Distributed Execution of Smartphone Workloads on Loosely Coupled Processors

Felix Xiaozhu Lin, Zhen Wang, and Lin Zhong

Rice University

Goal: energy-proportionality

Smartphones

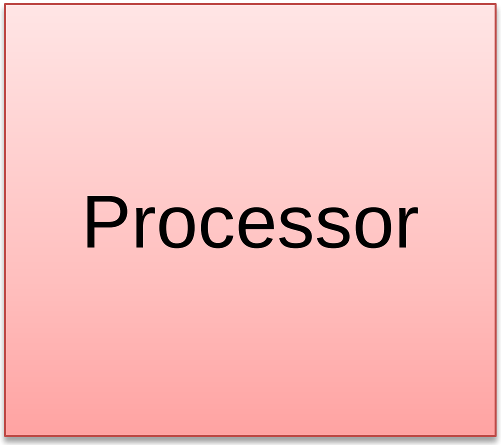
High performance for intensive workloads

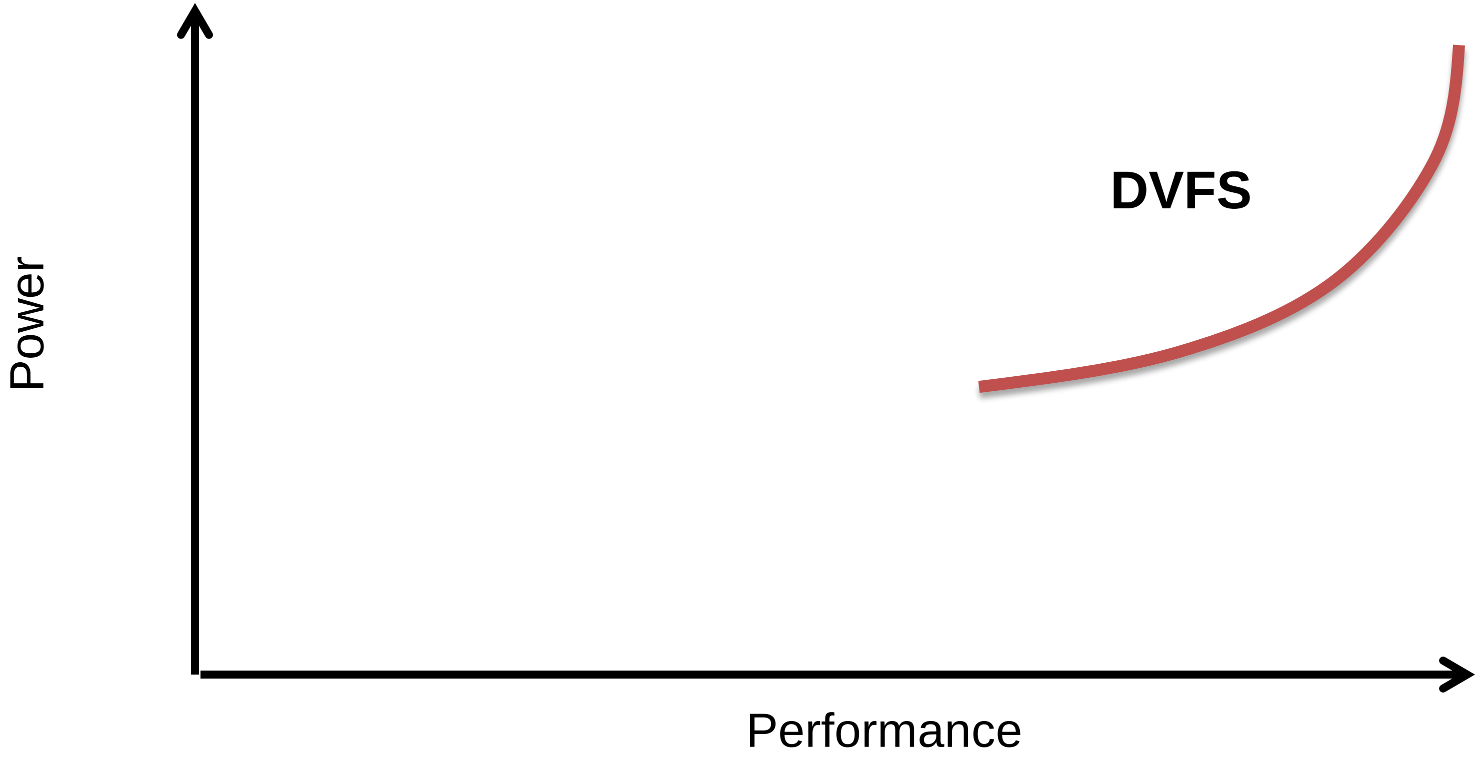
Conserve energy as workloads mitigate

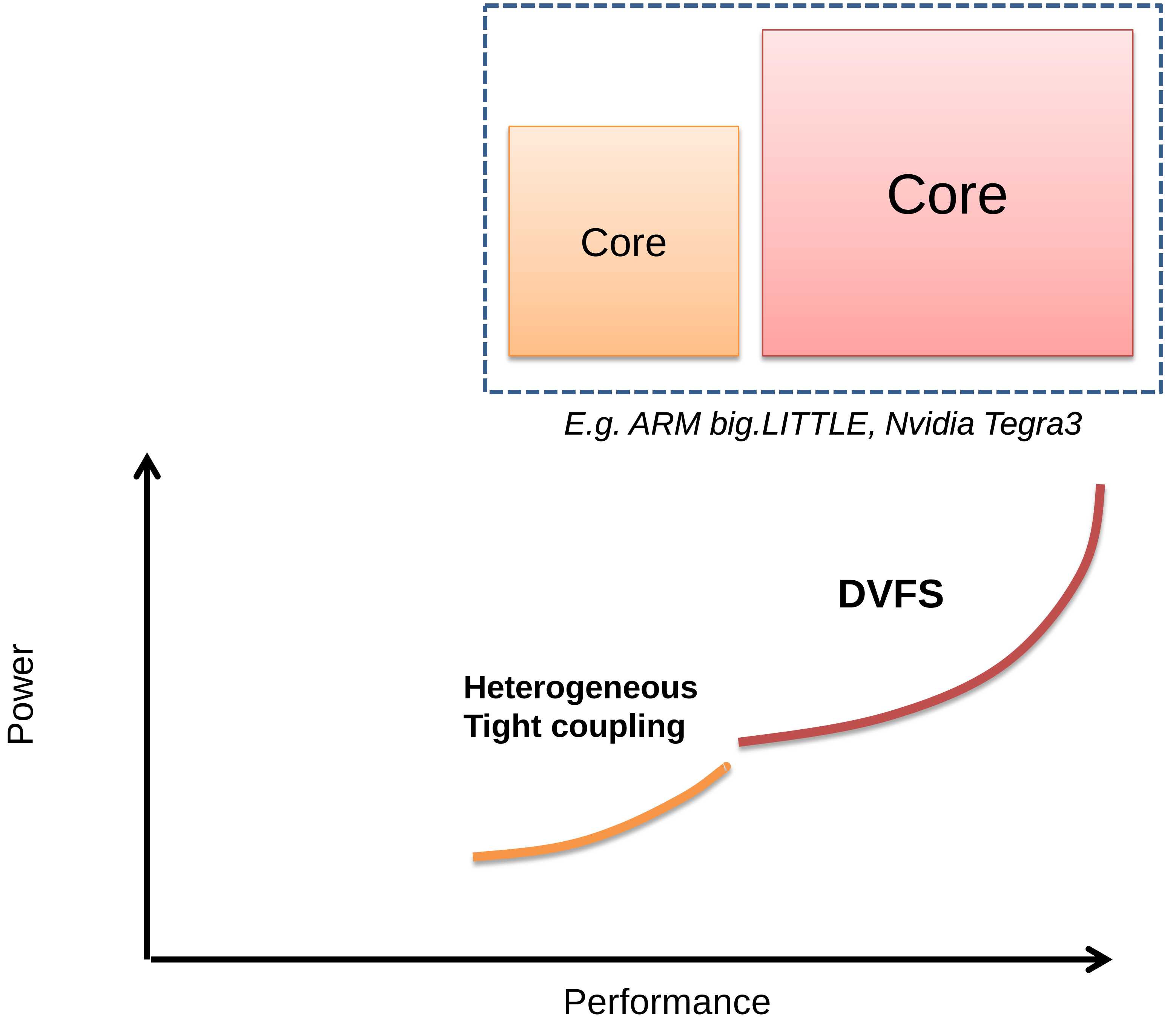
100x

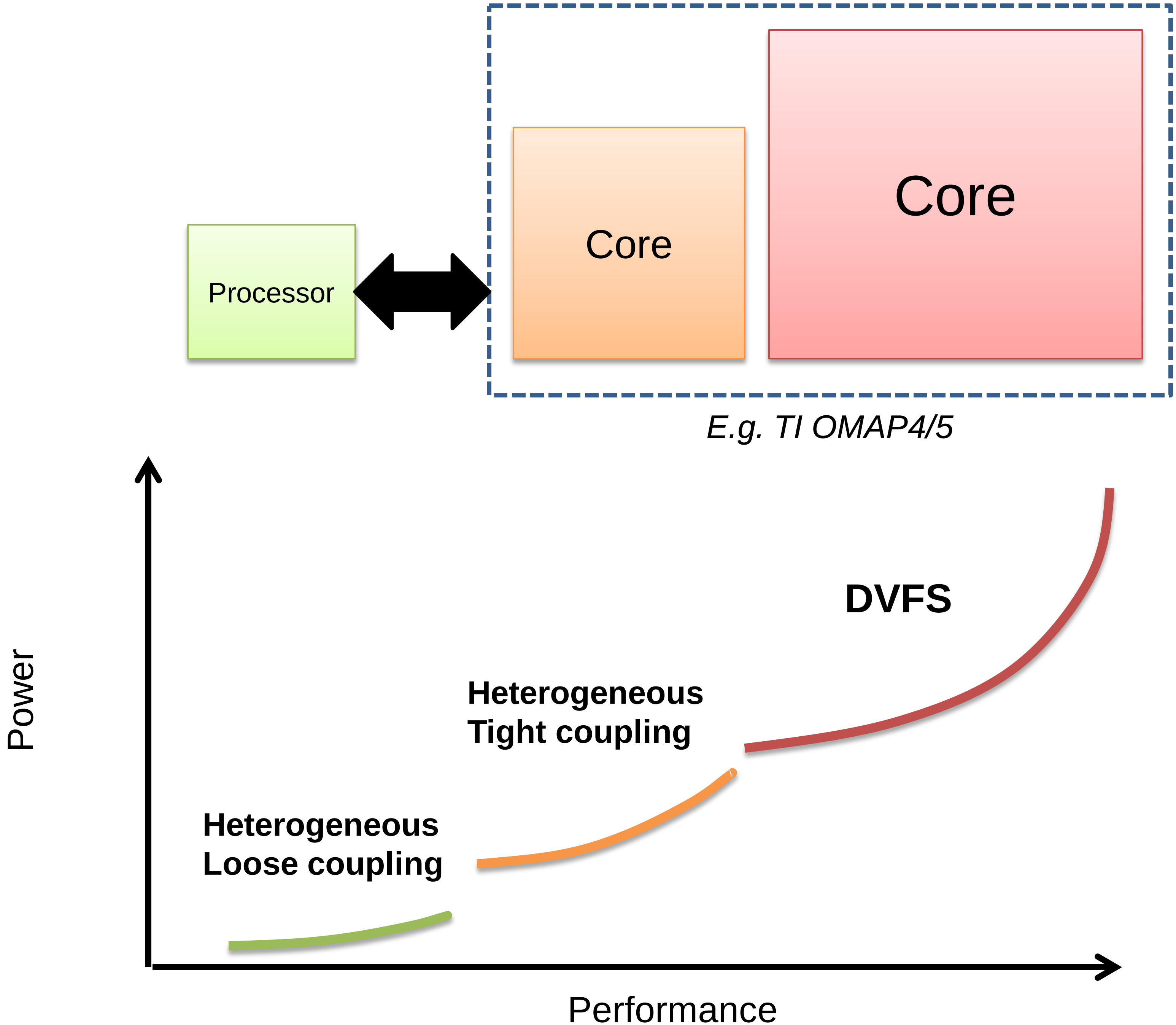
Tens of milli Watt ⟷ A few Watt

Architectural support for energy-proportionality






Core
Core
