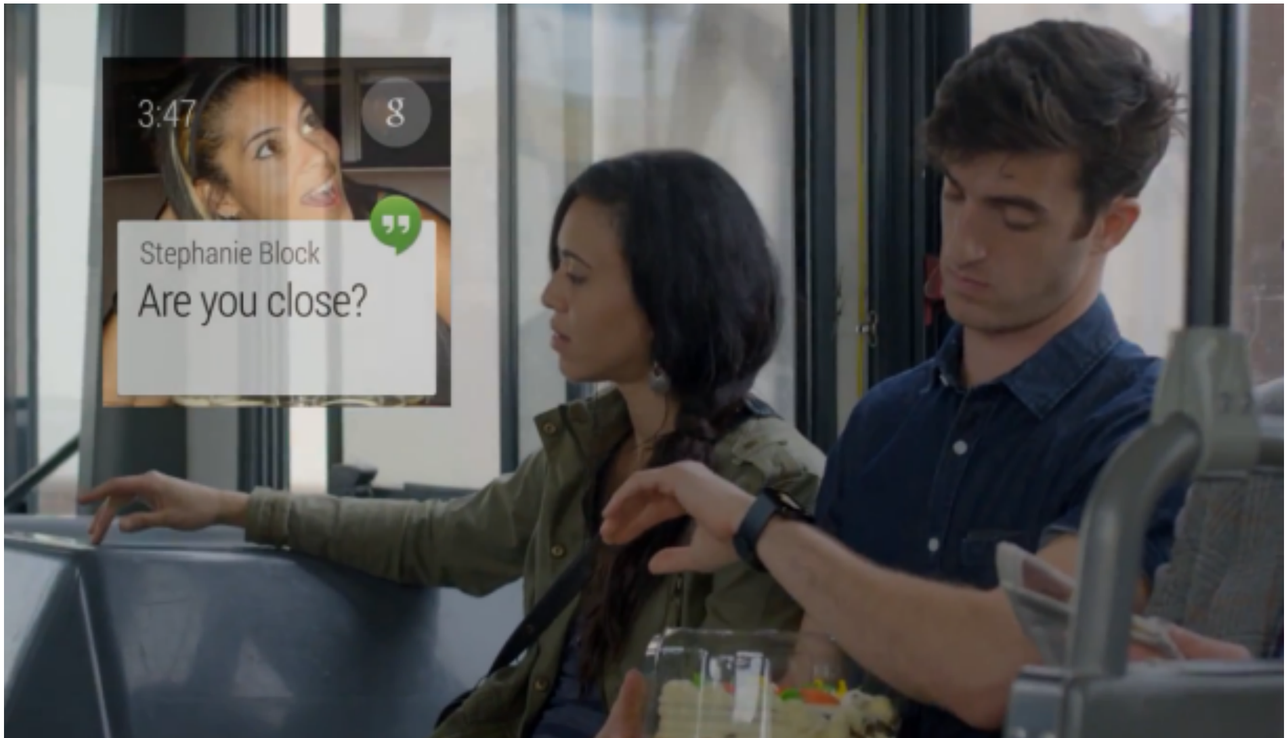
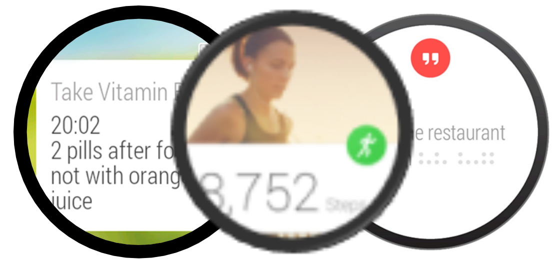
Understanding the Characteristics of Android Wear OS

Renju Liu and Felix Xiaozhu 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

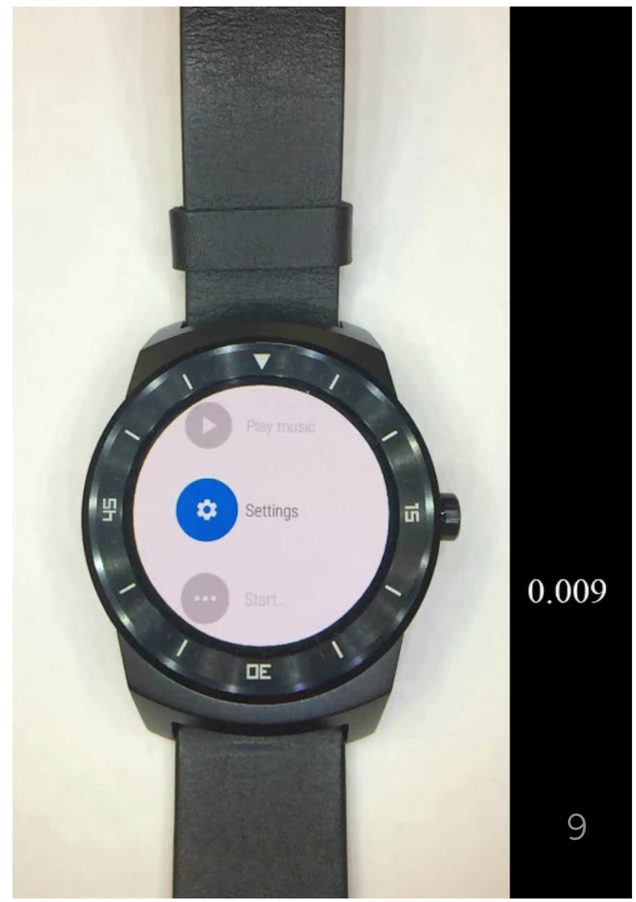
- **Pacing** – inefficient
  - face update: 400ms 88% busy

Clock face update

Observation -- Symptoms

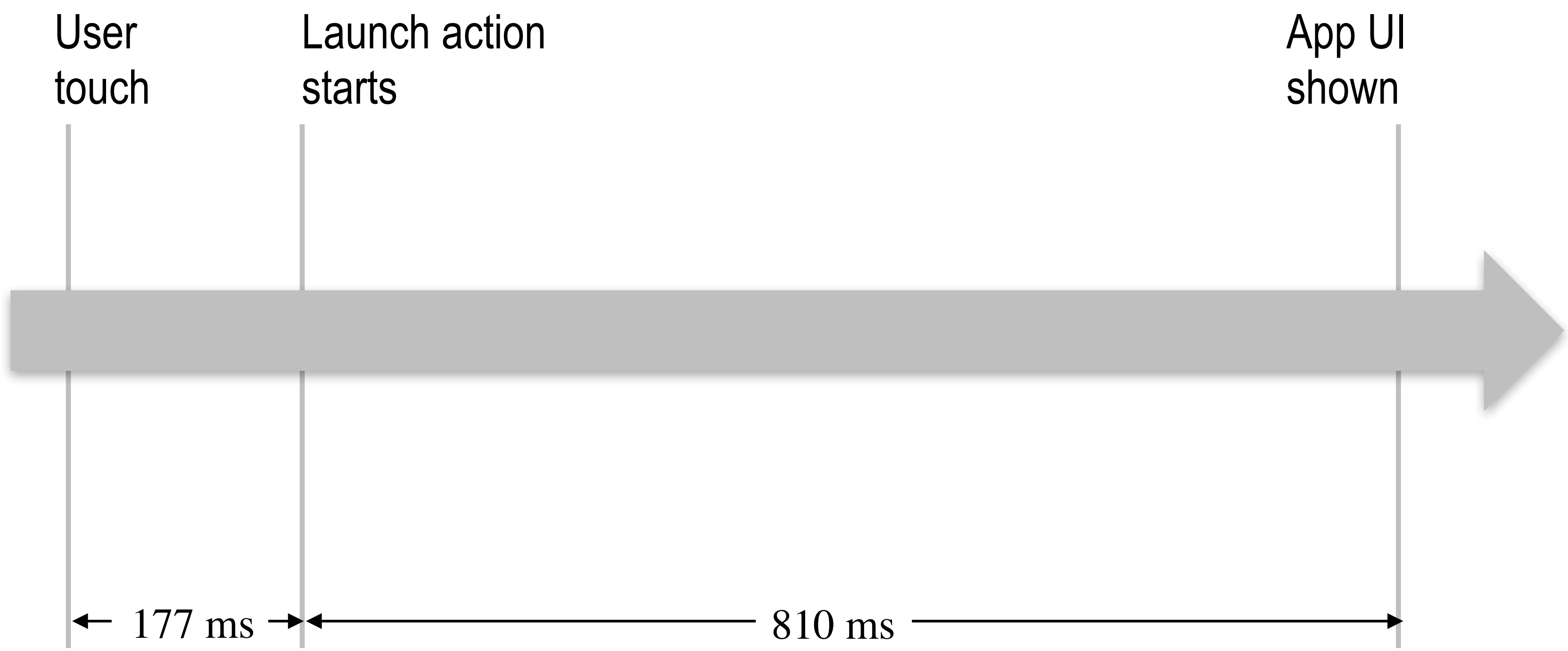
- The current performance & efficiency are far from baremetal

Launch “settings”

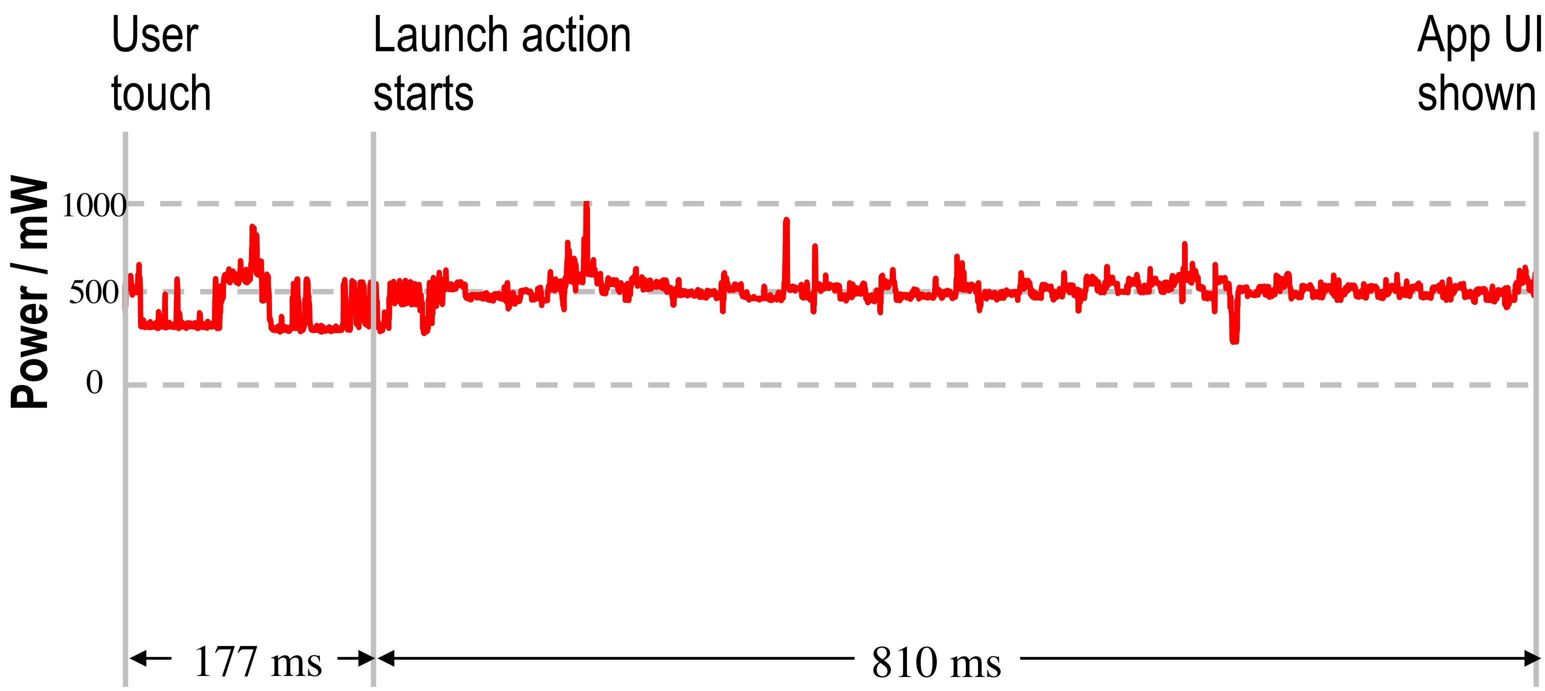


- **Pacing** – inefficient
  - face update: 400ms 88% busy
- **Racing** – slow
  - Launch an in-mem app: 1 sec

