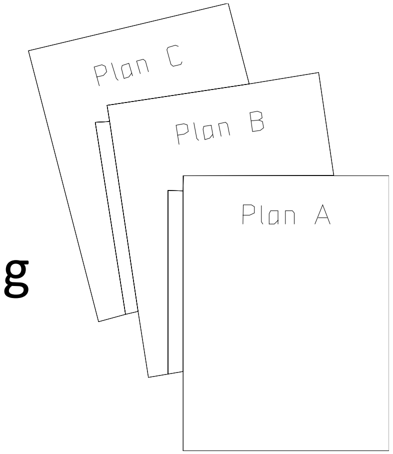
Mobile Cloud and Edge Systems

Overview

- Objective
  - To understand the use of cloud systems for mobile computing
- Content
  - Introduction to cloud systems for mobile computing
  - MAUI: Making Smartphones Last Longer With Code Offload



- After this module, you should be able to
  - Understand the importance of cloud systems to enable rich applications with resource-constrained mobile devices

Today's Computing Platforms

Trends:

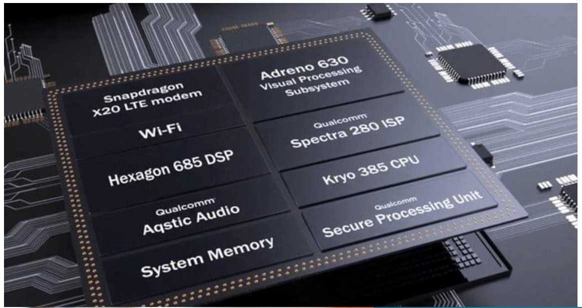
- parallel
- cloud
- mobile

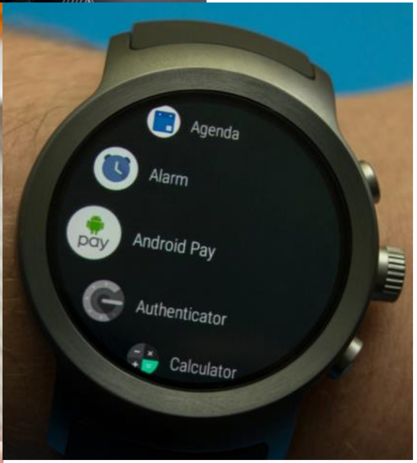
Traits:

- numerous
- diverse
- distributed

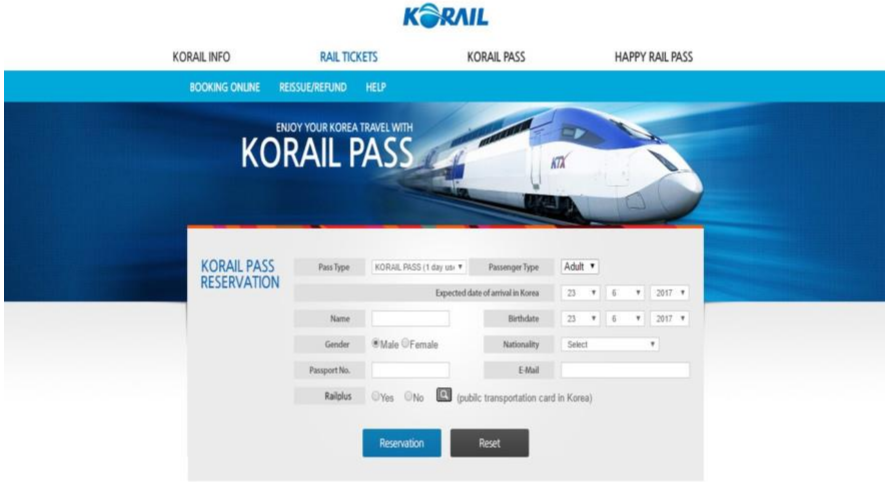
Unprecedented software engineering challenges in reliability, productivity, scalability, energy-efficiency

Why Cloud? (over local computation)





Why Cloud? (over its own server)



KORAIL PASS RESERVATION

Pass Type	KORAIL PASS (1 day us ▾	Passenger Type	Adult ▾
Expected date of arrival in Korea			23 ▾ 6 ▾ 2017 ▾
Name		Birthdate	23 ▾ 6 ▾ 2017 ▾
Gender	● Male ○ Female	Nationality	Select ▾
Passport No.		E-Mail	
Railplus	○ Yes ○ No (public transportation card in Korea)		

A Challenge in Mobile Computing

- Rich apps are hindered by resource-constrained mobile devices (battery, CPU, memory, ...)

How can we seamlessly partition mobile apps and offload compute-intensive parts to the cloud?

A Challenge in Cloud Computing

Windows Azure
Microsoft's Cloud Services Platform

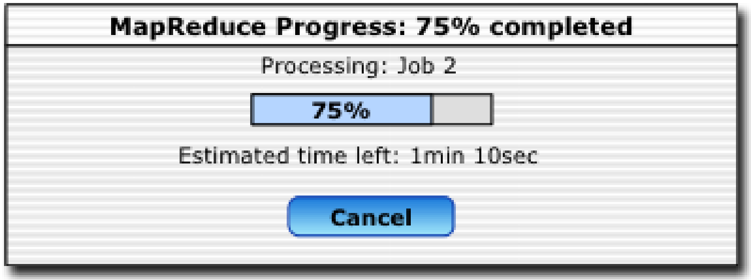
amazon web services™

hadoop

Google™ App Engine

service level agreements

resource isolation



MapReduce Progress: 75% completed

Processing: Job 2

75%

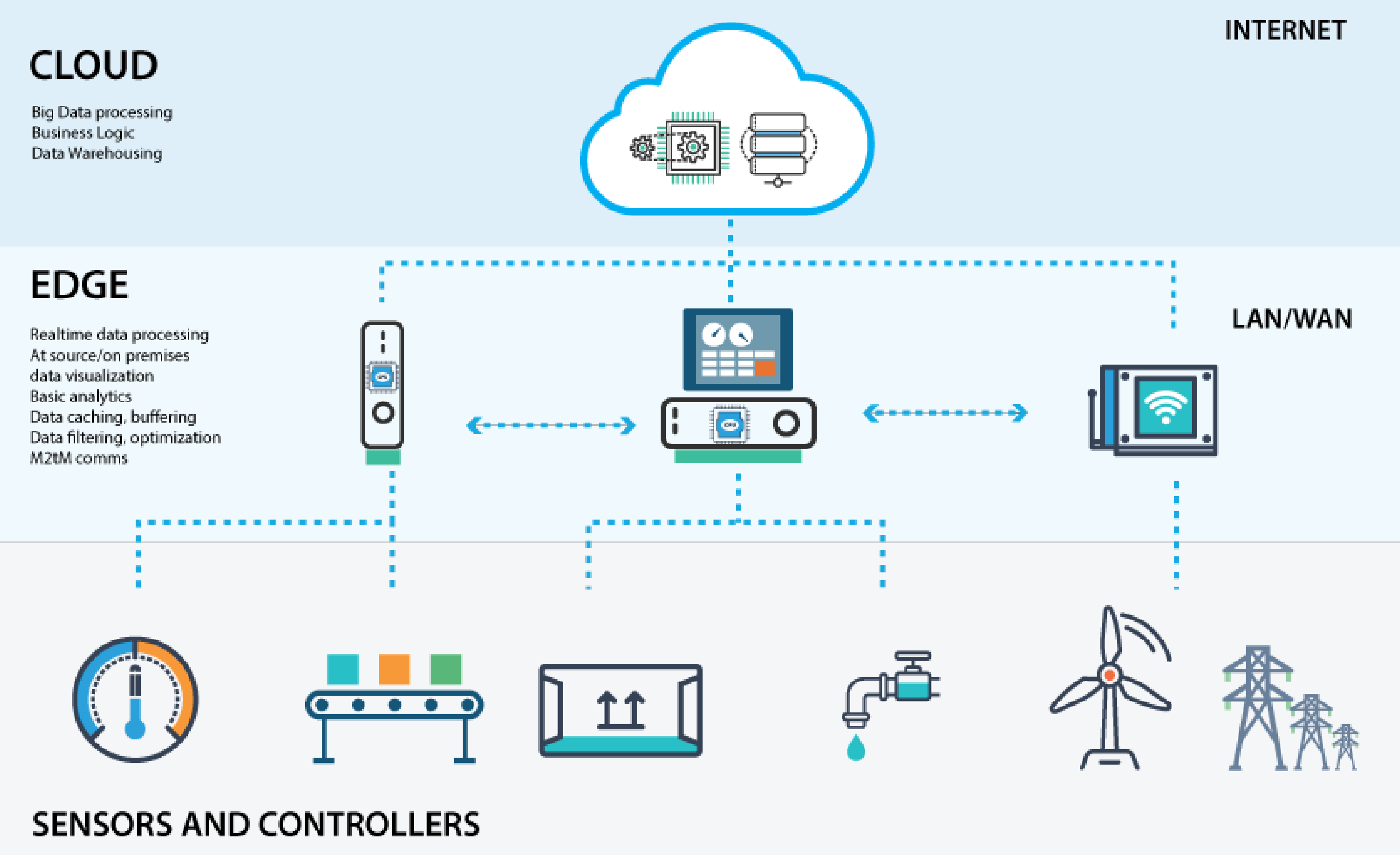
Estimated time left: 1min 10sec

energy efficiency

scheduling

How can we isolate the computational resource for multiple applications, schedule them, and optimize them?

