WALT vs PELT

Saravana Kannan
Sr Staff Software Engineer
Qualcomm Innovation Center, Inc.
Agenda

- WALT
- PELT
- Why?
- Where do we stand?
- Are we there yet?
- What next?
Introduction
Introduction

• Window Assisted Load Tracking
• QuIC authored solution
• Tracks thread's load as runtime within globally aligned windows
• Window is 20ms; can be changed
Introduction

• Per Entity Load Tracking
• Upstream solution
• Tracks a thread’s load as a continuous geometric average
• Quantized into 1 ms segments
• PELT32/16/8 indicates the load’s half life of 32ms, 16ms and 8ms respectively
Why WALT?

• Works well for interactive workloads
• Optimized for mobile devices
• Optimized for commercial device requirements

Why PELT?

• Upstream solution
• Lower maintenance cost
• Unified tuning interface for Android
Is it all about Windows?
No, this is Linux!

- Load tracking is just half the picture
- How you use the load is also important
- QuIC has several value adds built on top of WALT
- WALT = WALT + QuIC Value adds
Where do we stand?
Porting and profiling

- QuIC is evaluating PELT
- DVFS and Scheduler teams working on porting our value adds to PELT
- And profiling it, a lot!
- Work in progress, but here’s some early data
Baseline, assumptions and terminology

• 4.14 Kernel baseline

• PELT = PELT + util_est + backported RT runqueue tracking - jump to max for RT
  ◦ Jumping to max frequency just doesn't cut it for Android. Every rendered frame uses a RT thread.

• I’ll be grouping QuIC value adds into cumulative set of features:
  ◦ VA#1 = Value adds group #1
  ◦ VA#2 = VA #1 + Value add group #2
  ◦ ...  

• WALT = WALT with only value adds that have been ported to PELT

Apples to apples comparisons
We are done here

Geekbench is more like a measure of your car’s “top speed” than “time to 100 km/h”
Antutu
WALT vs PELT16

• Total score is almost never a good representation because of arbitrary weights to subtest scores
• Subtest scores give a better idea
• Subtests show non-trivial regression
PCMark performance
WALT vs PELT16

• Subtests show substantial regression
PCMark power
WALT vs PELT16

- Total power shows non-trivial savings
Real world use case power
WALT vs PELT16

- Chart shows CPU + infrastructure + memory power
- Doesn’t include components not affected by PELT vs WALT
- No consistent improvement or regression

Lower is better

<table>
<thead>
<tr>
<th>Application</th>
<th>WALT</th>
<th>PELT16</th>
</tr>
</thead>
<tbody>
<tr>
<td>YouTube</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>Light game #1</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>Light game #2</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>Light game #3</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>Browser</td>
<td>100%</td>
<td>100%</td>
</tr>
</tbody>
</table>
Value adds #1
Antutu
WALT vs PELT16 + VA#1

• VA#1 recovers most of the regressions due to PELT16
PCMark performance
WALT vs PELT16 + VA#1

• VA#1 brings substantial improvements to most subtests
• Yet some subtests are substantially lower than WALT
PCMark power
WALT vs PELT16 + VA#1

• Total power also gets better
PCMark Perf/mW
WALT vs PELT16 + VA#1

• So perf/mW also improves

• This is what you want for use cases not limited by display refresh rate

• Eg: Reducing frame time from 8ms to 5ms is pointless for a 60fps display
Real world use case power
WALT vs PELT16 + VA#1

• Light game #2 gets worse
• Everything else improves
• Browser power increases by 1%, but performance increases substantially
Value adds #2
Reminder: Includes VA#1
Antutu
WALT vs PELT16 + VA#2

- VA#2 brings PELT16 on par with WALT
- Delta from WALT is just run to run variation
PCMark performance
WALT vs PELT16 + VA#2

• Significant improvement in Web performance vs VA#1
• Writing is still worse than WALT
• Rest are on par with or better than WALT
PCMark power
WALT vs PELT16 + VA#2

• Total power shows pretty high regression compared to VA#1
• But pretty close to WALT
PCMark Perf/mW

WALT vs PELT16 + VA#2

• As expected, perf/mW also becomes on par with WALT
Real world use case power
WALT vs PELT16 + VA#2

• YouTube and browser regression are pretty bad compared to VA#1
• But browser at least gets a 24% performance increase
• Overall still worse than WALT
WALT vs PELT
Is PELT ready for mobile devices?

• PELT32 is not usable
• PELT16 looks good, but not ready for commercial use yet
• Upstream PELT still has some obvious gaps for use with Android:
  ◦ Can’t jump to max frequency for RT
  ◦ Need to track RT threads similar to CFS threads
  ◦ Needs util_est support for RT
  ◦ Need to do migration fix ups for RT threads

Is WALT worth maintaining?

• WALT is approximately 8000 lines of code
• VA#1 and VA#2 comes to about 1200 lines of code.
• After porting rest of the value adds, worst case is probably 4000 lines of code
• Not having to maintain WALT but still be on-par or better would be ideal
Can PELT get there?

- QuIC plans to try
- Need to test a lot more real world use cases
- Hope to get on par with WALT by:
  - Porting of remaining QuIC value adds
  - Tuning: Same workloads have slightly different utilization values reported by WALT and PELT. Eg: VA#2 seems a bit too aggressive for PELT