Understanding the Characteristics of Android Wear OS

Renju Liu and Felix Xiaozhuo Lin
Purdue ECE
The Wearable stack
Top questions

• Wearables should enjoy
  – Baremetal performance
  – Baremetal efficiency

• In this talk: Android Wear
  – Are we close to baremetal?
  – What is going on inside?
  – How should the OS evolve?
Observation -- Symptoms

• The current performance & efficiency are far from baremetal

• **Pacing** – *inefficient*
  • face update: 400ms 88% busy
Observation -- Symptoms

• The current performance & efficiency are far from baremetal

• **Pacing** – inefficient
  • face update: 400ms 88% busy

• **Racing** – slow
  • Launch an in-mem app: 1 sec

Launch “settings”
What happens underneath?

User touch
Launch action starts
App UI shown

177 ms 810 ms
What happens underneath?

- User touch
- Launch action starts
- App UI shown

Power / mW

- 177 ms
- 810 ms
What happens underneath?

Phase 1

User touch

Launch action starts

Phase 2

Idle

Busy with various tasks

App UI shown

Power / mW

CPU Exec.

177 ms

810 ms
What happens underneath?

User touch

Launch action starts

App UI shown

Phase 1

Phase 2

Power / mW

CPU Exec.

177 ms

810 ms

28 ms

130 ms

19 ms

Idle

Busy with various tasks
Four Aspects

- CPU busy?
- CPU idle?
- Thread-level parallelism (TLP)
- Microarchitectural behaviors

Won’t talk about our methodologies
Profiling – Core Use Scenarios

**Wakeup**
Update notification wrist...

**Single Input**
launch apps palming voice...

**Sensing**
Accel heart baro

**Interaction**
Game notes navigation
OS execution dominates CPU usage.

100%
75%
50%
25%
0%

<table>
<thead>
<tr>
<th>update</th>
<th>notif</th>
<th>wrist</th>
<th>touch</th>
</tr>
</thead>
<tbody>
<tr>
<td>lch.set</td>
<td>lch.calc</td>
<td>lch.game</td>
<td>palming</td>
</tr>
<tr>
<td>game</td>
<td>notes</td>
<td>navi</td>
<td>accel</td>
</tr>
</tbody>
</table>

Wakeup | Single In. | Interact. | Sensing
OS execution dominates CPU usage.
OS execution dominates CPU usage.
OS execution dominates CPU usage.

- Idle
- Apps
- OS:Clockwork
- OS:daemons

Wakeup
- Single In.
- Interact.
- Sensing

- Updates
- Notifications
- Wrist
- Touch
- Ich: set
- Ich: calc
- Ich: game
- Palming
- Voice
- Game
- Notes
- Navi
- Accel
- Heart
- Baro
OS execution dominates CPU usage.
OS execution dominates CPU usage.
OS execution dominates CPU usage.

The chart below shows the breakdown of CPU usage across different processes and activities:

- System Server
- Surface Flinger
- MediaServer
- logd
- Other

Activities include:
- Wakeup
- Single In.
- Interact.
- Sensing

The data is represented in a bar chart, where the x-axis represents different activities, and the y-axis shows the percentage of CPU usage.
Costly OS services are ...
Costly OS services are likely cruft.

**cruft** (kräft)

*n.*

1. Trash, debris, or other unwanted matter that accumulates over time.
2. Unnecessary digital information that accumulates over time, such as unneeded files or obsolete lines of code in software: *"By removing cruft, you can recover valuable disk space ... and reduce the chance of software conflicts" (Joe Kissell).*
Hot functions: highly skewed distribution

Top 5 $\rightarrow$ >20% CPU cycles
Top 50 $\rightarrow$ >50% CPU cycles
Hot functions: highly skewed distribution

Top 5 $\rightarrow$ >20% CPU cycles
Top 50 $\rightarrow$ >50% CPU cycles

Manipulating basic data structures
Legacy/improper OS designs
Hot functions: highly skewed distribution

Top 5 $\rightarrow$ >20% CPU cycles
Top 50 $\rightarrow$ >50% CPU cycles

Manipulating basic data structures
Legacy/improper OS designs

Anecdotes
Backlight  UI layout  low-mem killer
**Idle episodes: plentiful and of various lengths**