E.g. ARM big.LITTLE, Nvidia Tegra3
DVFS
Heterogeneous
Tight coupling
Power
Performance



Processor
Core
Core
E.g. TI OMAP4/5
DVFS
Heterogeneous
Tight coupling
Heterogeneous
Loose coupling
Power
Performance

Challenges

- Exploit loosely-coupled processors for smartphone workloads

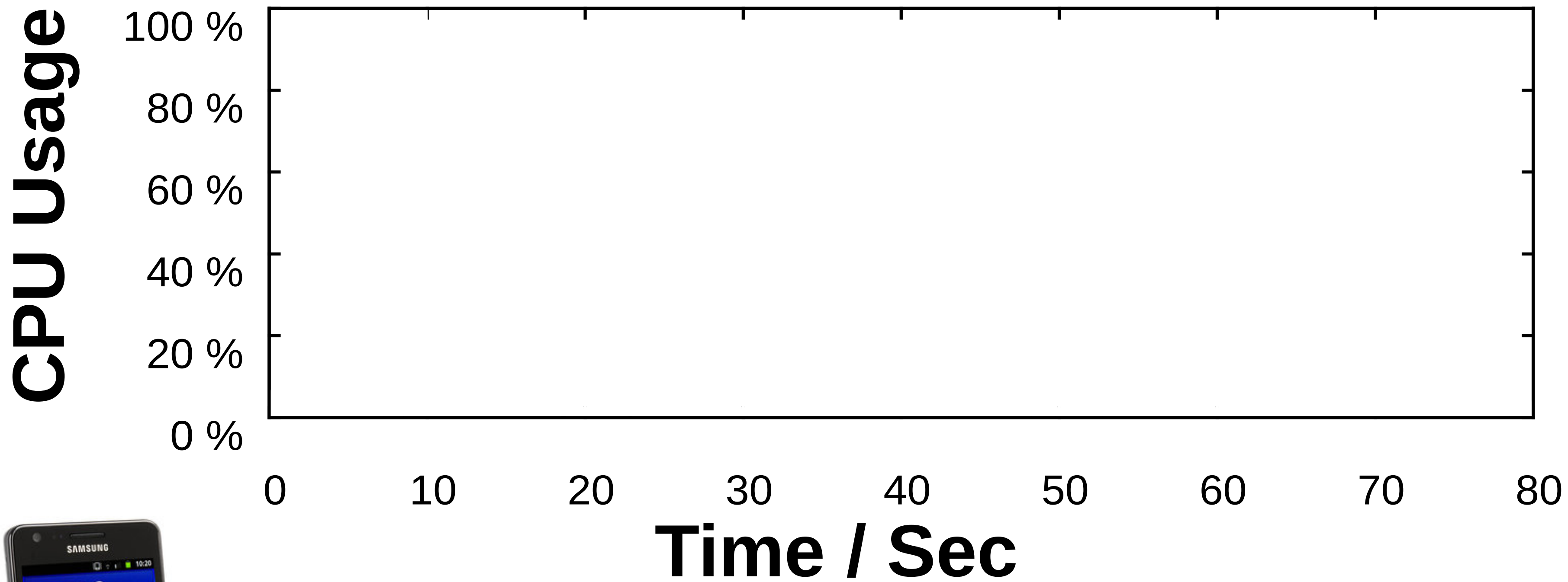
- Programmability
  - 800K apps, by 1 million programmers
  - Numerous legacy code: 1 million lines in Android

Characterize smartphone workloads

Temporal / spatial variations?

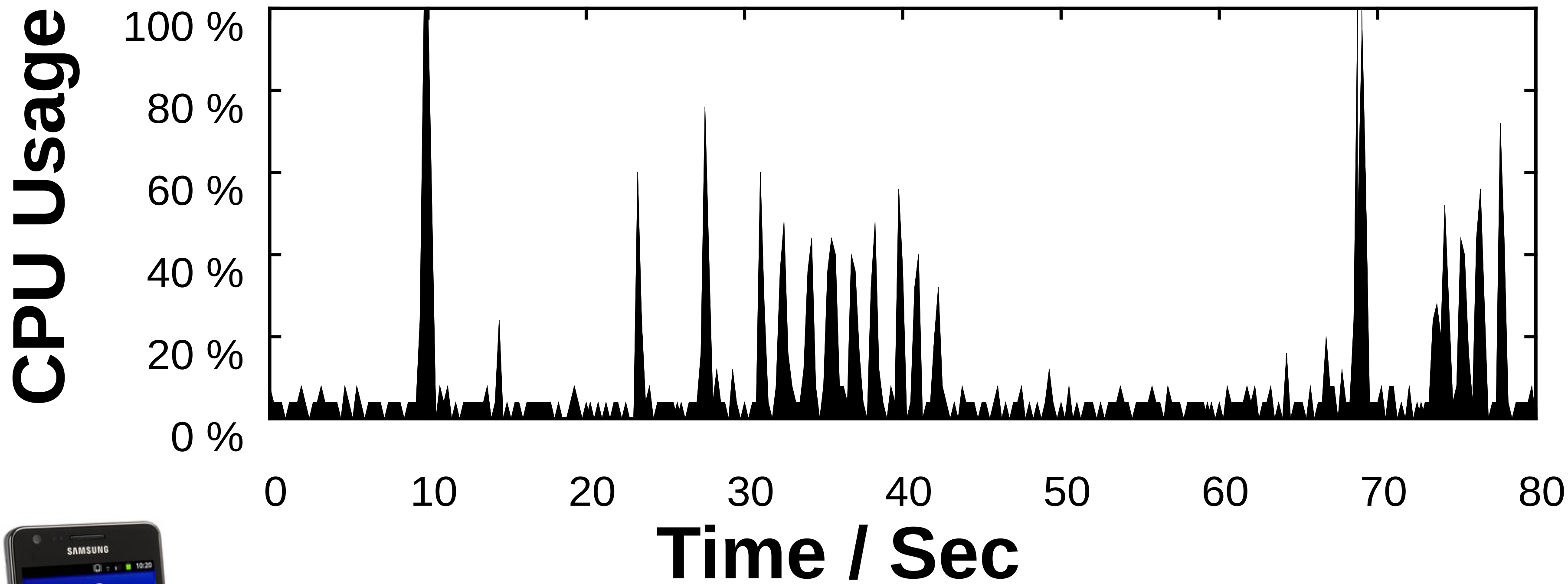
App / OS split?