Edge Computing and IoT



CLOUD

Big Data processing
Business Logic
Data Warehousing

EDGE

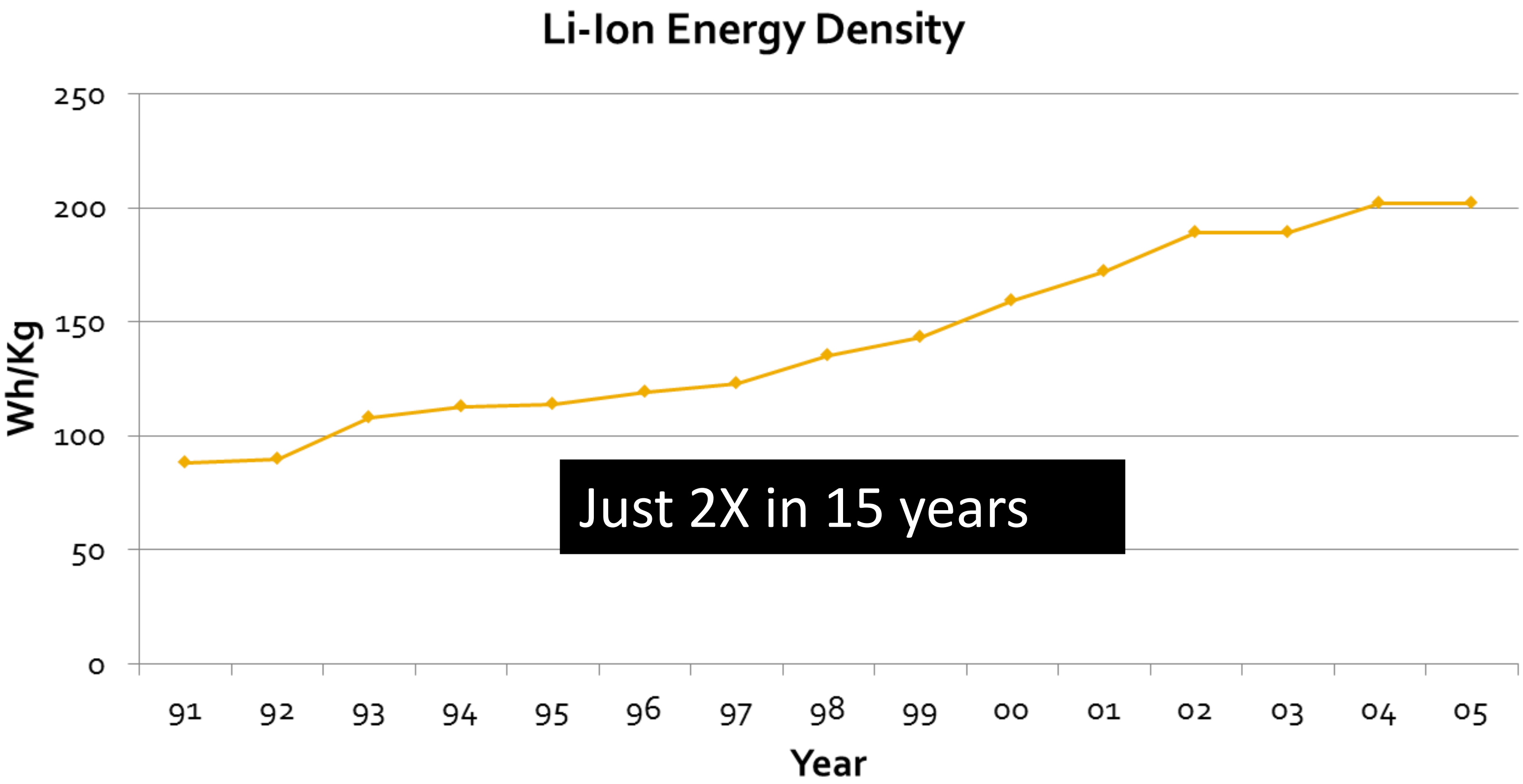
Realtime data processing
At source/on premises
data visualization
Basic analytics
Data caching, buffering
Data filtering, optimization
M2tM comms

SENSORS AND CONTROLLERS

MAUI: Making Smartphones Last Longer With Code Offload

MobiSys 2010

Battery is a Scarce Resource



Just 2X in 15 years

- CPU performance during same period: 246X
- A solution to the battery problem seems unlikely

