Cellular Networks and Mobile Computing COMS 6998-8, Spring 2012

Instructor: Li Erran Li
(lel2139@columbia.edu)

http://www.cs.columbia.edu/~coms6998-8/

4/2/2012: Mobile Cloud Computing

Mobile Cloud Computing (mCloud) today

- Apple iCloud
  - Store content in cloud and sync to all registered devices
  - Hosted by Windows Azure and Amazon AWS
  - iCloud Storage APIs support third-party app document syncing

Mobile Cloud Computing today (Cont'd)

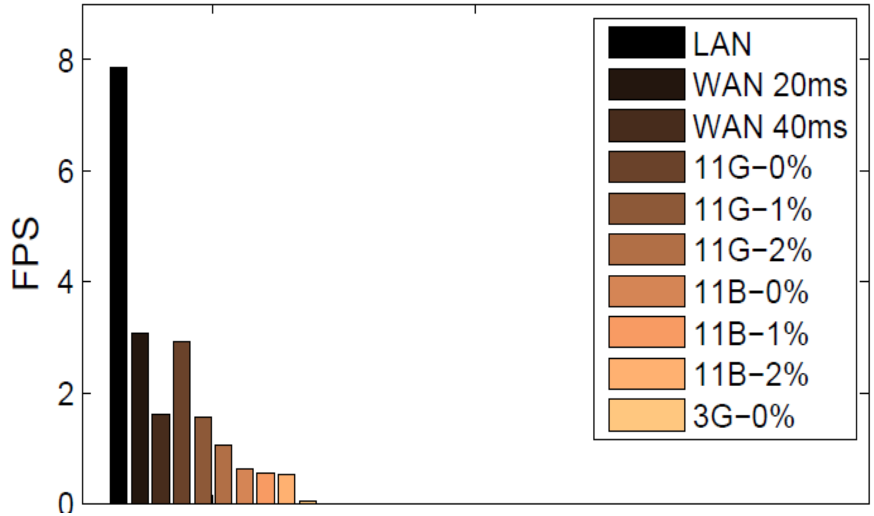
- Amazon Silk browser
  - Accelerates web access
  - Learns user behavior and precache
  - Intelligently partition work between local and Amazon cloud

mCloud Fundamental Challenges

- What architecture best supports mCloud?
- What programming model best enables client to tap mCloud resources?
- What are basic services or building blocks for mCloud?
- What best supports service interaction?

mCloud Architecture

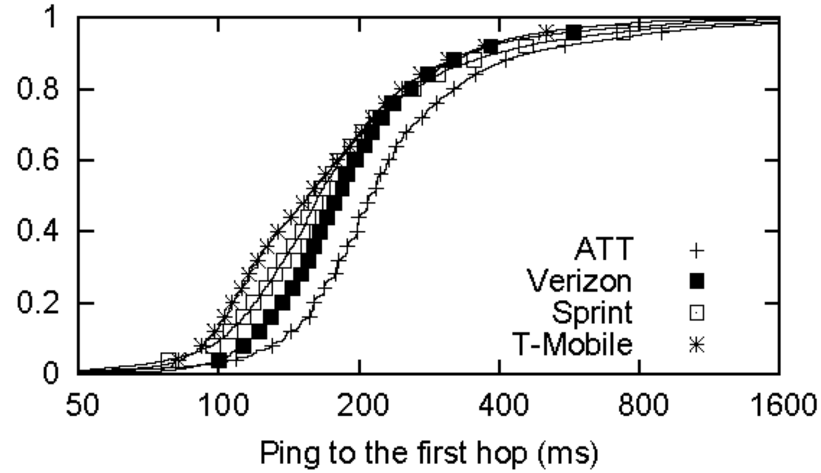
- Resource-intensive mobile applications
  - Face recognition for social networking app
  - Gesture recognition for control media app
  - Object and post recognition for augmented reality app
- End-to-end latency and througput matters for crisp interaction
  - Augmented reality need to display results within 1 sec
  - Need high data rate processing capability, low frame rate can miss gesture



Delay, loss on frame rate of video stream transfer [Odessa, 2011]

mCloud Architecture (Cont'd)

- WAN performance
  - First hop latency in 3G is 200ms
  - Verizon LTE :128ms, 6.44 Mbps downlink, 5Mbps uplink [Pcworld, March 2011]
- There is a need for a middle tier
- cloudlet = (compute cluster + wireless access point + wired Internet access + no battery limitations)
- → *"data center in a box"*

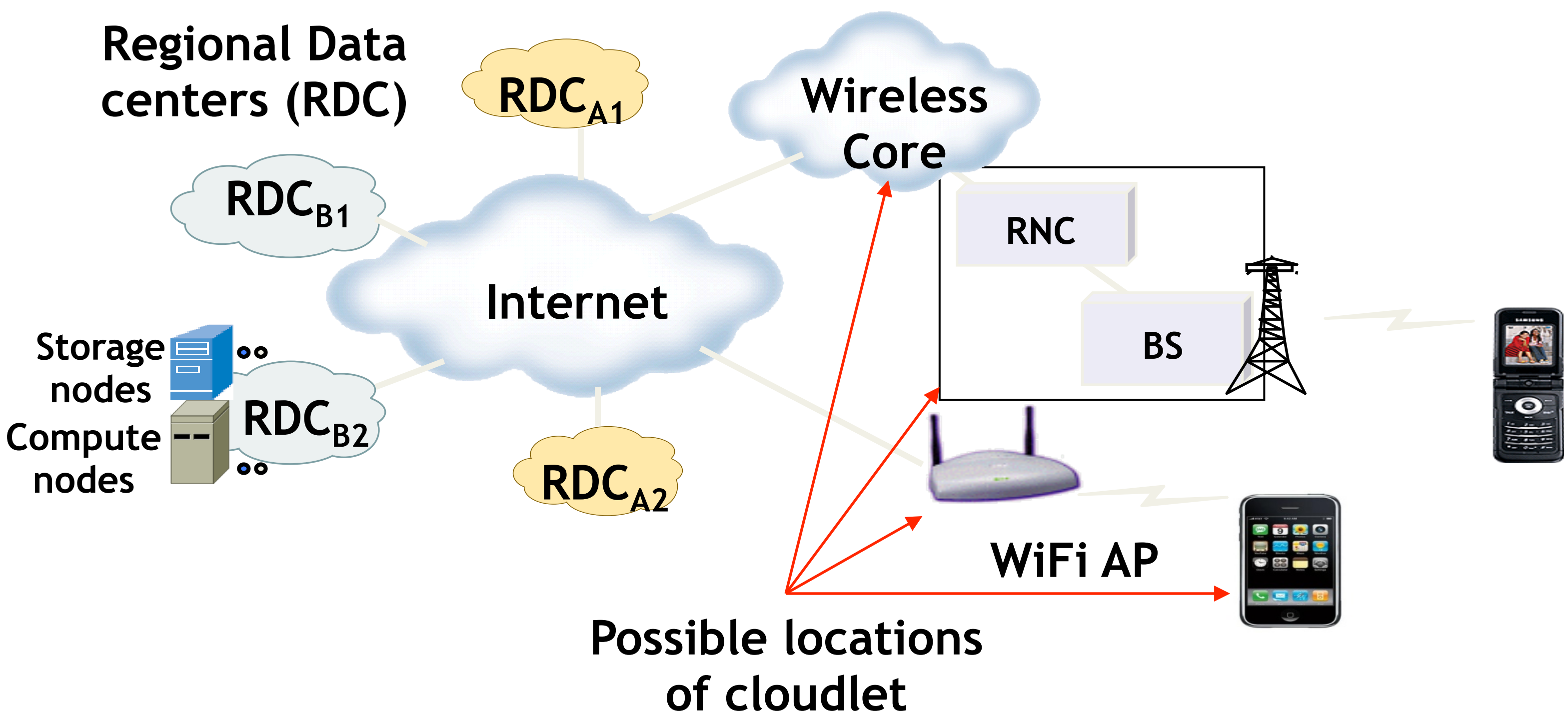


(f) CDF of Ping latency to the first hop

"Anatomizing Application Performance Differences on Smartphones", MobiSys 2010 (Huang et al)

mCloud Architecture (Cont'd)

- Cloudlet possible locations
- Cellular providers has a unique advantage



mCloud Programming Model

- MAUI: RPC based offloading architecture
- CloneCloud: tight synchronization between cloud and phone
- Odessa: data-flow graph to exploit parallelism in perception applications
- Orelans: a new programming model based on grains
- MAUI, CloneCloud , Odessa all have profiler, solver

	MAUI	CloneCloud	Odessa	Orleans
Remote execution unit	Methods (RMI)	Threads	Tasks	Grains

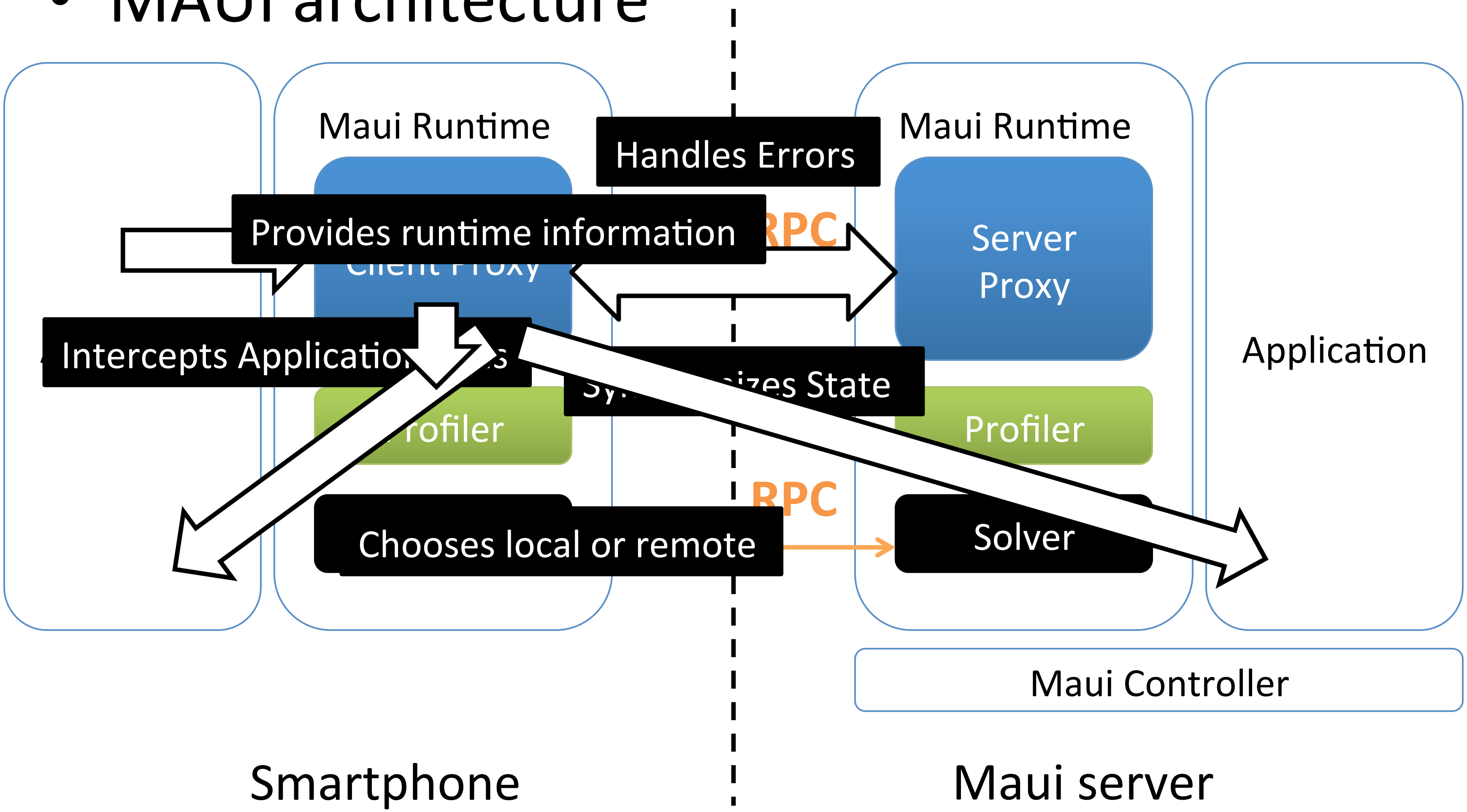
mCloud Programming Model: MAUI

- Combine extensive profiling with an ILP solver
  - Makes dynamic offload decisions
  - Optimize for energy reduction
  - Profile: device, network, application