All experiments done on Samsung Galaxy S2 + ICS


CPU Usage
100 %
80 %
60 %
40 %
20 %
0 %
0
10
20
30
40
50
60
70
80
Time / Sec


SAMSUNG
10:20
Fri.
06/05
Google
Gmail
Talk
Internet
Maps
Phone
Contacts
Messaging
Applications

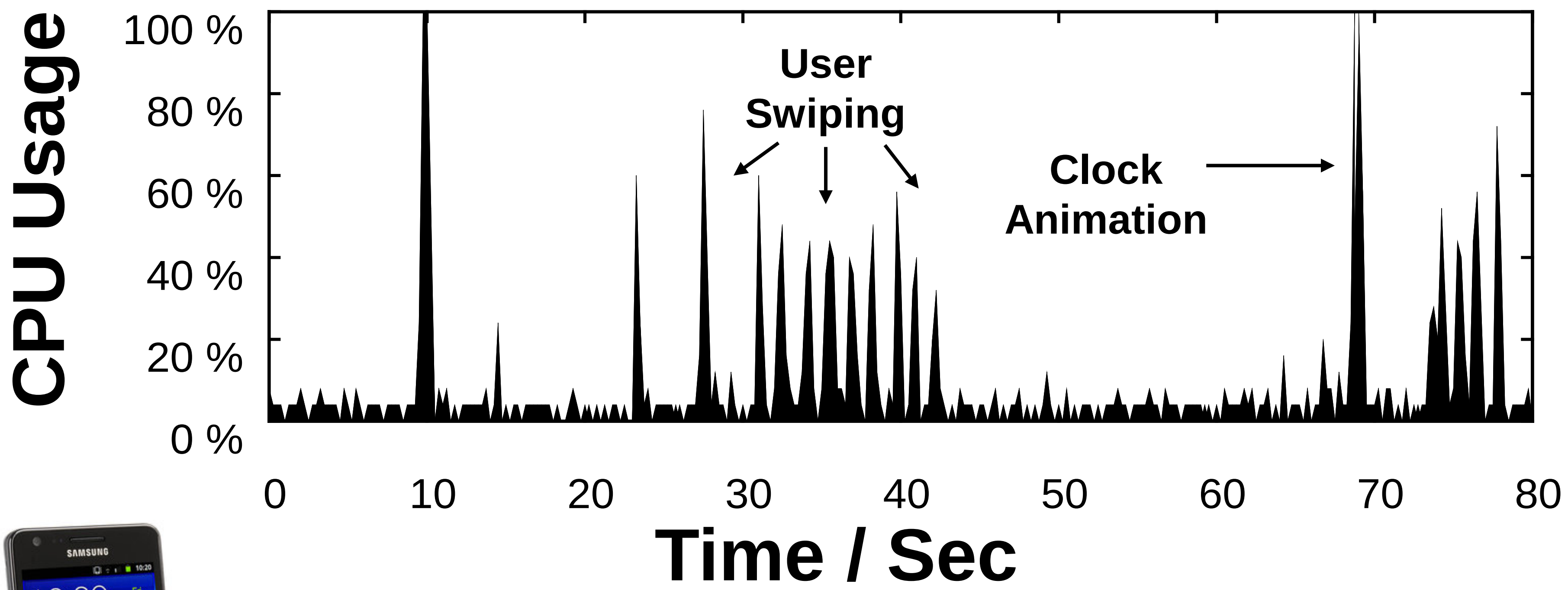
Home-idle



Home-idle

Observation 1

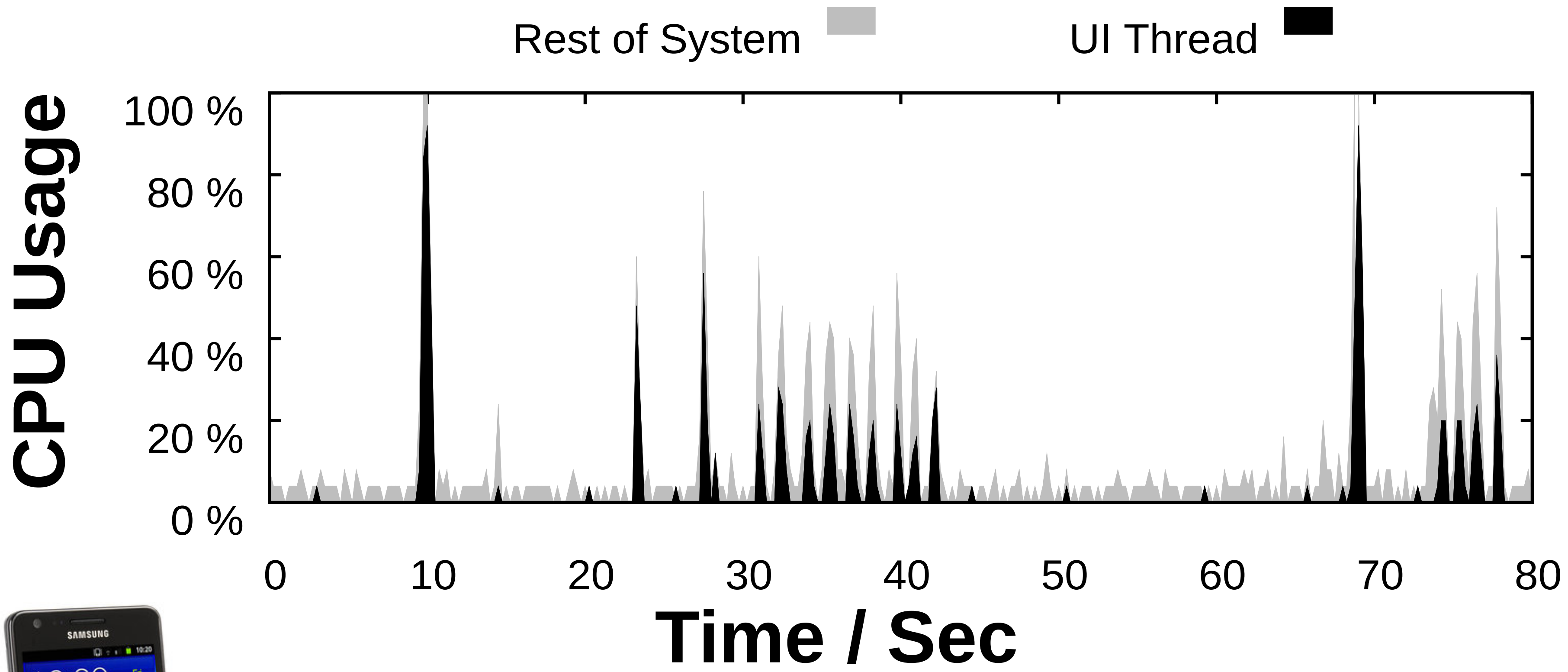
Workloads are time-varying



Home-idle

Observation 2

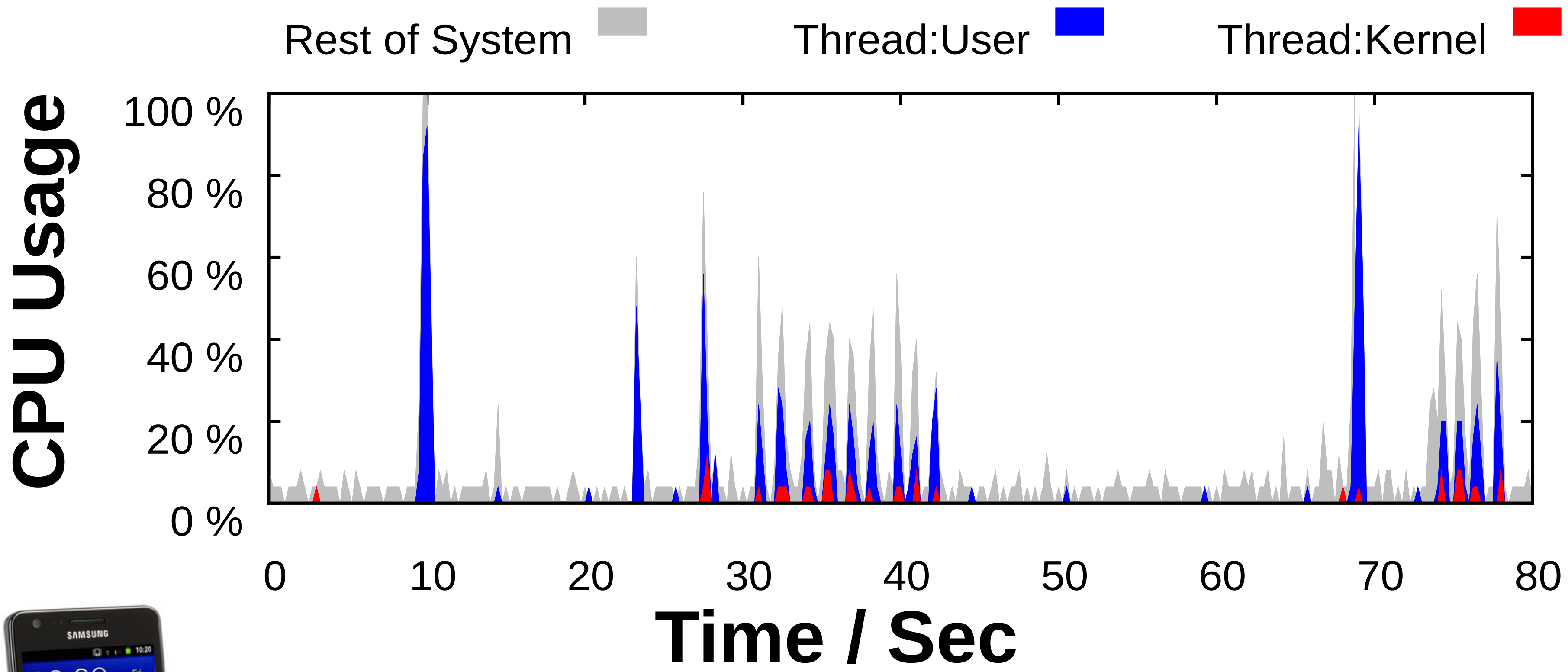
Same threads experience varying workloads