Mobile Apps Can’t Reach Their Full Potential

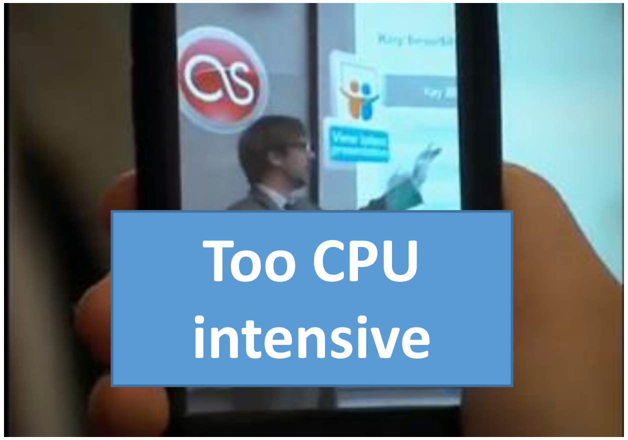
Slow, Limited or Inaccurate

Not on par with desktop counterparts

Speech Recogniti

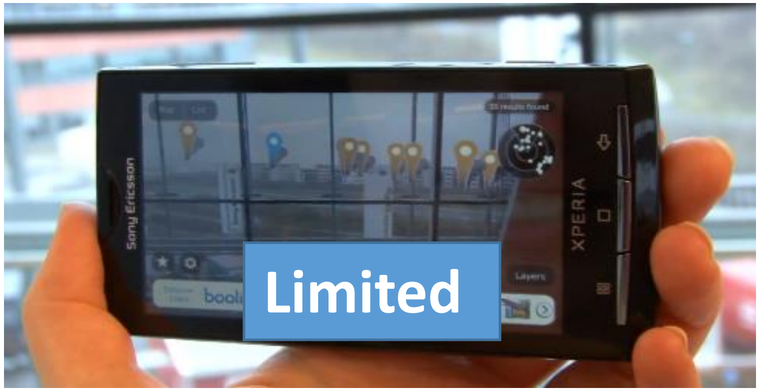
Power Intensive

ames



Too CPU intensive

Augmented Reality



Limited

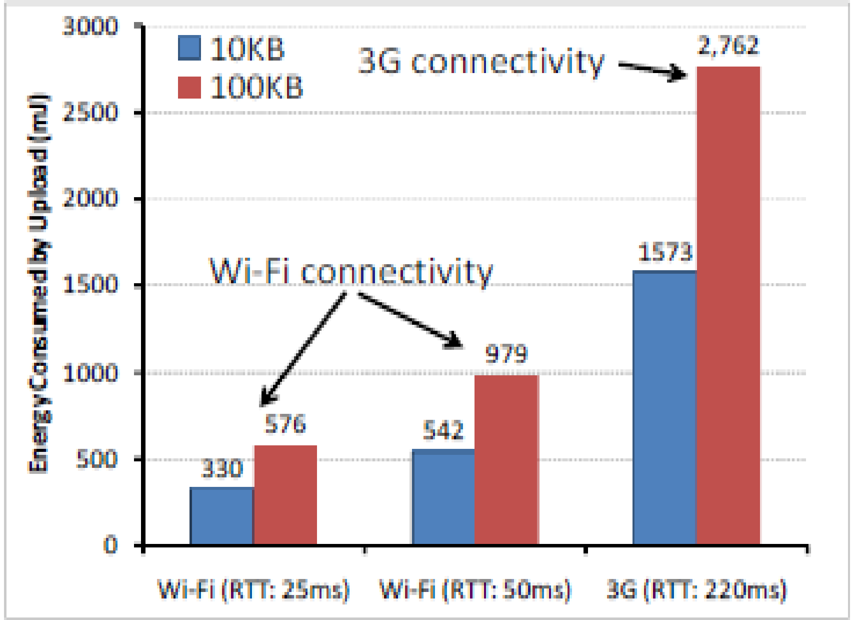
Approach: Remote Execution

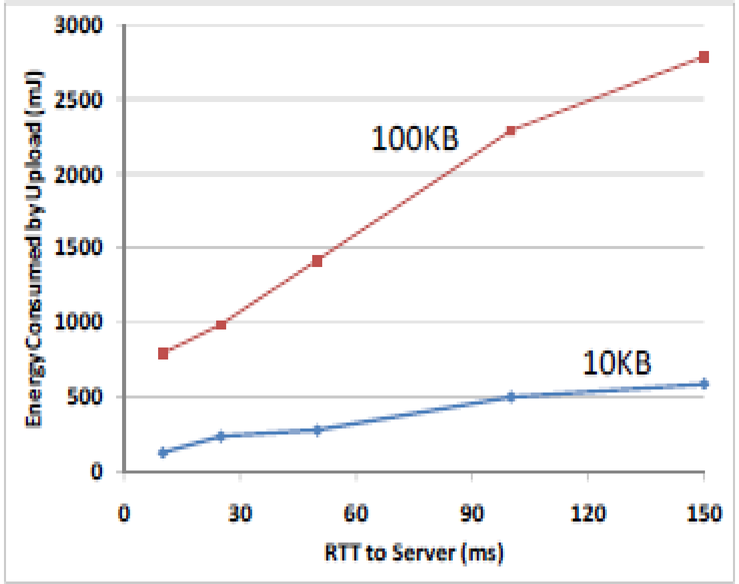
- **Remote execution can reduce energy consumption**

- Challenges:
  - What should be offloaded?
    - Leave to programmers? Full VM-migration?
  - How to dynamically decide when to offload?
    - Network conditions / program execution could change dynamically!
  - How to minimize the required programmer effort?

Key Observations

- Offloading Code works better when RTT is small.
- Volume of offloaded code→ sub-linear overhead.

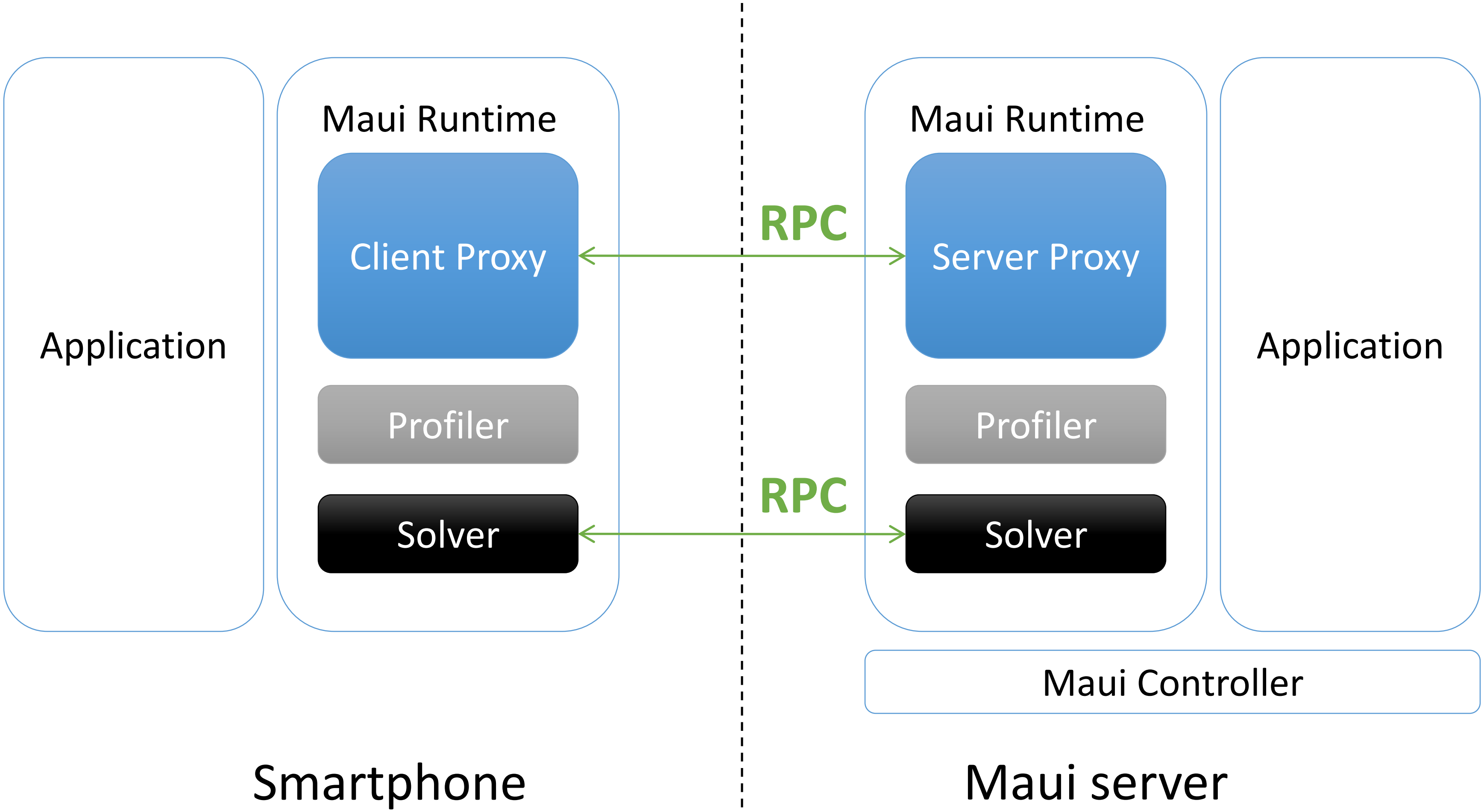




MAUI: Mobile Assistance Using Infrastructure

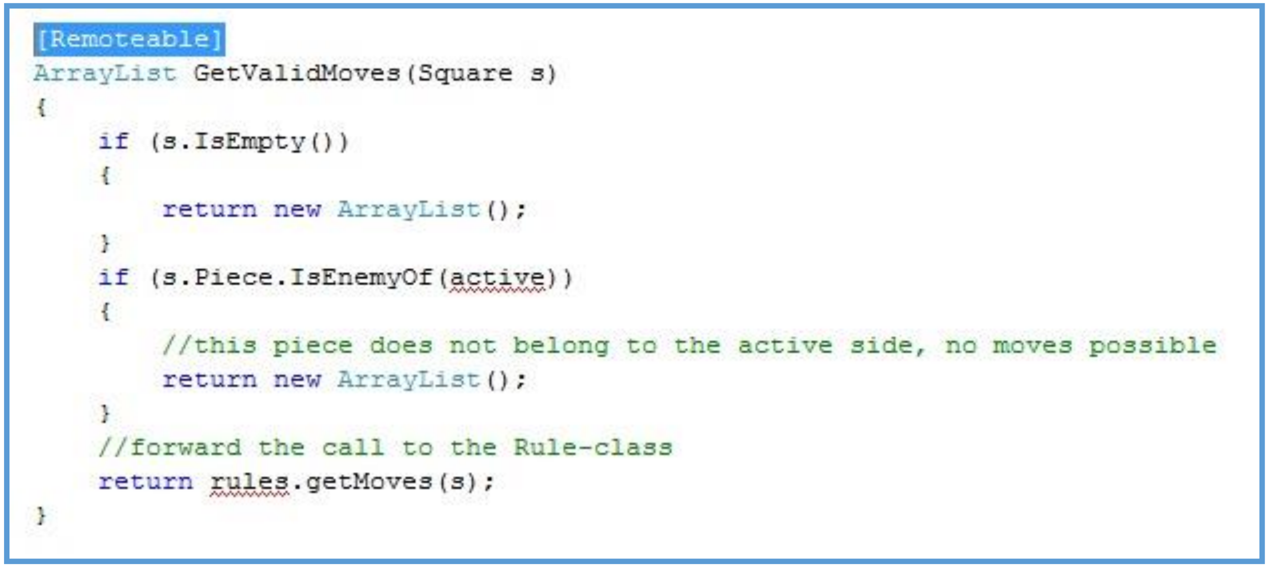
- Combine extensive profiling with an ILP (Integer Linear Programming) solver
  - Makes dynamic offload decisions
  - Optimize for energy reduction
  - Profile: energy for local execution vs. state transfer
- Leverage modern language runtime (.NET CLR)
  - Codes are portable between mobile and cloud
  - To simplify program partitioning

MAUI Architecture



How Does a Programmer Use MAUI?