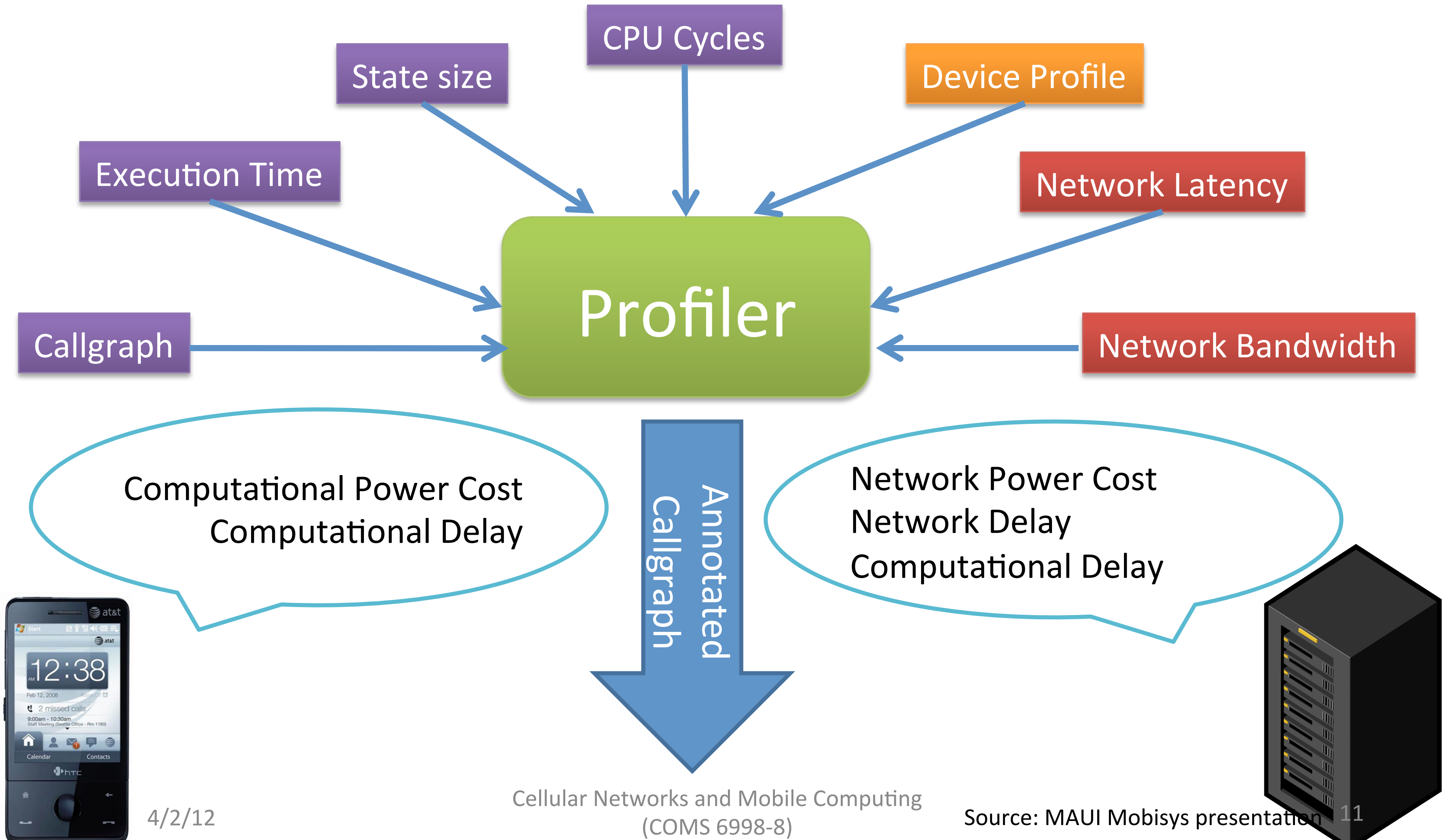
- Leverage modern language runtime (.NET CLR)
  - Portability:
    - Mobile (ARM) vs Server (x86)
    - .NET Framework Common Intermediate Language
  - Type-Safety and Serialization:
    - Automate state extraction
  - Reflection:
    - Identifies methods with [Remoteable] tag
    - Automates generation of RPC stubs

mCloud Programming Model: MAUI (Cont'd)

- MAUI architecture



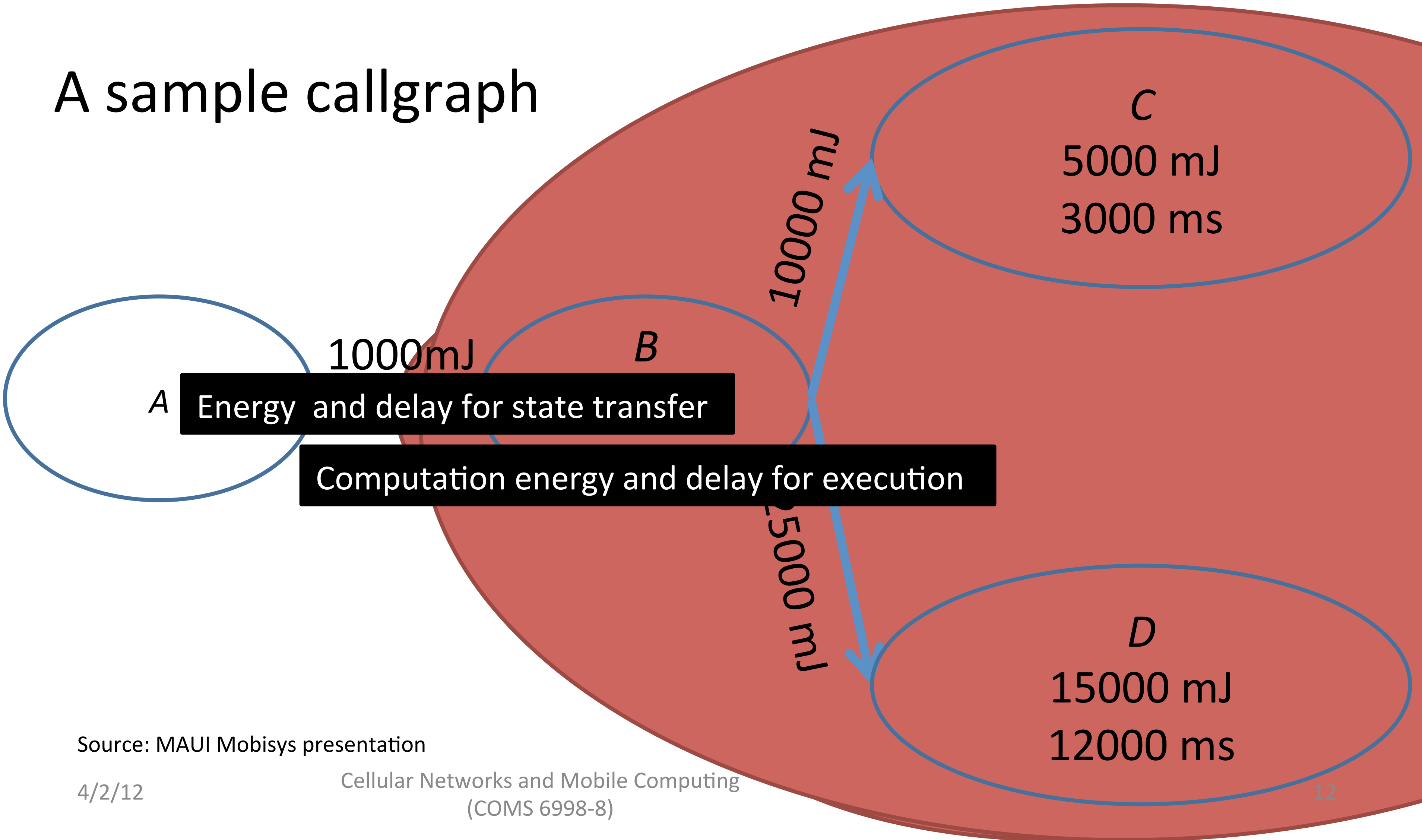
MAUI Profiler



Source: MAUI Mobisys presentation

MAUI Solver

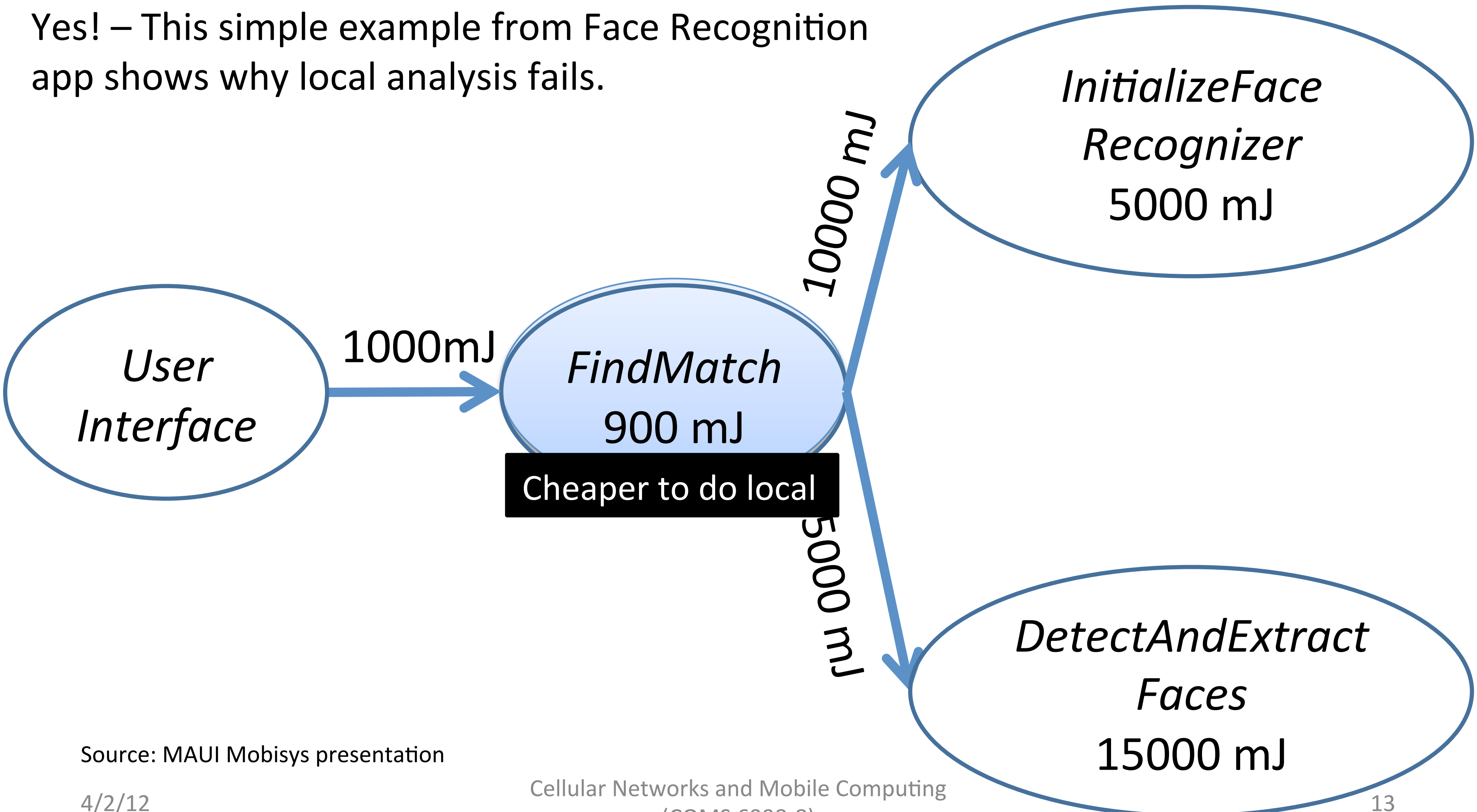
A sample callgraph



Source: MAUI Mobisys presentation

Is Global Program Analysis Needed?

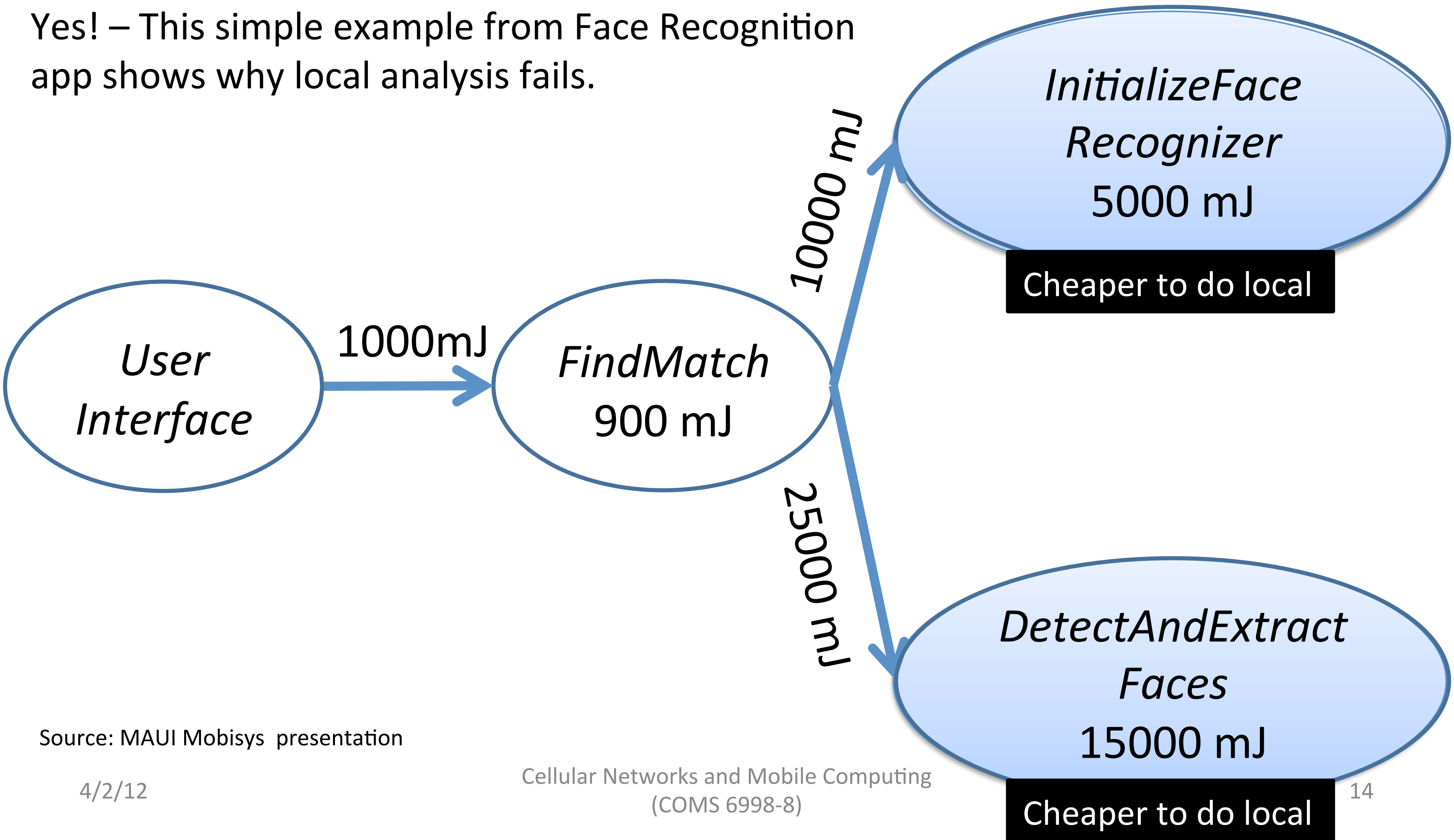
Yes! – This simple example from Face Recognition app shows why local analysis fails.



