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 throughput 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”**
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

<table>
<thead>
<tr>
<th></th>
<th>MAUI</th>
<th>CloneCloud</th>
<th>Odessa</th>
<th>Orleans</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remote execution unit</td>
<td>Methods (RMI)</td>
<td>Threads</td>
<td>Tasks</td>
<td>Grains</td>
</tr>
</tbody>
</table>

4/2/12
Cellular Networks and Mobile Computing (COMS 6998-8)
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

- Callgraph
- State size
- CPU Cycles
- Device Profile
- Network Latency
- Network Bandwidth
- Execution Time

Computed Power Cost
Computational Delay

Annotated Callgraph

Network Power Cost
Network Delay
Computational Delay

Source: MAUI Mobisys presentation 4/2/12
A sample callgraph

Energy and delay for state transfer

Computation energy and delay for execution

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

Cellular Networks and Mobile Computing (COMS 6998-8)
Is Global Program Analysis Needed?

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

User Interface 1000mJ FindMatch 900 mJ

InitializeFace Recognizer 5000 mJ Cheaper to do local

DetectAndExtract Faces 15000 mJ Cheaper to do local

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
Can MAUI Adapt to Changing Conditions?

*Missiles take around 60 bytes each*

Source: MAUI Mobisys presentation
Case 1

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

DoFrame \rightarrow DoLevel \rightarrow HandleEnemies
DoLevel \rightarrow HandleBonuses
DoLevel \rightarrow HandleMissiles

*Missiles take around 60 bytes each
Computation cost is close to zero

Source: MAUI Mobisys presentation
Case 2

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

*Missiles take around 60 bytes each

Source: MAUI Mobisys presentation
4/2/12

Cellular Networks and Mobile Computing (COMS 6998-8)
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?
How much can MAUI reduce energy consumption?

Face Recognizer

- Smartphone only
- MAUI (Wi-Fi, 10ms RTT)
- MAUI (Wi-Fi, 25ms RTT)
- MAUI (Wi-Fi, 50ms RTT)
- MAUI (Wi-Fi, 100ms RTT)

An order of magnitude improvement on Wi-Fi

Big savings even on 3G

Source: MAUI Mobisys presentation
How much can MAUI improve performance?

Face Recognizer

- Smartphone only
- MAUI (Wi-Fi, 10ms RTT)
- MAUI (Wi-Fi, 25ms RTT)
- MAUI (Wi-Fi, 50ms RTT)
- MAUI (Wi-Fi, 100ms RTT)
- MAUI* (3G, 220ms RTT)

Improvement of around an order of magnitude

Source: MAUI presentation
Latency to server impacts the opportunities for fine-grained offload

Arcade Game

Solver would decide not to offload

Up to 40% energy savings on Wi-Fi

Source: MAUI Mobisys presentation
Can MAUI Run Resource-Intensive Applications?

CPU Consumption (%)

Time

CPU Intensive even on a Core 2 Duo PC

Can be run on the phone with MAUI

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
CloneCloud v1 Architecture

Mobile Device

- Application
- App. VM
- Migration
- Profiler
- Runtime
- OS
- HW

Clone VM

- Application
- Migration
- Profiler
- App. VM
- OS
- Virtual HW
- VMM
- HW

Manager

Partition Analyzer

OS

Virtual HW

4/2/12

Cellular Networks and Mobile Computing (COMS 6998-8)

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
Partitioning

Where to partition code, satisfying any constraints for correctness, so as to optimize for current conditions
Automatic Partitioning Framework

Identify valid split points

Application binary

Profiling inputs

Static analysis

Dynamic profiling

Optimization solver

Optimal partitioning points

Cost-annotated executions

Pick best choice for an objective

Construct cost models

Partition Analyzer

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
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
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
Distributed Execution

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

Mobile Device

<table>
<thead>
<tr>
<th>Application</th>
<th>Migrator</th>
<th>App-VM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Migrator</td>
<td>App-VM</td>
</tr>
<tr>
<td>Migrator</td>
<td></td>
<td></td>
</tr>
<tr>
<td>App-VM</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Clone

