WALT vs PELT : Redux

Pavankumar Kondeti
Engineer, Staff
Qualcomm Technologies, Inc
09/27/2017

pkondeti@qti.qualcomm.com

Agenda

- Why we need load tracking?
- A quick recap of PELT and WALT
- Results

Why we need load tracking?

- Load balancer/group scheduling
- Task placement
  - Task load and CPU utilization tracking is essential for energy aware scheduling.
  - Heavy tasks can be placed on higher capacity CPUs. Small tasks can be packed on a busy CPU without waking up an idle CPU.
- Frequency guidance
  - CPU frequency governors like ondemand and interactive uses a timer and compute the CPU busy time by subtracting the CPU idle time from the timer period.
  - Task migrations are not accounted. If a task execution time is split across two CPUs, governor fails to ramp up the frequency.

PELT recap

- The PELT (Per Entity Load Tracking) is introduced in 3.8 kernel. The load is tracked at the sched entity and cfs_rq/rq level
- The load is accounted using a decayed geometric series with runnable time in 1 msec period as coefficients. The decay constant is chosen such that the contribution in the 32 msec past is weighted half as strongly as the current contribution.
- The blocked tasks also contribute to the rq load/utilization. The task load is decayed during sleep.
- Currently the load tracking is done only for the CFS class. There are patches available to extend it to RT/DL class.

PELT signals

- se->avg.load_avg - The load average contribution of an entity. The entity weight and frequency scaling are factored into the runnable time.
- cfs_rq->runnable_load_avg - Sum of the load average contributions from all runnable tasks. Used in load balancer.
- cfs_rq->avg.load_avg - Represents the load average contributions from all runnable and blocked entities. Used in CFS group scheduling
- se->avg.util_avg - The utilization of an entity. The CPU efficiency and frequency scaling are factored into the running time. Used in EAS task placement
- cfs_rq->avg.util_avg - Represents the utilization from all runnable and blocked entities. Input to the schedutil governor for frequency selection.

WALT Introduction

- WALT keeps track of recent N windows of execution for every task. Windows where a task had no activity are ignored and not recorded.
- Task demand is derived from these N samples. Different policies like max(), avg(), max(recent, avg) are available.
- The wait time of a task on the CPU rq is also accounted towards its demand.
- Task utilization is tracked separately from the demand. The utilization of a task is the execution time in the recently completed window.
- The CPU rq's utilization is the sum of the utilization of all tasks ran in the recently completed window.

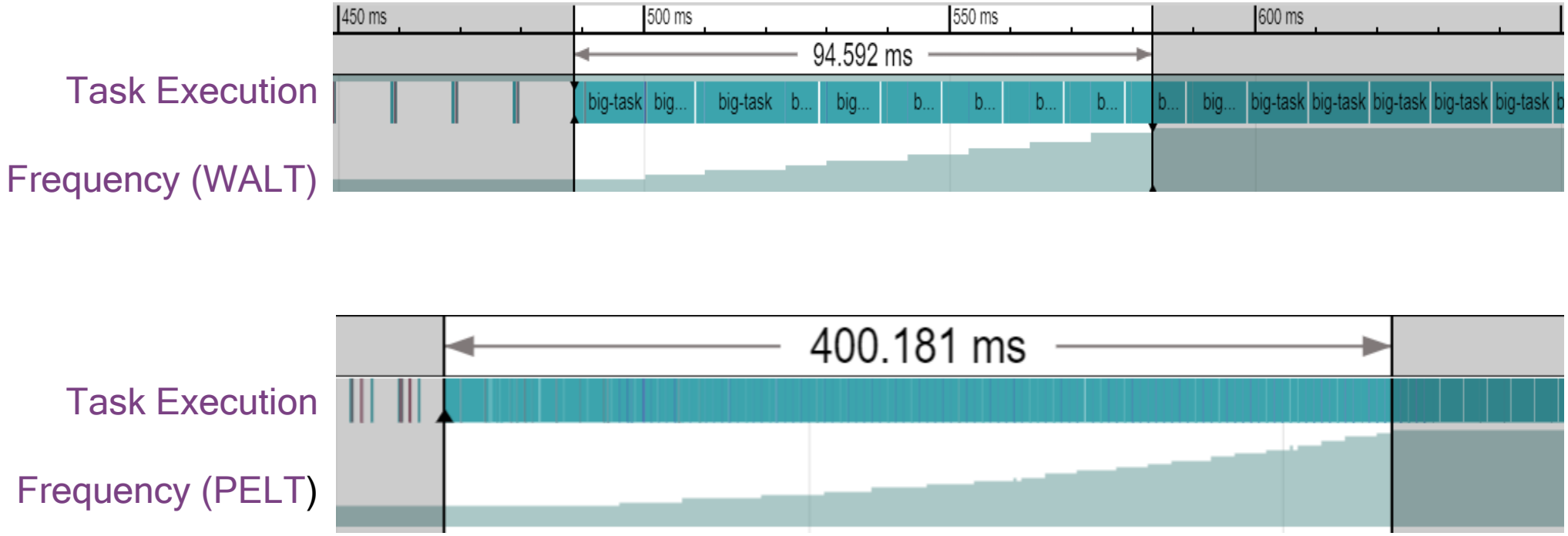
WALT signals

- p->ravg.demand - Task demand. Used in the EAS task placement.
- rq->cumulative_runnable_avg - Sum of the demand of all runnable tasks on this CPU. Represent the Instantaneous load. Used in the EAS task placement
- rq->prev_runnable_sum - CPU utilization in the recent complete window. Input to the schedutil.
- rq->cum_window_demand - Sum of the demand of tasks ran in the current window. This signal is used to estimate the frequency. Used in the EAS task placement for evaluating the energy difference