Source: MAUI Mobisys presentation

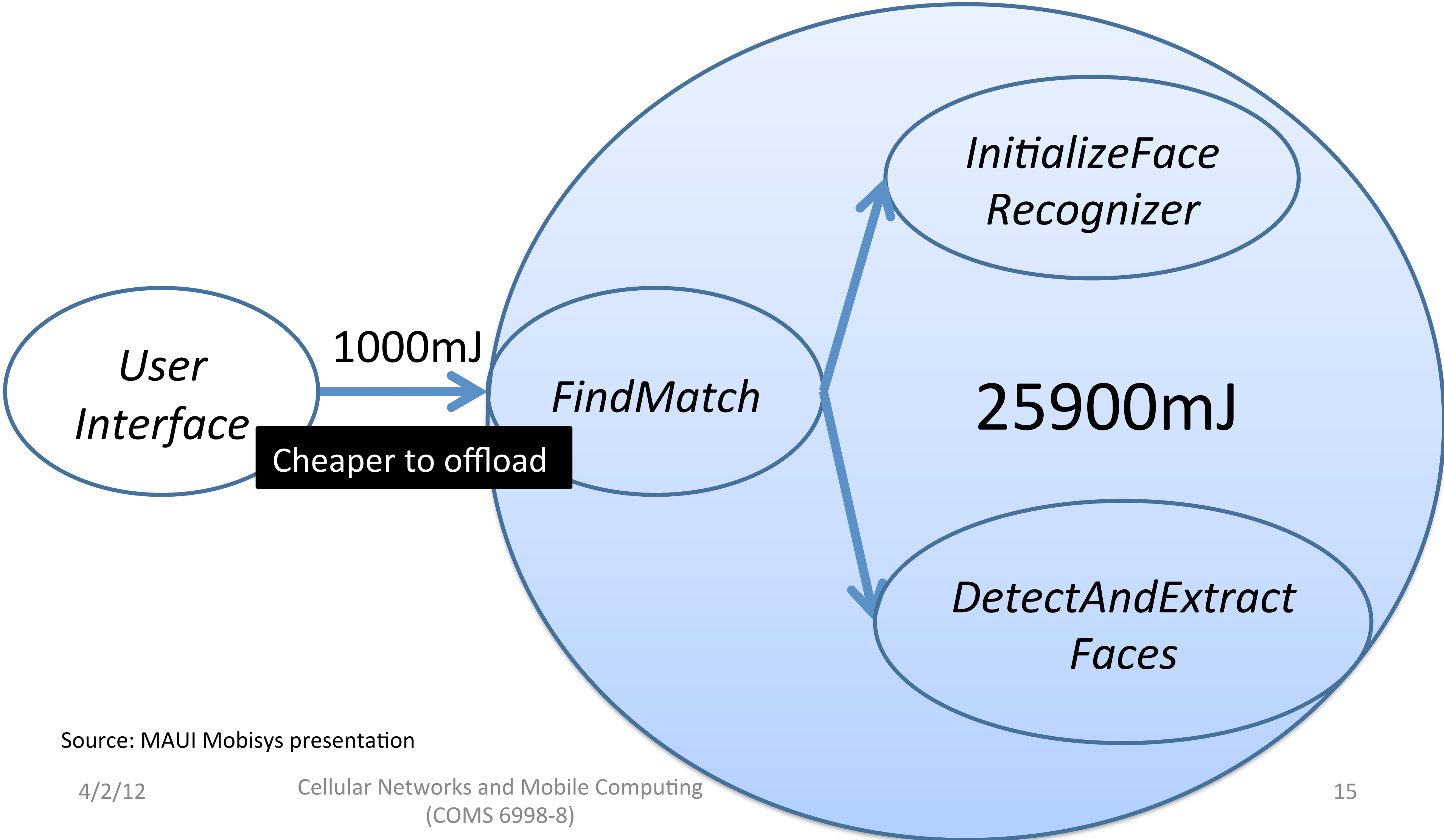
Is Global Program Analysis Needed?

Yes! – This simple example from Face Recognition app shows why local analysis fails.



Source: MAUI Mobisys presentation

Is Global Program Analysis Needed?



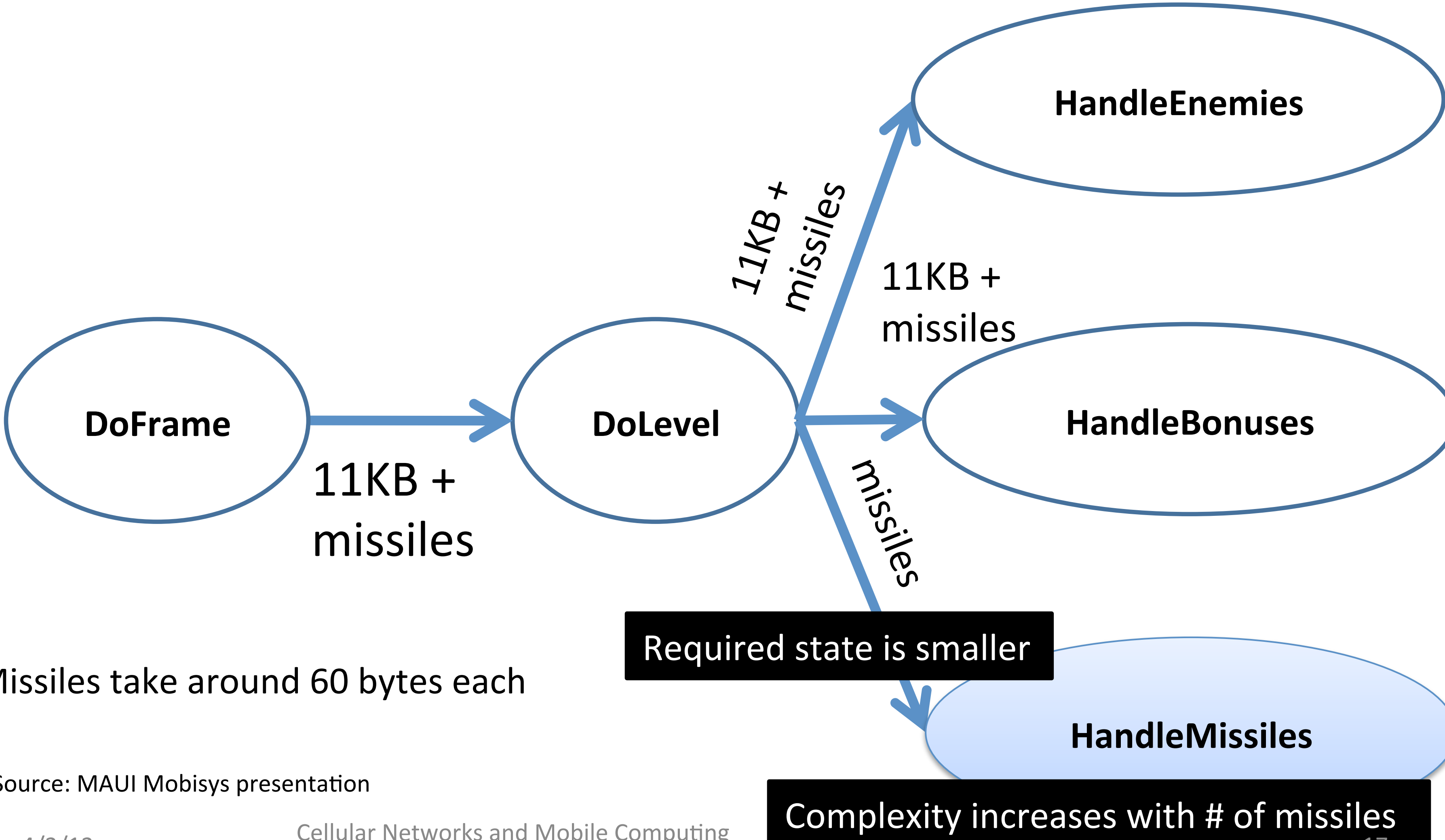
Source: MAUI Mobisys presentation

Can MAUI Adapt to Changing Conditions?

- Adapt to:
  - Network Bandwidth/Latency Changes
  - Variability on method's computational requirements
- Experiment:
  - Modified off the shelf arcade game application
  - Physics Modeling (homing missiles)
  - Evaluated under different latency settings

Source: MAUI Mobisys presentation

Can MAUI Adapt to Changing Conditions?

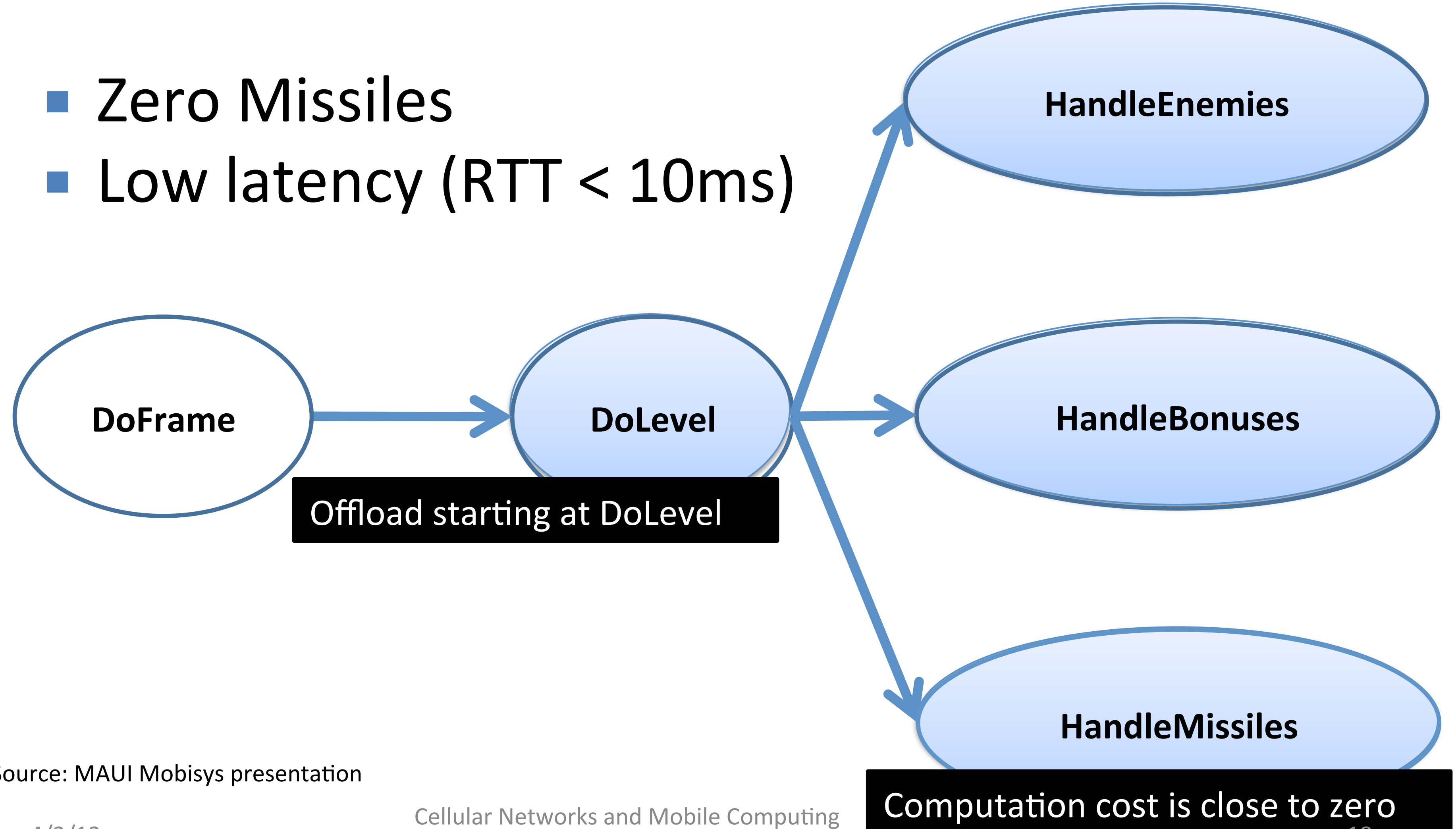


*Missiles take around 60 bytes each

Source: MAUI Mobisys presentation

Case 1

- Zero Missiles
- Low latency (RTT < 10ms)

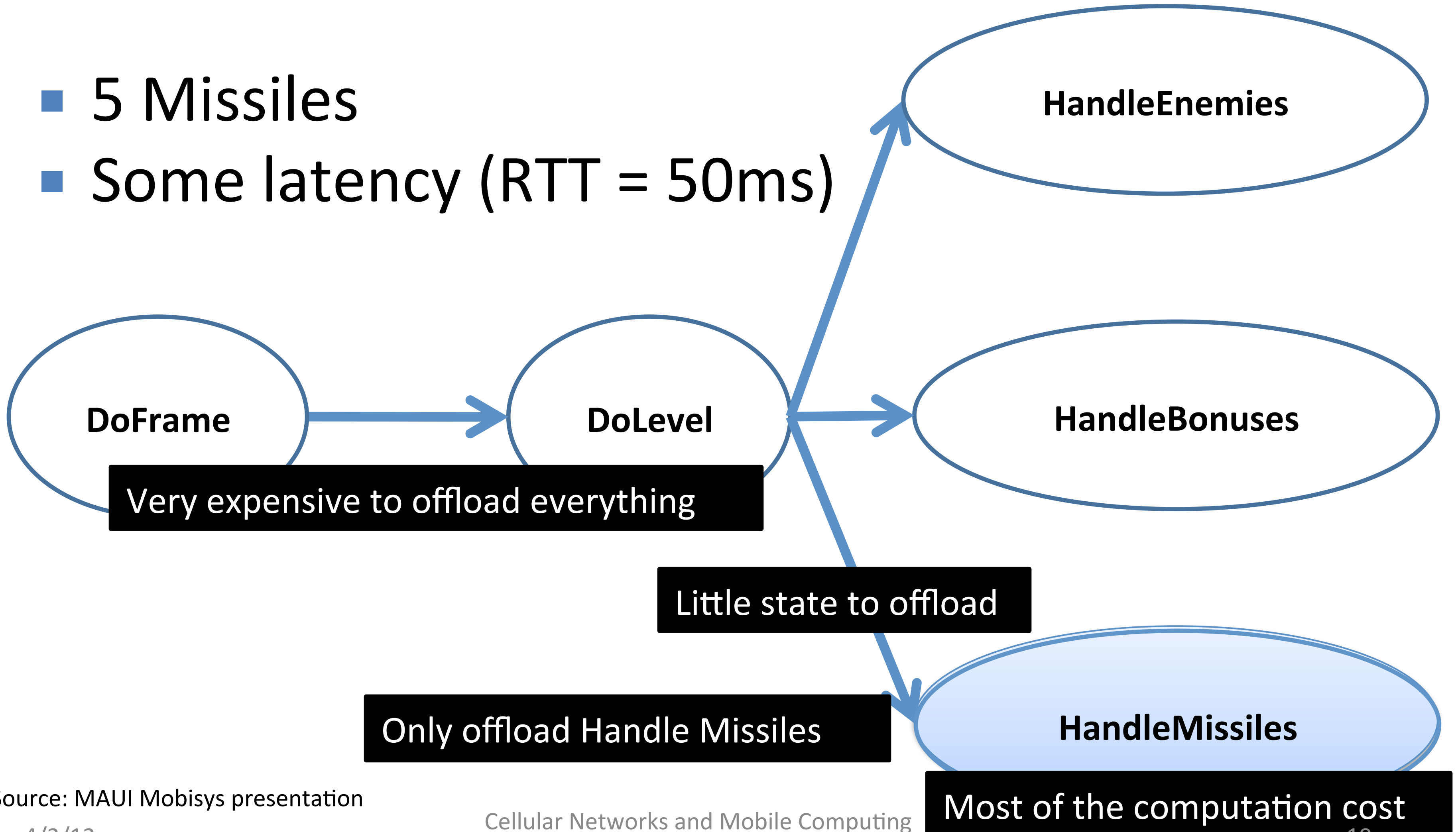


*Missiles take around 60 bytes each

Source: MAUI Mobisys presentation

Case 2

- 5 Missiles
- Some latency (RTT = 50ms)