- Goal: make it dead-simple to MAUI-ify apps
  - Build app as a standalone phone app
  - Add .NET attributes to indicate “remoteable”
    - UI and sensing can’t be remoteable.
  - Follow a simple set of rules

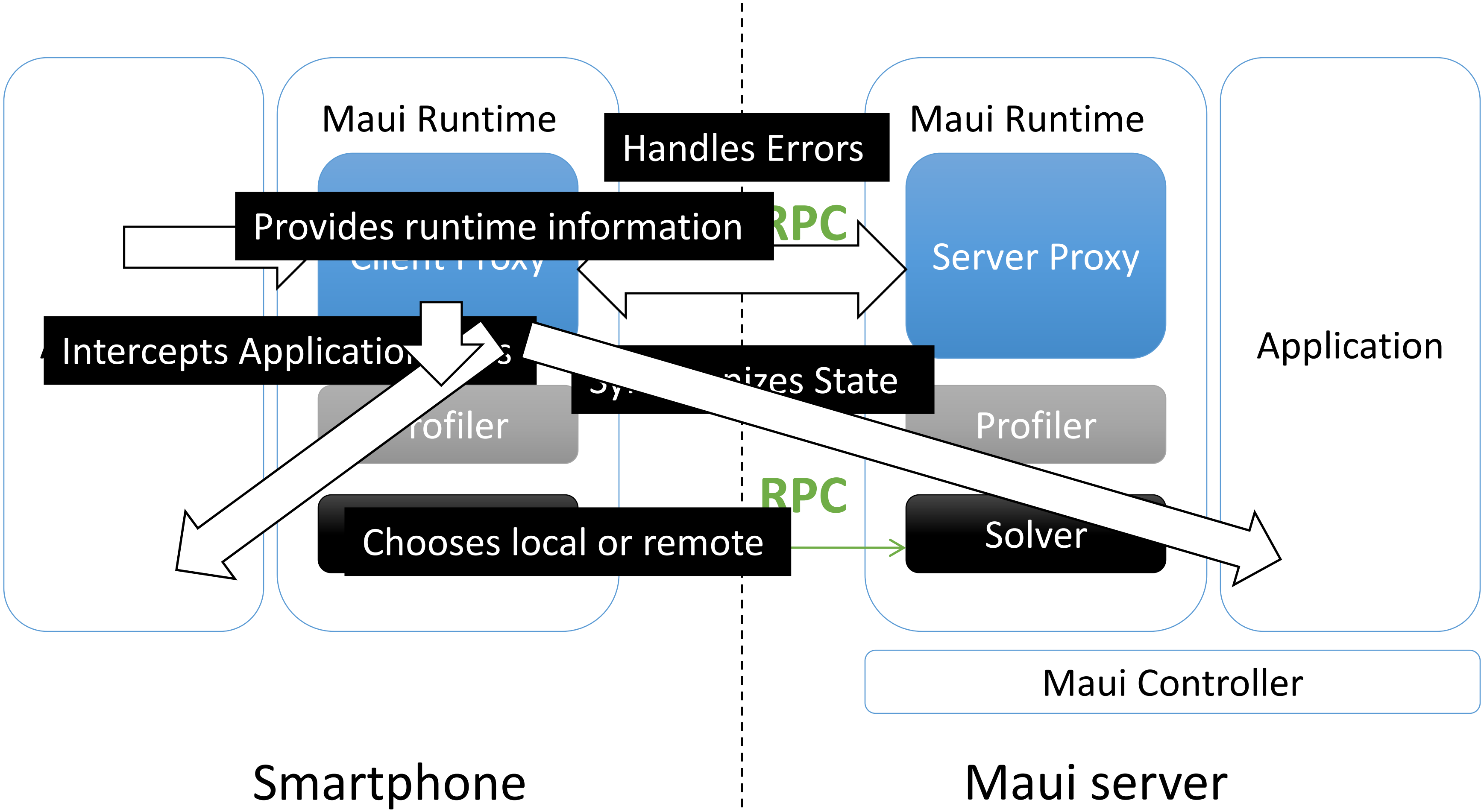


```
[Remoteable]
ArrayList GetValidMoves(Square s)
{
    if (s.IsEmpty())
    {
        return new ArrayList();
    }
    if (s.Piece.IsEnemyOf(active))
    {
        //this piece does not belong to the active side, no moves possible
        return new ArrayList();
    }
    //forward the call to the Rule-class
    return rules.getMoves(s);
}
```

Language Run-Time Support For Partitioning

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

MAUI Proxy



Maui Runtime

Handles Errors

Provides runtime information

RPC

Client Proxy

Server Proxy

Intercepts Application Calls

Synchronizes State

Profiler

Profiler

RPC

Chooses local or remote

Solver

Application

Maui Runtime

Maui Controller

Smartphone

Maui server

MAUI Proxy: Control and Data Transfer

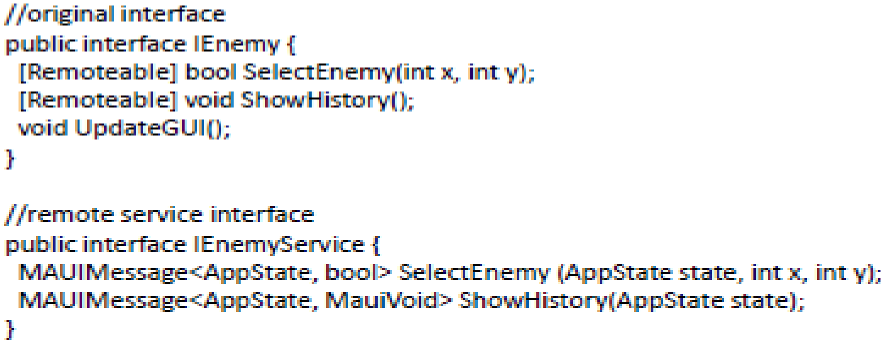
MAUI supports fine-grained offload at the method-level

At compile time:

- Find [remoteable] methods
- Produce client- and server-side stubs for all remoteable methods

At run time:

- Decide whether to invoke local or remote method
- Perform state synchronization when control transfers (in either direction)
  - Identify what program state to transfer
  - Serialize (deep copy): method parameters, class member variables, public sta tic members
  - Use deltas to reduce the data transfer overhead



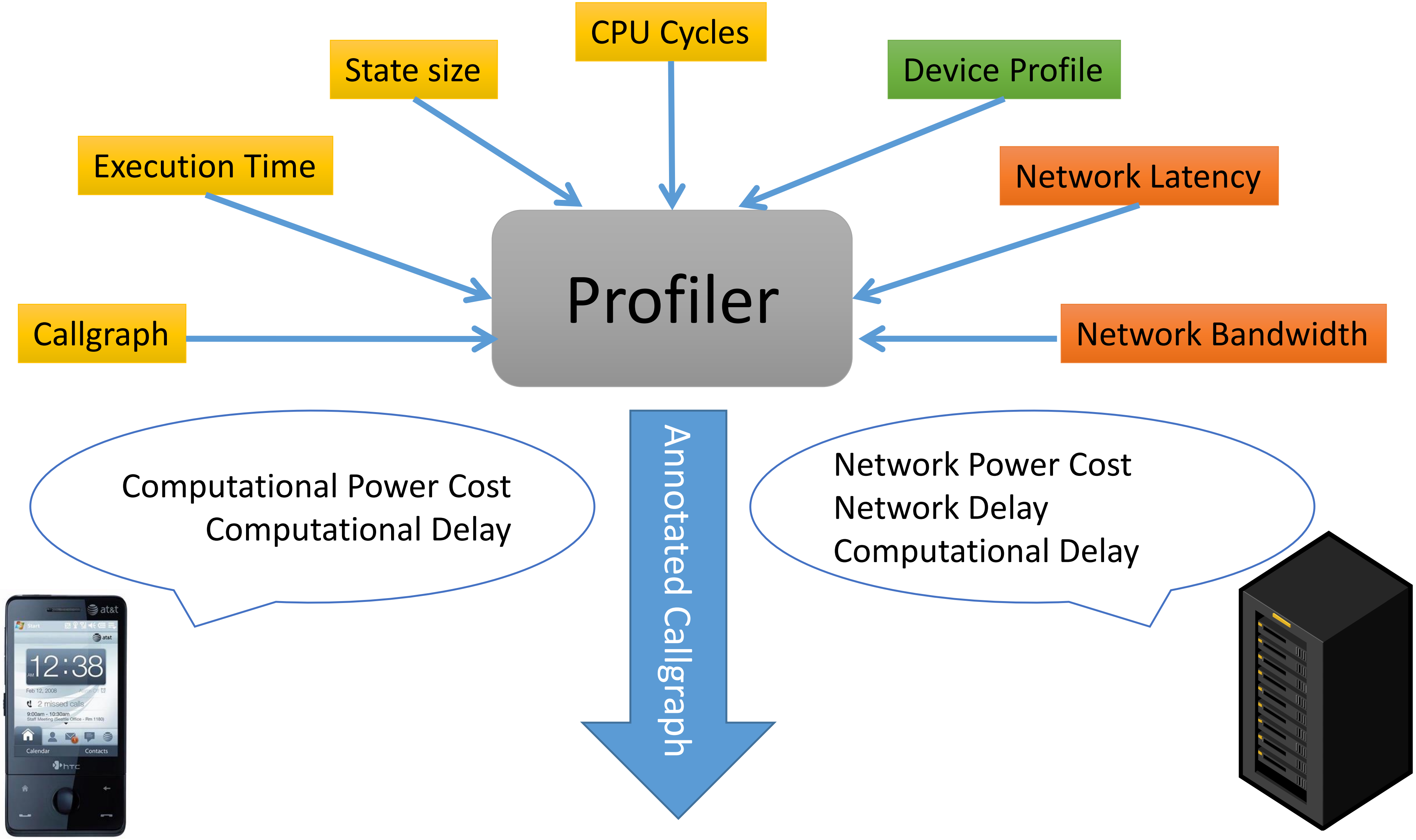
```
//original interface
public interface IEnemy {
  [Remoteable] bool SelectEnemy(int x, int y);
  [Remoteable] void ShowHistory();
  void UpdateGUI();
}

//remote service interface
public interface IEnemyService {
  MAUIMessage<AppState, bool> SelectEnemy (AppState state, int x, int y);
  MAUIMessage<AppState, MauiVoid> ShowHistory(AppState state);
}
```