PELT vs WALT

- WALT has more dynamic demand/utilization signal than provided by PELT, which is subjected to the geometric series.
- PELT takes more time to detect a heavy task or re-classification of a light task as heavy. Thus the task migration to a higher capacity CPU is delayed. As example, it takes ~138msec for a task with 0 utilization to become a 95% utilization task.
- PELT decays the utilization of a task when it goes to sleep. As an example, a 100% utilization task would become 10% utilization task just after 100 msec sleep.
- Similar to task utilization, PELT takes more time to build up the CPU utilization, thus delaying the frequency ramp up.
- PELT's blocked utilization decay implies that the underlying cpufreq governor would have to delay dropping frequency for longer than necessary

Frequency ramp up

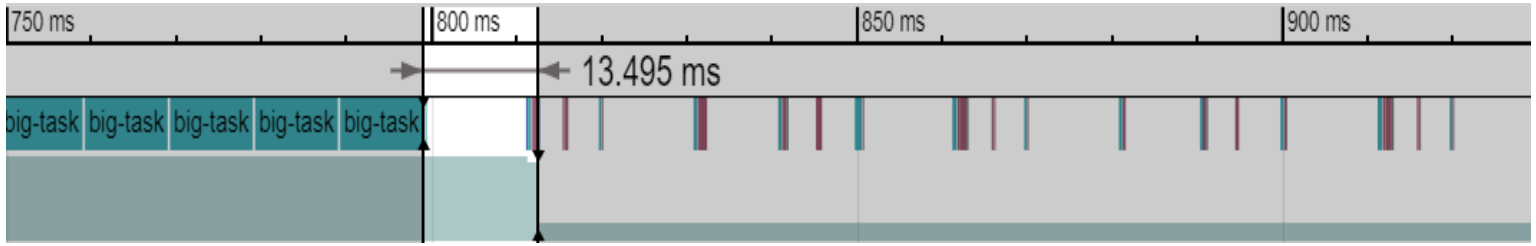


- The frequency ramp up is 4 times faster on WALT
- The CPU utilization is limited by the current capacity. If the frequency is not ramped up fast, the utilization accumulation is delayed which in turn delays ramping up to the Fmax.

Frequency ramp down

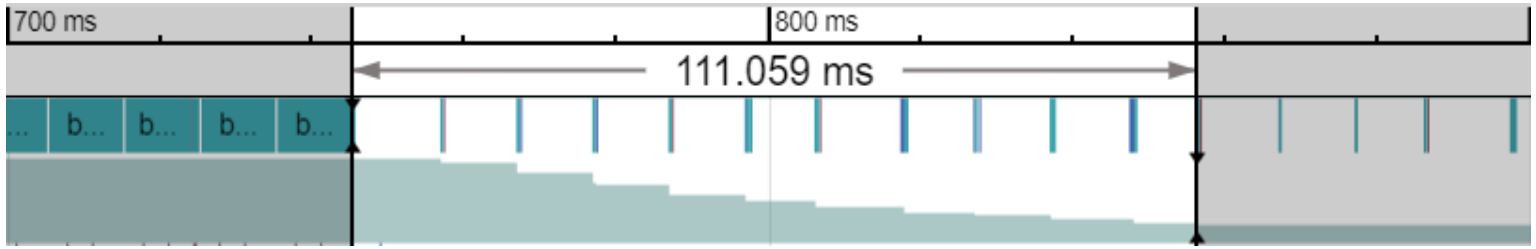
Task Execution

Frequency (WALT)



Task Execution

Frequency (PELT)



- The frequency ramp down is 8 times faster on WALT
- The blocked tasks also contribute to the CPU utilization in PELT. This delays frequency ramp down. It hurts power but may boost the performance in some use cases.

Results

- Benchmarks
- JankBench
- Ux (Twitter, Facebook, Youtube)
- Games

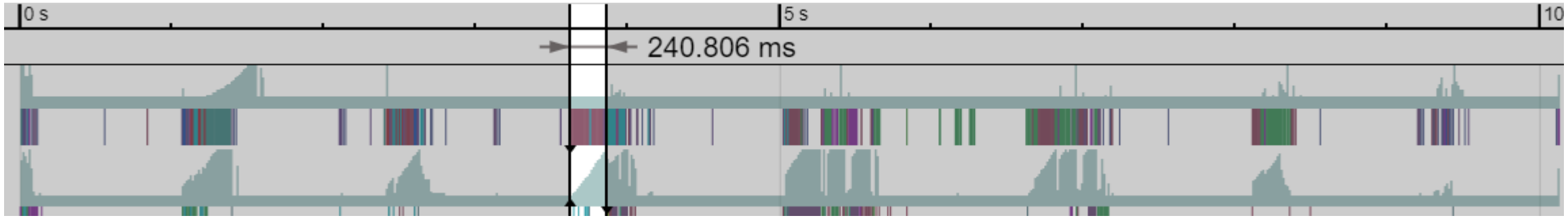
Testing platform

- The tests are run on the SDM835 platform
- The EAS + schedutil is based on the AOSP common kernel. Few additional patches which are available in AOSP gerrit review are included.
- HZ = 300 and WALT window size is 10 msec
- The schedtune boost for top-app is 10% by default. It is boosted to 50% upon touch event for 3 seconds.