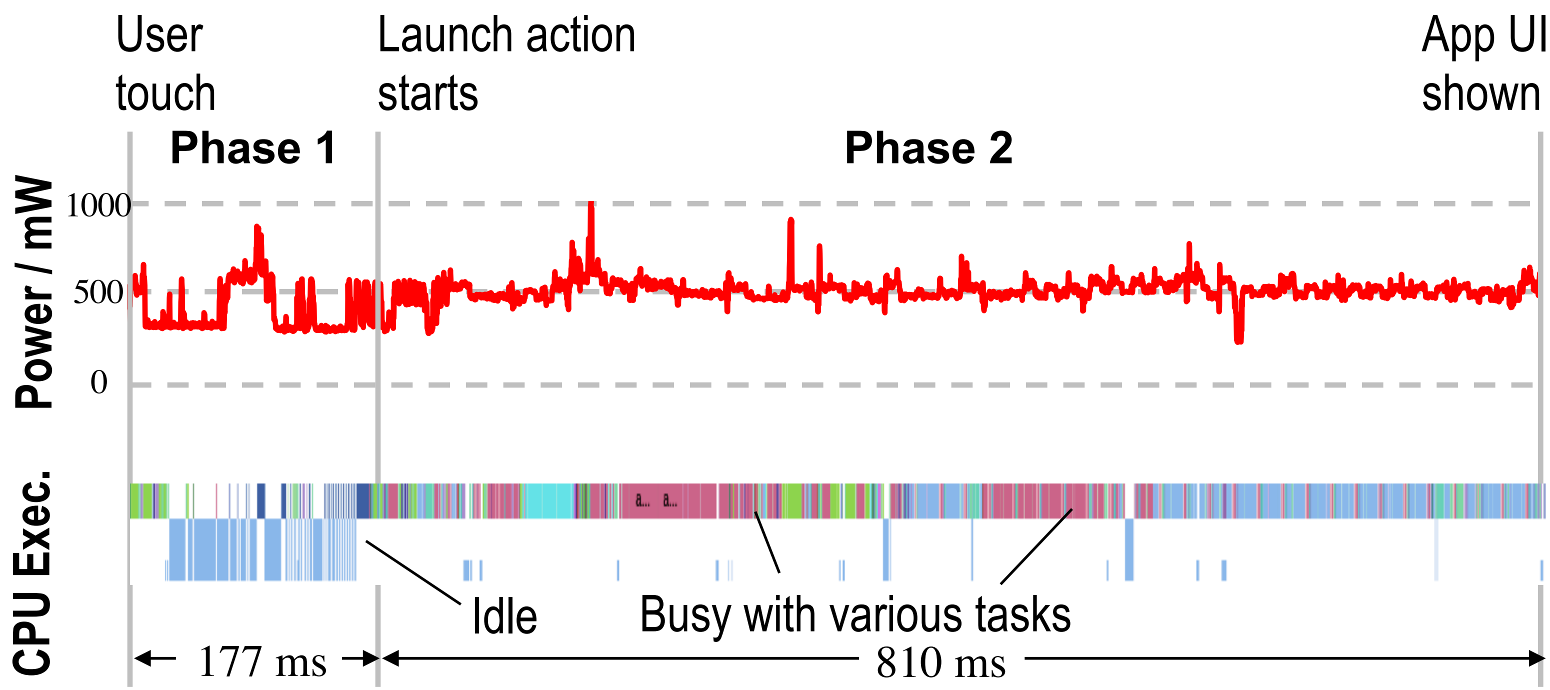
What happens underneath?



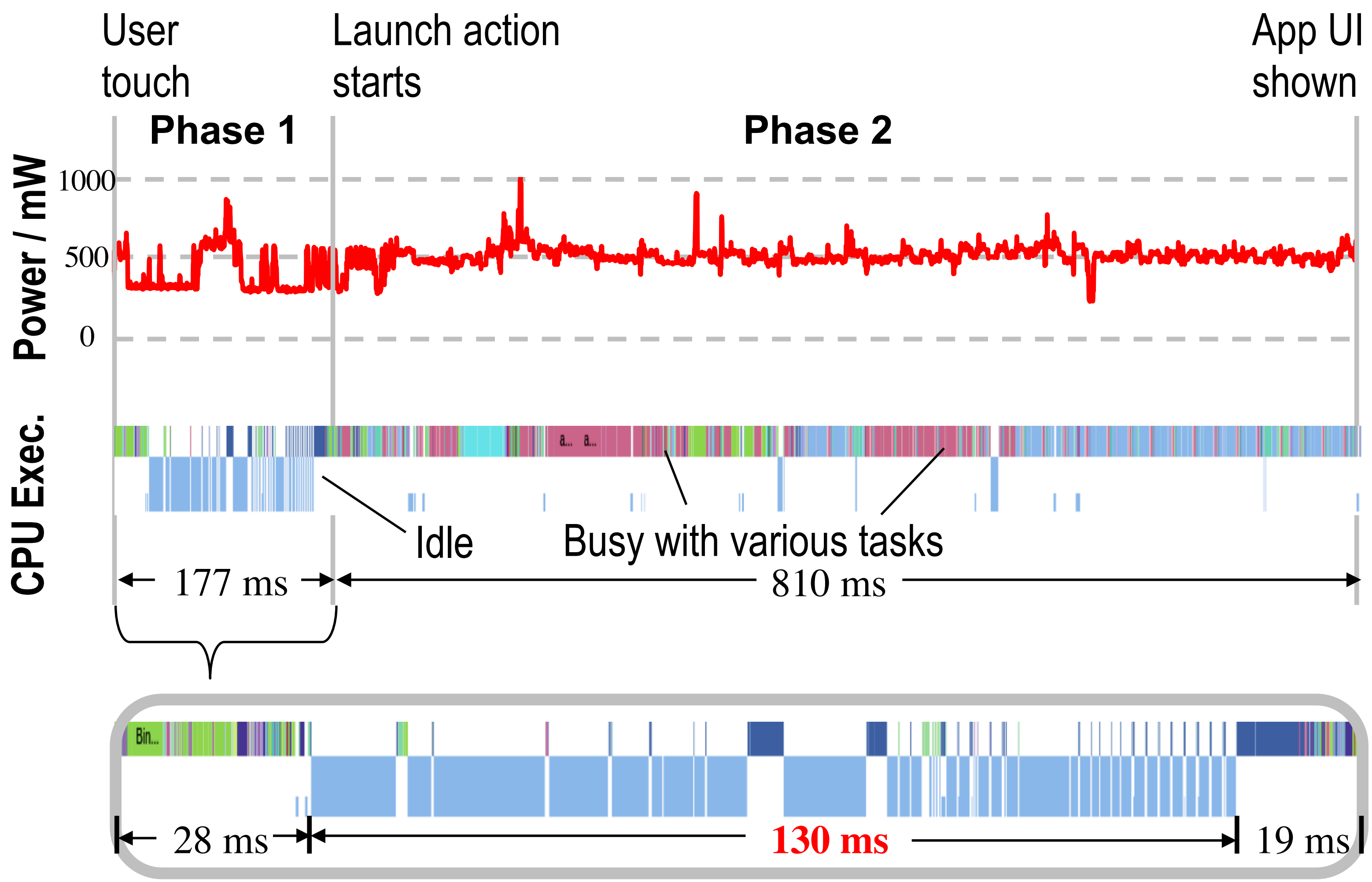
What happens underneath?



What happens underneath?



What happens underneath?



Four Aspects

CPU busy?

CPU idle?

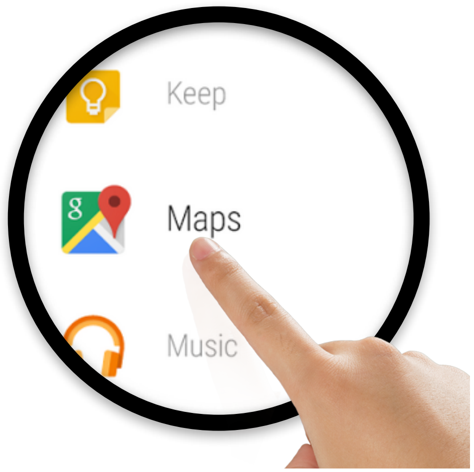
Thread-level parallelism (TLP)

Microarchitectural behaviors

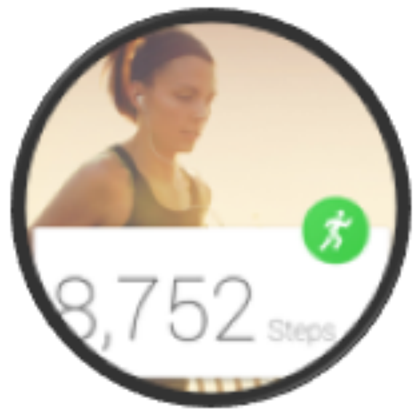
Won't talk about our methodologies

Profiling – Core Use Scenarios

Wakeup
Update
notification
wrist...



Single Input
launch apps
palming
voice...

