Mobile Cloud Computing

Cloudlets, Offloading, and other Platform Specific Concerns

Main Benefits of MCC

- Increased computational speed
  - Better responsiveness
- Decreased energy expenditure
  - Better/longer battery life
- General increase in ability
  - A resource poor device can run resource intensive programs
  - Access to large data sets that can't be stored on the device constantly

Briefly: Why This Matters to Non-Nerds

- Besides battery designers and network engineers, who cares? What cool things can we do?
- Mobile devices are with us constantly.
  - Leveraging MCC allows for more computationally intensive games
  - We can go to a new doctor and pull up our medical records without them being on the device constantly (and therefore at risk if lost or stolen)
  - Translation services for tourists (common *ad hoc* example)

The Cloud Archetypes

- Three basic types of clouds
  - *ad hoc* local cloud
    - Share the cost
    - Don't have to send data far
    - Big overhead
    - You share the cost
  - Cloudlet
    - Nearby
    - Powerful
    - Requires additional infrastructure
    - Variable overhead
  - Distant Cloud – what we deal with most of the time

ad hoc Mobile Clouds

- Good for localized tasks
  - Throw more processors at the problem
  - Better for parallelizable than serial tasks
- Requires overhead
  - Devices need to communicate to each other
  - Security concerns – Huge!
  - Technological constraints due to number of users
- Time savings vs. Energy Savings
  - You can farm out, and be farmed out, tasks
  - Can cost more energy than doing your tasks by yourself, but results are returned quicker
  - You pay energy cost for tasks others farm out to you

Cloudlets

- Distributed infrastructure
  - Minimize hops from device
  - Expensive (money)
- More powerful than *ad hoc*
  - Speedy processors
  - You aren't paying energy cost for others from your device; TANSTAAFL
  - Can effectively offload more (in terms of size) as well as a wider variety of tasks (serialized as well as parallel)
- Security concerns
  - Same ones we generally deal with
  - Tracking!

Distant Clouds (in MCC)

- Big and Fast
  - Can handle the most resource intensive of tasks
- Far away
  - More hops = lower QoS
- Allows for task-specific optimization
  - Case dependent, but cloud might have data and methods stored so you don't need to transmit as much as with other cloud types
- Generally where you will access data from

Wi-Fi: Users, Hops, and QoS

- The amount of traffic on the wireless access point can negatively impact QoS.
  - 10 MB file used in study.
  - The transfer time doesn't change much until about 20 users.
  - Between 20 and 25 big jump; architecture based. We lose packets due to interference or have to wait for our turn, depending on the implementation.
- The number of hops to the cloud also affects the transfer time of application and its running states.
  - "The scenario is simulated with 15 wireless devices for transferring the 10 MB file." (E. Ahmed, *et al.*)