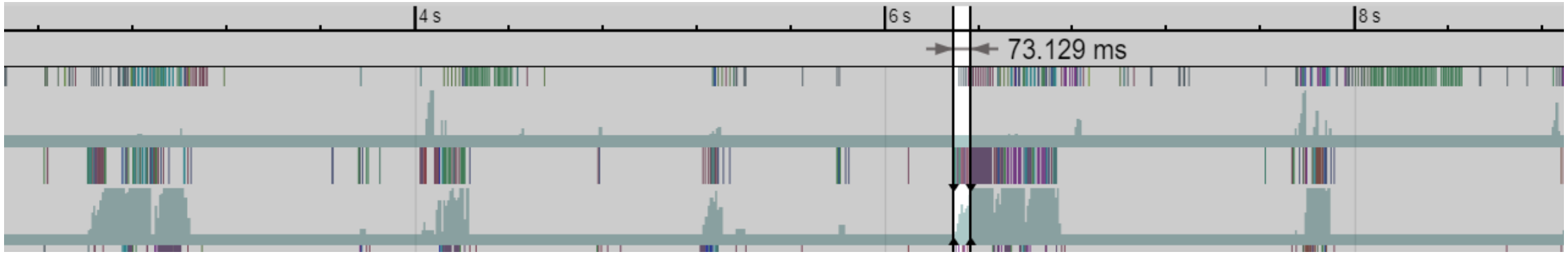
Benchmark		Result
GeekBench (v4)	ST	same
	MT	same
PCMark (v1.2)	Total	WALT is 23% better
	Web browsing	WALT is 38% better
	Video Playback	PELT is 4% better
	Writing	WALT is 33% better
	Photo Editing	WALT is 32% better
Antutu (v6)		same
AndroBench (v5)	Sequential Read	same
	Sequential Write	Same
	Random Read	WALT is 51% better
	Random Write	WALT is 11% better

WALT accounts task wait time towards demand. This results in task spreading which improved performance. With schedtune.prefer_idle = 1, the both PELT and WALT are giving same result.

Frequency ramp up in PCMark



PELT

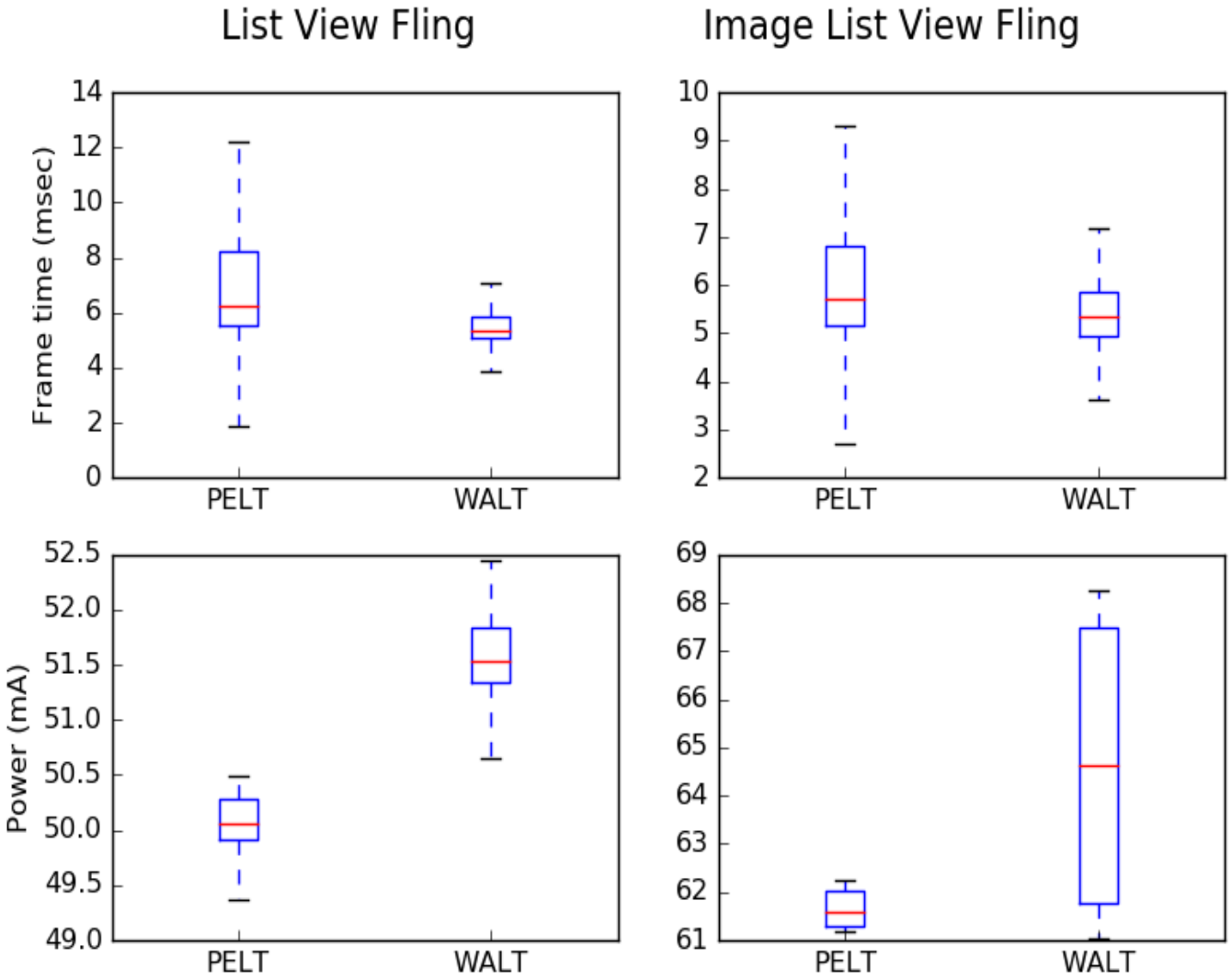


WALT

- The faster frequency ramp up is helping WALT in PCMark
- Better results are observed with Patrick's util_est series for PELT. The gap is reduced to 10% from 23%