*Missiles take around 60 bytes each

Source: MAUI Mobisys presentation

MAUI Implementation

- Platform
  - Windows Mobile 6.5
  - .NET Framework 3.5
  - HTC Fuze Smartphone
  - Monsoon power monitor
- Applications
  - Chess
  - Face Recognition
  - Arcade Game
  - Voice-based translator

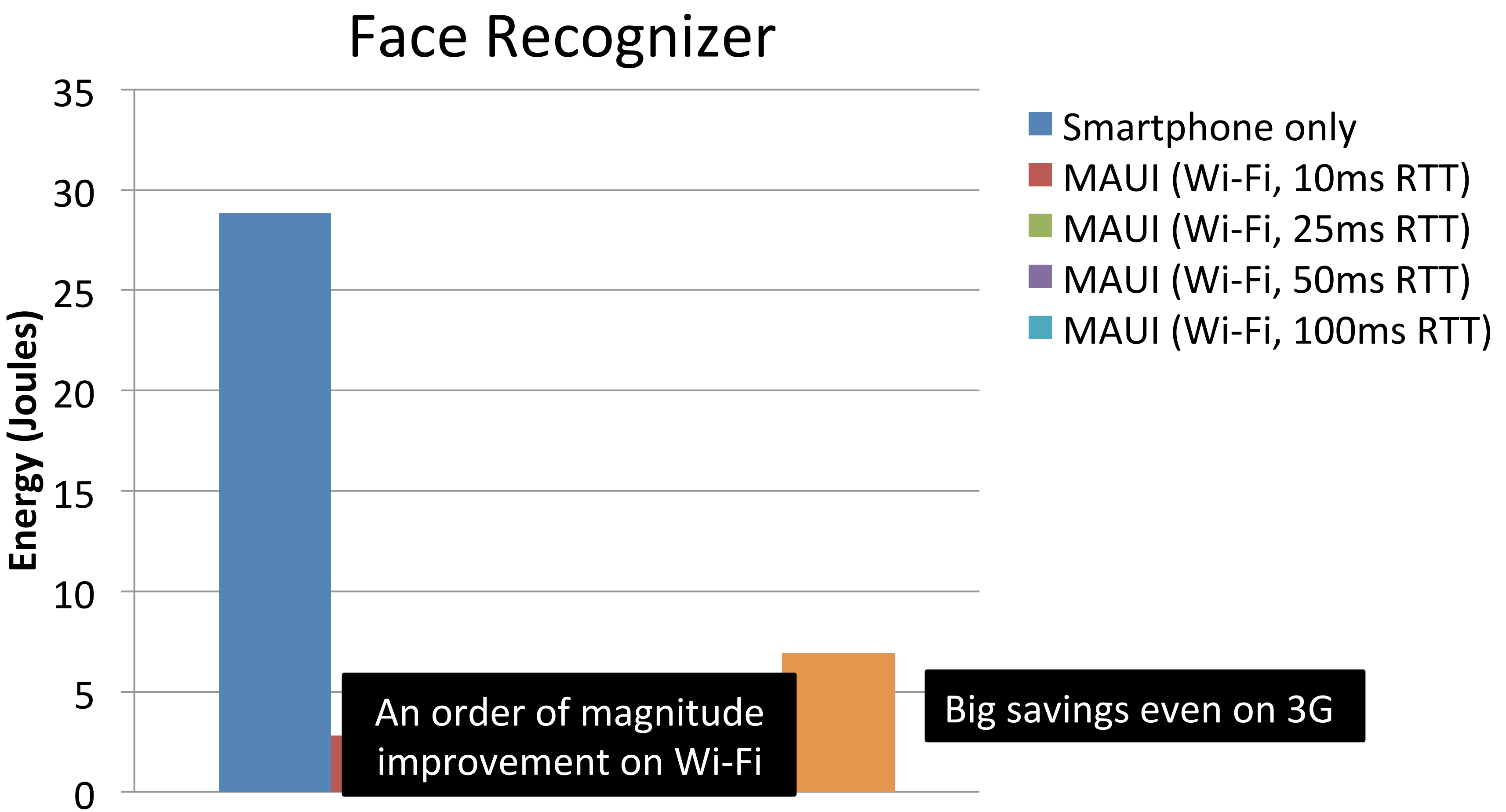
Source: MAUI Mobisys presentation

Questions

- How much can MAUI reduce energy consumption?
- How much can MAUI improve performance?
- Can MAUI Run Resource-Intensive Applications?

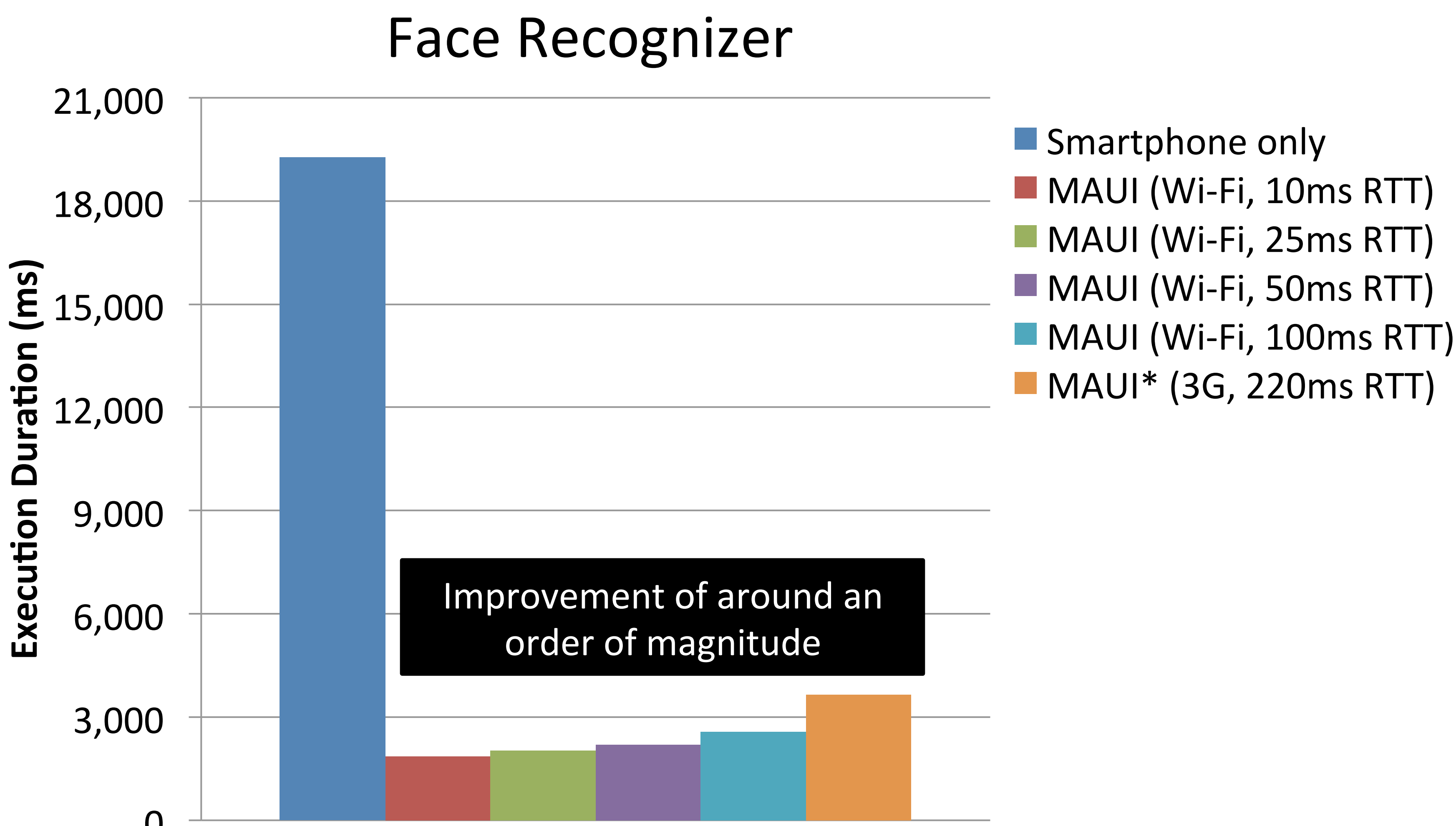
Source: MAUI Mobisys presentation

How much can MAUI reduce energy consumption?



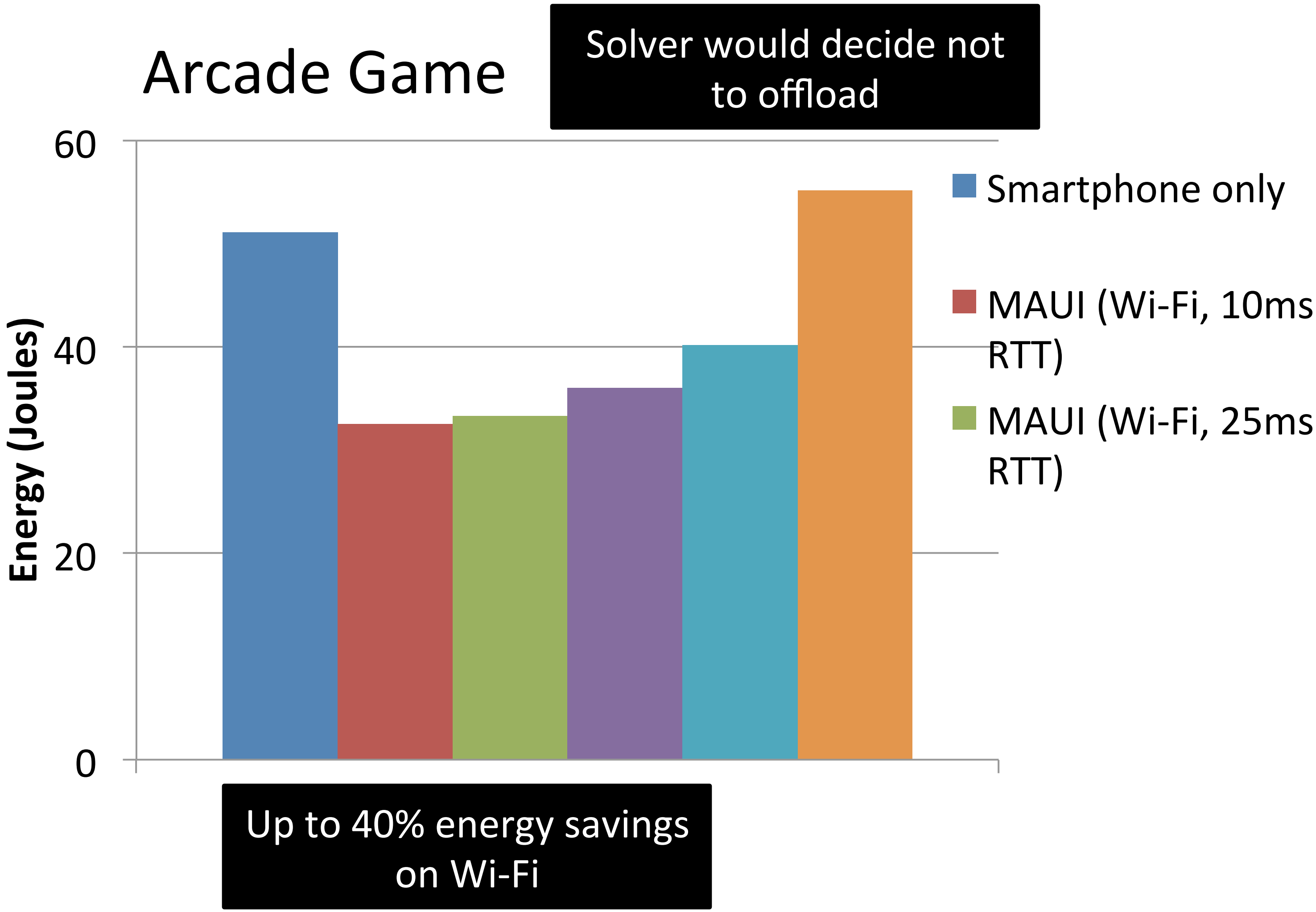
Source: MAUI Mobisys presentation

How much can MAUI improve performance?