Wi-Fi: Users and QoS

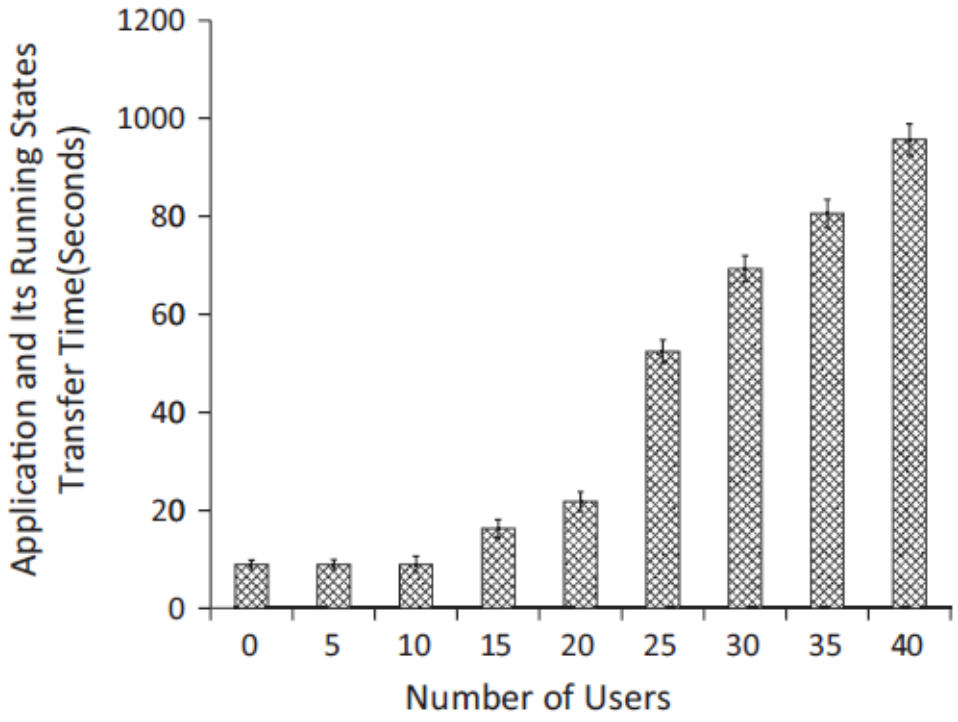


Fig. 4. Number of users effect on application and its states transfer time in WLAN.

Wi-Fi: Hops and QoS

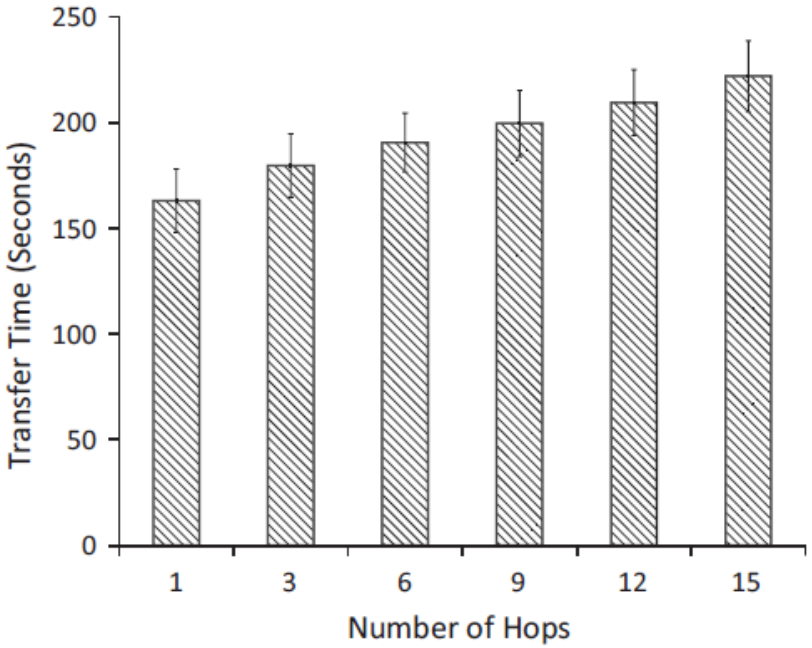


Fig. 12. Effect of number of hops to the cloud on transfer time of file containing the application and its states.

E. Ahmed *et al.*

Computational Offloading

- Generally follows the middle road
  - We gain no benefit from keeping everything local
  - The problem of full virtualization
    - Device intrinsic methods (gps, camera, etc.)
    - Responsiveness
    - Bandwidth (costs energy and time)
- Approaches to partitioning
  - Systems of linear equations
  - Solving optimal graphs
  - Hybrid (combinations of linear and graph)
  - Exceptional (none of the above)

Application Partitioning: Overview

Table 5
Comparison of application partitioning algorithms for MCC.