List View Fling	PELT (msec)	WALT (msec)
25%	5.541621	5.046939
50%	6.235489	5.364740
75%	8.211237	5.858221
90%	9.225127	6.928594
99%	11.848338	10.306270

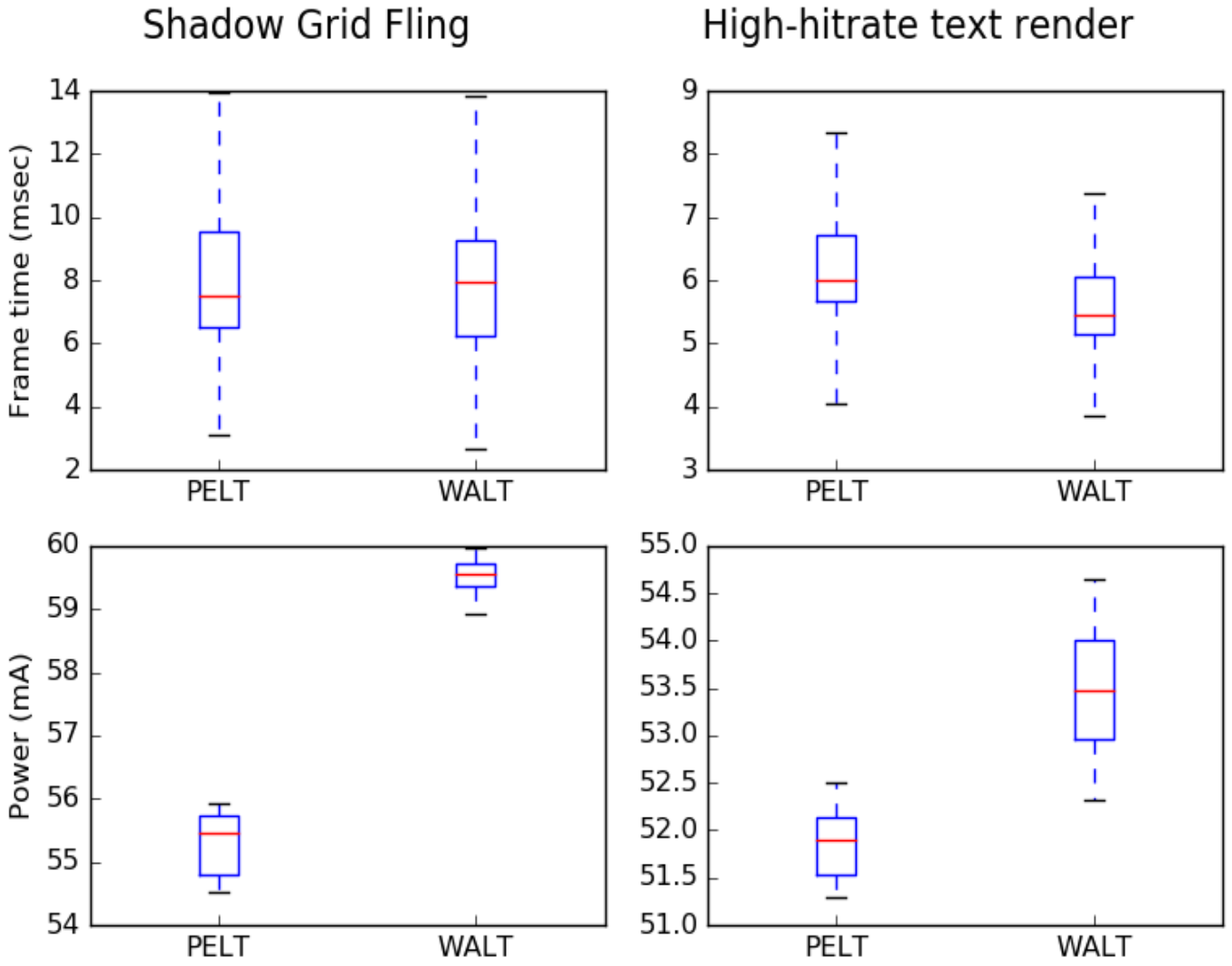
Image List View Fling	PELT (msec)	WALT (msec)
25%	5.170975	4.950513
50%	5.715942	5.348545
75%	6.817395	5.850904
90%	8.662801	6.625814
99%	11.443108	9.915054



Test	Power
List View Fling	PELT is 2% better
Image List View Fling	PELT is 4% better

Shadow Grid Fling	PELT (msec)	WALT (msec)
25%	6.500720	6.219871
50%	7.525394	7.932580
75%	9.514976	9.256707
90%	10.416724	10.443205
99%	12.531525	13.334770