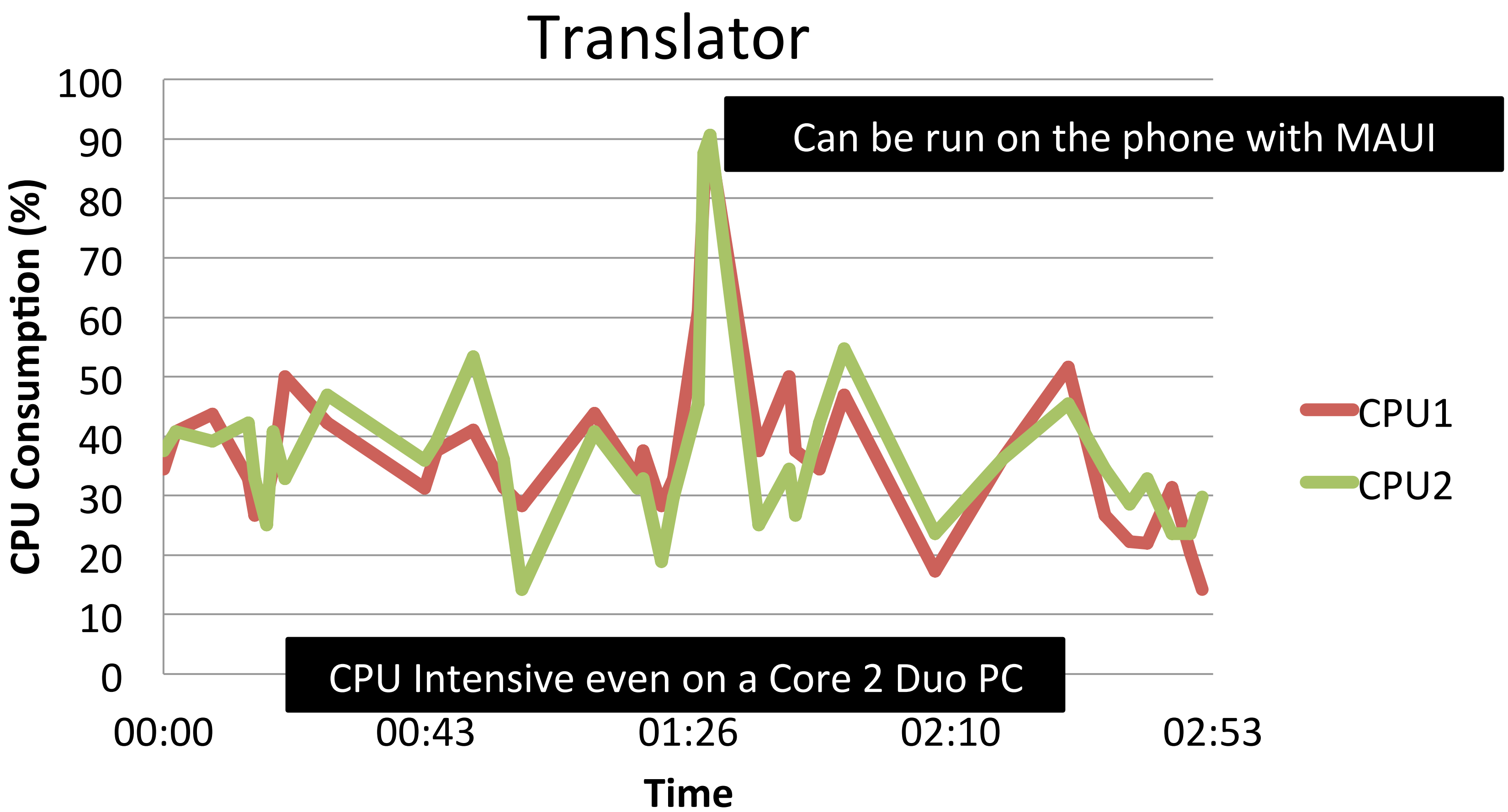
Source: MAUI presentation

Latency to server impacts the opportunities for fine-grained offload



Source: MAUI Mobisys presentation

Can MAUI Run Resource-Intensive Applications?



Source: MAUI Mobisys presentation

Conclusions

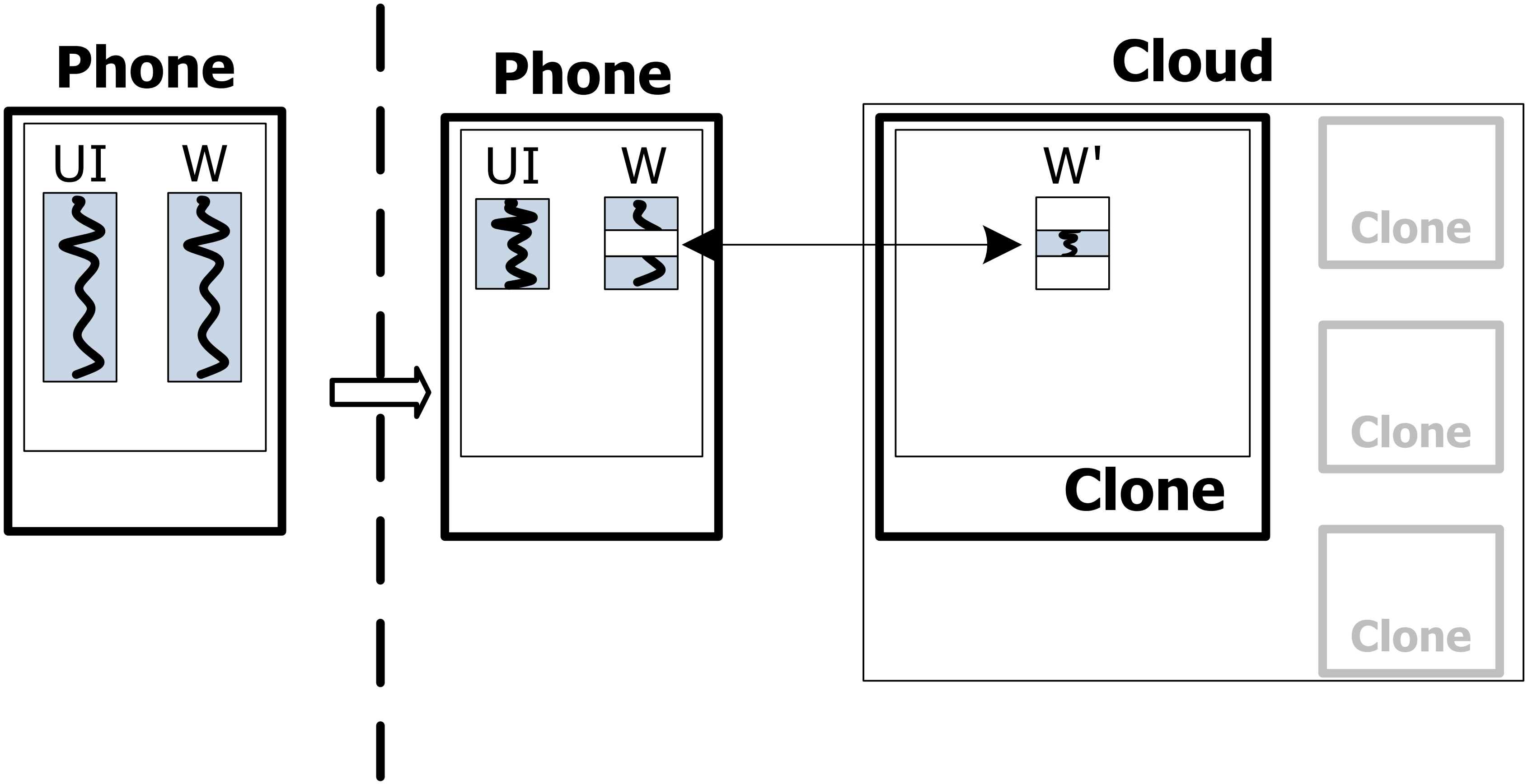
- MAUI enables developers to:
  - Bypass the resource limitations of handheld devices
  - Low barrier entry: simple program annotations
- For a resource-intensive application
  - MAUI reduced energy consumed by an order of magnitude
  - MAUI improved application performance similarly
- MAUI adapts to:
  - Changing network conditions
  - Changing applications CPU demands

Source: MAUI Mobisys presentation

mCloud Programming Model: CloneCloud

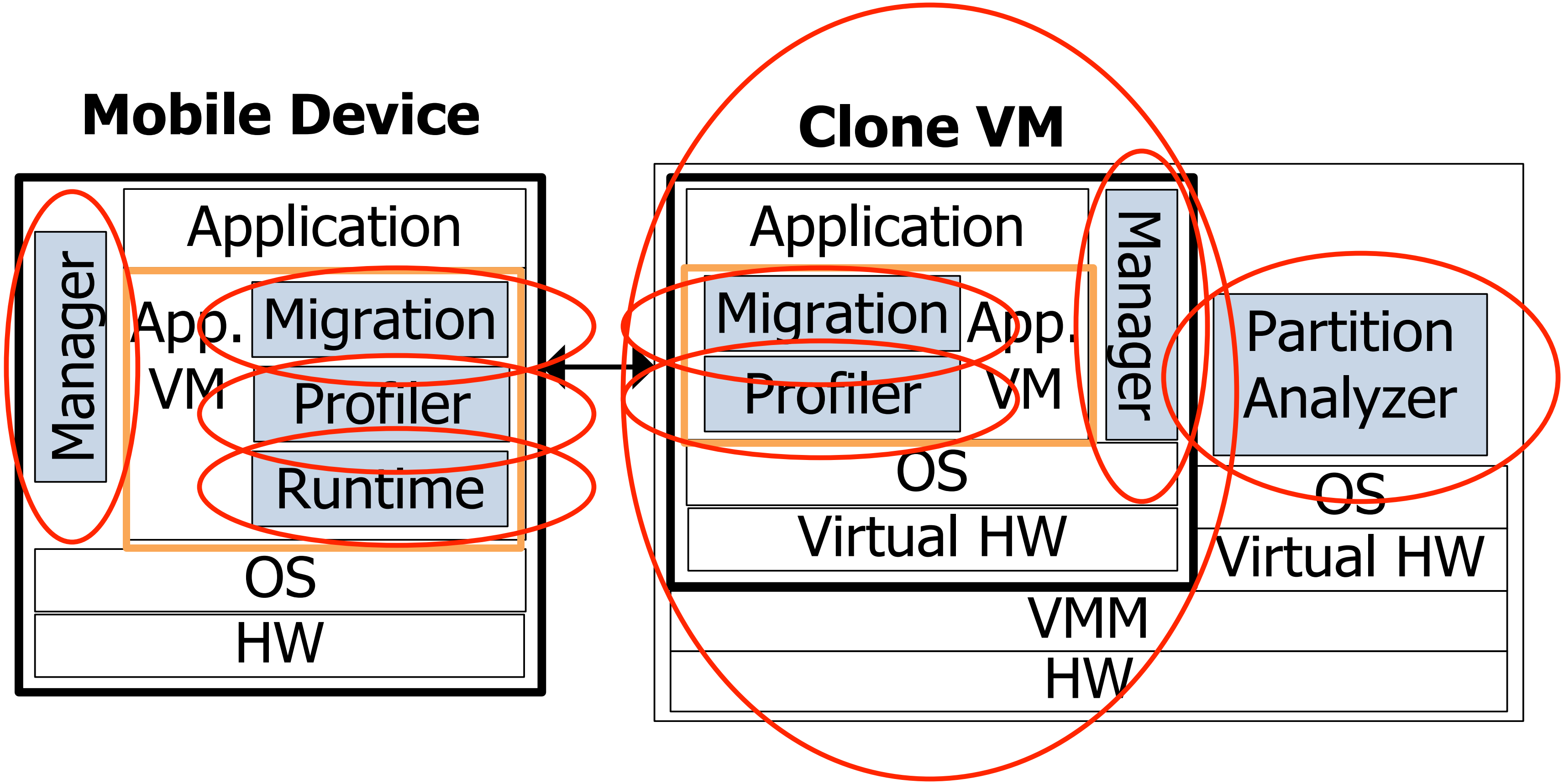
- Offloading decision done at beginning of execution

The Before-After Picture



Courtesy: Byung-Gon Chun et. al.

CloneCloud v1 Architecture



Courtesy: Byung-Gon Chun et. al.

AppVM Clones

- Virtual machine of entire client device
  - Android **x86** VM natively executed on VMWare in an x86 machine

- Synchronized file systems

Courtesy: Byung-Gon Chun et. al.

Partitioning

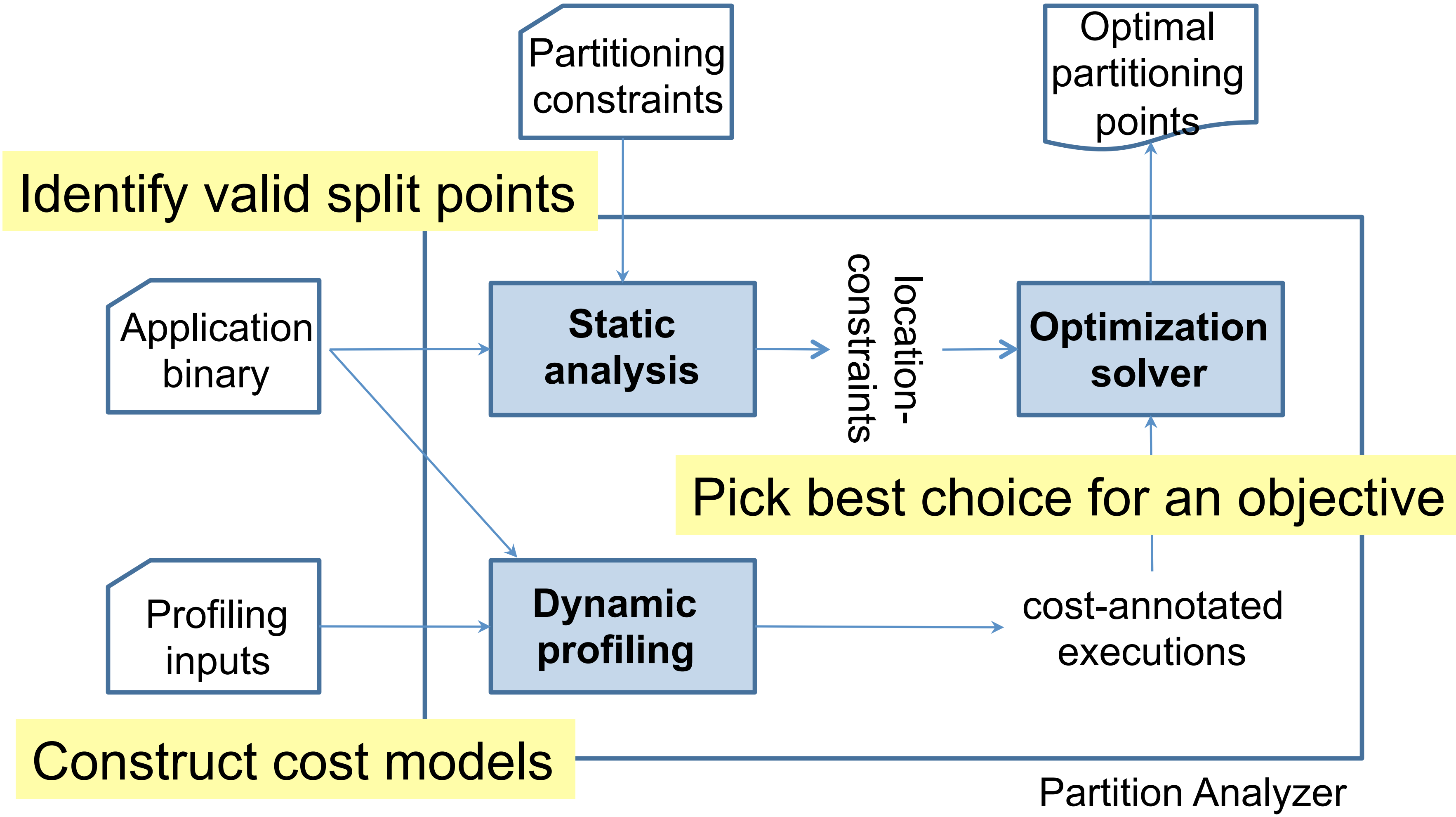
Where to partition code,

satisfying any constraints for correctness,

so as to optimize for current conditions

Courtesy: Byung-Gon Chun et. al.