MAUI: Why Not Static Partitioning?

- **Failure model:** when phone is disconnected, or even intermittently connected, applications don’t work
- **Device Scaling:** Developers need to revisit application structure as device characteristics change
- **Dynamic Context:** The portion of an app that makes sense to download changes based on the latency to the MAUI server

MAUI Profiler



CPU Cycles

State size

Device Profile

Execution Time

Network Latency

Callgraph

Profiler

Network Bandwidth

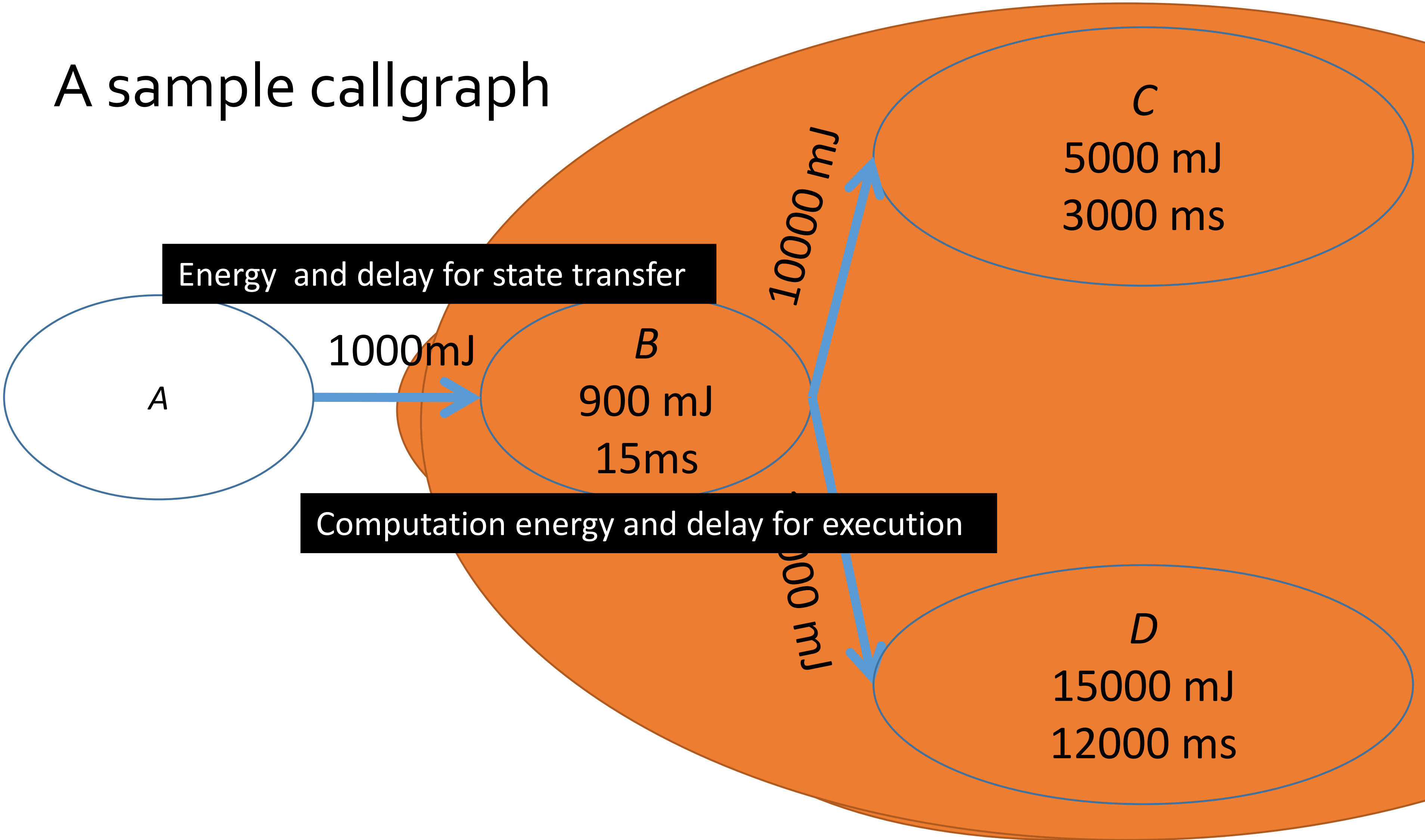
Annotated Callgraph

Computational Power Cost
Computational Delay

Network Power Cost
Network Delay
Computational Delay

MAUI Solver

A sample callgraph

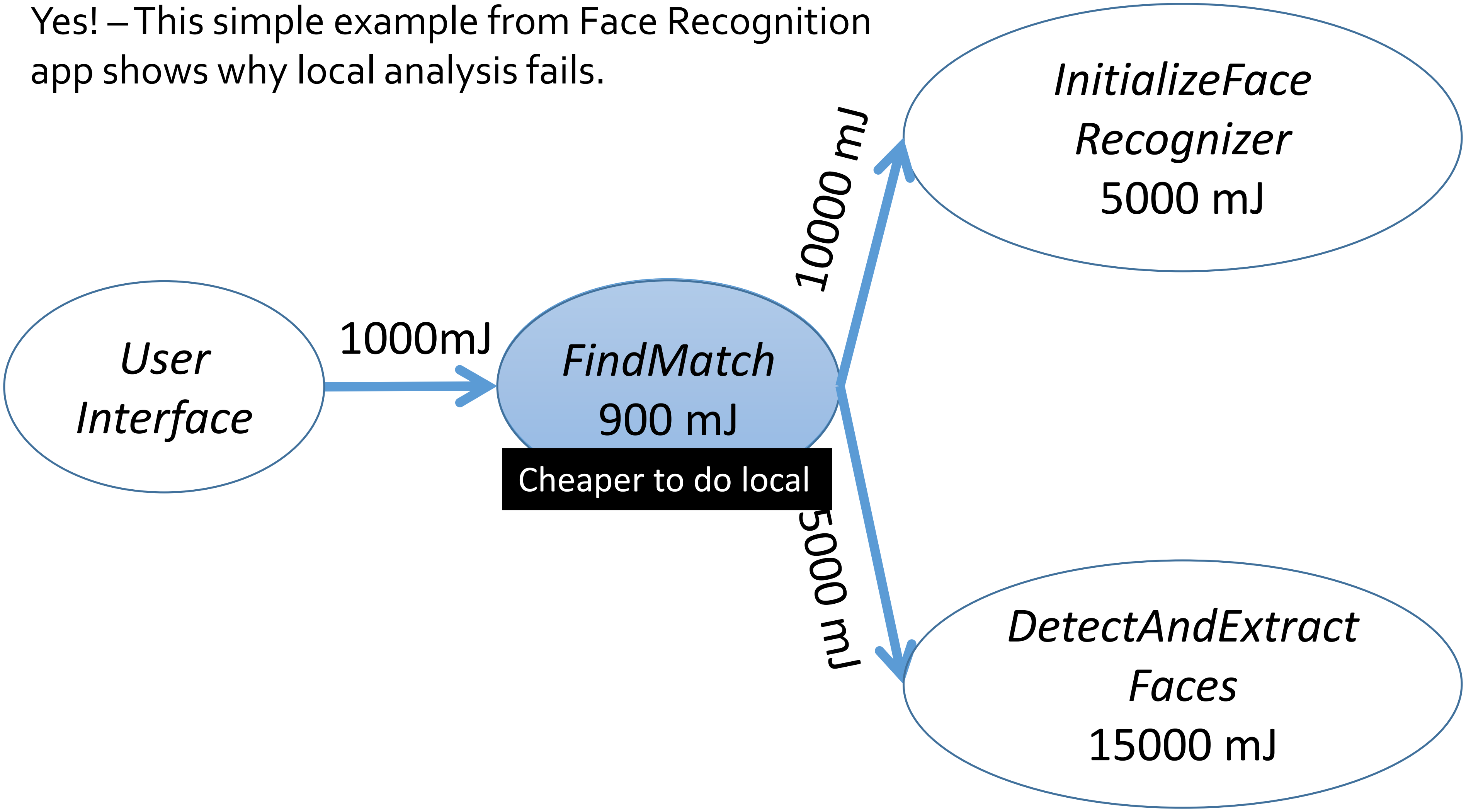


Energy and delay for state transfer

Computation energy and delay for execution

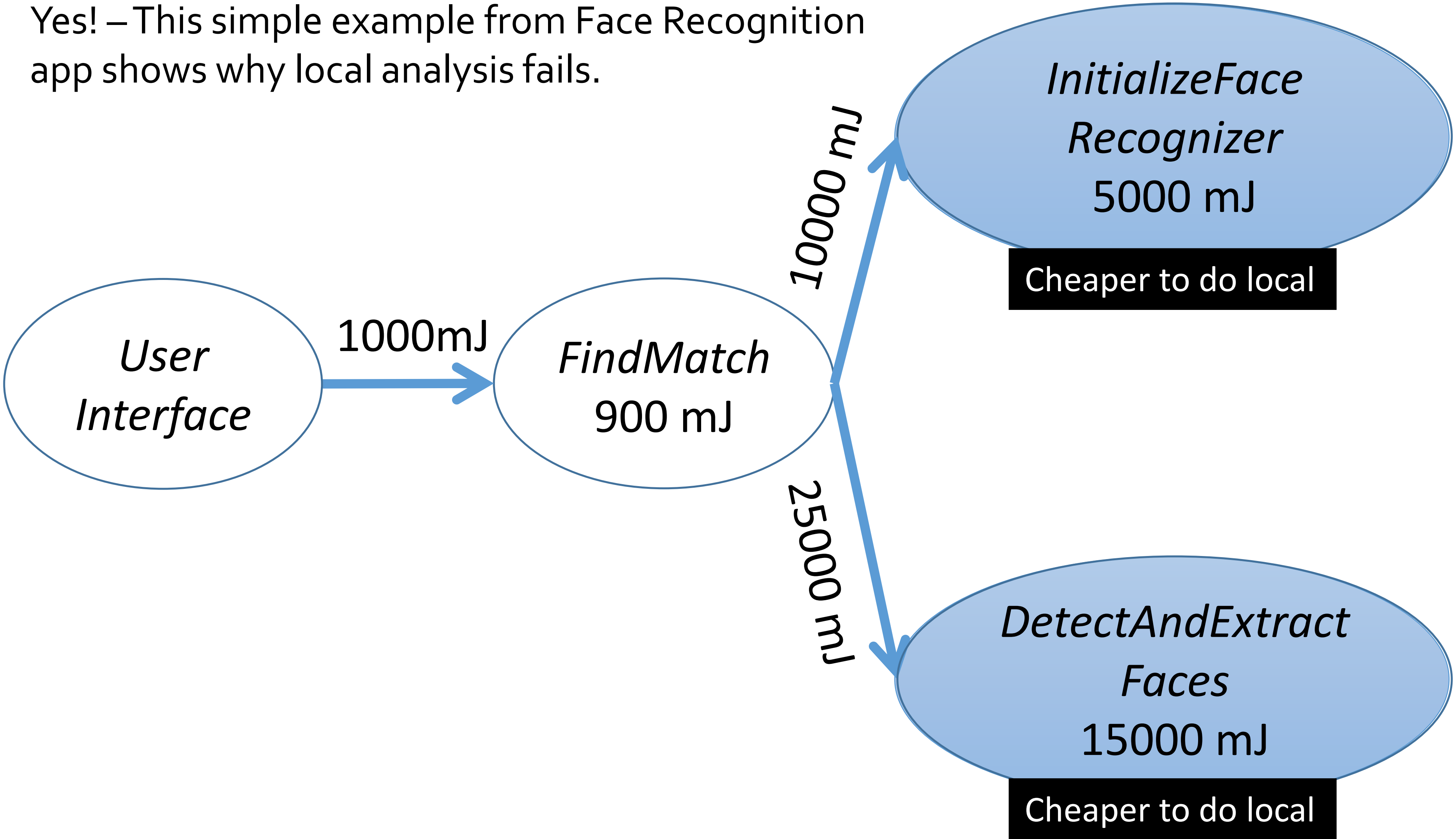
Is Global Program Analysis Needed?

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

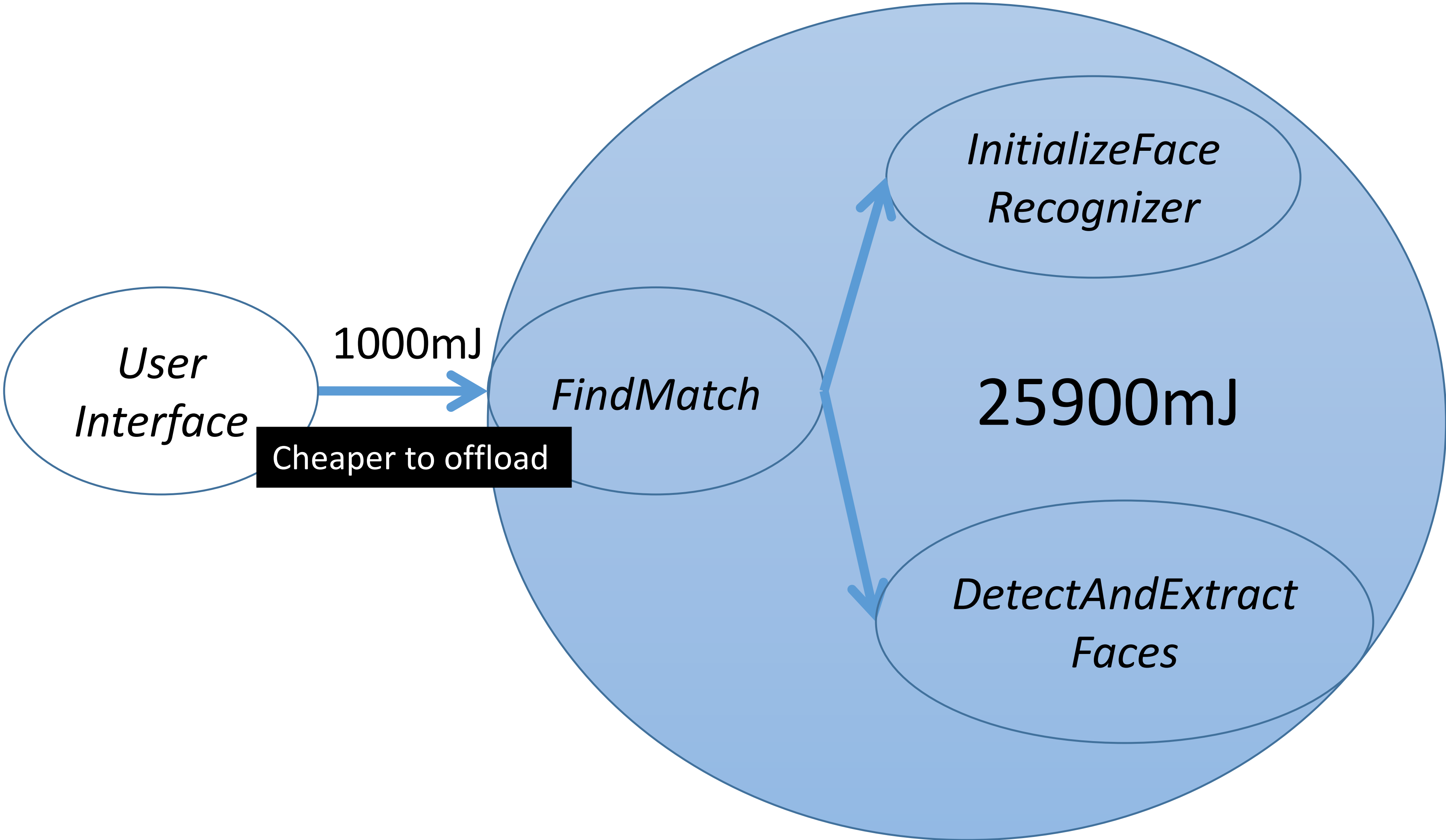


Is Global Program Analysis Needed?