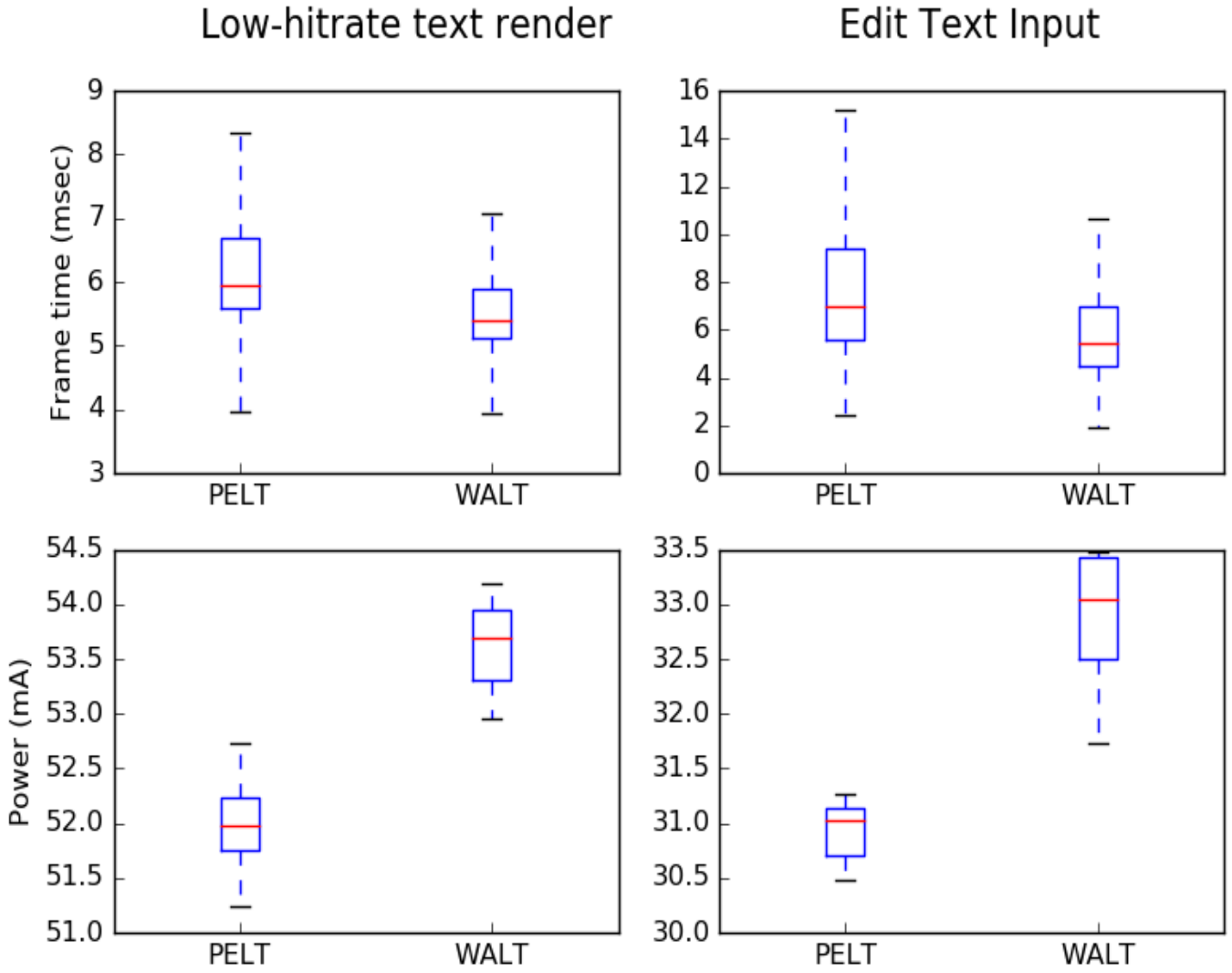
High hitrate text render	PELT (msec)	WALT (msec)
25%	5.663946	5.161756
50%	6.003140	5.444418
75%	6.726914	6.053425
90%	8.065195	7.008964
99%	12.561744	11.277686



Test	Power
Shadow Grid Fling	PELT is 7% better
High hitrate text render	PELT is 3% better