APAs	PG	PM	PLS	PR	AD	AT	AN
Data Stream Application (Yang et al., 2012b)	Component	Hybrid	Multiple	Network	Online	Dynamic	Automatic
HGG (Abebe and Ryan, 2011)	Hybrid	Graph	Single	Software, hardware	Online	Static	Automatic
Distributed Abstract Class Graph (Abebe and Ryan, 2012)	Class	Graph	Single	N/A	Offline	Dynamic	N/A
OLIE (Gu et al., 2003)	Class	Graph	Single	software, network	Online	Dynamic	Manual
Adaptive Multi-Constrained Partitioning (Ou et al., 2006)	Class	Graph	Single	All	Online	Dynamic	N/A
Automated Refactoring (Jamwal and Iyer, 2005)	Hybrid	Graph	Single	N/A	Offline	Dynamic	Manual
Dynamic Software Deployment (Giurgiu et al., 2012)	Bundle	Graph	Single	All	Hybrid	Dynamic	Automatic
J-Orchestra (Tilevich and Smaragdakis, 2006)	Object	Exceptional	Single	Software	Offline	Dynamic	Manual
Parametric Analysis (Wang and Li, 2004)	Task	Graph	Universal	N/A	Online	Dynamic	Manual
RCMCPP (Yang et al., 2012a)	Module	LP	N/A	Network, hardware	Online	Dynamic	Automatic
Multi-site Computation Offloading (Sinha and Kulkarni, 2011)	Allocation-site	Hybrid	Single	N/A	Online	Static	Automatic
Calling the Cloud (Giurgiu et al., 2009)	Bundle	Graph	Single	Software, hardware	Hybrid	Dynamic	Automatic
CloneCloud (Chun et al., 2011)	Thread	Hybrid	Multiple	Network, hardware	Online	Static	Automatic
MACS (Kovachev and Klamma, 2012)	Bundle	LP	Single	Software, hardware	Online	Dynamic	Automatic
Improving Energy Efficiency (Ra et al., 2012)	Component	LP	Universal	N/A	Offline	Dynamic	Manual
MAUI (Cuervo et al., 2010)	Method	Hybrid	Single	All	Online	Dynamic	Manual
Partitioning Application (Smit et al., 2012)	Component	Graph	Multiple	N/A	Offline	Static	Manual
Dynamic Updates (Bialek et al., 2004)	Component	Exceptional	Single	N/A	Offline	Dynamic	Manual
Roam (Chu et al., 2004)	Component	Exceptional	Single	N/A	Online	Static	Manual
Complexity Prediction (Pedrosa et al., 2012)	Component	Graph	Single	N/A	Online	Static	Manual
Simplifying Cyber Foraging (Balan et al., 2007)	Module	Exceptional	Universal	N/A	Offline	Static	Manual
Energy-Optimal Software Partitioning (Goraczko et al., 2008)	Task	Hybrid	N/A	Software, hardware	Offline	Dynamic	Automatic
Wishbone (Newton et al., 2009)	Thread	Hybrid	Multiple	Network, hardware	Offline	Static	Manual
WORG (Niu et al., 2014)	Object	Graph	Single	Network	Online	Static	N/A
Graph Partitioning (Verbelen et al., 2013)	Component	Graph	Single	N/A	Hybrid	Dynamic	Manual

Note: PG: Partitioning Granularity, PM: Partitioning Model, PLS: Programming Language Support, PR: Profiler, AD: Allocation Decision, AT: Analysis Technique, AN: Annotation.

Graph Partitioning

Table 1
Strengths and weaknesses of graph-based application partitioning algorithms.

APAs	Graph type	Strengths	Weaknesses
Partitioning Application (Smit et al., 2012)	Dependency	Does not affect application development Does not require manual migration	Less visible for maintenance task
Parametric Analysis (Wang and Li, 2004)	Task control flow	Useful for determining necessary user annotation	Demands programmers effort for dummy parameter annotation
Complexity Prediction (Pedrosa et al., 2012)	Component	Demands low input effort from programmers	Needs programmers to balance metric utility and specify lightweight metric function
Calling the Cloud (Giurgiu et al., 2009)	Data flow	Does not need new infrastructure	Does not support platform-independent cooperative interaction Lacks of privacy, provenance and dynamic adaptation
OLIE (Gu et al., 2003)	Class	Minimizes component interactions between partitions	Might not offer the best partitioning solutions
Adaptive Multi-Constrained Partitioning (Ou et al., 2006)	Multi-cost	Provides better performance and effective adaptation	Not suitable for applications with a large number of components Incurs more resources which can cause performance degradation
Dynamic Software Deployment (Giurgiu et al., 2012)	Consumption graph	Makes it possible for an application to run in a resource-efficient manner	Needs modularized applications Demands prior accurate measurements
HGG (Abebe and Ryan, 2011)	Hybrid granularity	Provides finer level adaptation and more effective object topologies Computationally feasible	Only allowed to migrate partition to the same device
Distributed Abstract Class Graph (Abebe and Ryan, 2012)	Class	Allows cloud partitioning result to be on-loaded back to device Reduces remote object coupling and migration cost	Depends on the behavior of an application
Automated Refactoring (Jamwal and Iyer, 2005)	Internal dependency	Has better interoperability and context-awareness	Demands more effort from applicationdevelopers
WORG (Niu et al., 2014)	Object relational	Optimal bandwidth-adaptive partitioning	Applied to specific situations
Graph Partitioning (Verbelen et al., 2013)	Component	Higher probability to reach global optimum	Demands annotation input from application developers