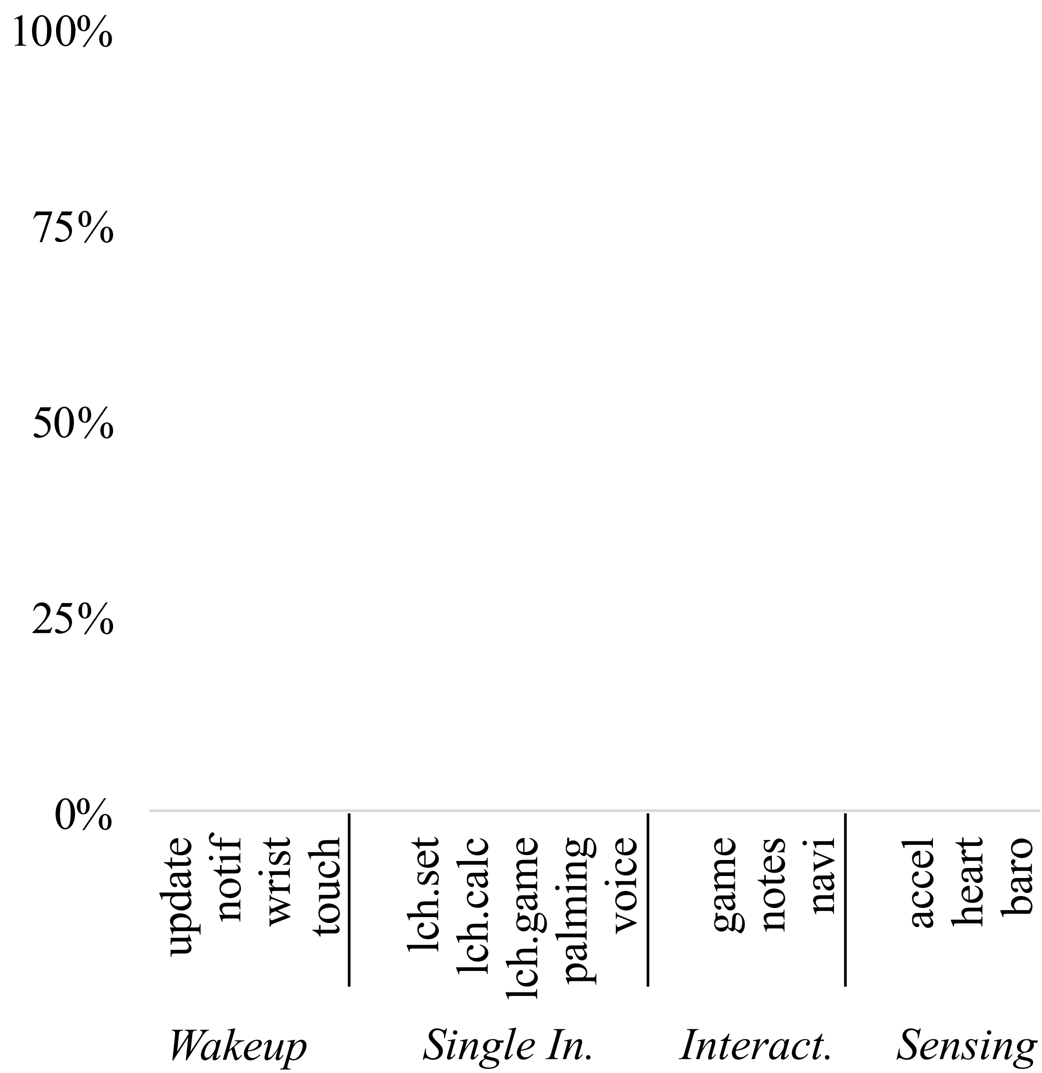


Sensing
Accel
heart
baro

Interaction
Game
notes
navigation

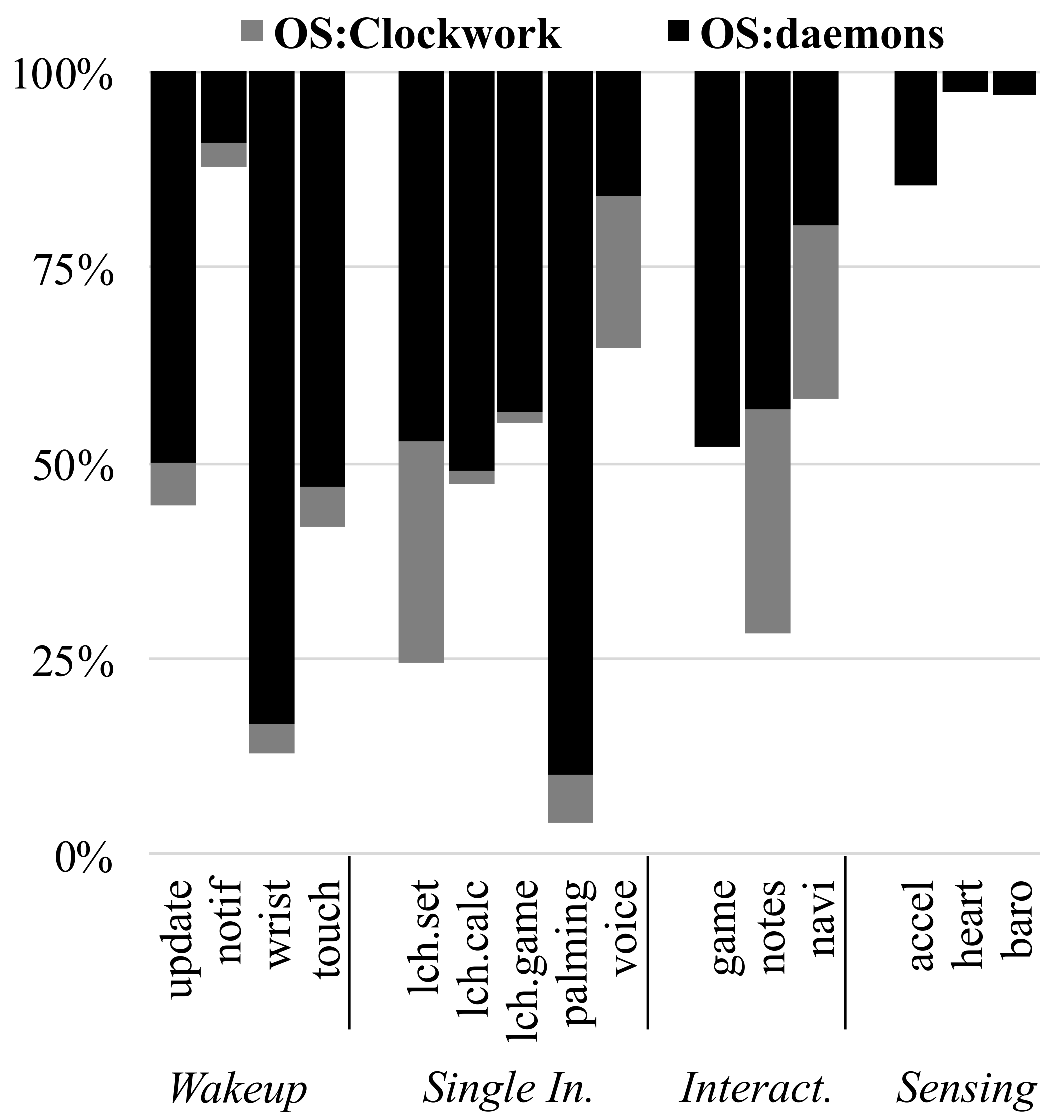
CPU busy
CPU idle
TLP
uArch

OS execution dominates CPU usage.



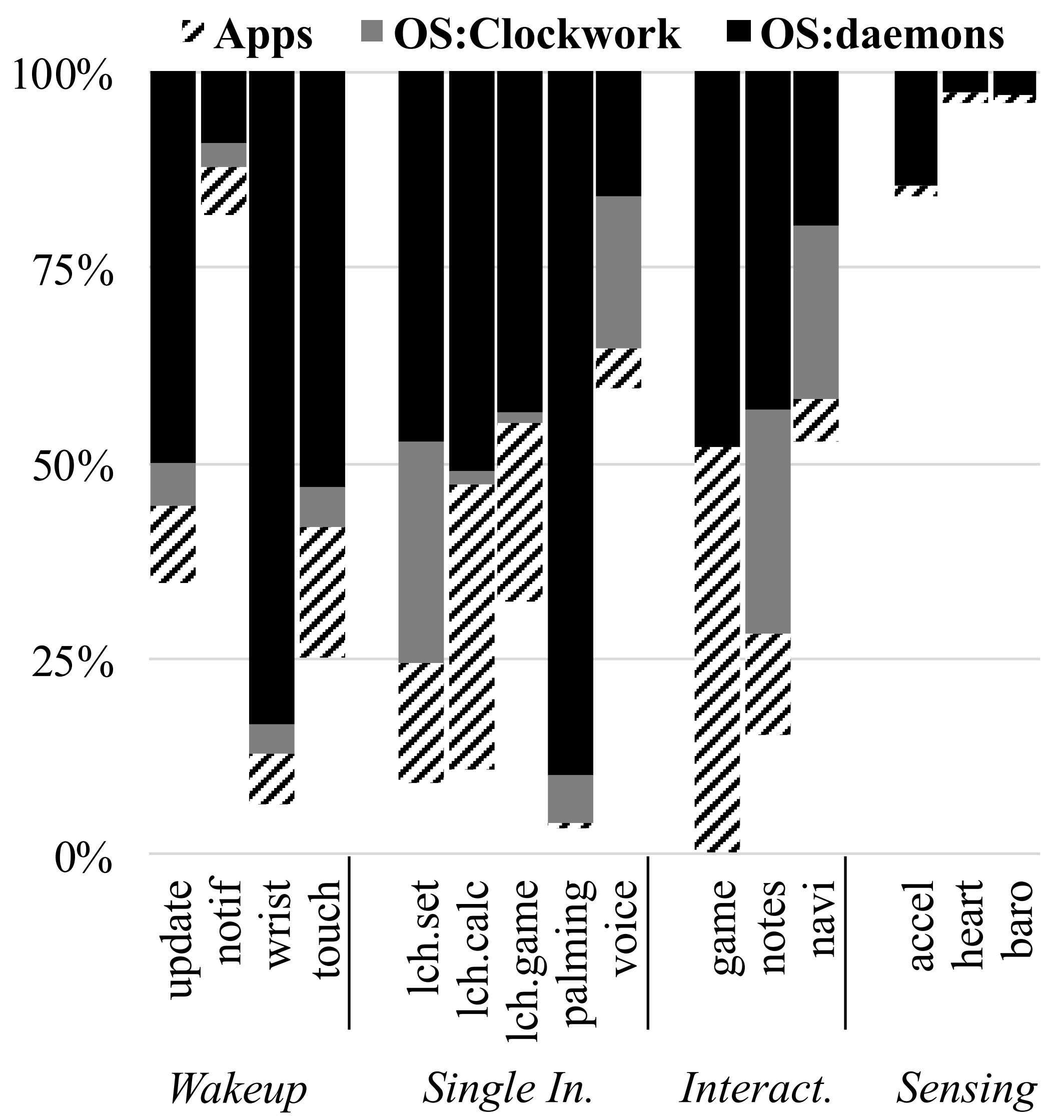
CPU busy
CPU idle
TLP
uArch

OS execution dominates CPU usage.

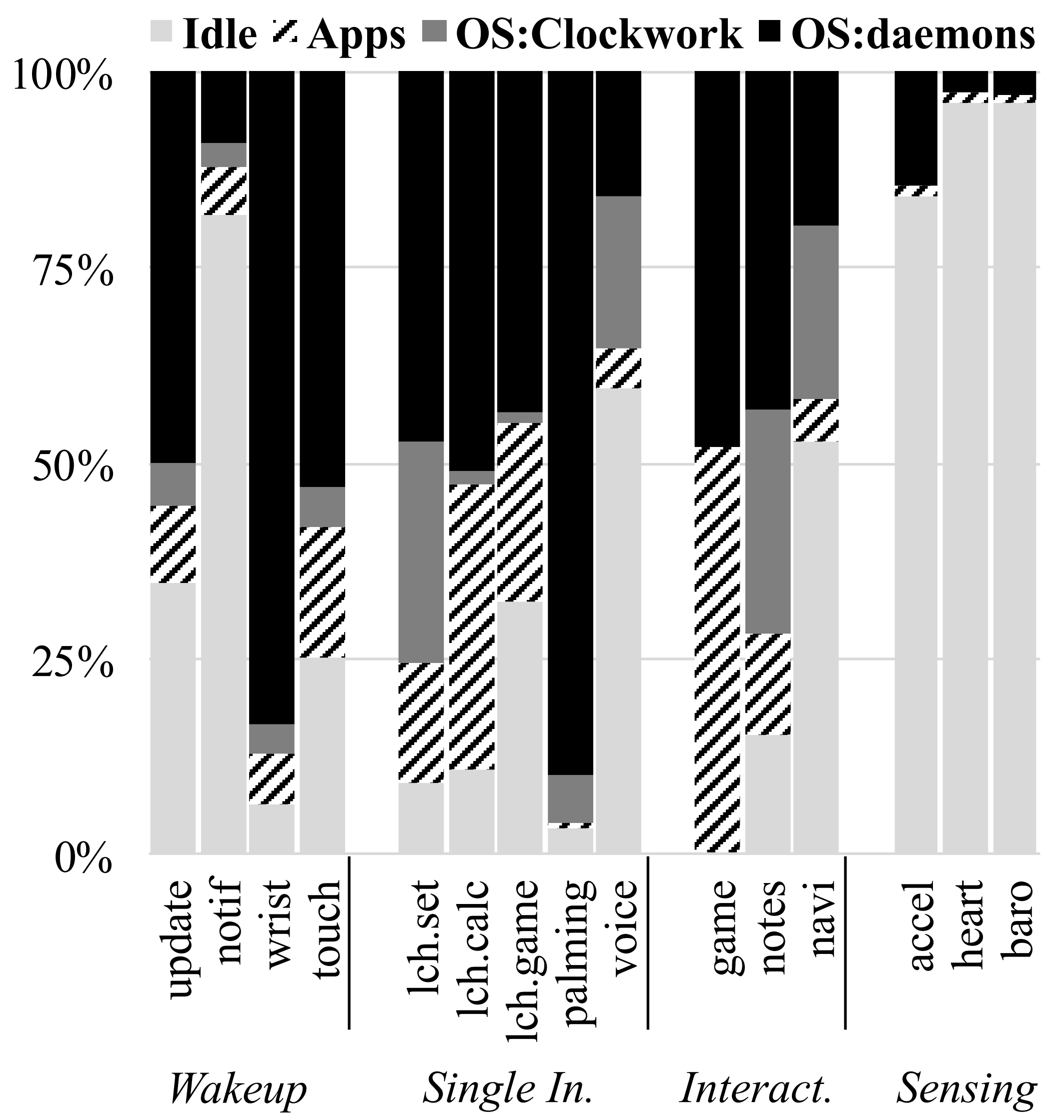


CPU busy
CPU idle
TLP
uArch

OS execution dominates CPU usage.