Home-idle

Observation 2

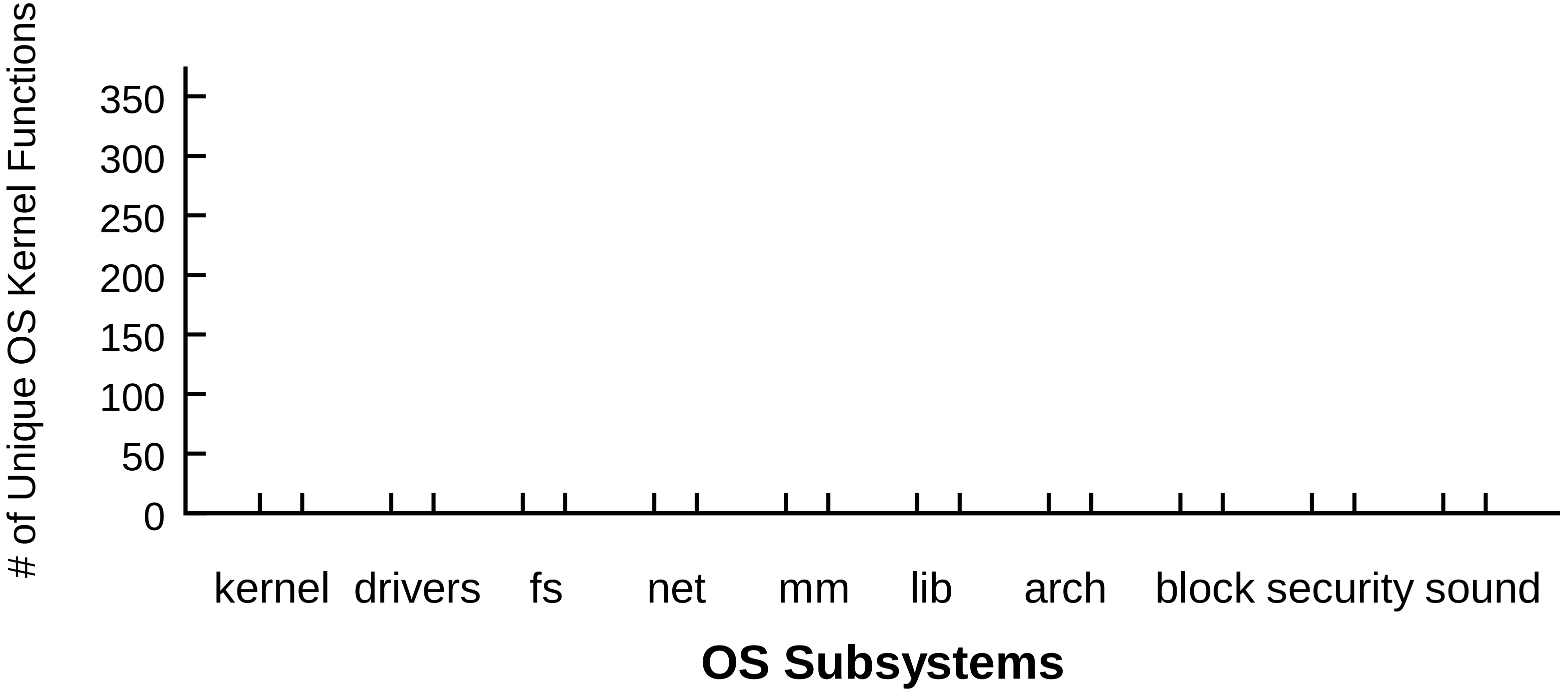
Same threads experience varying workloads