J. Liu *et al.*

Linear Partitioning

Table 2
Strengths and weaknesses of linear programming-based application partitioning algorithms.

APAs	LP type	Strengths	Weaknesses
RCMCPP (Yang et al., 2012a)	MILP	Able to deal with an unpredictable number of users Saves cloud providers' operational cost	Incurs high overhead for identifying number of users and resources required
MACS (Kovachev and Klamma, 2012)	ILP	Dynamic and lightweight	Needs extra profiling and resource monitoring
Improving Energy Efficiency (Ra et al., 2012)	ILP	Derives accurate result that is closer to reality	Needs dynamic job scheduling technique

Hybrid Partitioning

Table 3
Strengths and weaknesses of hybrid application partitioning algorithms.

APAs	Graph type	LP type	Strengths	Weaknesses
MAUI (Cuervo et al., 2010)	Call	ILP	Reduces programmers burden Maximizes energy-saving Enables application to bypass the memory limit of SMDs	Demands high overhead due to continuous profiling
CloneCloud (Chun et al., 2011)	Control flow	ILP	Requires low overhead due to static analysis	Needs offline pre-processing for every newly built application Difficult to customize application functionality
Energy-Optimal Software Partitioning (Goraczko et al., 2008)	Task	ILP	Suitable for loosely coupled multiprocessor system	Ignores latency and energy consumption Inefficient for fast computation of task mapping
Wishbone (Newton et al., 2009)	Data flow	ILP	Uses the limited network resources efficiently	Not robust enough for dynamic sensor environments No flexibility for application re-engineering
Data Stream Application (Yang et al., 2012b)	Data flow	ILP	Has better scalability Likely to converge to the global optimal partition	Unsuitable for batch computation system
Multi-site Computation Offloading (Sinha and Kulkarni, 2011)	Object interaction	0–1LP	Allows finer offloading decision	Incurs unnecessary overhead

J. Liu *et al.*

Exceptional Partitioning

Table 4
Strengths and weaknesses of exceptional application partitioning algorithms.

APAs	Strengths	Weaknesses
Simplifying Cyber Foraging (Balan et al., 2007)	Easy to learn to create tactics file	Needs to create tactics file once for each application
	Language independent Has for runtime adaptation support	Tedious and prone to human errors
Roam (Chu et al., 2004)	Allows migration among heterogeneous devices in an effortless manner	Difficult to customize a device-independent representation for a particular device
Dynamic Updates (Bialek et al., 2004)	Enables transparent partitioning Simplifies and optimizes application updates	Needs partitioning guidelines
J-Orchestra (Tilevich and Smaragdakis, 2006)	Requires low effort input from programmers	Needs more efficient distribution middleware
	Excels on generality, flexibility, and degree of automation	Lack of automation to infer detailed object migration strategies in response to synchronous events

Algorithmic Goals

- Researchers have developed these different algorithms with different goals in mind
  - What we've talked about
    - Better performance
    - Energy savings
  - Additional common goals
    - Reducing network overhead
    - Eliminating memory constraints
    - Making things easier for the programmer
  - Special Cases
    - Multi-site offloading
    - Dynamic application updating (aimed at mirroring across devices)

Partitioning Objectives

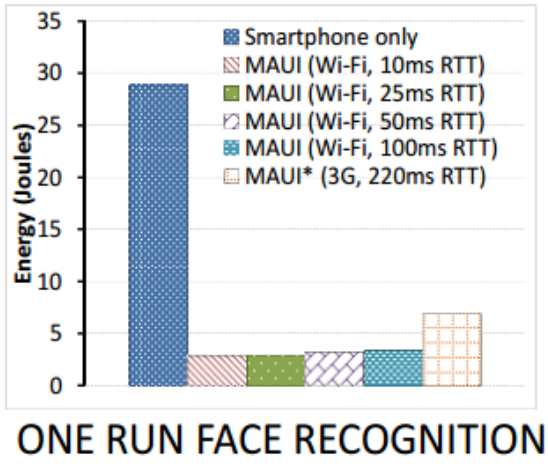
Table 6
Comparison of application partitioning algorithms based on the partitioning objective.