<table>
<thead>
<tr>
<th></th>
<th>Time (ms)</th>
<th>Pct. Overall</th>
<th>Episodes</th>
<th>Pct. Explained</th>
</tr>
</thead>
<tbody>
<tr>
<td>notes</td>
<td>614.1</td>
<td>17.1%</td>
<td>376</td>
<td>100.0%</td>
</tr>
<tr>
<td>voice</td>
<td>843.3</td>
<td>50.5%</td>
<td>352</td>
<td>100.0%</td>
</tr>
<tr>
<td>lch.game</td>
<td>722.6</td>
<td>50.9%</td>
<td>205</td>
<td>99.9%</td>
</tr>
<tr>
<td>lch.set</td>
<td>185.2</td>
<td>25.6%</td>
<td>110</td>
<td>92.9%</td>
</tr>
<tr>
<td>lch.calc</td>
<td>153.6</td>
<td>15.6%</td>
<td>120</td>
<td>91.4%</td>
</tr>
<tr>
<td>touch</td>
<td>16.8</td>
<td>10.6%</td>
<td>6</td>
<td>100.0%</td>
</tr>
<tr>
<td>update</td>
<td>223.0</td>
<td>61.2%</td>
<td>44</td>
<td>100.0%</td>
</tr>
<tr>
<td>navi</td>
<td>2173.0</td>
<td>52.80%</td>
<td>912</td>
<td>100.0%</td>
</tr>
<tr>
<td>notif</td>
<td>4035.6</td>
<td>86.80%</td>
<td>277</td>
<td>100.0%</td>
</tr>
</tbody>
</table>
Idle anomalies are caused by ...

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>notes</td>
<td>614.1</td>
<td>17.1%</td>
<td>376</td>
<td>100.0%</td>
</tr>
<tr>
<td>voice</td>
<td>843.3</td>
<td>50.5%</td>
<td>352</td>
<td>100.0%</td>
</tr>
<tr>
<td>lch.game</td>
<td>722.6</td>
<td>50.9%</td>
<td>205</td>
<td>99.9%</td>
</tr>
<tr>
<td>lch.set</td>
<td>185.2</td>
<td>25.6%</td>
<td>110</td>
<td>92.9%</td>
</tr>
<tr>
<td>touch</td>
<td>153.6</td>
<td>15.6%</td>
<td>120</td>
<td>91.4%</td>
</tr>
<tr>
<td>User think</td>
<td>16.8</td>
<td>10.6%</td>
<td>6</td>
<td>100.0%</td>
</tr>
<tr>
<td>Bluetooth tail time</td>
<td>223.0</td>
<td>61.2%</td>
<td>44</td>
<td>100.0%</td>
</tr>
<tr>
<td>OS shell policy</td>
<td>2173.0</td>
<td>52.80%</td>
<td>912</td>
<td>100.0%</td>
</tr>
<tr>
<td>lch.calc</td>
<td>4035.6</td>
<td>86.80%</td>
<td>277</td>
<td>100.0%</td>
</tr>
</tbody>
</table>

- Device suspend
- Voice UI
- Cont. interaction
- Cont. interact.+NetI/O
- Storage I/O
- User think
- Bluetooth tail time
- OS shell policy
- App policy
Idle anomalies are caused by ...

- Device suspend
- Voice UI
- Cont. interaction
- Cont. interact.+NetI/O
- Storage I/O
- User think
- Bluetooth tail time
- OS shell policy
- App policy

Legacy/improper OS designs
Performance overprovisioning

Anecdote
Voice UI
Subject: Substantial TLP on a par with desktop

- **CPU busy**
- **CPU idle**
- **TLP**
- **uArch**

# of concurrent threads:

- C0
- C1
- C2
- C3
- C4

Tasks:
- Wakeup
- Single In.
- Interact.
- Sensing

Percentage:
- 100%
- 75%
- 50%
- 25%
- 0%
Substantial TLP on a par with desktop

# of concurrent threads
Substantial TLP on a par with desktop

TLP: avg. busy CPU cores (over non-idle time)