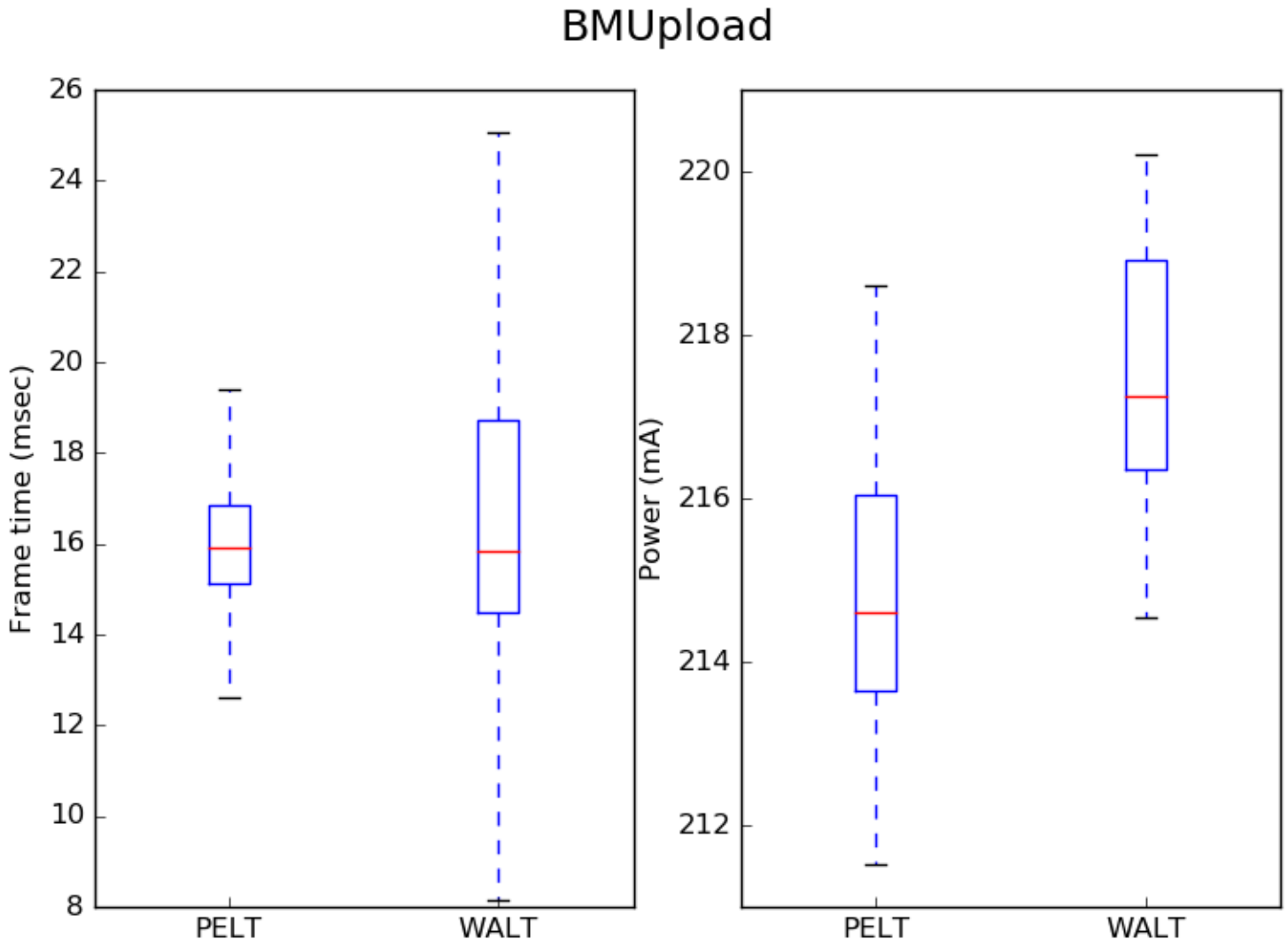
Low hitrate text render	PELT (msec)	WALT (msec)
25%	5.596932	5.106558
50%	5.945192	5.388022
75%	6.695878	5.898033
90%	8.143912	6.888017
99%	12.639909	10.600410

Edit text input	PELT (msec)	WALT (msec)
25%	5.566935	4.477202
50%	6.977525	5.445546
75%	9.407826	6.955690
90%	14.892524	9.764885
99%	21.150100	16.515769