Home-idle

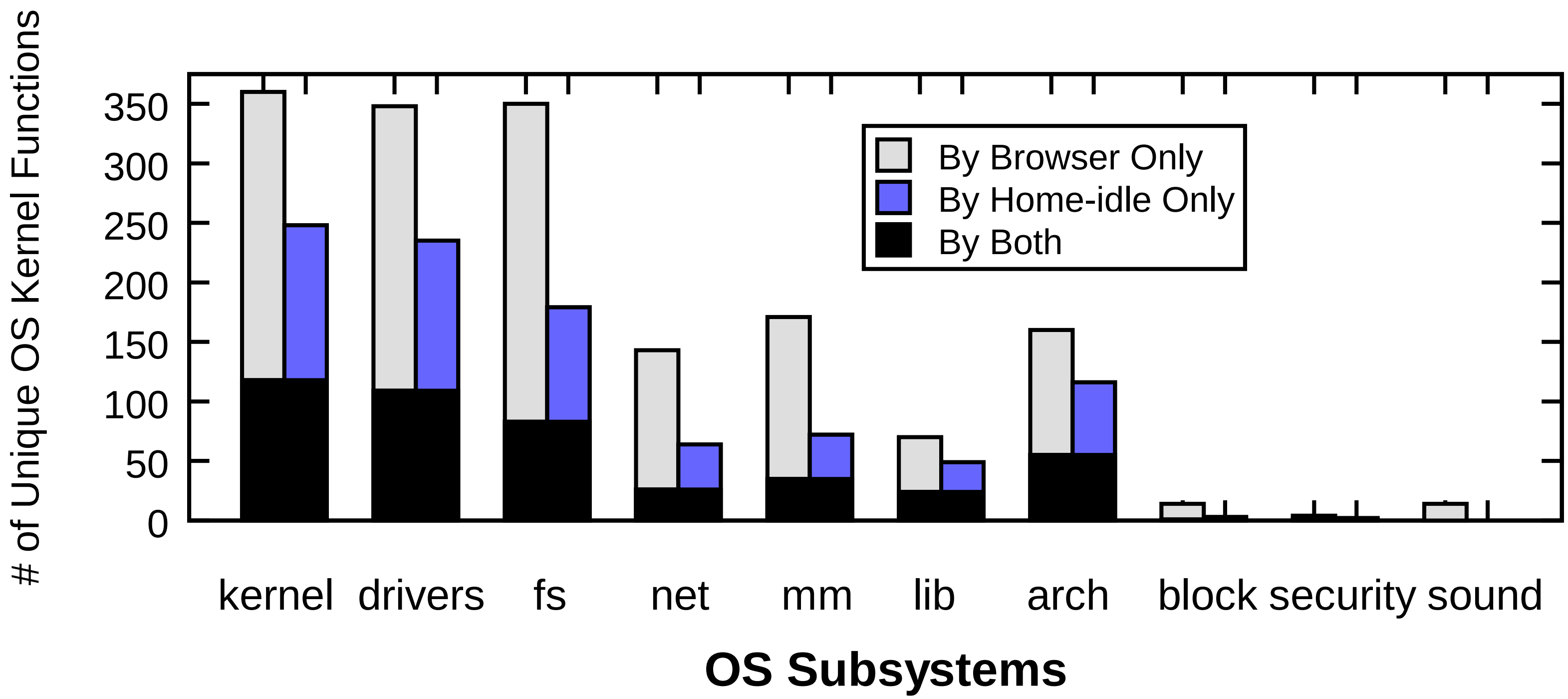
Use of OS Services

***Home-idle* (light)** **VS** ***Browser* (heavy)**



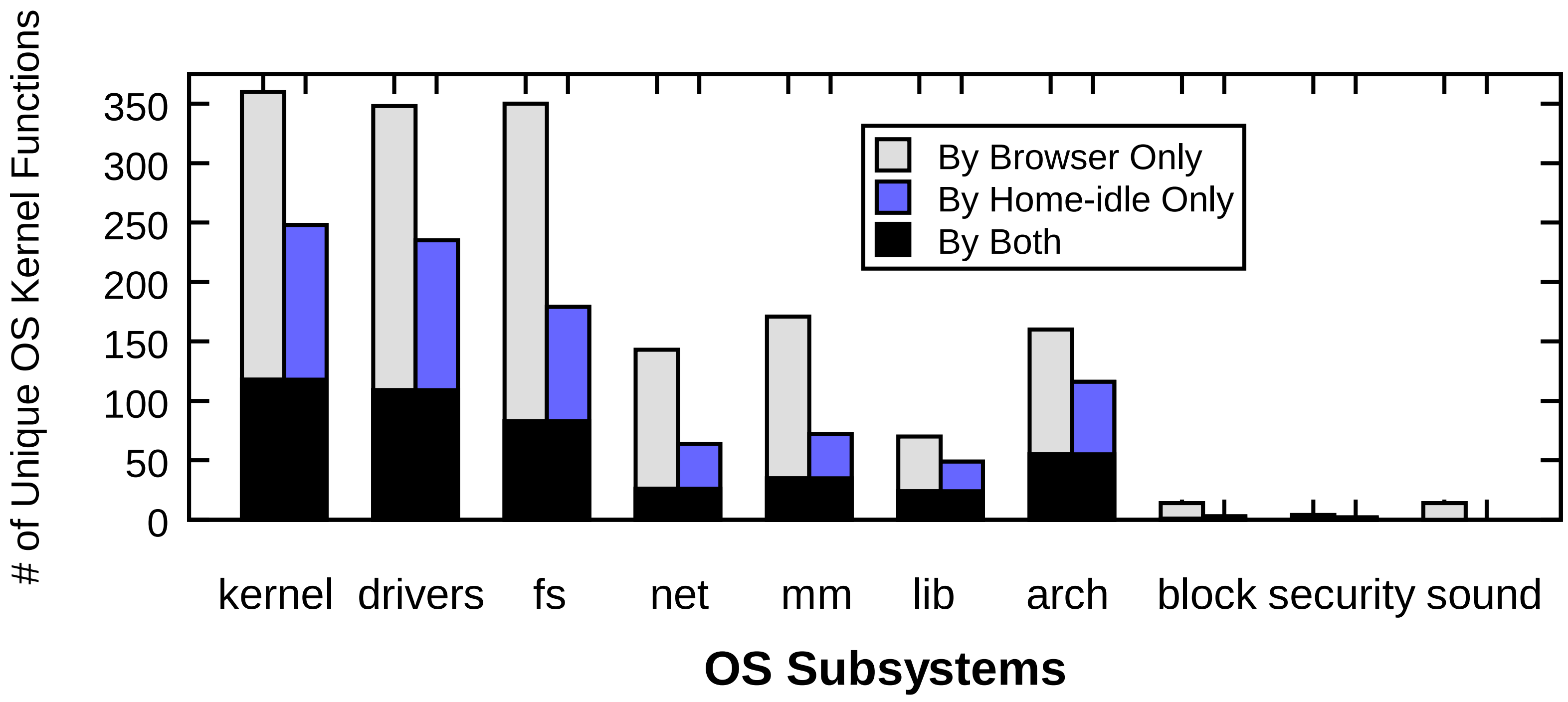
# of Unique OS Kernel Functions
350
300
250
200
150
100
50
0
kernel
drivers
fs
net
mm
lib
arch
block
security
sound
OS Subsystems

Browser : Heavy
Home-idle: Light



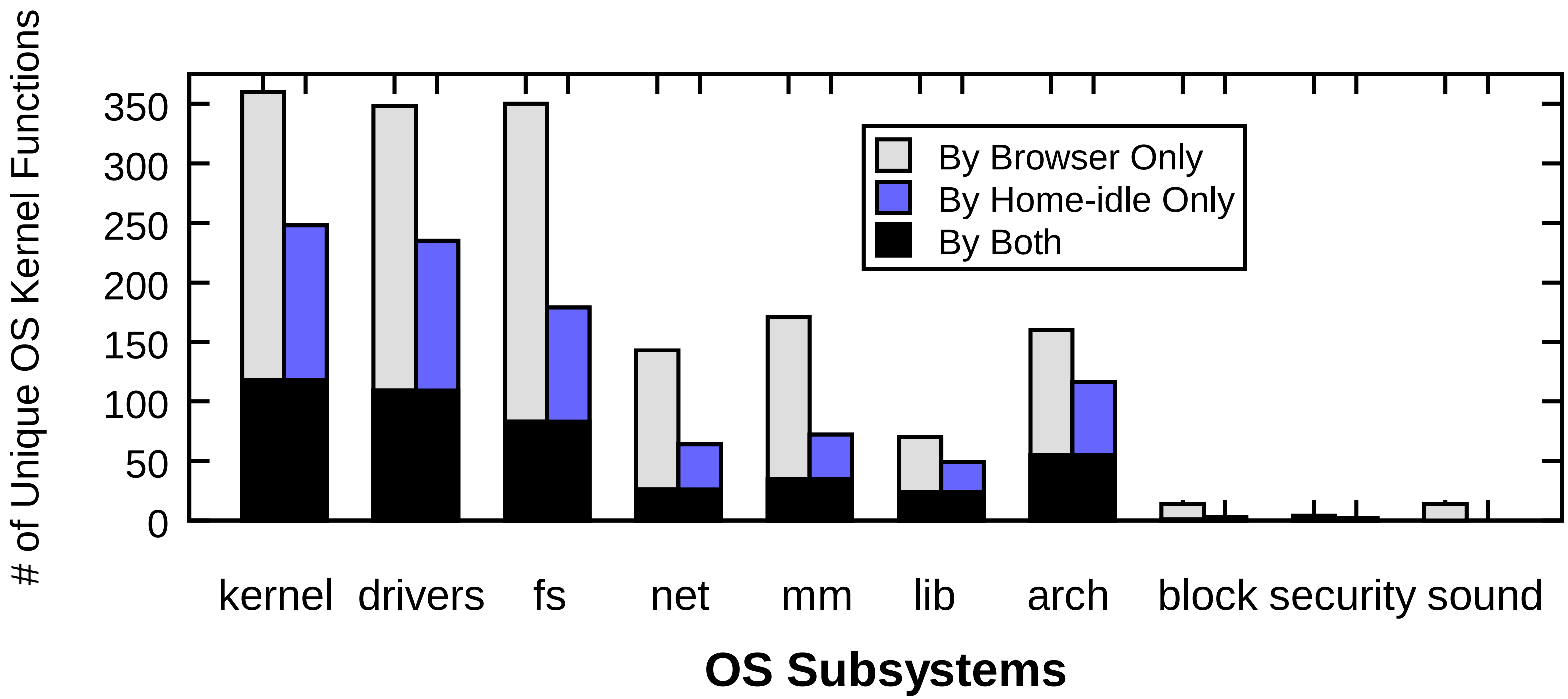
Observation 3

Same OS functions experience varying workloads



Observation 4

Workloads invoke diverse OS functions

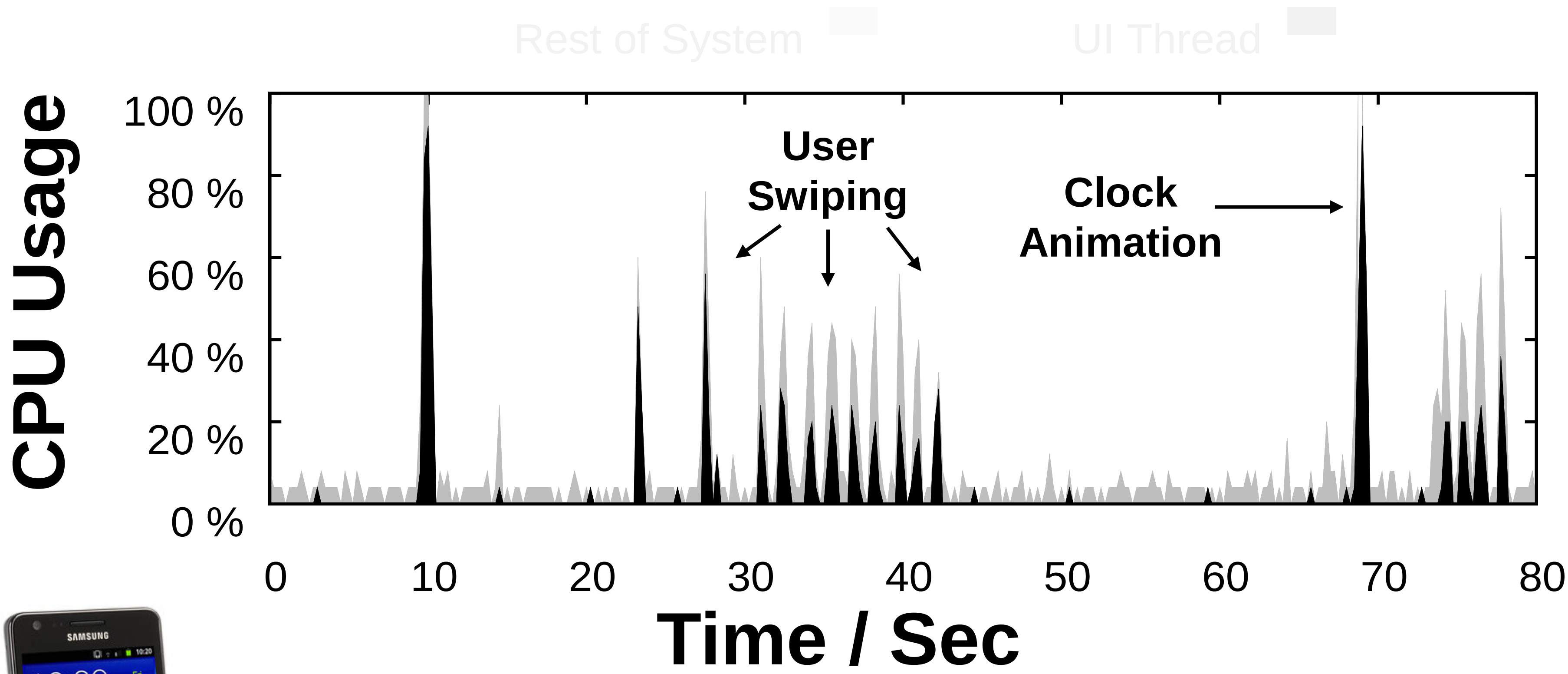


Implications: for user threads

– **Ob1: Workloads are time-varying**
– **Ob2: Same threads experience varying workloads**