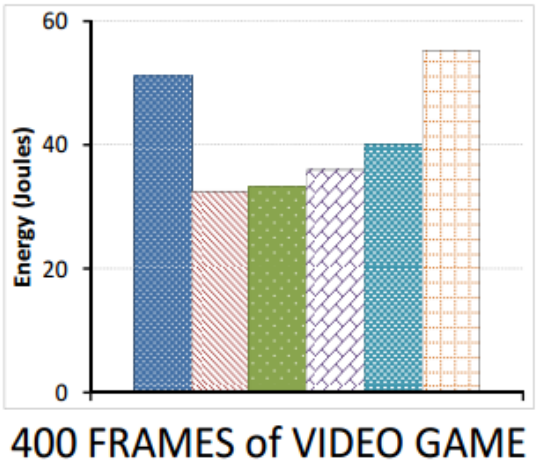
APAs	RNO	SE	IP	RMC	RPB	UAD	MO
Data Stream Application (Yang et al., 2012b)	No	No	Yes	No	No	No	No
HGG (Abebe and Ryan, 2011)	Yes	No	No	No	No	No	No
Distributed Abstract Class Graph (Abebe and Ryan, 2012)	Yes	Yes	Yes	Yes	No	No	No
OLIE (Gu et al., 2003)	No	No	No	Yes	No	No	No
Adaptive Multi-Constrained Partitioning (Ou et al., 2006)	No	No	Yes	No	No	No	No
Automated Refactoring (Jamwal and Iyer, 2005)	No	No	No	No	Yes	No	No
Dynamic Software Deployment (Giurgiu et al., 2012)	Yes	Yes	No	No	No	No	No
J-Orchestra (Tilevich and Smaragdakis, 2006)	No	No	No	No	Yes	No	No
Parametric Analysis (Wang and Li, 2004)	No	Yes	No	No	No	No	No
RCMCPP (Yang et al., 2012a)	No	No	Yes	No	No	No	No
Multi-site Computation Offloading (Sinha and Kulkarni, 2011)	No	No	No	No	No	No	Yes
Calling the Cloud (Giurgiu et al., 2009)	No	Yes	No	No	No	No	No
CloneCloud (Chun et al., 2011)	No	Yes	Yes	No	Yes	No	No
MACS (Kovachev and Klamma, 2012)	No	Yes	Yes	No	No	No	No
Improving Energy Efficiency (Ra et al., 2012)	No	Yes	No	No	No	No	No
MAUI (Cuervo et al., 2010)	No	Yes	Yes	No	Yes	No	No
Partitioning Application (Smit et al., 2012)	No	No	No	No	Yes	No	No
Dynamic Updates (Bialek et al., 2004)	No	No	No	No	No	Yes	No
Roam (Chu et al., 2004)	No	No	Yes	No	Yes	No	No
Complexity Prediction (Pedrosa et al., 2012)	No	Yes	No	No	No	No	No
Simplifying Cyber Foraging (Balan et al., 2007)	No	No	No	No	Yes	No	No
Energy-Optimal Software Partitioning (Goraczko et al., 2008)	No	Yes	No	No	No	No	No
Wishbone (Newton et al., 2009)	Yes	No	Yes	No	No	No	No
WORG (Niu et al., 2014)	No	Yes	Yes	No	No	No	No
Graph Partitioning (Verbelen et al., 2013)	No	Yes	No	No	Yes	No	No

Note: RNO: Reducing Network Overhead, SE: Saving Energy, IP: Improving Performance, RMC: Reducing Memory Constraints, RPB: Reducing Programmers Burden, UAD: Updating Application Dynamically, MO: Multi-site Offloading.