<table>
<thead>
<tr>
<th>Clone</th>
<th>Migrator</th>
<th>App-VM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone</td>
<td>Migrator</td>
<td>App-VM</td>
</tr>
</tbody>
</table>

Partitions

Net

(1) (2) (3) (4)

Cellular Networks and Mobile Computing (COMS 6998-8)

Courtesy: Byung-Gon Chun et. al.
Migration: Suspend-Transmit-Resume

At migration point:
- Suspend thread
- Capture context
- Transmit
- Launch process
- Patch migrated context
- Resume thread

At reintegration point:
- Transmit
- Merge migrated context
- Resume thread
- Suspend thread
- Capture context

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
Migration: References in Motion

(1) Mobile Device

<table>
<thead>
<tr>
<th>Reference</th>
<th>MID</th>
<th>CID</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>1</td>
<td>null</td>
</tr>
<tr>
<td>0x02</td>
<td>2</td>
<td>null</td>
</tr>
<tr>
<td>0x03</td>
<td>3</td>
<td>null</td>
</tr>
</tbody>
</table>

(2) Clone

<table>
<thead>
<tr>
<th>Reference</th>
<th>MID</th>
<th>CID</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x21</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>0x22</td>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>0x23</td>
<td>3</td>
<td>13</td>
</tr>
<tr>
<td>0x24</td>
<td>null</td>
<td>14</td>
</tr>
<tr>
<td>0x22</td>
<td>null</td>
<td>15</td>
</tr>
</tbody>
</table>

(3) Mobile Device

<table>
<thead>
<tr>
<th>Reference</th>
<th>MID</th>
<th>CID</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>0x02</td>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>0x03</td>
<td>3</td>
<td>13</td>
</tr>
<tr>
<td>0x04</td>
<td>4</td>
<td>14</td>
</tr>
<tr>
<td>0x05</td>
<td>5</td>
<td>15</td>
</tr>
</tbody>
</table>
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
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
Execution Time Comparison (Image Search)

- **Phone (Local)**
- **CloneCloud (Wi-Fi)**
- **CloneCloud (3G)**

Execution time (seconds)

- 1 image: 1x
- 10 images: 20x
- 100 images: 20x

Courtesy: Byung-Gon Chun et al.
Energy Consumption Comparison
(Image Search)

- **Phone (Local)**
- **CloneCloud (Wi-Fi)**
- **CloneCloud (3G)**

Energy consumption (J)

- **1 image**
  - Phone (Local): 1x
  - CloneCloud (Wi-Fi): 1x
  - CloneCloud (3G): 1x

- **10 images**
  - Phone (Local): 20x
  - CloneCloud (Wi-Fi): 20x
  - CloneCloud (3G): 20x

- **100 images**
  - Phone (Local): 20x
  - CloneCloud (Wi-Fi): 20x
  - CloneCloud (3G): 20x

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
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
mCloud Programming Model: Odessa

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

Source: Odessa Mobisys presentation
mCloud Programming Model: Odessa (Cont’d)

- Offloading DEcision System for Streaming Applications
mCloud Programming Model: Odessa (Cont’d)

Cloud Infrastructure

Network

B2
B1
C
A

Network

B
C
A

Screen

Incremental decisions adapt quickly to input and platform variability.

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

Application

Orleans

.NET

Azure

Servers
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)

Customer Grain

Grain (actor)

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>“John Doe”</td>
</tr>
<tr>
<td>Email</td>
<td>“<a href="mailto:john.doe@hotmail.com">john.doe@hotmail.com</a>”</td>
</tr>
<tr>
<td>Address</td>
<td>“123 Main St., Anywhere UR 01234”</td>
</tr>
<tr>
<td>Products</td>
<td></td>
</tr>
</tbody>
</table>

Methods

Checkout

AddProduct

RemoveProduct

Grain ID1

Grain ID2

State

Message Queue

Messages

4/2/12

Cellular Networks and Mobile Computing (COMS 6998-8)

Courtesy Jim Larus
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