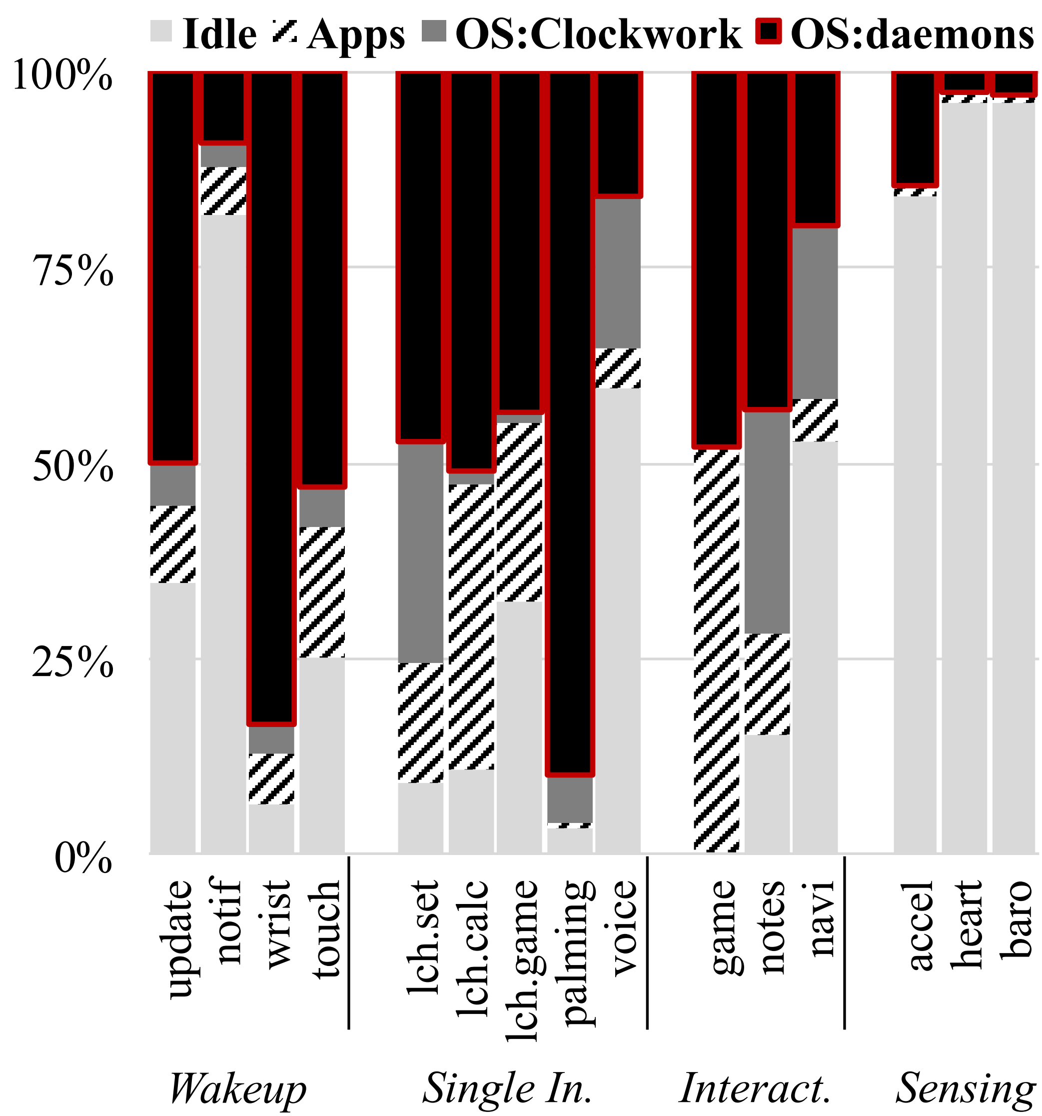


OS execution dominates CPU usage.



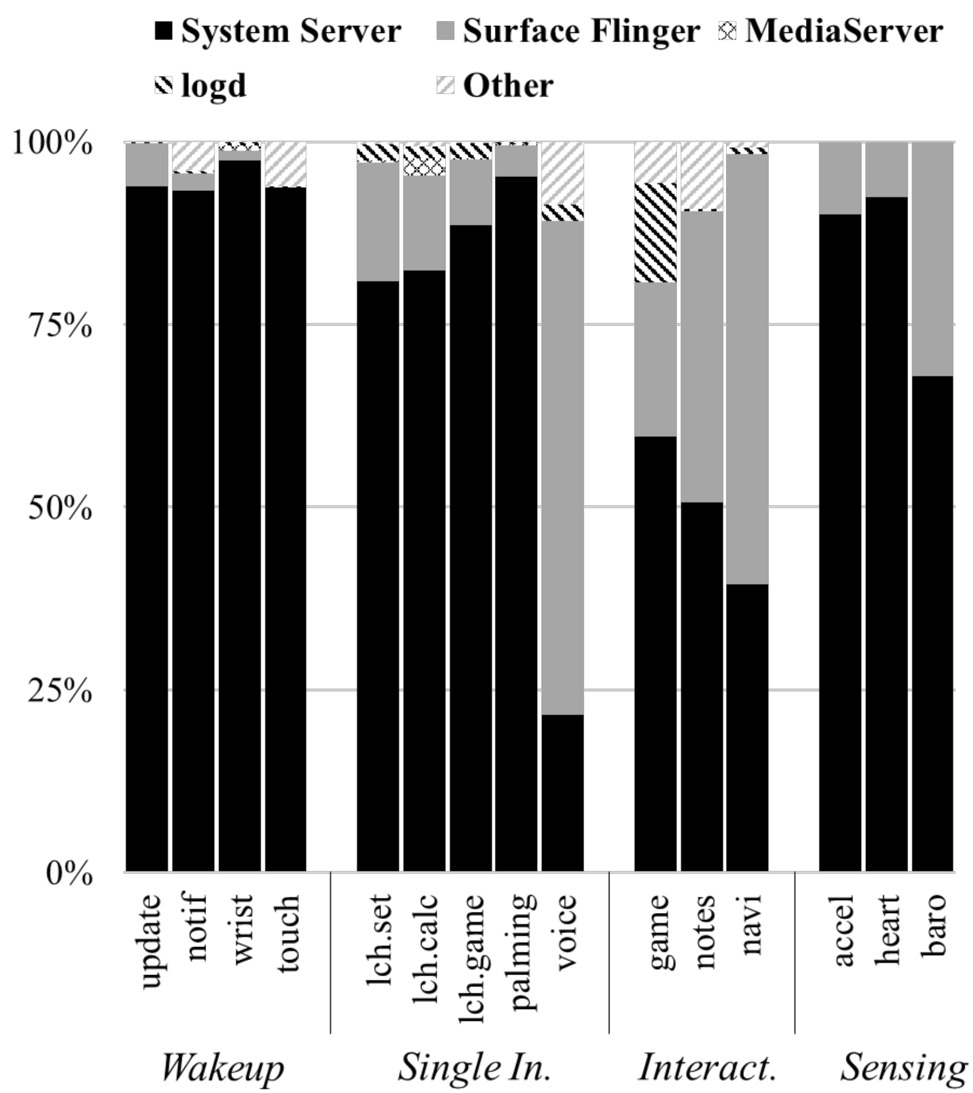
CPU busy
CPU idle
TLP
uArch

OS execution dominates CPU usage.



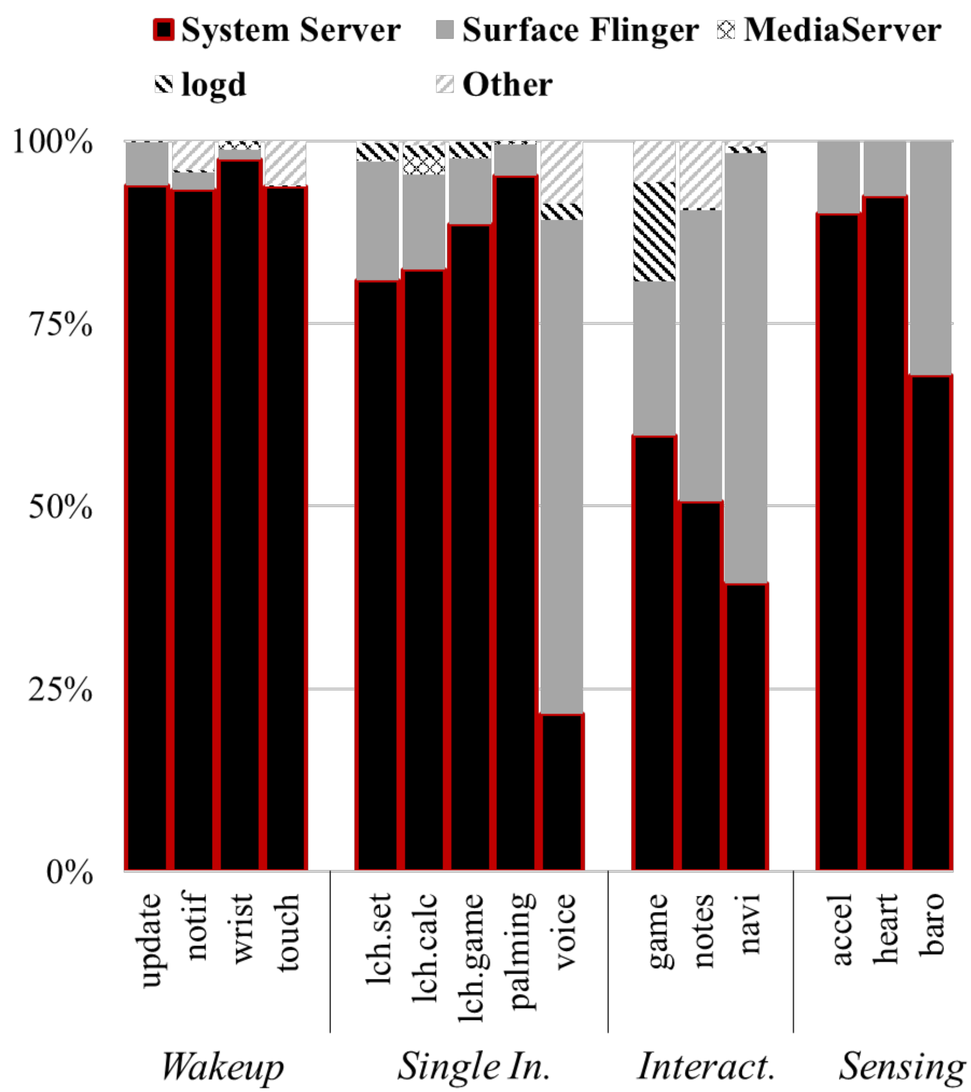
CPU busy
CPU idle
TLP
uArch

OS execution dominates CPU usage.