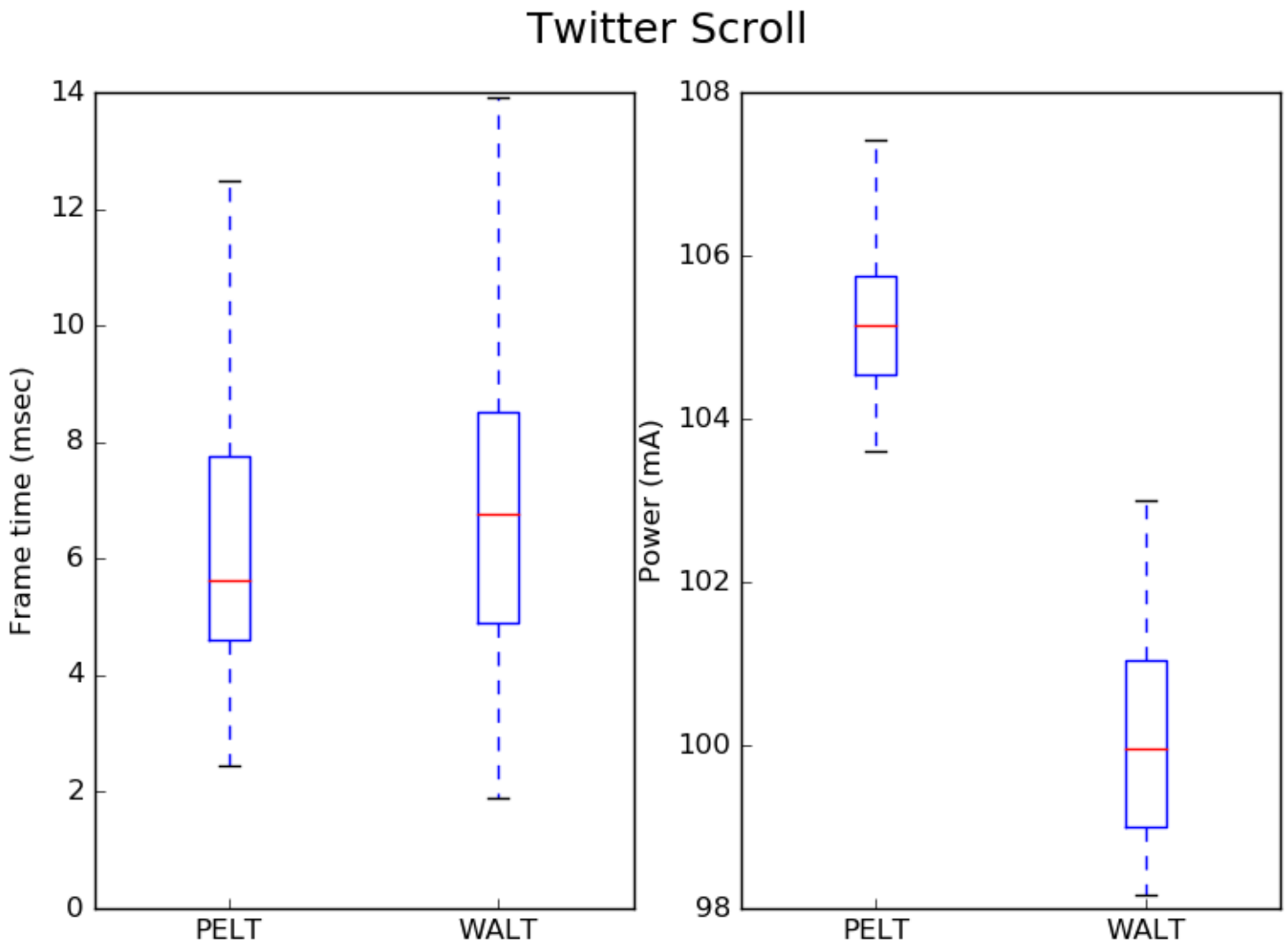


Test	Power
Low hitrate text render	PELT is 3% better
Edit text input	PELT is 6% better

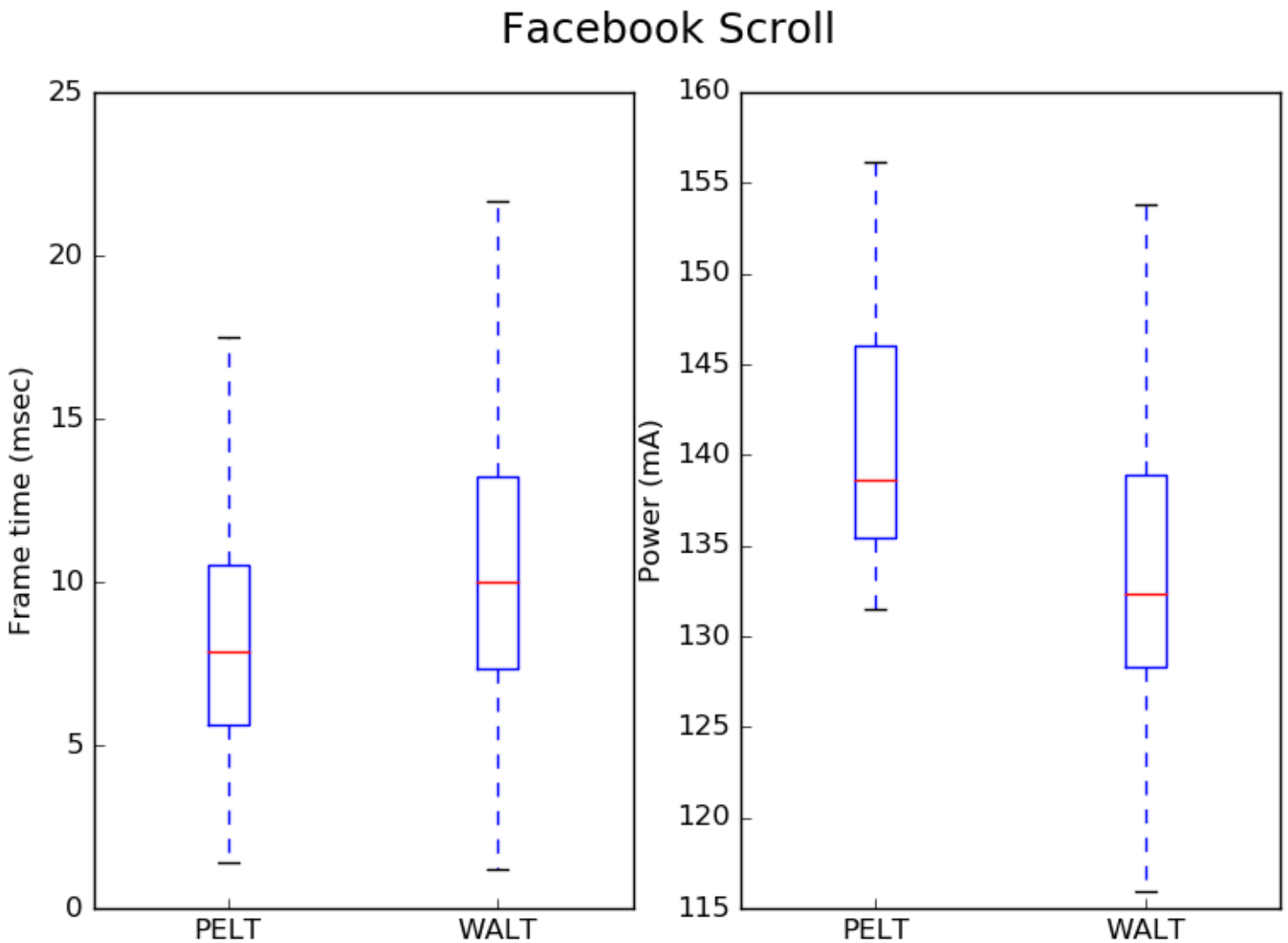
BitMap upload (overdraw)	PELT (msec)	WALT (msec)
25%	15.136069	14.489167
50%	15.908737	15.844335
75%	16.836846	18.725650
90%	18.500224	21.474493
99%	22.062164	28.835235