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



Is Global Program Analysis Needed?



Cheaper to offload

The Actual Optimization Problem

- An ILP Formulation
  - Solved on the cloud for efficiency reasons

$$\text{maximize} \sum_{v\in V} I_v \times E_v^l - \sum_{(u,v)\in E} |I_u - I_v| \times C_{u,v}$$

I_v: Indicator=1 if remote execution

$$\text{such that:} \sum_{v\in V} ((1 - I_v) \times T_v^l) + (I_v \times T_v^r))$$

$$+ \sum_{(u,v)\in E} (|I_u - I_v| \times B_{u,v}) \le L$$

Time to execute method in a location
Different than its upstream neighbor

$$\text{and} \qquad I_v \le r_v, \forall v \in V$$

Only remoteable methods can
be executed remotely

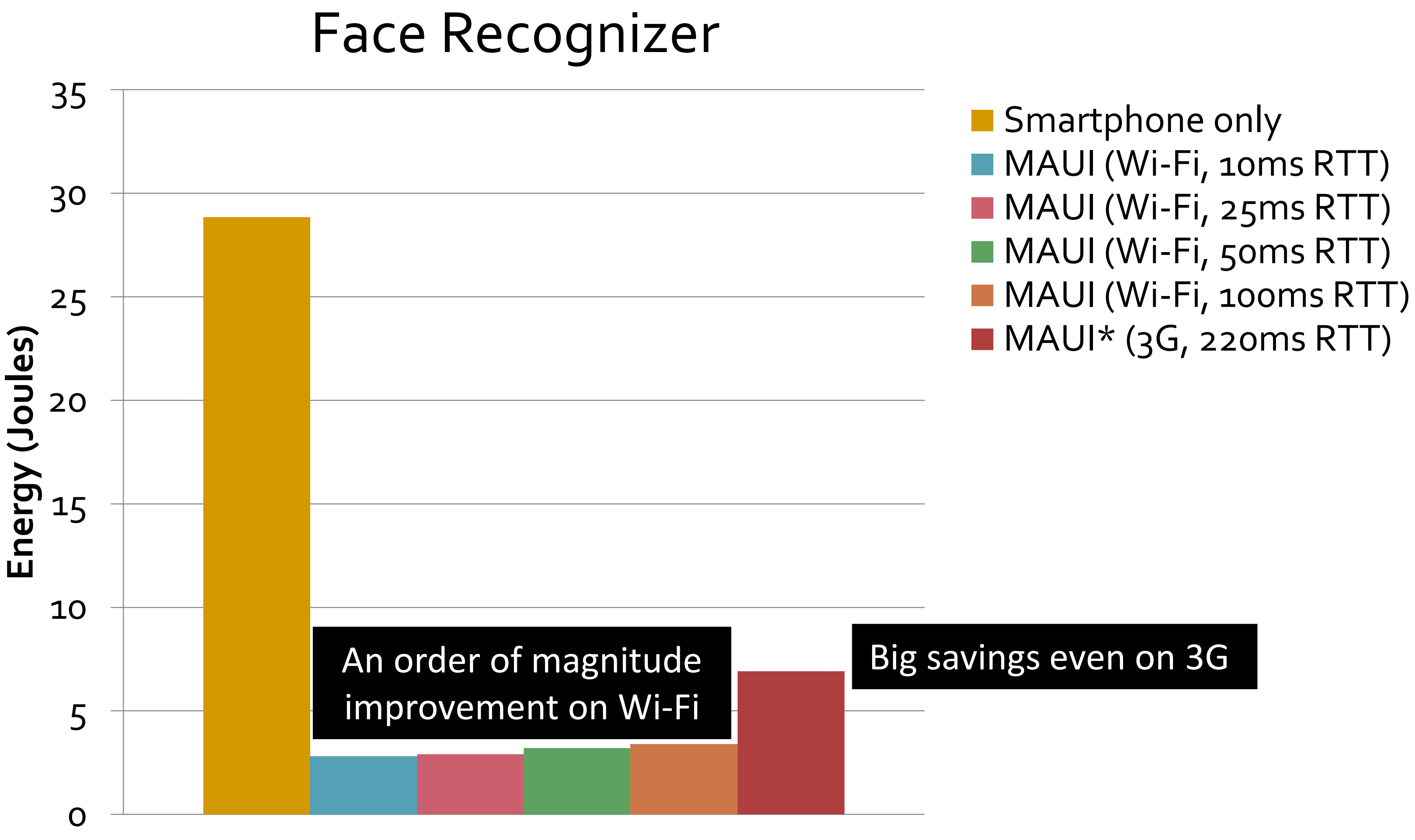
MAUI Implementation

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

Questions

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

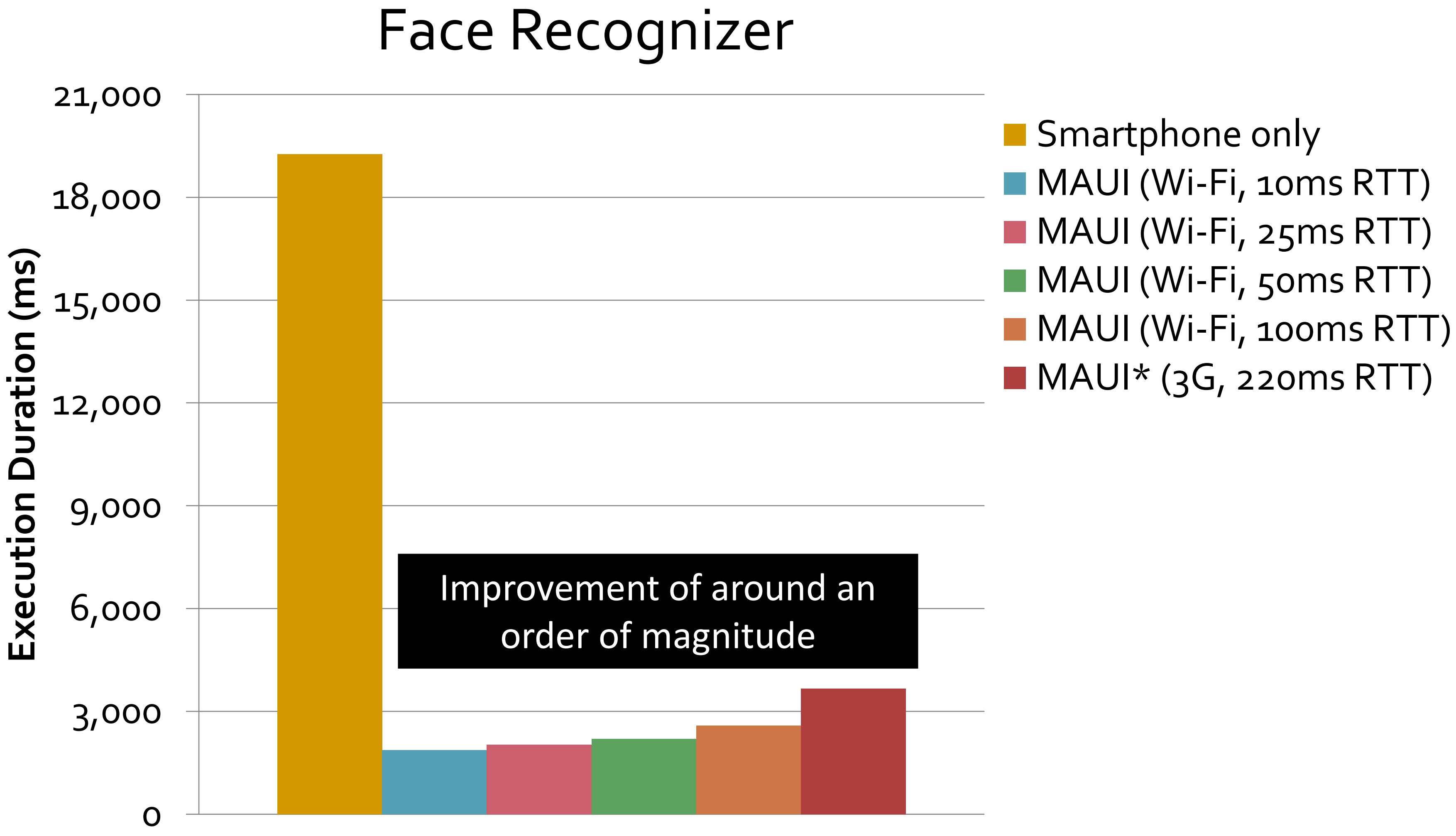
How Much can MAUI Reduce Energy Consumption?