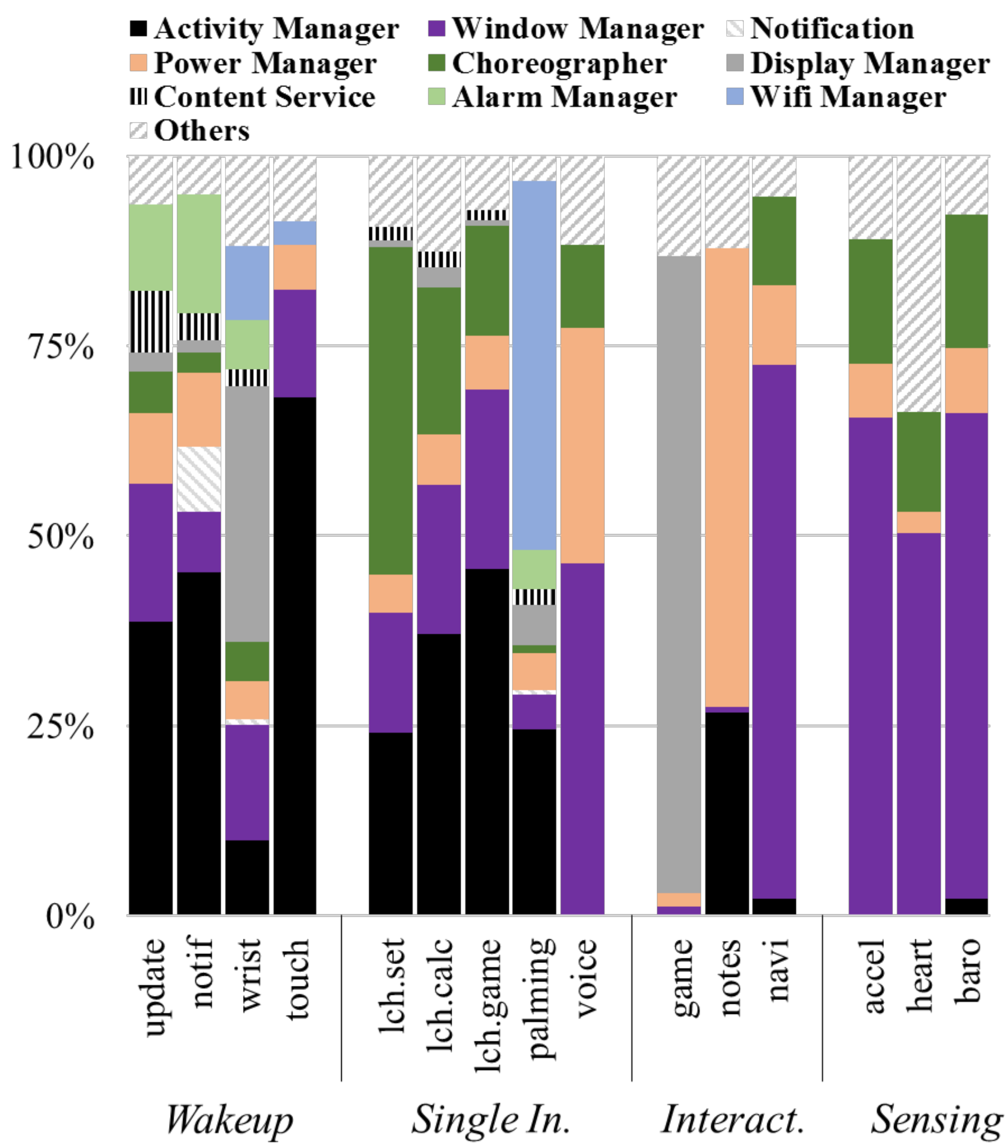
CPU busy
CPU idle
TLP
uArch

OS execution dominates CPU usage.



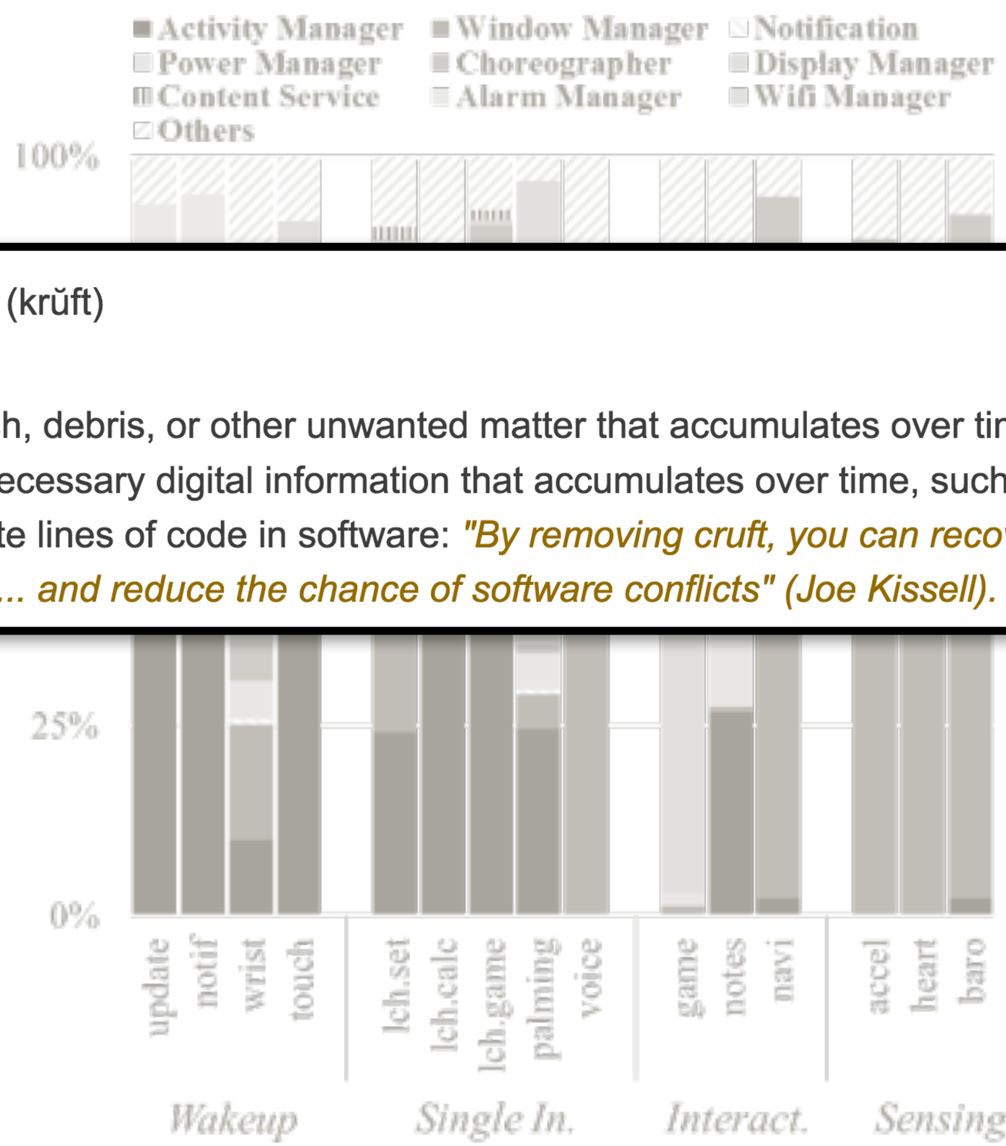
CPU busy
CPU idle
TLP
uArch

Costly OS services are ...



CPU busy
CPU idle
TLP
uArch

Costly OS services are likely cruft.



cruft (krŭft)

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 → >20% CPU cycles

Top 50 → >50% CPU cycles

Hot functions: highly skewed distribution

Top 5 → >20% CPU cycles

Top 50 → >50% CPU cycles

Manipulating basic data structures

Legacy/improper OS designs

Hot functions: highly skewed distribution

Top 5 → >20% CPU cycles

Top 50 → >50% CPU cycles

Manipulating basic data structures

Legacy/improper OS designs

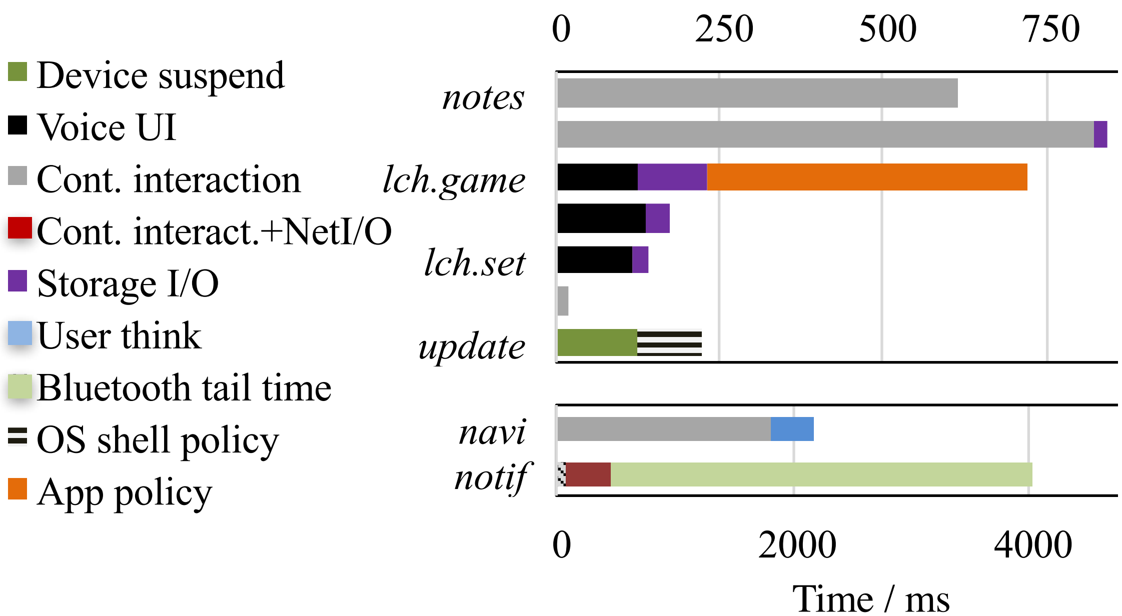
Anecdotes

Backlight UI layout low-mem killer

Idle episodes: plentiful and of various lengths

Time (ms)	Pct. Overall	Episodes	Pct. Explained	
614.1	17.1%	376	100.0%	*notes*
843.3	50.5%	352	100.0%	*voice*
722.6	50.9%	205	99.9%	*lch.game*
185.2	25.6%	110	92.9%	*lch.calc*
153.6	15.6%	120	91.4%	*lch.set*
16.8	10.6%	6	100.0%	*touch*
223.0	61.2%	44	100.0%	*update*
2173.0	52.80%	912	100.0%	*navi*
4035.6	86.80%	277	100.0%	*notif*