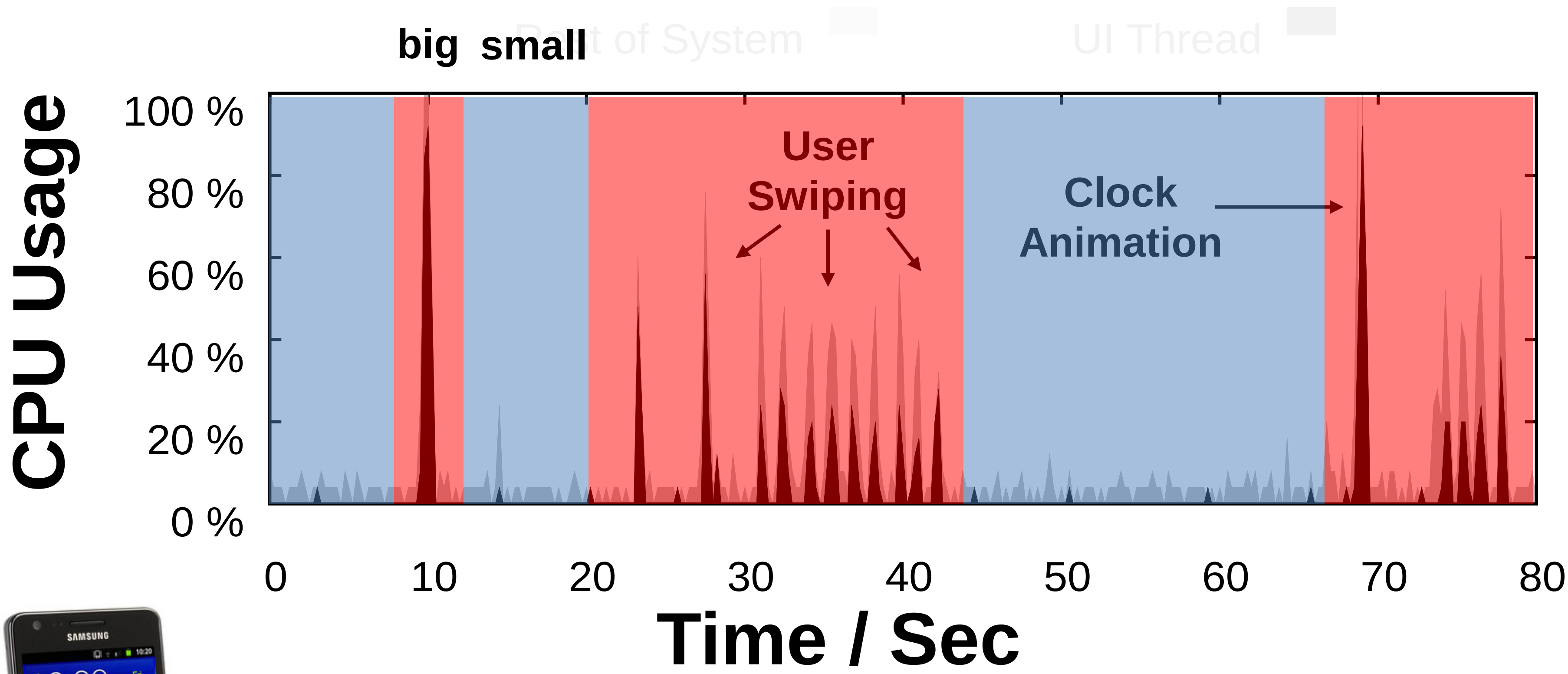
➢ Enable dynamic use of different processors
➢ Retain existing program structure

Example: UI thread of Home



Home-idle

Example: UI thread of Home



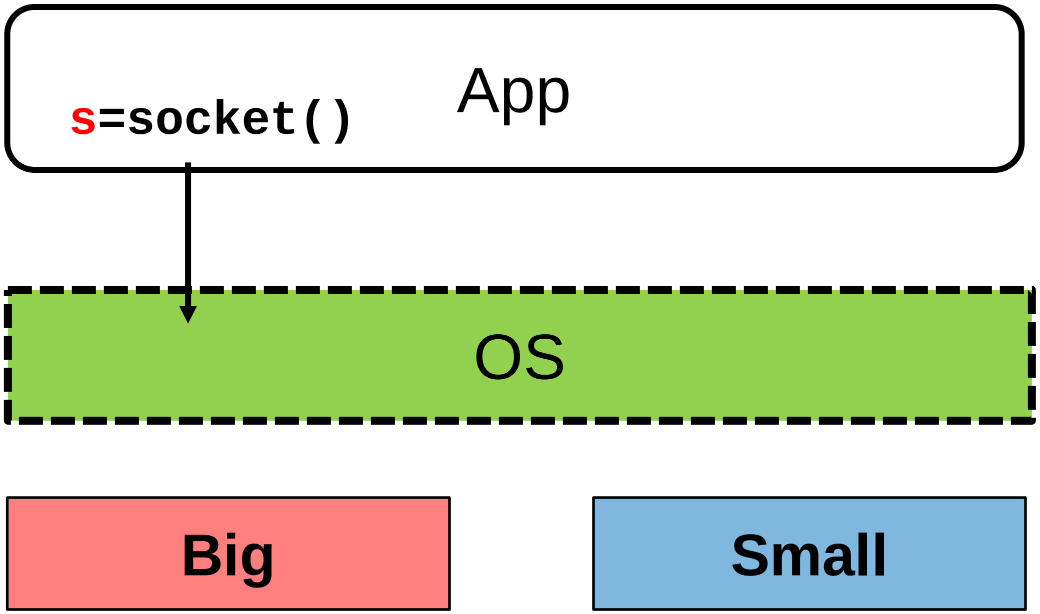
Home-idle

Implications: for OS functions

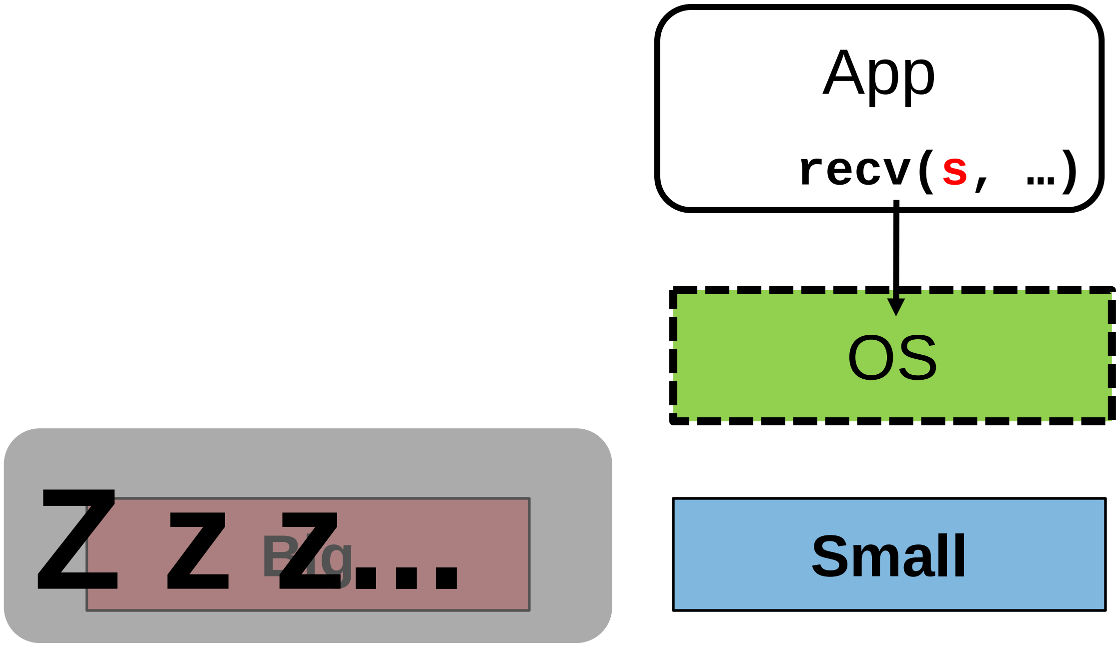
– **Ob3: Same OS functions experience varying workloads**
– **Ob4: Workloads invoke diverse OS functions**

- ➢ Execute same OS services on multiple processors
- ➢ Preserve a single system image