# of concurrent threads

<table>
<thead>
<tr>
<th>C0</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.8</td>
<td>1.4</td>
<td>1.4</td>
<td>1.4</td>
<td>1.4</td>
</tr>
<tr>
<td>2.2</td>
<td>2.0</td>
<td>1.5</td>
<td>1.5</td>
<td>1.5</td>
</tr>
<tr>
<td>1.8</td>
<td>1.4</td>
<td>1.5</td>
<td>1.4</td>
<td>1.6</td>
</tr>
<tr>
<td>1.5</td>
<td>1.5</td>
<td>1.1</td>
<td>1.1</td>
<td>1.1</td>
</tr>
</tbody>
</table>

Wakeup | Single In. | Interact. | Sensing

CPU busy
CPU idle
TLP
uArch
...due to short interactions.
Apps are mostly single-threaded; OS contributes to TLP significantly.
Wearable suffers from uArch inefficiency

Cycles-per-instruction (lower is better)  
2 -- 5 (high!)
Wearable suffers from uArch inefficiency

Cycles-per-instruction (lower is better)

2 -- 5 (high!)

Smartphone as a comparison

1.3 -- 2.5 web rendering

<2 SPEC INT
Wearable suffers from uArch inefficiency

**Cycles-per-instruction** (lower is better)

2 -- 5 (high!)

Smartphone as a comparison

1.3 -- 2.5 web rendering

<2 SPEC INT
Wearable suffers from uArch inefficiency

Cycles-per-instruction (lower is better)

2 -- 5 (high!)

Smartphone as a comparison

1.3 -- 2.5 web rendering

<2 SPEC INT
The major cause: complex OS code (L1 icache, iTLB, and branch predictor)
The major cause: complex OS code (L1 icache, iTLB, and branch predictor)

uArch problem will NOT be gone with future wearable CPUs
Four Aspects

CPU busy
- OS dominates
- Lots of cruft
- Skewed hot functions
- Legacy bottlenecks

CPU idle
- Anomalous
- OS flaws
- Too much performance

Thread-level parallelism
- Desktop-like
- OS-contributed

Microarchitectural behaviors
- Mismatch
- OS code complexity
Repair, don’t overhaul (yet)

- **CPU busy**
  - OS dominates
  - Lots of cruft
  - Skewed hot functions
  - Legacy bottlenecks

- **CPU idle**
  - Anomalous
  - OS flaws
  - Too much performance

- **Thread-level parallelism**
  - Desktop-like
  - OS-contributed

- **Microarchitectural behaviors**
  - Mismatch
  - OS code complexity
How about after that? (i.e. “next-gen wearable OS”)

We probably will reach a point when OS overhaul/redesign is justified.

Specializing OS for common, single-app scenarios
Restructuring OS for Wearable

Specializing OS for common, single-app scenarios

OS Daemons

Kernel

Full
Simple

Full
Simple

Activity
Manager

Window
Manager
Restructuring OS for Wearable

- Apps
  - Activity Manager
  - Window Manager
- OS Daemons
  - Full
  - Simple
  - Simple
- Kernel
  - Full

...
Restructuring OS for Wearable

Apps
- Simple
- Full

OS Daemons
- Activity Manager
- Window Manager

Kernel

60
Final takeaway

• Wearables: unique usage and hardware
• Many mobile OS tradeoffs are invalid
  – efficiency v.s. flexibility & programming ease
• Immediate actions: fixing individual OS components
• Future: OS specialization may be needed
FAQ

• You forgot Apple Watch or Samsung Tizen.
• Isn’t your discovery just some oversight of Google engineers?
• Aren’t these things easy to fix?
• Doesn’t multicore wearable sound crazy?
• Power! I want to learn about power.
• I bet the Android Wear team already fixed these!

xsel.rocks/p/wear
Has Android Wear improved?

**Android Wear 2.0 Developer Preview**

**New User Interface**
- Material design for wearables
- Expanded notifications
- Darker UI

**Standalone Apps**
- Direct network access to cloud
- Apps run on watch even when your phone (Android or iOS) isn’t with you

**Watch Face**
- Complications API: any watch face can show data from any app

**Messaging**
- New input methods: handwriting, keyboard, Smart Reply, and 3rd party IMEs

**Fitness**
- Google Fit platform: automatic activity recognition and data API

[g.co/wearpreview](g.co/wearpreview)