An order of magnitude improvement on Wi-Fi

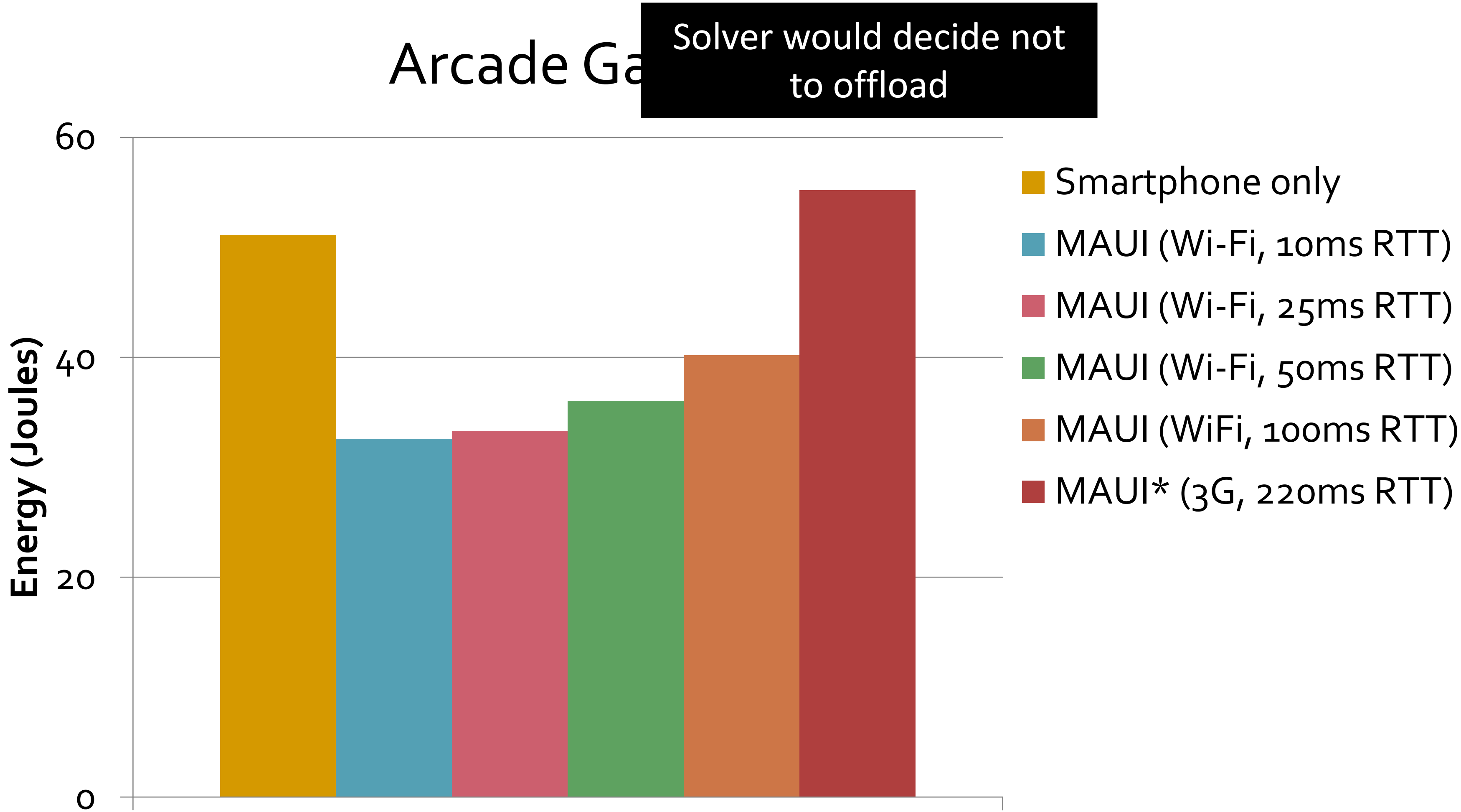
Big savings even on 3G

How Much can MAUI Improve Performance?



Improvement of around an order of magnitude

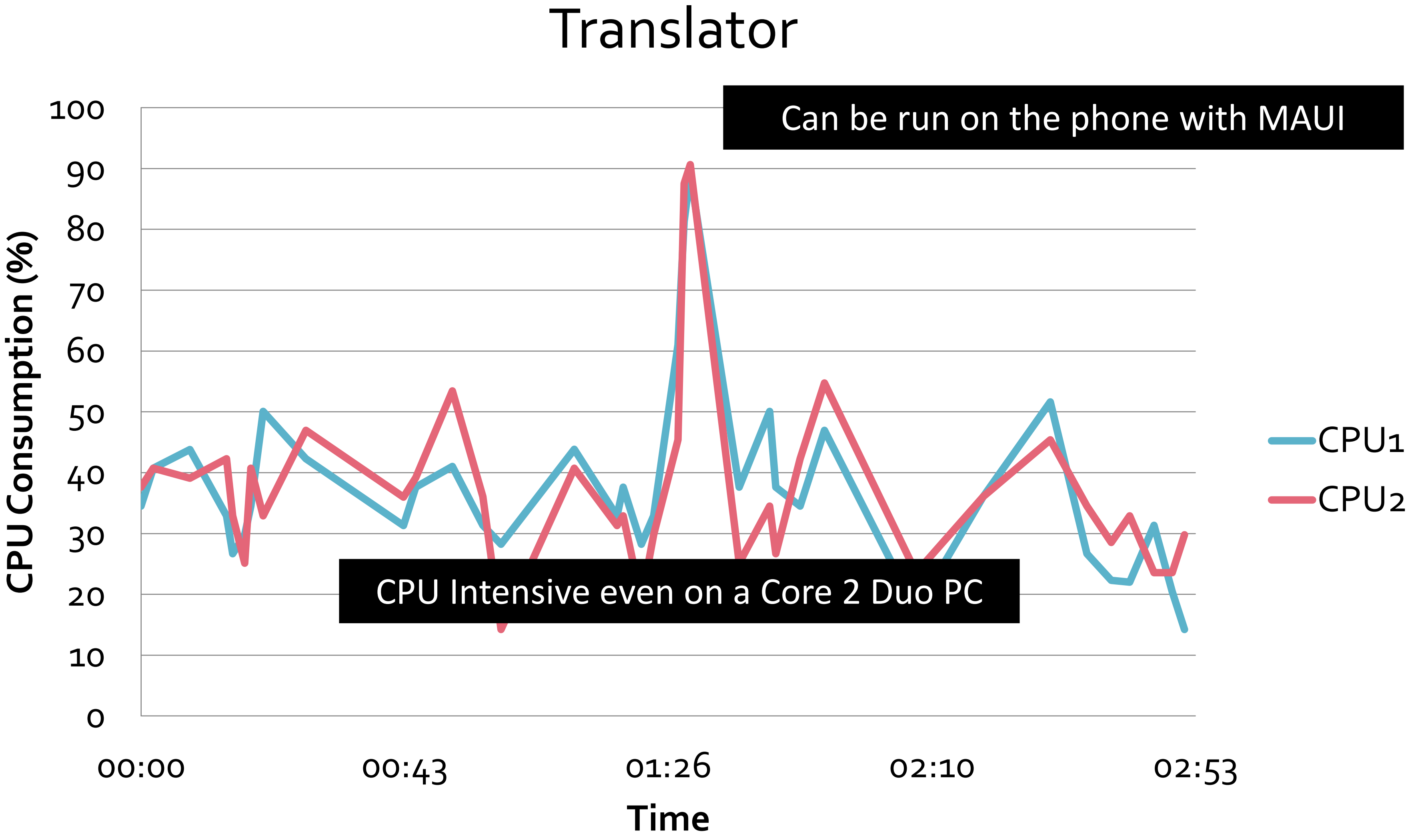
Latency to Server Impacts the Opportunities for Fine-Grained Offload



Solver would decide not to offload

Up to 40% energy savings on Wi-Fi

Can MAUI Run Resource-Intensive Applications?



Can be run on the phone with MAUI

CPU Intensive even on a Core 2 Duo PC

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