Real Results: MAUI

- MAUI is a hybrid partitioning algorithm
  - Energy savings





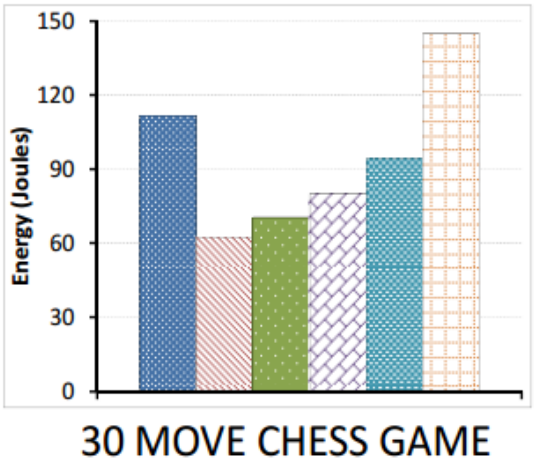
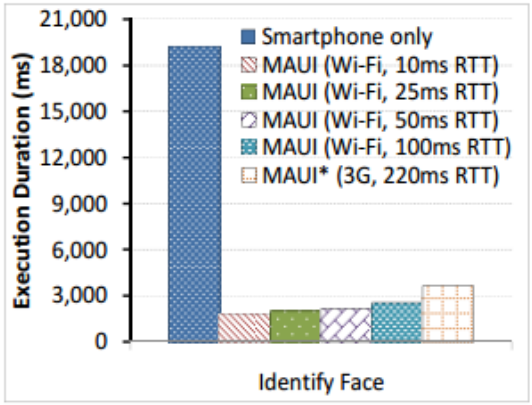


Figure 9: A comparison of MAUI's energy consumption. ***We compare the energy consumption of running three applications standalone on the smartphone versus using MAUI for remote execution to servers that are successively further away (the RTT is listed for each case).***

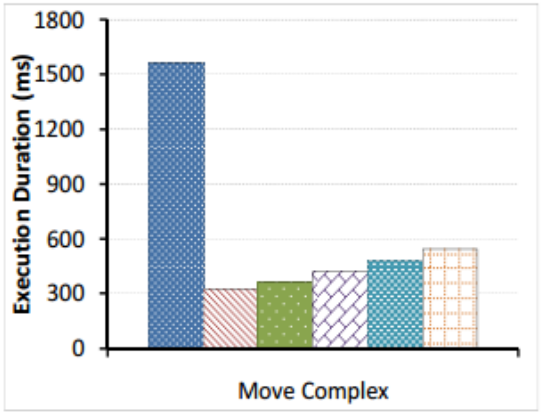
E. Cuervo *et al.*

Real Results: MAUI

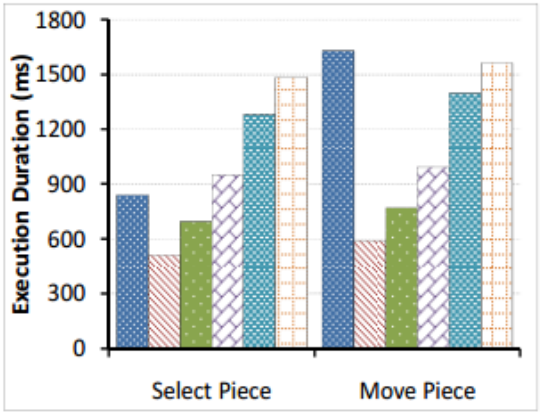
- MAUI is a hybrid partitioning algorithm
  - Better performance