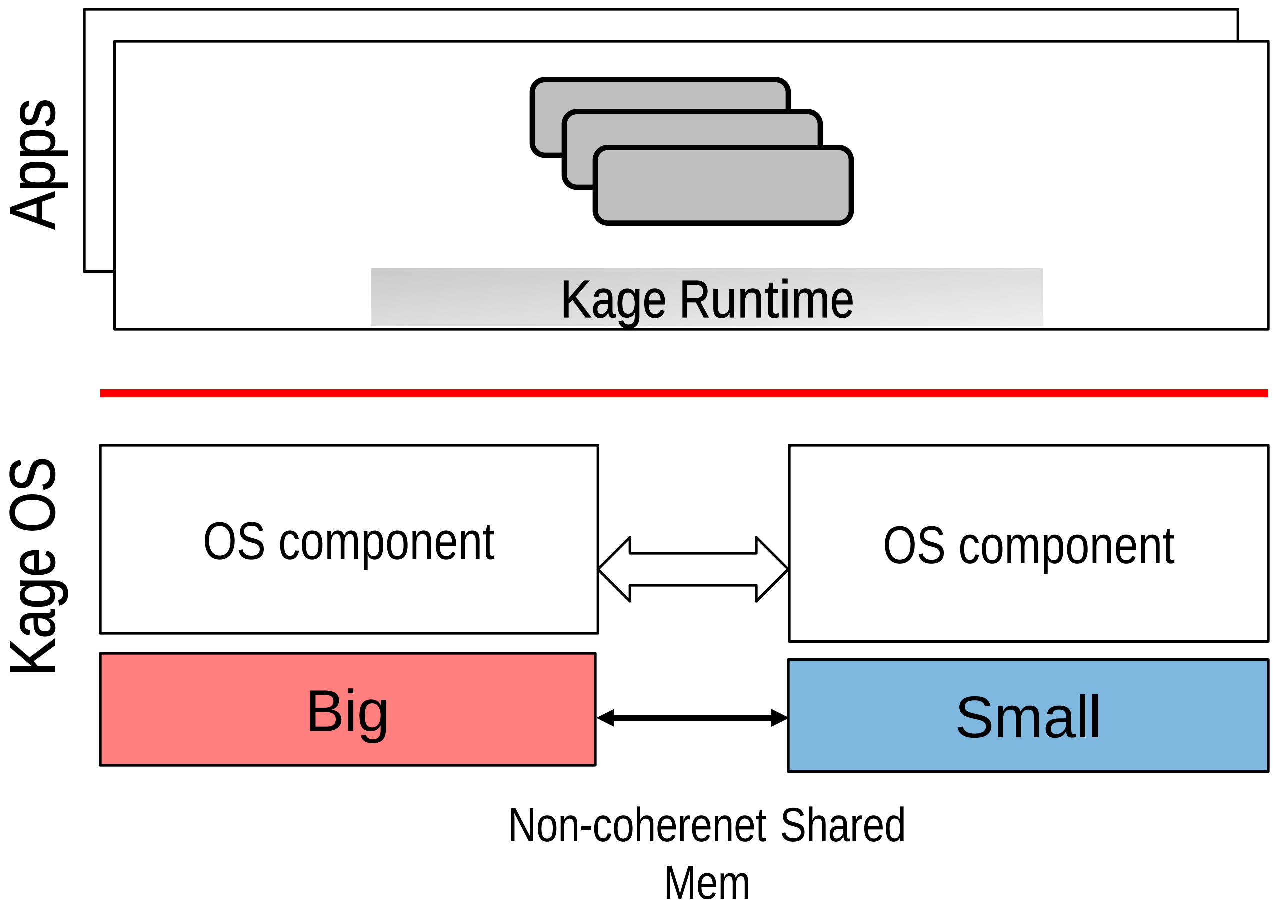
Example: Instant Messenger



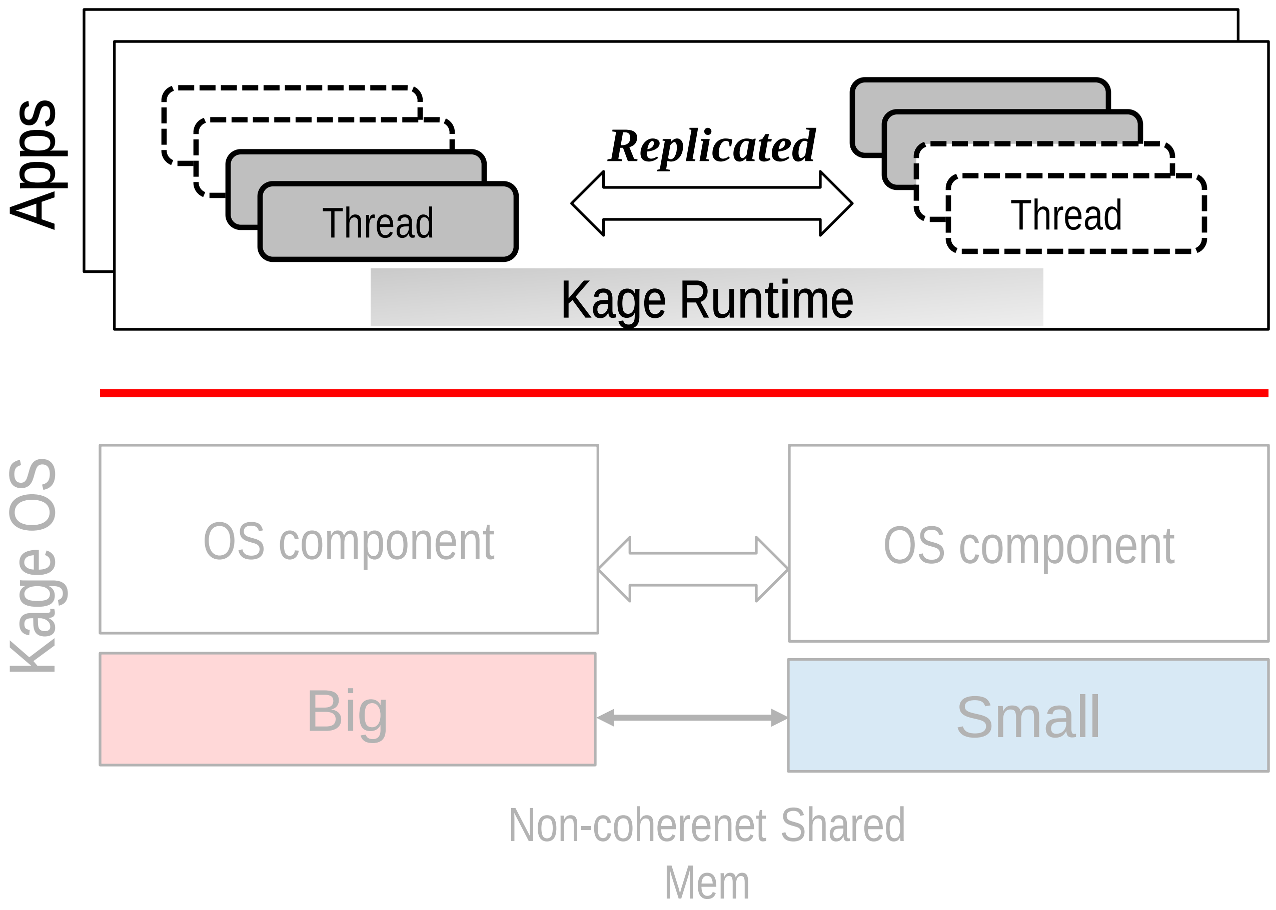
Example: Instant Messenger



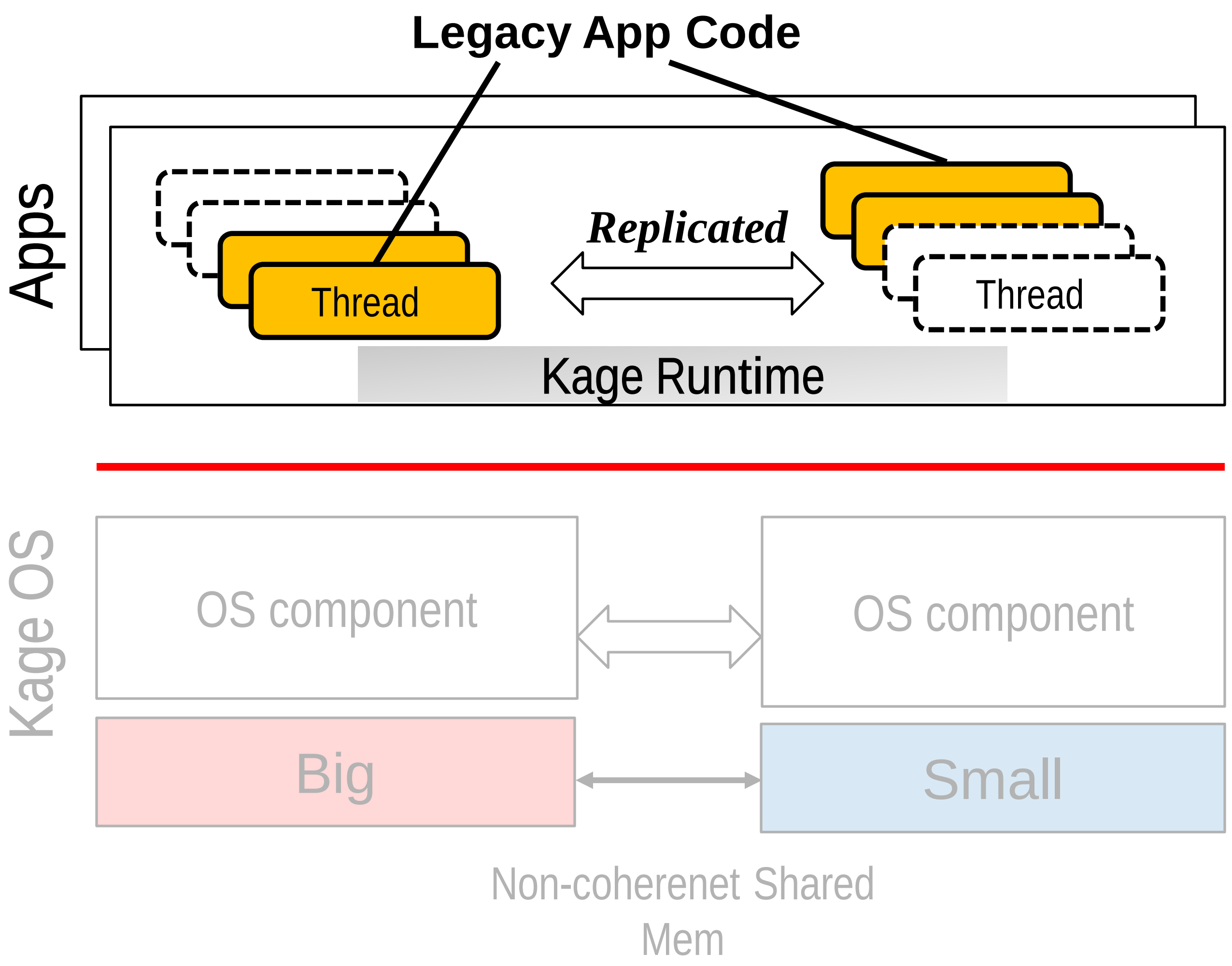
Our solution: Kage runtime lib + OS



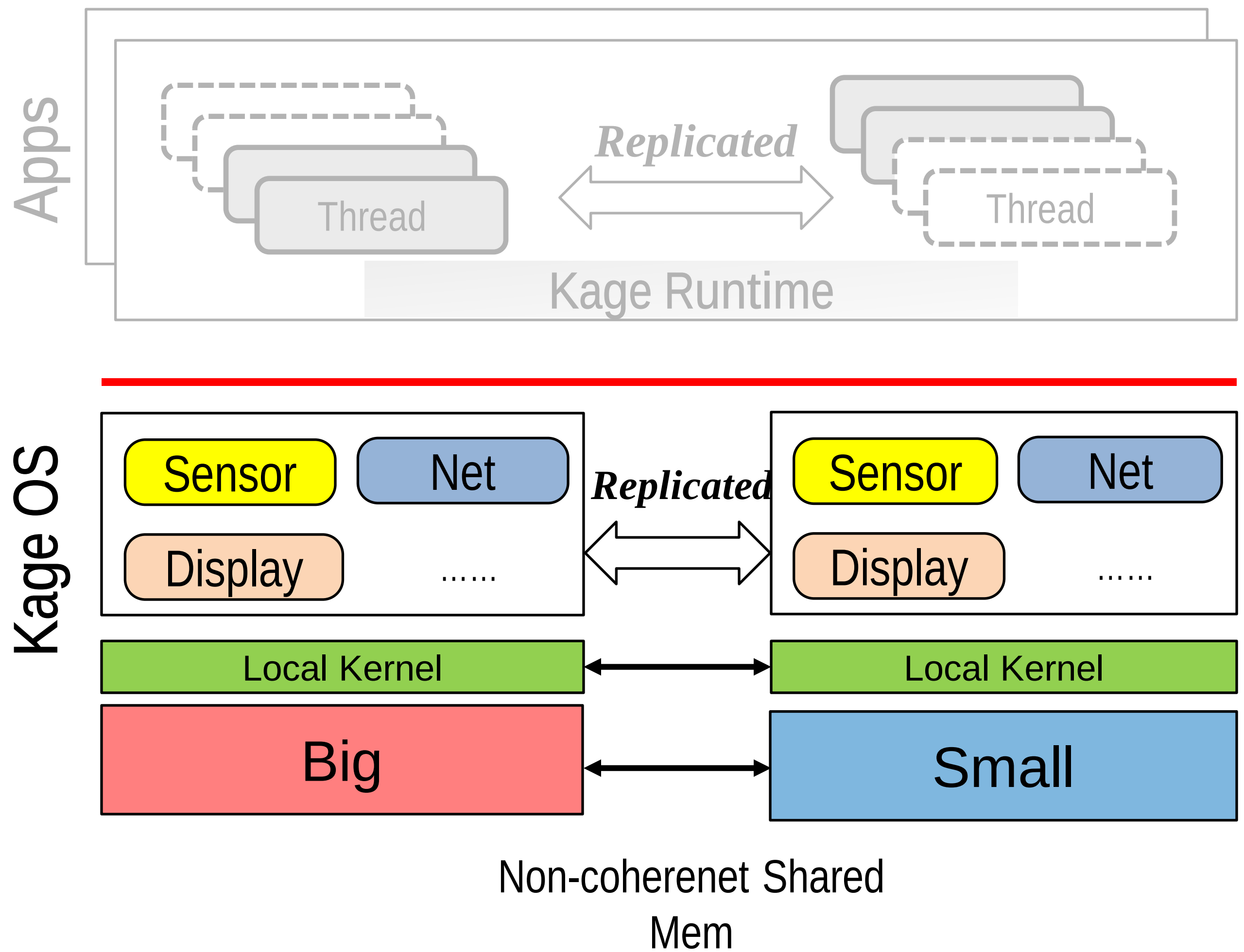
Kage: runtime lib



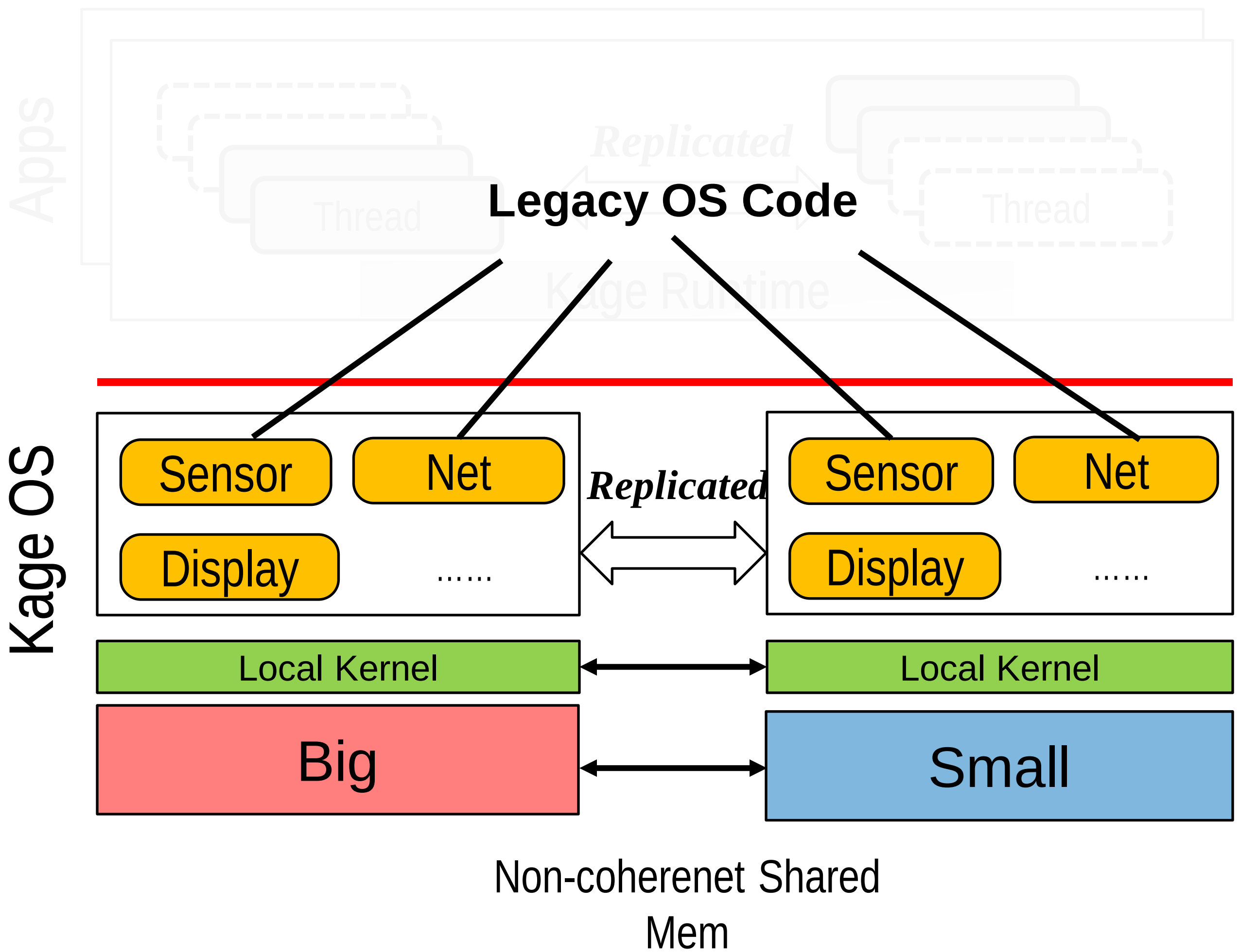
Kage: runtime lib



Kage: OS structure



Kage: OS structure



Open questions and plans

- State consistency
- Limiting inter-processor concurrency

- Implementation: TI OMAP4 (two types of processors)
- Two local kernels with separate memory allocators

Related Work

Heterogeneity with HW cache-coherence

- big.LITTLE

OS with partitioned functions

- Helios, Barrelfish (for scalability)

OS with replicated functions

- V, fos

Summary

Goal: energy proportionality for smartphone workloads

Observations:

– Architecture: loosely coupling, asymmetry
– Workloads: high variations, in both app and OS

Challenges:

– Exploiting the architecture for smartphone workloads
– Maximizing reuse of legacy code

Solution: Kage, a suite of runtime lib and OS

– Replicating user thread execution
– Replicating OS functions