Automatic Partitioning Framework



Courtesy: Byung-Gon Chun et. al.

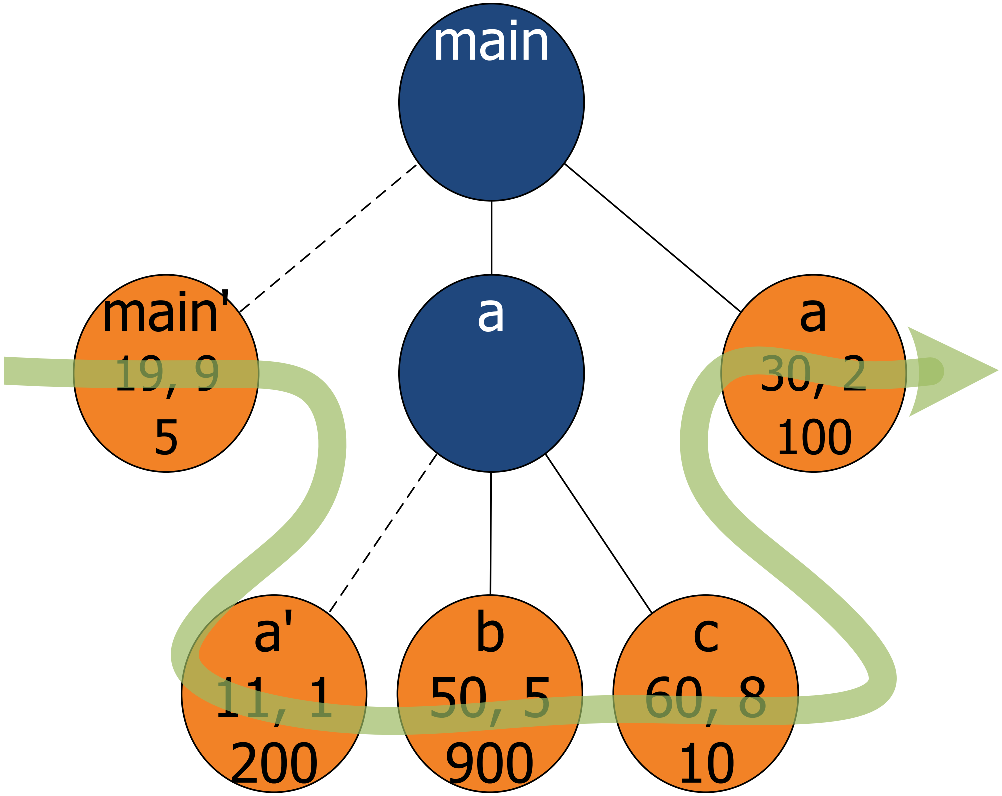
Partitioning: Static Analysis

- Partitioning points
  - Restrict to method entry/exit
- Identify “pinned” methods
  - Identify framework library methods
- Identify mutually dependent native state
  - Class natives stay together
- Collect static control flow
  - Who calls whom

Courtesy: Byung-Gon Chun et. al.

Partitioning: Profiling

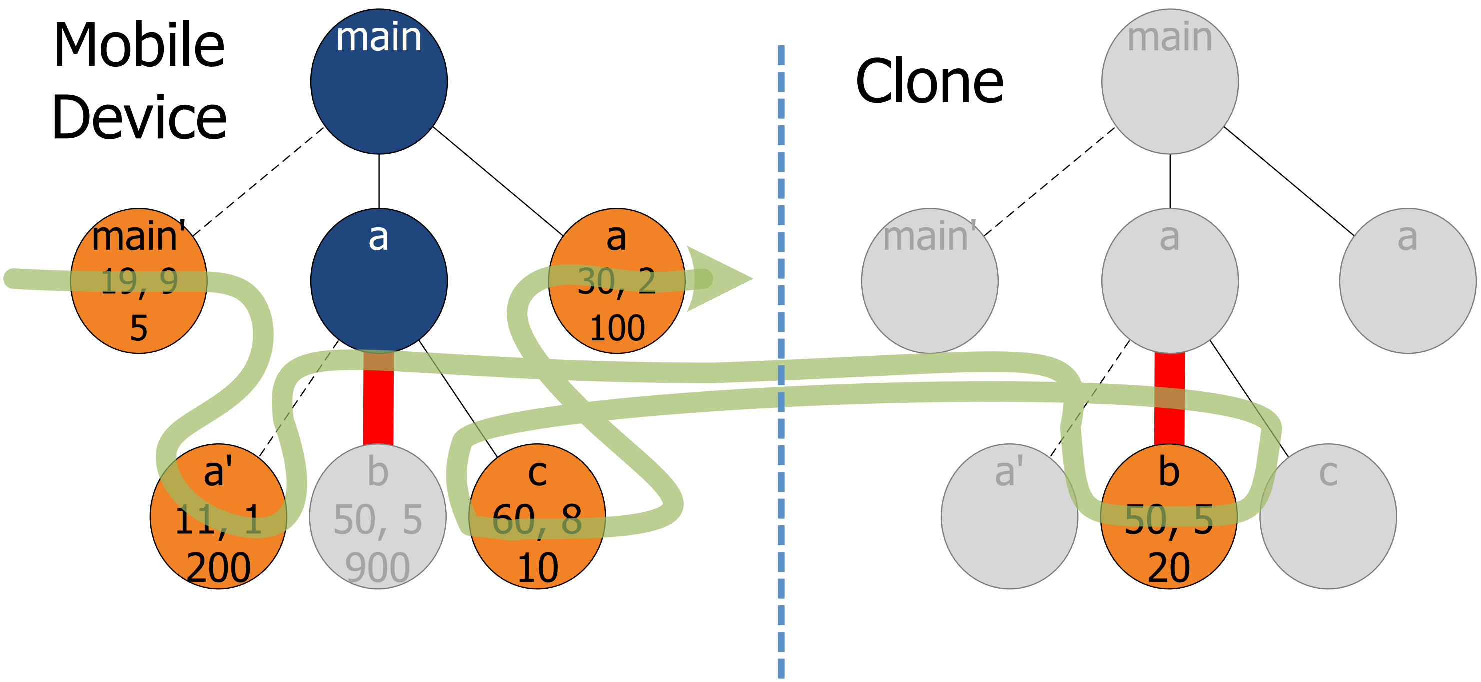
- Pick k invocations
- For each invocation
  - Run app on mobile device
  - Run app on clone
- For each method
  - Execution time
  - Context size entry/exit
  - Estimate of energy consumption



Courtesy: Byung-Gon Chun et. al.

Partitioning: Intuition

- Partitioning points splice profile trees together
- Context size used to estimate network cost



Partitioning: Optimization

- List all partitioning choices
- Remove partitioning choices that do not meet the constraints
- For each choice, compute total time
- Find the minimum

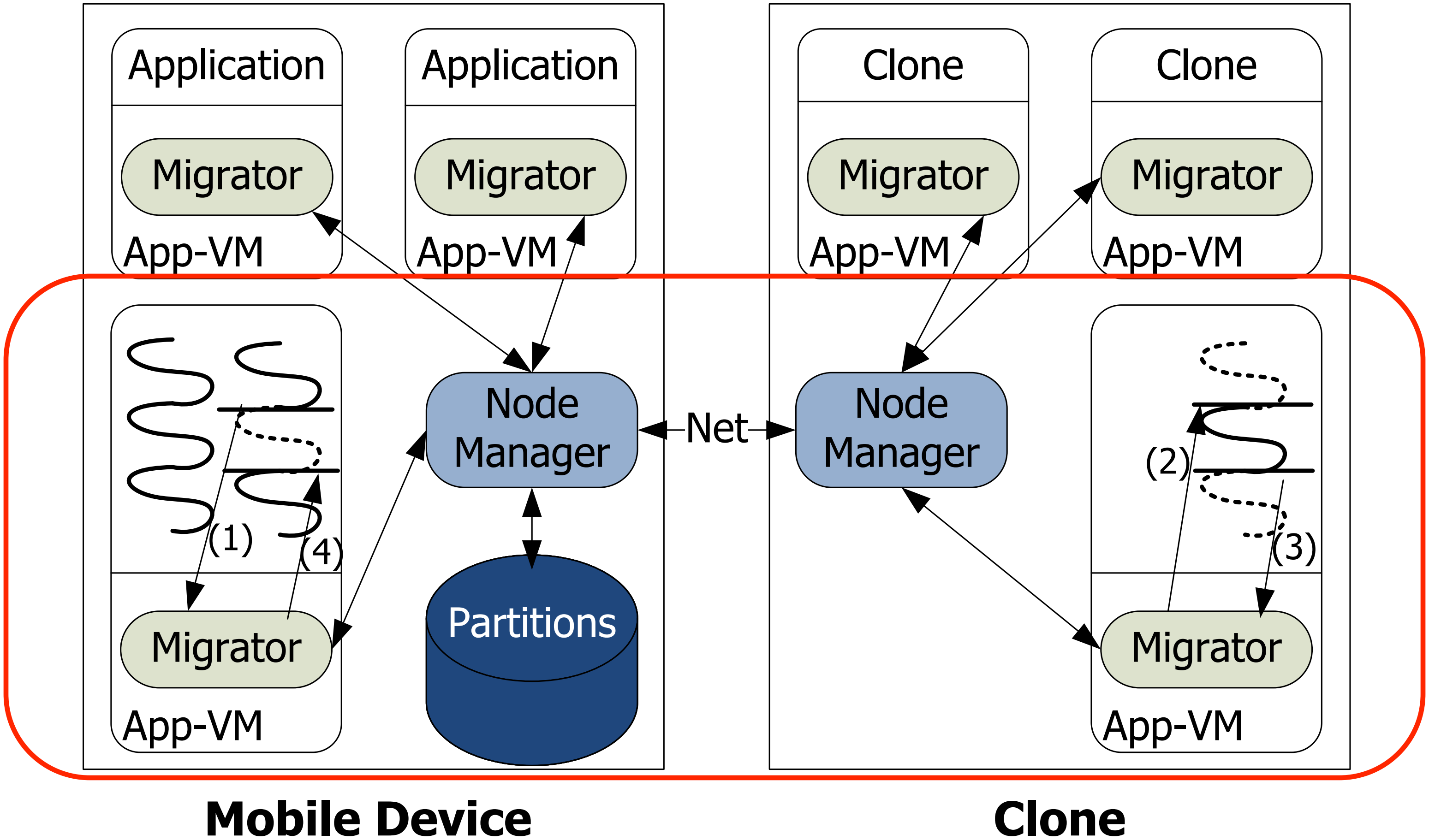
Courtesy: Byung-Gon Chun et. al.

Distributed Execution

How to seamlessly execute the partitions of an application locally and remotely

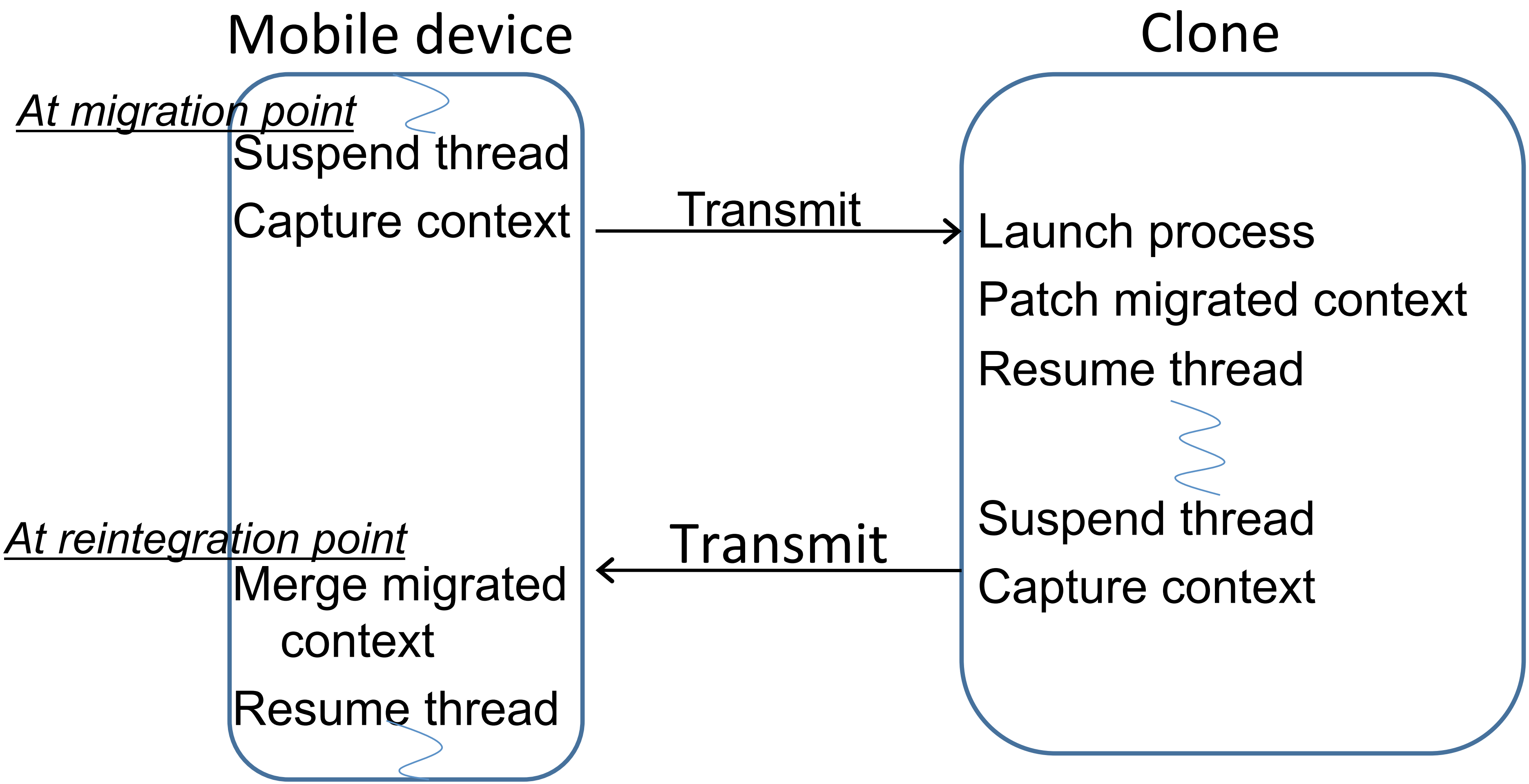
Courtesy: Byung-Gon Chun et. al.

Migration Architecture



Courtesy: Byung-Gon Chun et. al.