ONE RUN FACE RECOGNITION



400 FRAMES of VIDEO GAME



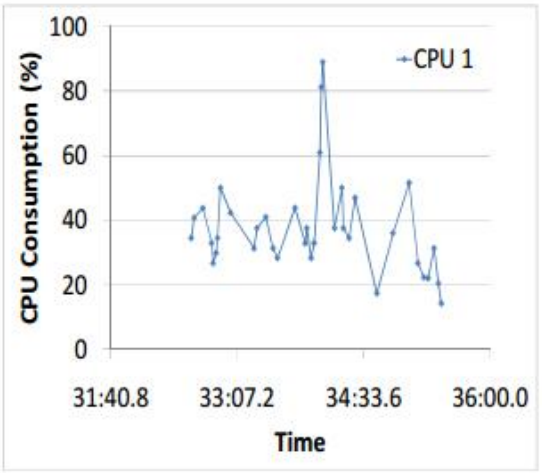
30 MOVE CHESS GAME

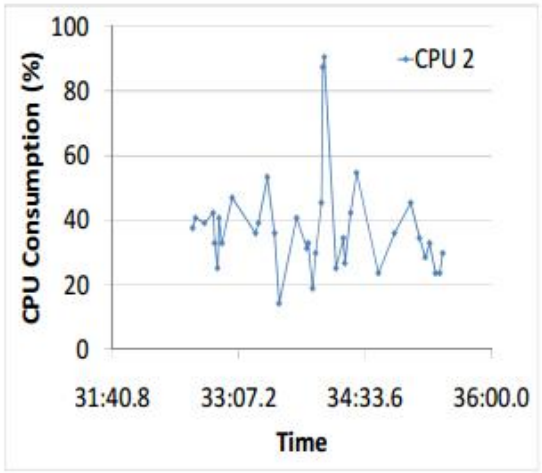
Figure 10: A comparison of MAUI's performance benefits. ***We compare the performance of running three application standalone on the smartphone versus using MAUI for remote execution to servers that are successively further away (the RTT is listed for each case).***

E. Cuervo *et al.*

Real Results: MAUI

- MAUI is a hybrid partitioning algorithm
  - Resource intensive operations for resource poor devices





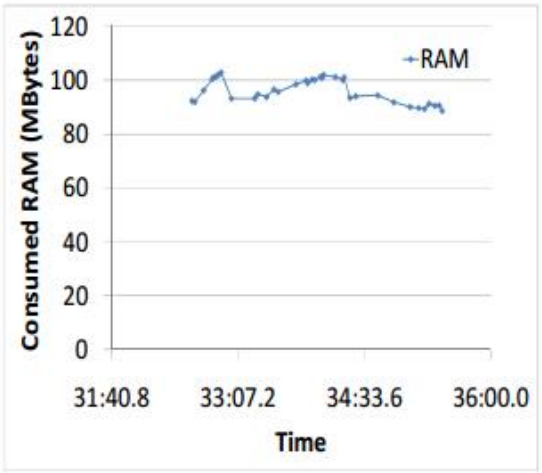


Figure 11: Profiling the Speech Recognition Engine on the MAUI Server. ***The MAUI server is a 2 GHz Intel Core2 CPU with 2 GB of RAM. The load of one CPU is shown in the graph on the left, whereas the other CPU is shown in the middle. The graph on the right shows the RAM consumed by the speech engine.***

E. Cuervo *et al.*

Questions?

References

Ejaz Ahmed, Adnan Akhunzada, Md Whaiduzzaman, Abdullah Gani, Siti Hafizah Ab Hamid, Rajkumar Buyya, Network-centric performance analysis of runtime application migration in mobile cloud computing, *Simulation Modelling Practice and Theory*, Volume 50, January 2015, Pages 42-56, ISSN 1569-190X, http://dx.doi.org/10.1016/j.simpat.2014.07.001.

Eduardo Cuervo, Aruna Balasubramanian, Dae-ki Cho, Alec Wolman, Stefan Saroiu, Ranveer Chandra, and Paramvir Bahl. 2010. MAUI: making smartphones last longer with code offload, *Proceedings of the 8th international conference on Mobile systems, applications, and services*(MobiSys '10). ACM, New York, NY, USA, 49-62. http://dx.doi.org/10.1145/1814433.1814441.

Jieyao Liu, Ejaz Ahmed, Muhammad Shiraz, Abdullah Gani, Rajkumar Buyya, Ahsan Qureshi, Application partitioning algorithms in mobile cloud computing: Taxonomy, review and future directions, *Journal of Network and Computer Applications*, Volume 48, February 2015, Pages 99-117, ISSN 1084-8045, http://dx.doi.org/10.1016/j.jnca.2014.09.009.