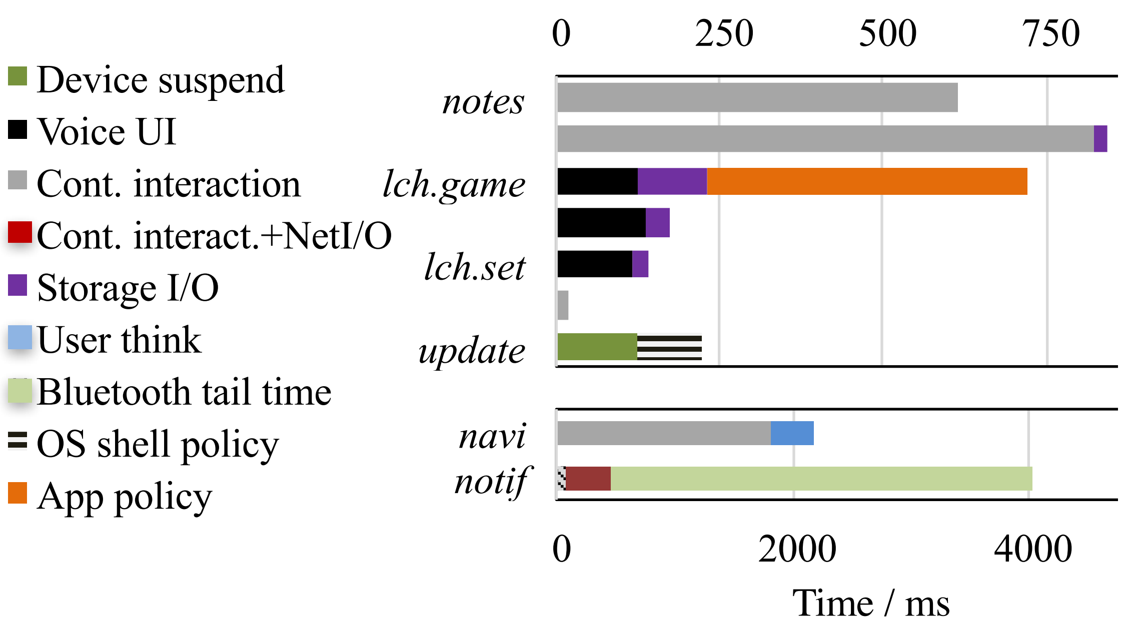
Idle anomalies are caused by …



Time (ms)	Pct. Overall	Episodes	Pct. Explained	
614.1	17.1%	376	100.0%	*notes*
843.3	50.5%	352	100.0%	*voice*
722.6	50.9%	205	99.9%	*lch.game*
185.2	25.6%	110	92.9%	*lch.calc*
153.6	15.6%	120	91.4%	*lch.set*
16.8	10.6%	6	100.0%	*touch*
223.0	61.2%	44	100.0%	*update*
2173.0	52.80%	912	100.0%	*navi*
4035.6	86.80%	277	100.0%	*notif*

CPU busy
CPU idle
TLP
uArch

Idle anomalies are caused by …



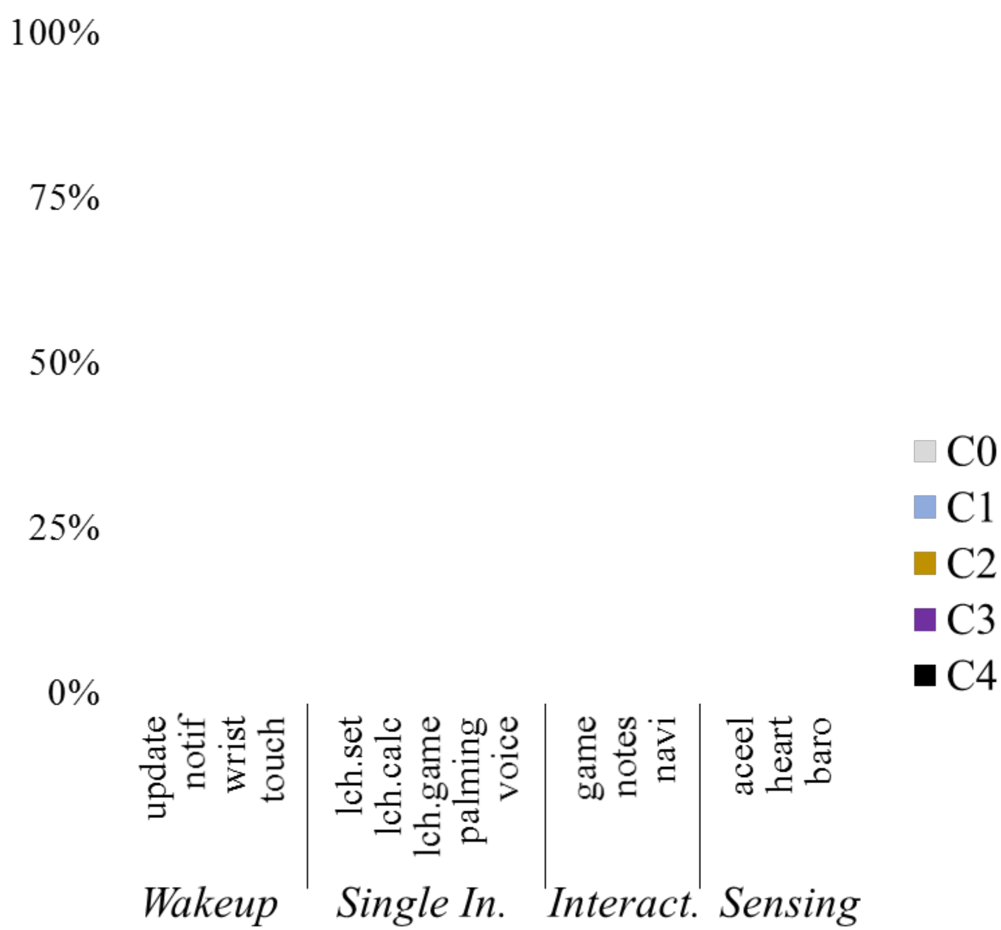
Time (ms)	Pct. Overall	Episodes	Pct. Explained	
614.1	17.1%	376	100.0%	*notes*
843.3	50.5%	352	100.0%	*voice*
722.6	50.9%	205	99.9%	*lch.game*
185.2	25.6%	110	92.9%	*lch.calc*
153.6	15.6%	120	91.4%	*lch.set*
16.8	10.6%	6	100.0%	*touch*
223.0	61.2%	44	100.0%	*update*
2173.0	52.80%	912	100.0%	*navi*
4035.6	86.80%	277	100.0%	*notif*