Migration: Suspend-Transmit-Resume



Courtesy: Byung-Gon Chun et. al.

Migration Challenges

- Regular IDs (references) not globally unique
  - Address space difference between mobile device and clone

- System objects created at boot everywhere

Courtesy: Byung-Gon Chun et. al.

Migration: References in Motion

Reference	MID	CID
0x01	1	null
0x02	2	null
0x03	3	null

(1) Mobile Device

migration

Reference	MID	CID
0x21	1	11
~~0x22~~	~~2~~	~~12~~
0x23	3	13
0x24	null	14
0x22	null	15

GC'ed (row 0x22, 2, 12)

New objects (rows 0x24 and 0x22, null)

(2) Clone

reintegration

(3) Mobile Device

Reference	MID	CID
0x01	1	11
~~0x02~~	~~2~~	~~12~~
0x03	3	13
0x04	4	14
0x05	5	15

Courtesy: Byung-Gon Chun et. al.

The CloneCloud v1 Prototype

- On Android platform + x86 VM clones
- Modified DalvikVM to support thread-level migration and dynamic profiling
- Implemented NodeManager to handle registration and communication
- Static analysis using JChord

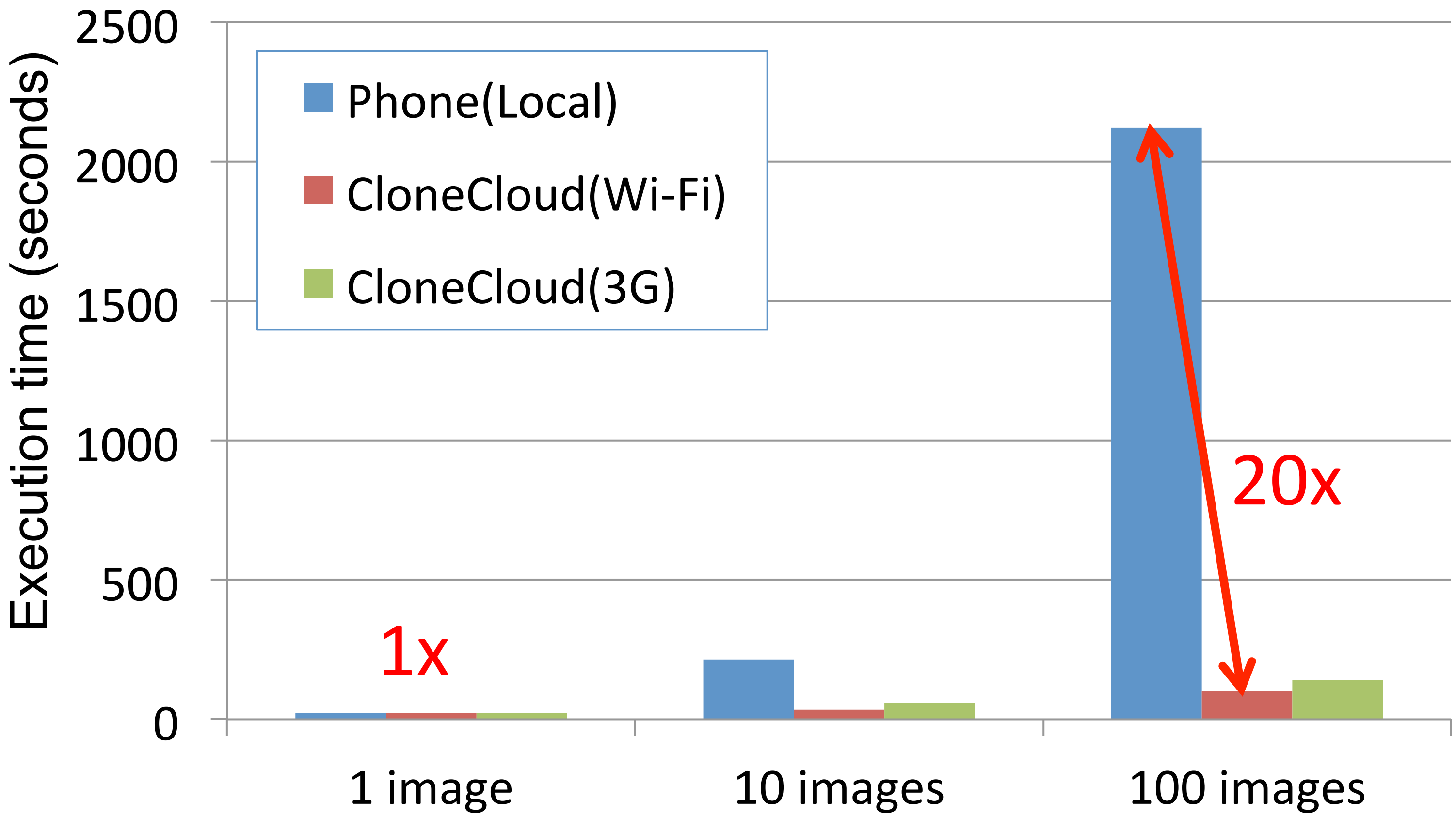
Courtesy: Byung-Gon Chun et. al.

Evaluation Setup

- Test applications
  - Image search
  - Virus scanning
  - Privacy-preserving user profiling
- Phone: Android Dev Phone 1
- Clone VM: a server with a 3.0GHz Xeon CPU running VMWare ESX 4.1 in Intel IT infrastructures
- Wireless connection: WiFi, T-mobile 3G

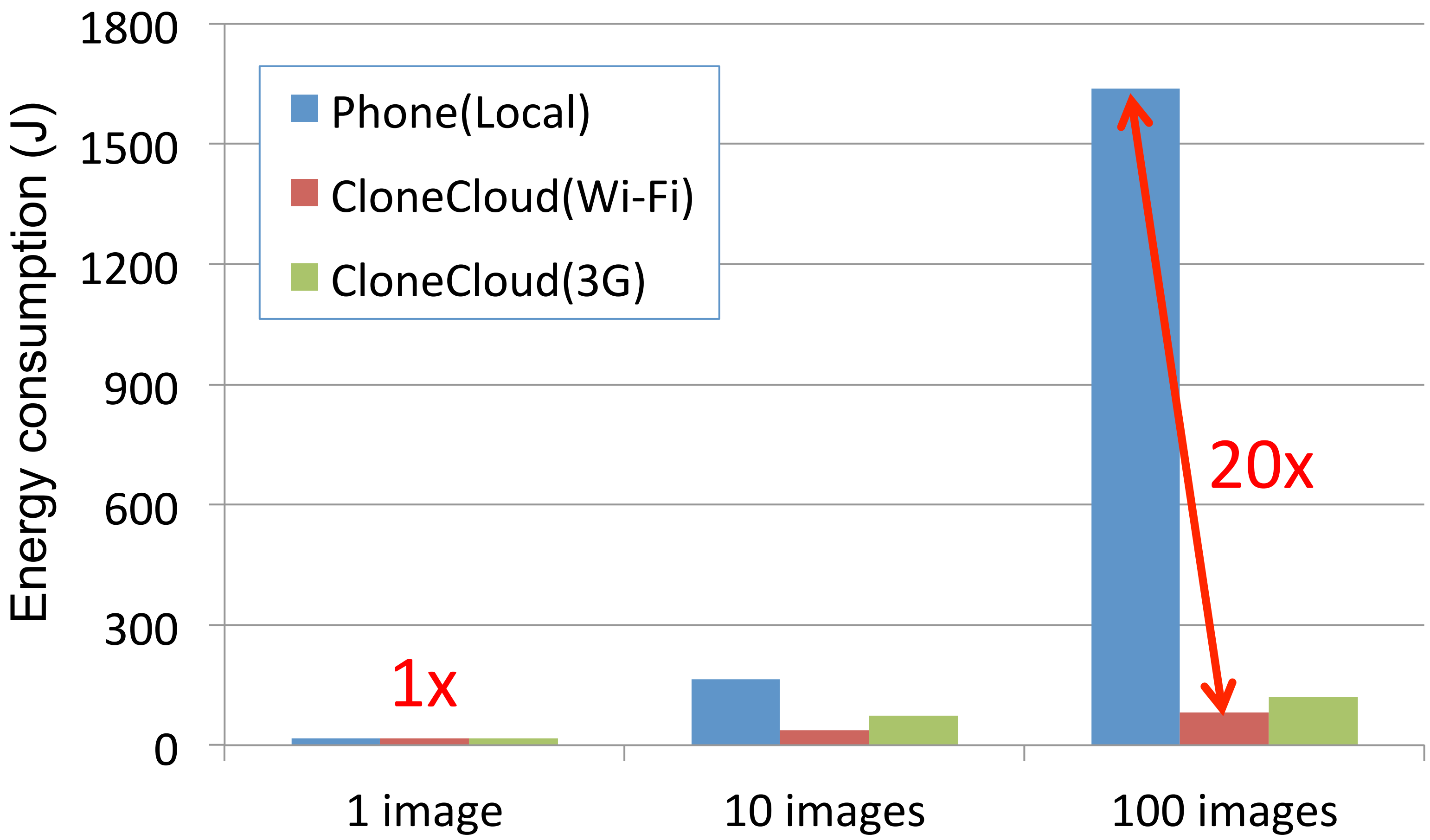
Courtesy: Byung-Gon Chun et. al.

Execution Time Comparison (Image Search)



Courtesy: Byung-Gon Chun et. al.

Energy Consumption Comparison (Image Search)



Courtesy: Byung-Gon Chun et. al.

Discussion

- Applicability is application-specific
  - Application characteristics
  - Execution mechanism
- A design point with focus on automation
  - Apps structured to be more amenable to migration
- Clone VMs in untrusted environments

Courtesy: Byung-Gon Chun et. al.

Conclusion

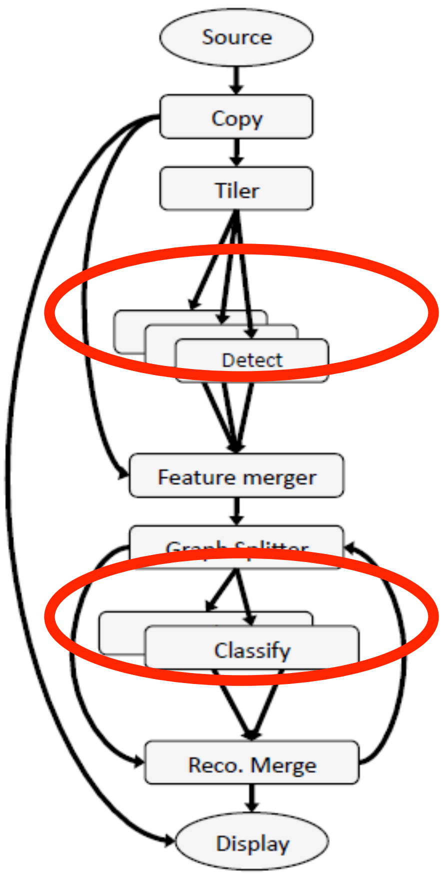
- CloneCloud: automatic partitioning and migration to enhance mobile applications
  - Use of clones
  - Flexible app partitioning
  - Seamless migration

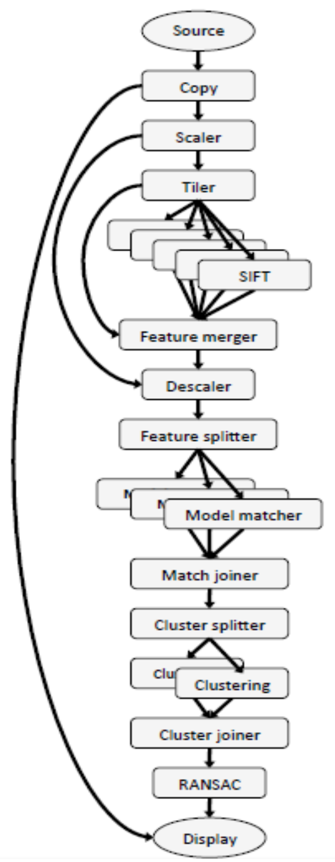
Lots of excitements in mobile cloud computing

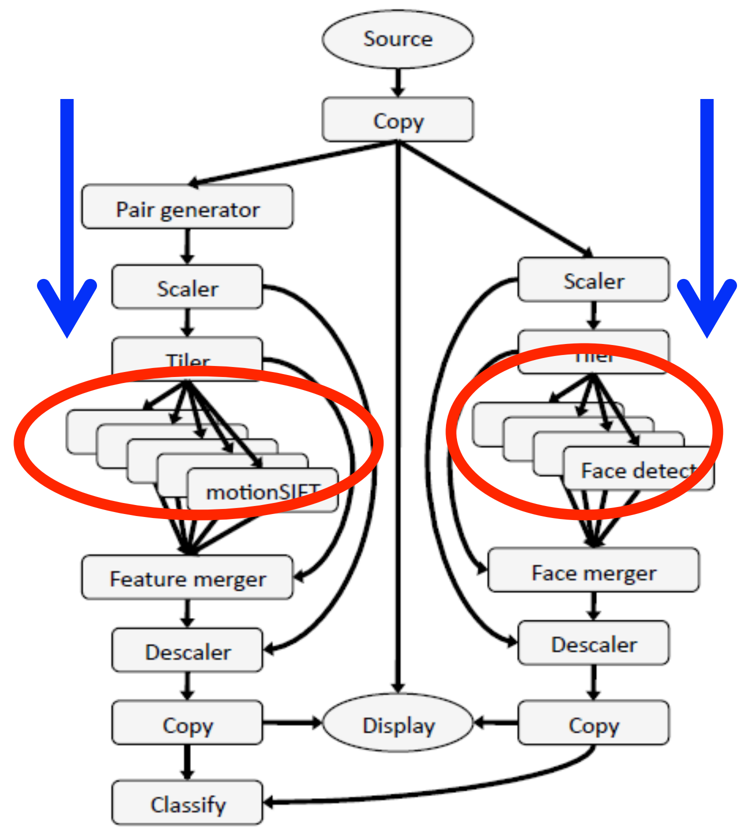
Courtesy: Byung-Gon Chun et. al.

mCloud Programming Model: Odessa

- Data flow graph: vertices are stages; edges are connectors; stages share nothing