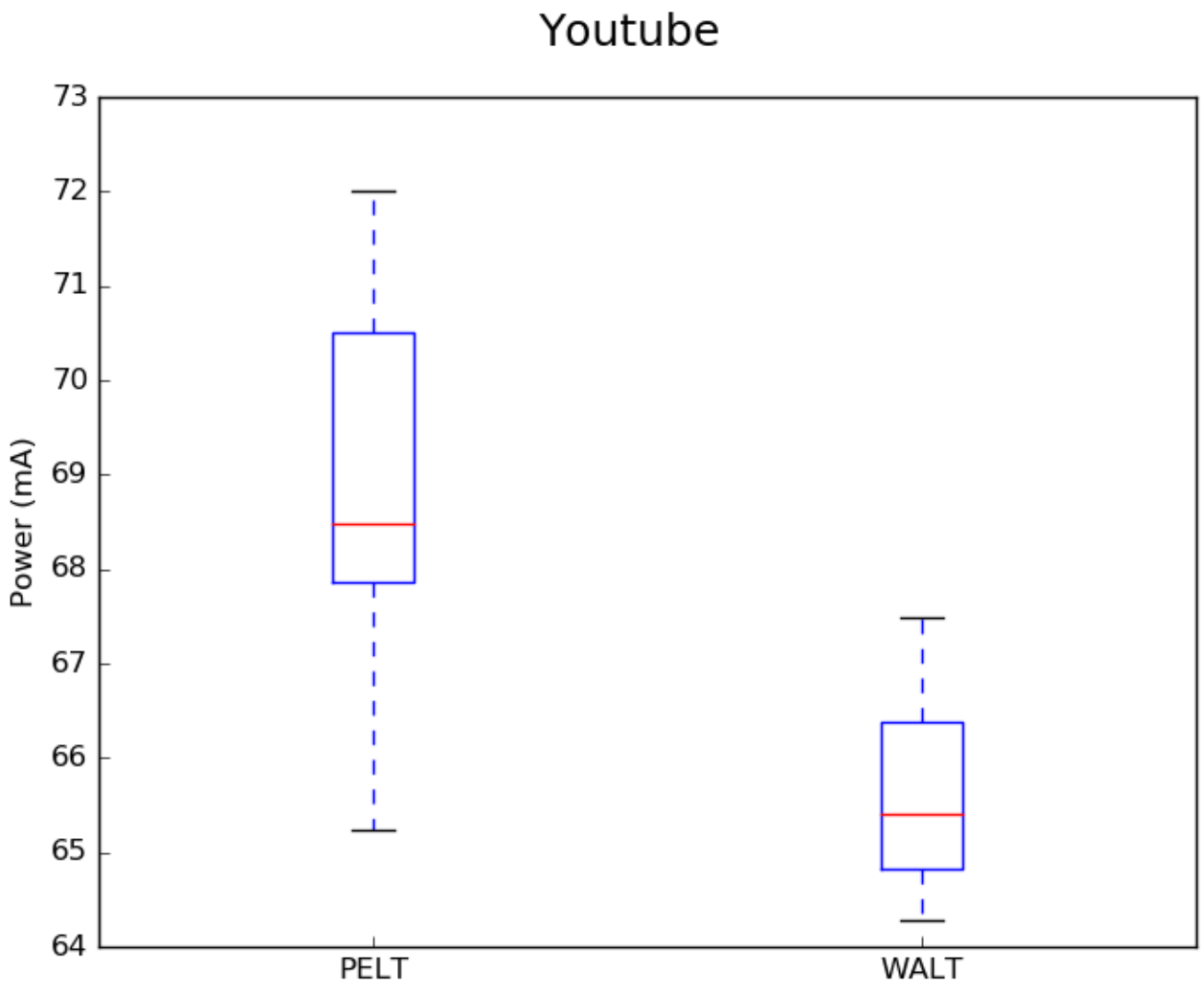
Test	Power
BitMap upload (OverDraw)	PELT is 1% better



WALT has 5% better power with similar performance (no of janks). Frame rendering times are better with PELT



WALT has 5% better power with similar performance (no of janks). Frame rendering times are better with PELT



Both PELT and WALT played the video with 30 Fps. WALT has 4% better power

Games

Game	Result
Subway Surfers	same With touchboost disable, Power is 10% better for WALT. PELT has better performance (~2FPs)
Thunder Cross	Power is 3% better for WALT for the same performance measured in FPS.
Candy crush soda saga	same

Conclusion

- There is no clear winner in UX use cases like JankBench. The performance (especially 90% and 99%) is better with WALT. The touchboost settings can be tuned down to save power. Note that for UX use cases, rendering frame with in 16 msec is critical. The power optimizations are not possible if we can’t meet this deadline.
- Clear Power savings with WALT for Youtube streaming.
- WALT performance is better in Benchmarks like PCMark.
- We will do this comparison again with Patrick’s util_est patches, which are meant to address the load decay and slower ramp up problems with PELT.

Thank you

Follow us on:

For more information, visit us at:
www.qualcomm.com & www.qualcomm.com/blog