Legacy/improper OS designs
Performance overprovisioning

Anecdote

Voice UI

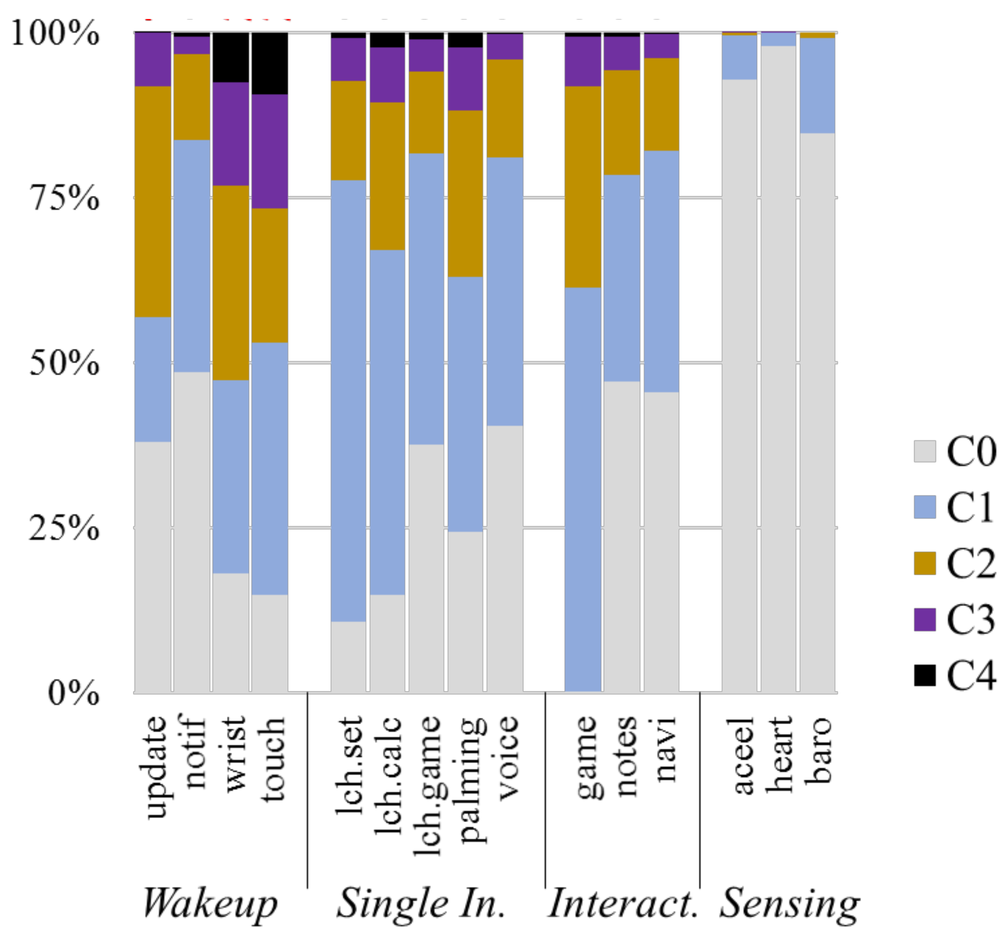
Substantial TLP on a par with desktop



# of concurrent threads

CPU busy
CPU idle
TLP
uArch

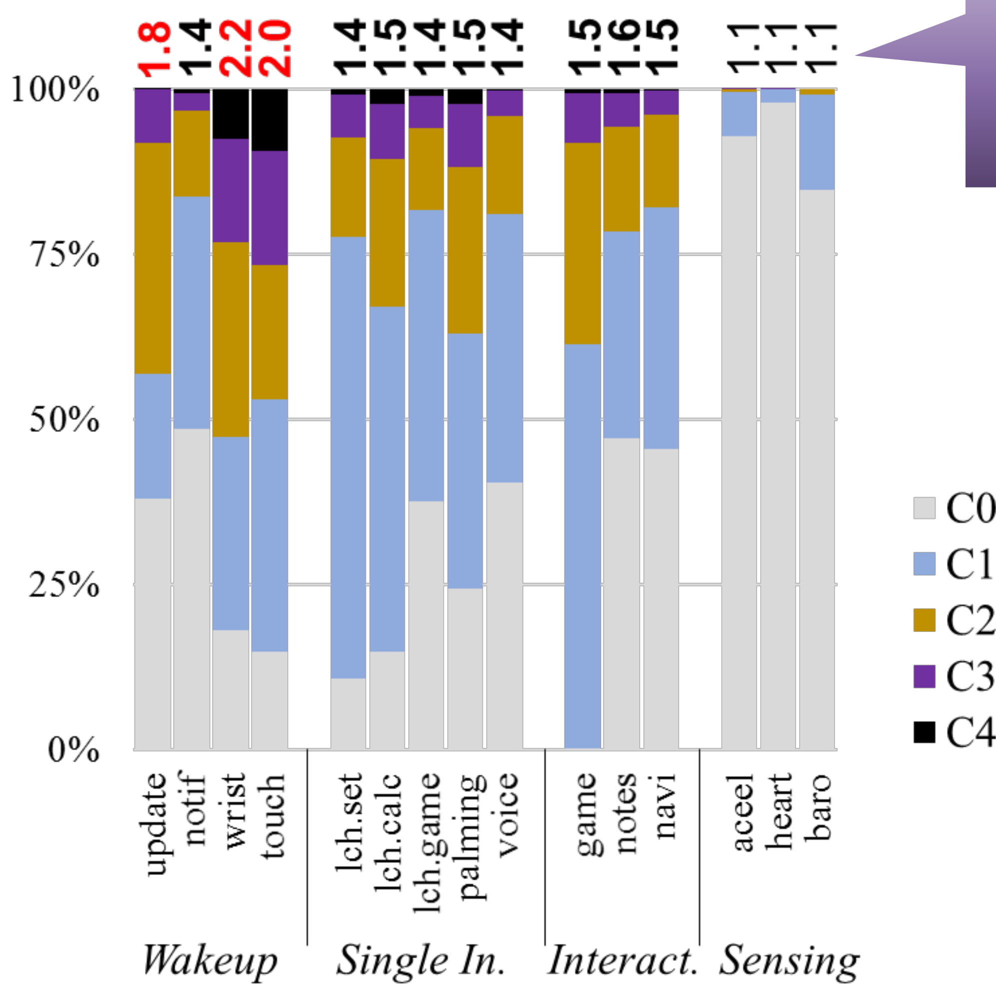
Substantial TLP on a par with desktop



# of concurrent threads

CPU busy
CPU idle
TLP
uArch

Substantial TLP on a par with desktop

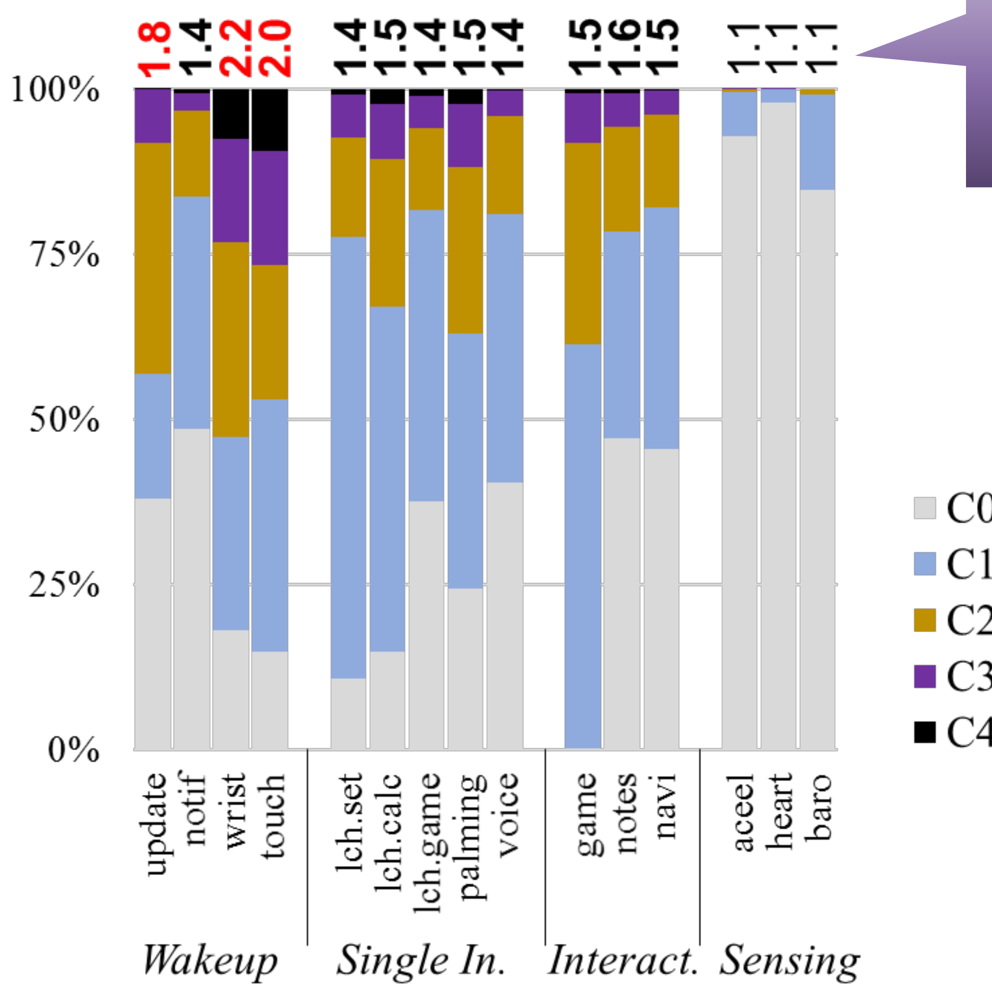


TLP: avg. busy CPU cores (over non-idle time)

# of concurrent threads

CPU busy
CPU idle
TLP
uArch

...due to short interactions.

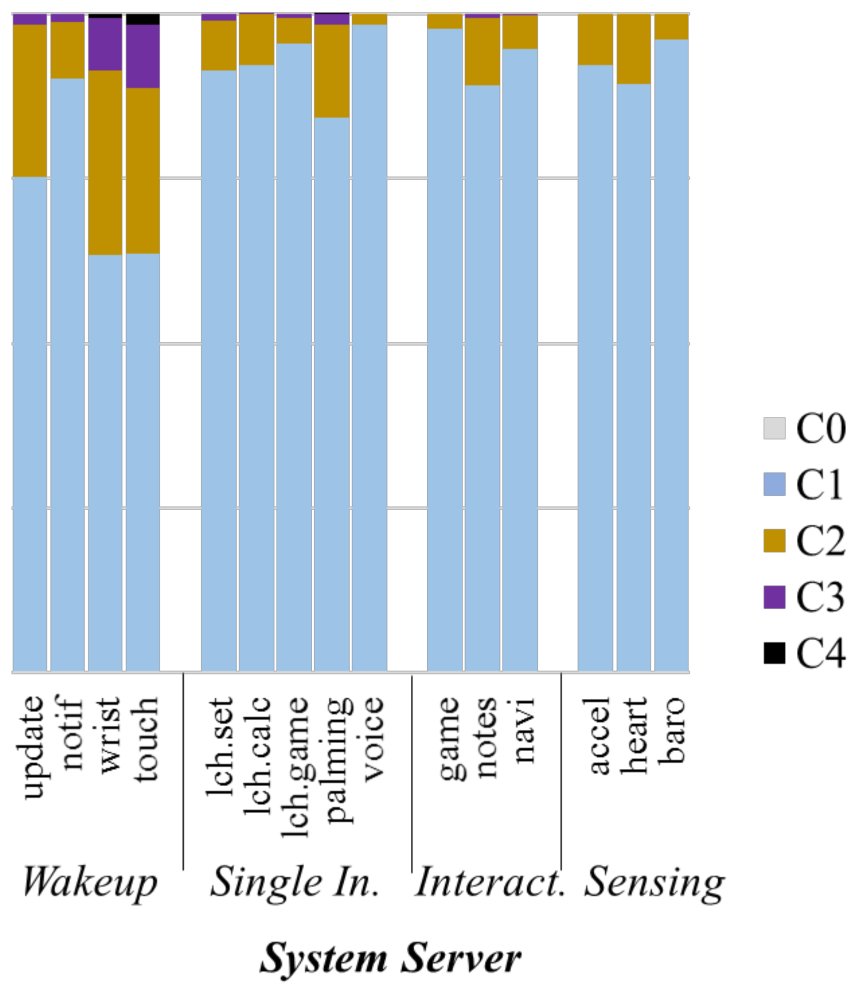


TLP: avg. busy CPU cores
(over non-idle time)

# of concurrent threads

CPU busy
CPU idle
TLP
uArch

Apps are mostly single-threaded; OS contributes to TLP significantly.

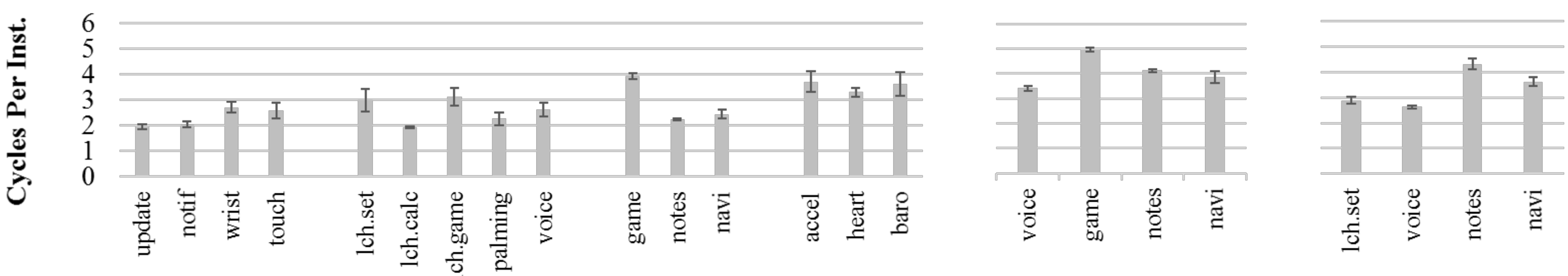


CPU busy
CPU idle
TLP
uArch

Wearable suffers from uArch inefficiency

Cycles-per-instruction (lower is better)

2 -- 5 (high!)



CPU busy
CPU idle
TLP
uArch

Wearable suffers from uArch inefficiency

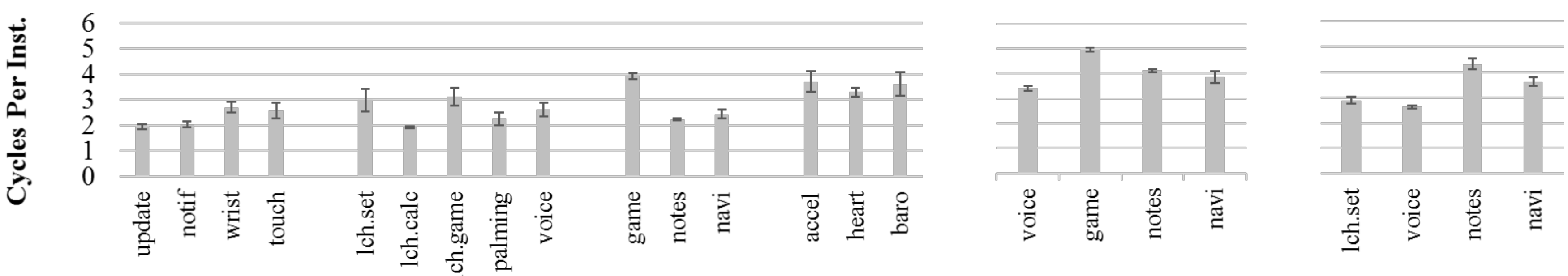
Cycles-per-instruction (lower is better)

2 -- 5 (high!)

Smartphone as a comparison

1.3 -- 2.5 web rendering

<2 SPEC INT



CPU busy
CPU idle
TLP
uArch

Wearable suffers from uArch inefficiency

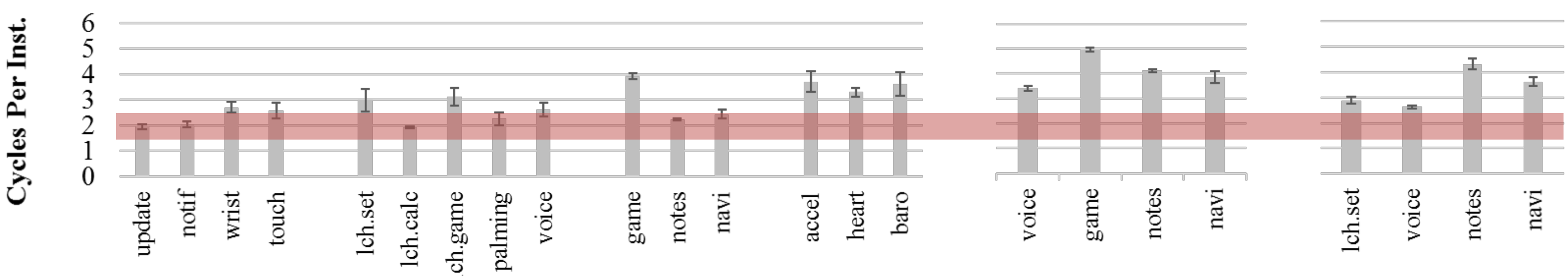
Cycles-per-instruction (lower is better)

2 -- 5 (high!)

Smartphone as a comparison

1.3 -- 2.5 web rendering

<2 SPEC INT



CPU busy
CPU idle
TLP
uArch

Wearable suffers from uArch inefficiency

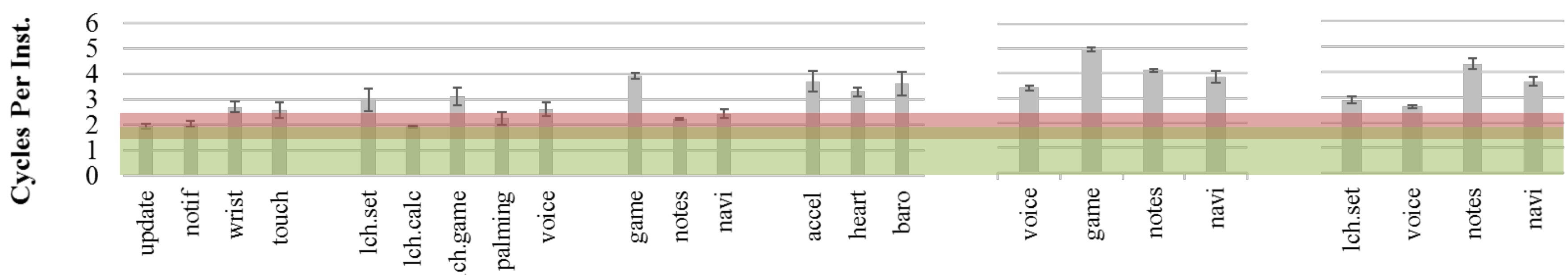
Cycles-per-instruction (lower is better)

2 -- 5 (high!)