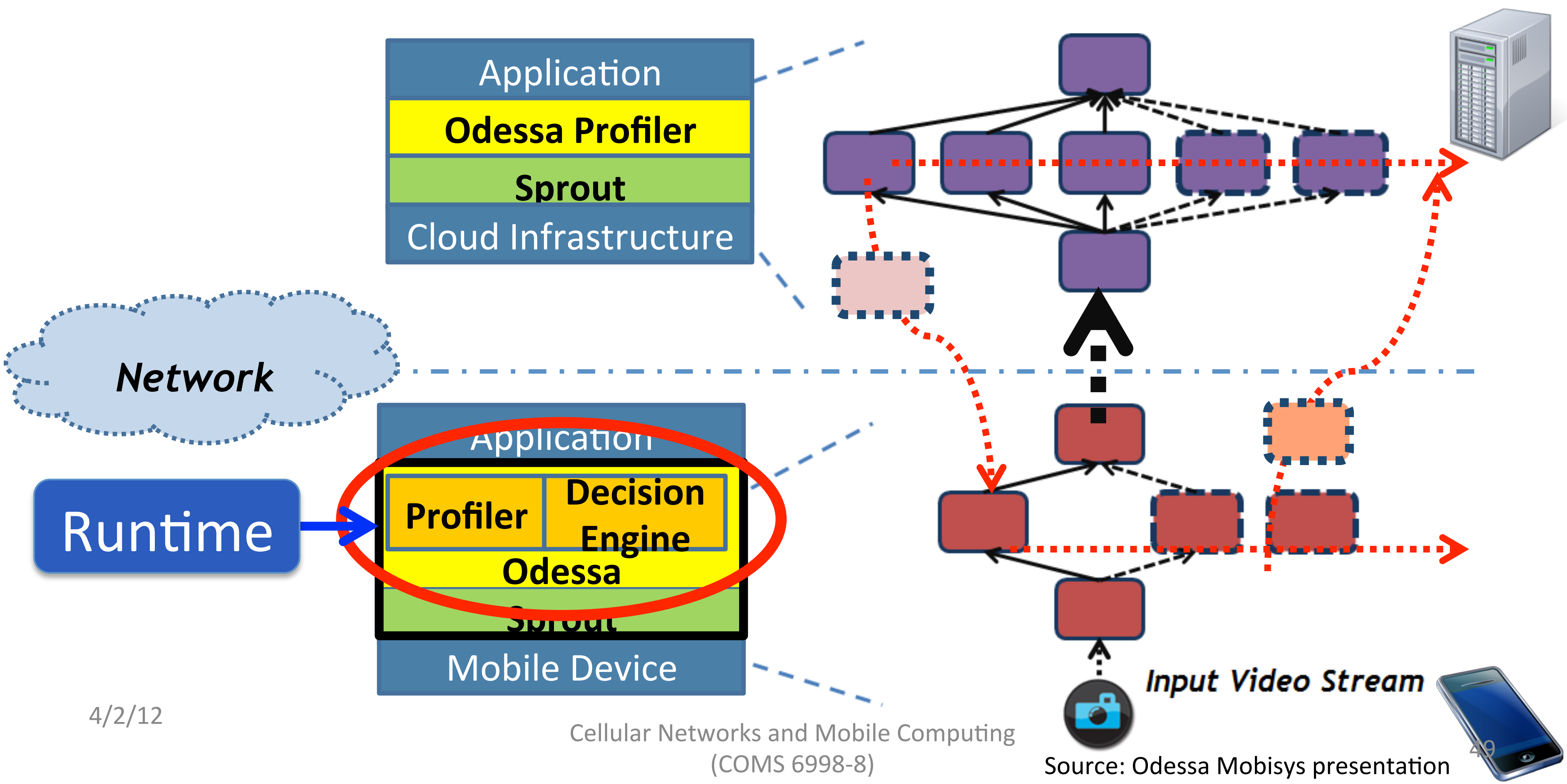


Source: Odessa Mobisys presentation

Face Recognition **Object Pose Estimation** **Gesture Recognition**

mCloud Programming Model: Odessa (Cont'd)

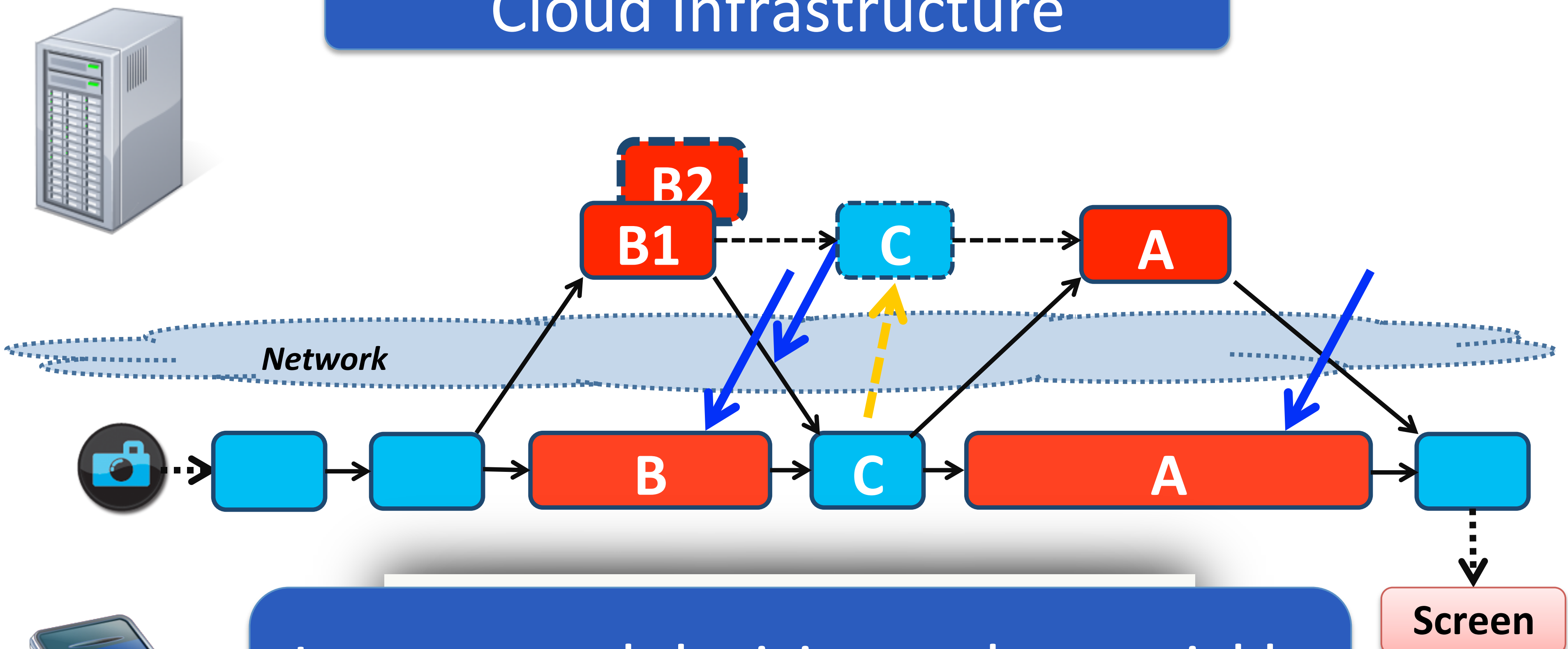
- Offloading DEcision System for Streaming Applications



Source: Odessa Mobisys presentation

mCloud Programming Model: Odessa (Cont'd)

Cloud Infrastructure



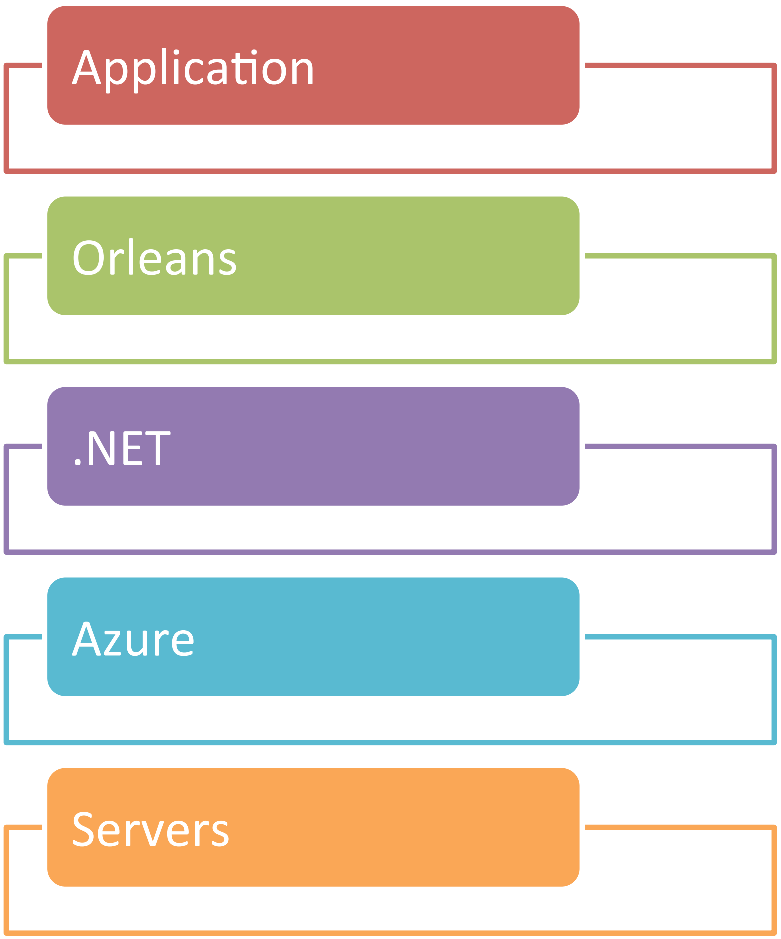
Incremental decisions adapt quickly to input and platform variability.

Smartphone

Source: Odessa Mobisys presentation

mCloud Programming Model: Orleans

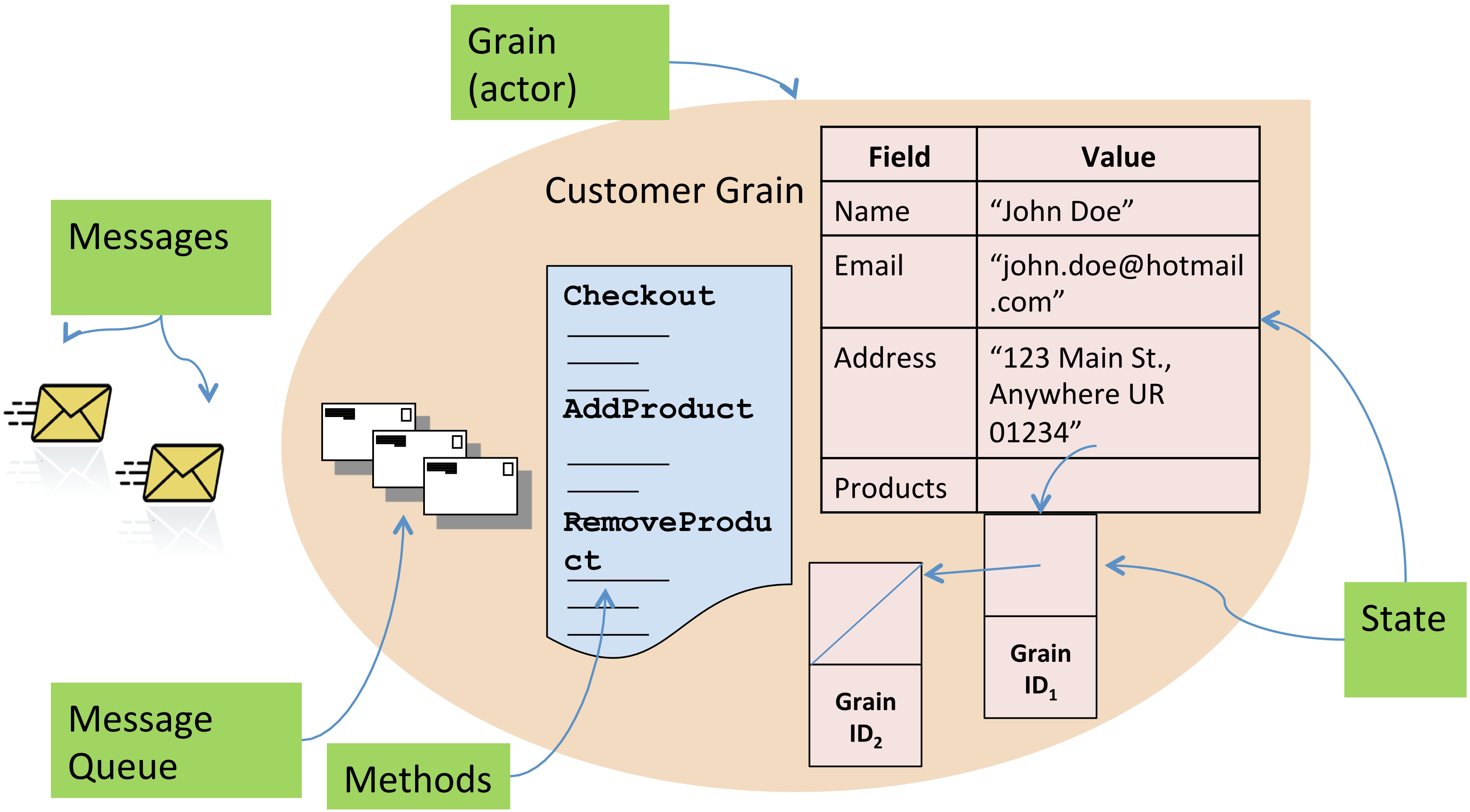
- Software framework and runtime to make cloud programming easier and more productive
- Shift burden of correctness and performance from developer to Orleans system
- Experimental system from Microsoft Research
- Radically simplified, prescriptive programming model
  - Actors
  - Asynchronous messaging
  - Lightweight transactions
  - Persistence
  - Adaptive performance management



mCloud Programming Model: Orleans (Cont'd)

- Actor based: fine-grain distributed objects are natural abstraction
- Grains partition data
- Secure and isolated computation with clear points of communications
  - Enable computation replication
- Natural integration with persistent storage
  - Grain resides on disk until activated

mCloud Programming Model: Orleans (Cont'd)



Support for Service Interaction

- Synergy of mCloud services
  - Shared basic services enables more efficient usage of cloud resources(e.g. shared memcache eliminates data duplication of individual services)
  - Co-location makes mashed-up applications to achieve native performance (file transfer becomes an object reference)
- mCloud offers shared platform services
- mCloud optimizes service interaction through active VM migration

Closing Thoughts

- Cloud computing needs to advance in architecture, programming model, platform services to revolutionize mobile computing

Useful Tools

- Hprof: a heap/CPU profiling tool
- Jchord: a static analysis tool on Java bytecode
- Dummynet (ipfw): simulates/enforces queue and bandwidth limitations, delays, packet losses
- OpenCV: a library of programming functions for real time computer vision