985	1.31	13.25	0.15
kworker/4:1	118	118	0.02	0.17	0.12
kworker/5:2	740	740	0.02	0.2	0.12

Easy to identify bad runs

Good run: 1117 mWs

Area Selection CPU by thread CPU by process Wattson by thread Wattson by process Wattson by package

Process Name	PID	Average power (estimated mW)	Total energy (estimated mWs)	Idle transitions overhead (estimated mWs)
		111.67	1117	127.53
NULL	0	39.88	398.84	0
com.prefabulated.touchlatency	9005	20.68	206.78	20.49
/system/bin/surfaceflinger	492	18	179.99	35.41
system_server	1238	6.72	67.22	0.44
/vendor/bin/hw/android.hardware.graphics.composer@2.4-service	494	6.37	63.73	3.53
decon0_kthread	260	2.37	23.75	2.62

Bad run: 2908 mWs

Area Selection CPU by thread CPU by process Wattson by thread Wattson by process Wattson by package

Process Name	PID	Average power (estimated mW)	Total energy (estimated mWs)	Idle transitions overhead (estimated mWs)
		290.88	2908.39	208.16
NULL	0	64.71	647.05	0
system_server	1060	43.81	438.06	3.57
com.google.android.apps.nexuslauncher	2270	32.81	328.04	10.83
kworker/u32:3	84	31.3	312.97	3.53
zygote64	9391	27.93	279.32	13.64
/system/bin/surfaceflinger	490	22.41	224.07	30.71
/apex/com.android.adbd/bin/adbd	3502	9.66	96.57	8.54

Command line: Wattson metrics

Summarizes energy/power estimates for period of interest in the trace:

- Per thread estimate
- Per (virtual) power rail estimate


```
"wattson_trace_threads": {
  "metric_version": 2,
  "task_info": [
    {
      "estimated_mws": 437.572479,
      "estimated_mw": 39.725395,
      "thread_name": "swapper",
      "thread_id": 0,
      "process_id": 0
    },
    {
      "estimated_mws": 213.184097,
      "estimated_mw": 19.354101,
      "thread_name": "commands.monkey",
      "process_name": "app_process",
      "thread_id": 4174,
      "process_id": 4174
    },
    {
      "estimated_mws": 196.183197,
      "estimated_mw": 17.810659,
      "thread_name": "Shutdown thread",
      "process_name": "app_process",
      "thread_id": 4232,
      "process_id": 4232
    },
    {
      "estimated_mws": 167.009415,
      "estimated_mw": 15.162094,
      "thread_name": "surfaceflinger",
      "process_name": "/system/bin/surfaceflinger",
      "thread_id": 534,
      "process_id": 534
    },
```

```
"wattson_trace_rails": {
  "metric_version": 3,
  "period_info": [
    {
      "period_id": 1,
      "period_dur": 11014931067,
      "cpu_subsystem": {
        "estimated_mw": 274.486938,
        "policy0": {
          "estimated_mw": 53.386322,
          "cpu0": {
            "estimated_mw": 18.123945
          },
          "cpu1": {
            "estimated_mw": 11.202857
          },
          "cpu2": {
            "estimated_mw": 11.298664
          },
          "cpu3": {
            "estimated_mw": 12.760857
          }
        },
        "policy4": {
          "estimated_mw": 33.656109,
          "cpu4": {
            "estimated_mw": 16.248100
          },
          "cpu5": {
            "estimated_mw": 17.408009
          }
        },
        "policy6": {
          "estimated_mw": 135.341263,
```

How to use Wattson for kernel development?

Hardware support:

- Pixel 6
- Open to adding your SoC of choice if you provide the power curves

Suggested workloads:

- Android
  - BouncyBall has been a good analog for Android apps
- Linux
  - Pick a workload that's more real world and has some CPU idle time
  - Don't use any benchmark that maxes out CPU frequency
  - Pick one that's repeatable in the amount of work that's done

Getting started on Perfetto:

Excellent Quickstart guide by John:
https://gist.github.com/johnstultz-work/0ec4974e0929c4707bfd89c876ae4735

Perfetto for device:

- Android: Comes preinstalled.
- Or for a static Linux binary (called tracebox):
  - `curl -O https://raw.githubusercontent.com/google/perfetto/main/tools/tracebox`

Trace processor for your PC:

- Android: Built as part of any android build
- Can also download from: https://github.com/google/perfetto/releases
- Or for a static Linux binary:
  - `curl -O https://raw.githubusercontent.com/google/perfetto/main/tools/trace_processor`
  - `chmod +x trace_processor`

Collect Perfetto trace for Wattson: Android

```
// Collect Perfetto trace on device
perfetto --txt -c min_wattson.cfg -o output_trace.pb

// Insert Wattson markers around the period of interest
echo 'I|0|wattson_start' >/sys/kernel/tracing/trace_marker
<use case runs>
echo 'I|0|wattson_stop' >/sys/kernel/tracing/trace_marker

// Flush Perfetto trace to buffer
killall -w perfetto
```

Collect Perfetto trace for Wattson: Linux

```
// Collect Perfetto trace on device
tracebox --txt -c min_wattson.cfg -o output_trace.pb

// Insert Wattson markers around the period of interest
echo 'I|0|wattson_start' >/sys/kernel/tracing/trace_marker
<use case runs>
echo 'I|0|wattson_stop' >/sys/kernel/tracing/trace_marker

// Flush Perfetto trace to buffer
killall -w tracebox
```

Wattson via cmdline

```
// Post process Perfetto trace via cmdline (or upload trace to https://ui.perfetto.dev/ for GUI)
curl -O https://raw.githubusercontent.com/google/perfetto/main/tools/trace_processor
chmod +x trace_processor

// For getting thread level power attribution
trace_processor --run-metrics wattson_markers_threads output_trace.pb

// For getting CPUSS estimates
trace_processor --run-metrics wattson_markers_rails output_trace.pb
```

min_wattson.cfg - Minimum config for Wattson

```
write_into_file: true
flush_period_ms: 30000
file_write_period_ms: 30000
buffers: {
    size_kb: 200000
    fill_policy: DISCARD
}
buffers: {
    size_kb: 2048
    fill_policy: DISCARD
}
data_sources: {
    config {
        name: "linux.process_stats"
        target_buffer: 1
        process_stats_config {
            scan_all_processes_on_start: true
        }
    }
}
data_sources: {
    config {
        name: "linux.ftrace"
        ftrace_config {
            ftrace_events: "ftrace/print"
            ftrace_events: "power/cpu_frequency"
            ftrace_events: "power/cpu_idle"
            ftrace_events: "power/suspend_resume"
        }
    }
}
```

Discussion

Do you find Wattson useful?

What will encourage you to integrate Wattson into your development workflow?

Can we start using Wattson to check power impact of major sched/DVFS changes?

What additional capabilities would you like to see added to Wattson?

Thank you!

Contact:
wattson-external@google.com