Smartphone as a comparison

1.3 -- 2.5 web rendering

<2 SPEC INT



CPU busy
CPU idle
TLP
uArch

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

CPU busy
CPU idle
TLP
uArch

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

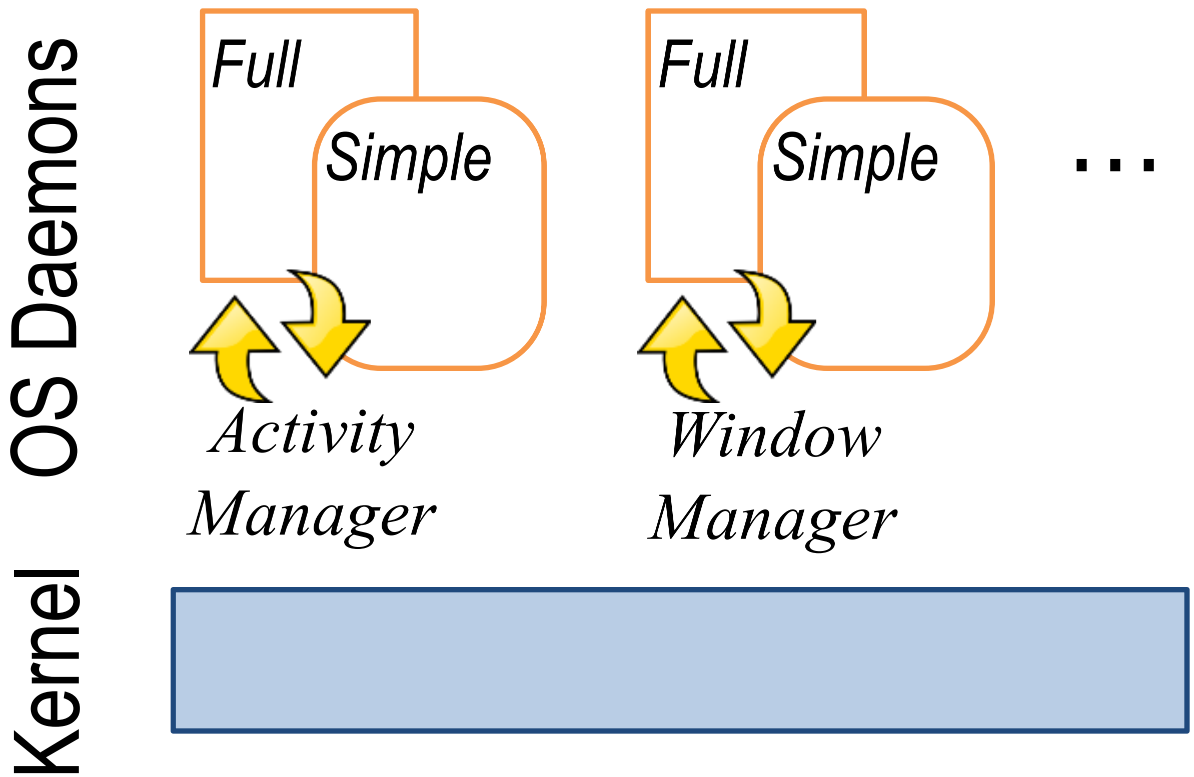
How about after that?
(i.e. “next-gen wearable OS”)

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

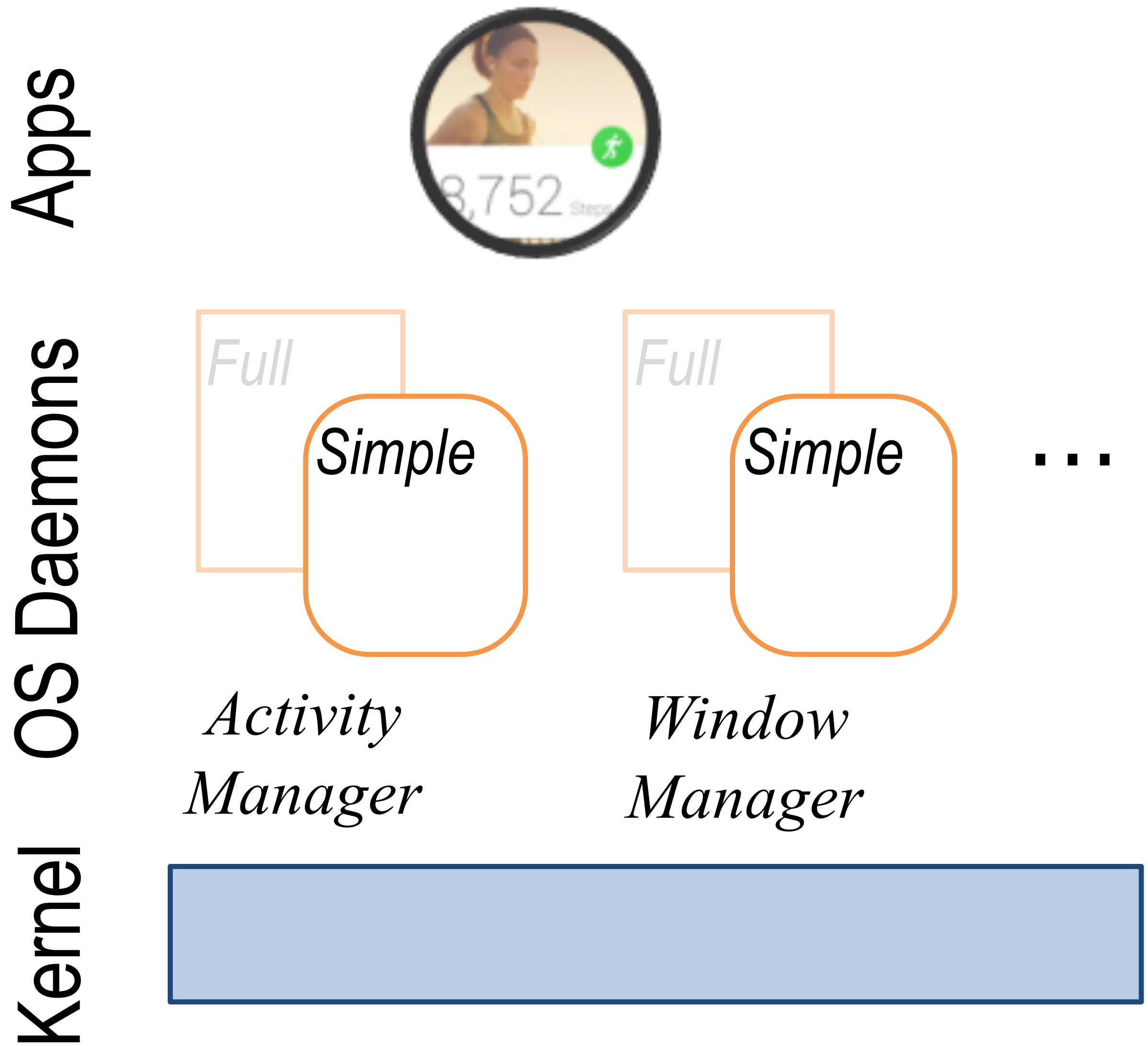
Specializing OS
for common, single-app scenarios

Restructuring OS for Wearable

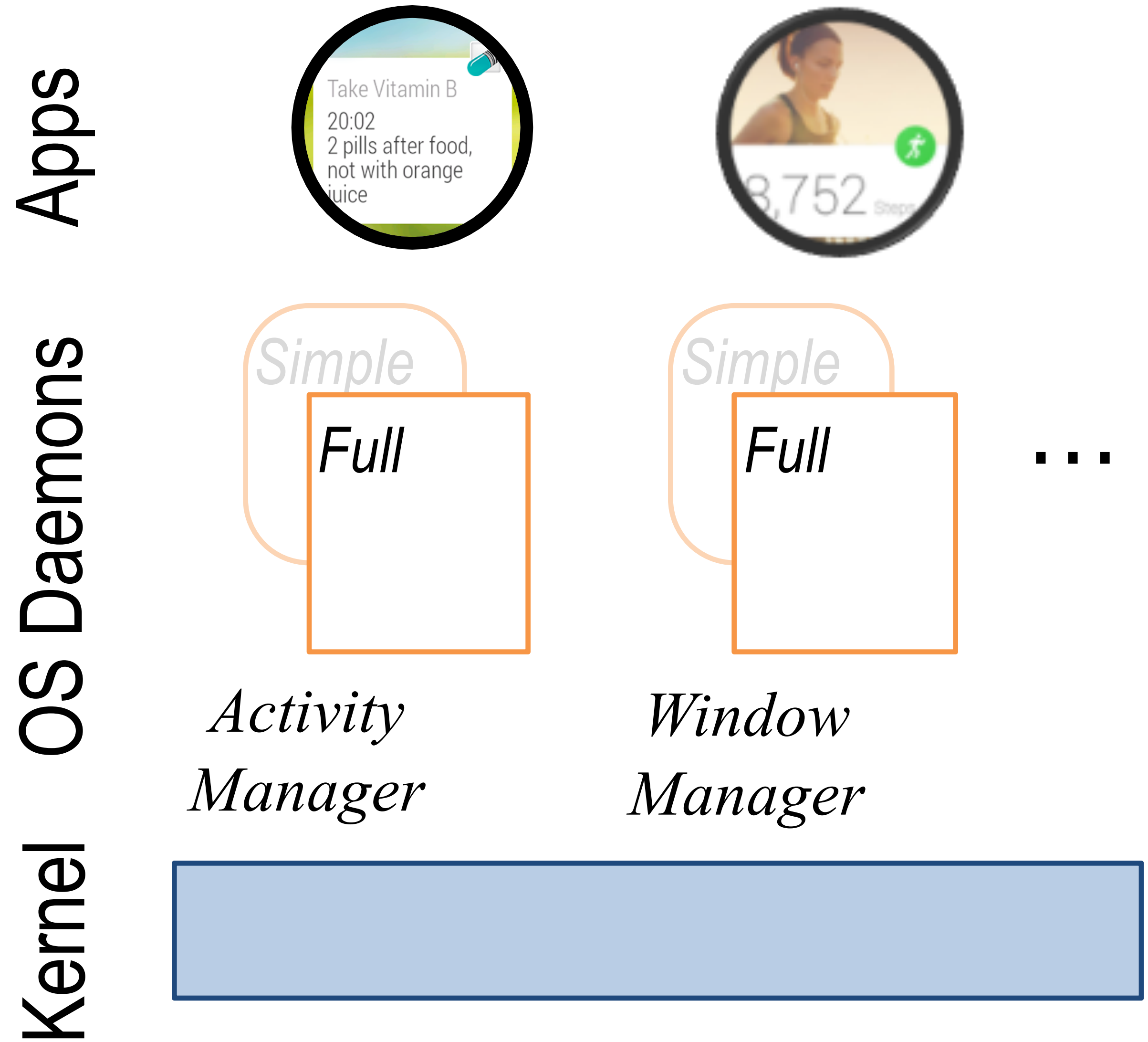
Specializing OS
for common, single-app scenarios



Restructuring OS for Wearable



Restructuring OS for Wearable



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

PURDUE ENGINEERING

Tools, data, and benchmark videos

`xsel.rocks/p/wear`

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?