CLONECLOUD: ELASTIC EXECUTION BETWEEN MOBILE DEVICE AND CLOUD

Byung-Gon Chun

Joint work with Sunghwan Ihm (Princeton)
Petros Maniatis (Intel), Mayur Naik (Intel), Ashwin Patti (Intel)
Mobile Apps Gain Rapid Popularity
Users Increasingly Expect More from Apps

• Examples
  – Mobile augmented reality
  – Visual/audio search
  – Image/video/speech processing
  – Citizen science
  – Intrusion detection, exfiltration protection
The Mobile-Resource Disparity

• Handheld devices still in the ’90s
  – Constrained by thermal profile & size
Project Goal

Enable expensive, exotic applications on mobile devices without increase in device footprint or required capabilities
Observations

• Powered devices more powerful than ever
  – Have them at home, and there’s cloud

• Lots of faster wireless connectivity
Use Ambient Resources

• Use accessible clouds, computing hotspots (e.g., cloudlets), available desktops or plugged-in laptops nearby

• **Augment** mobile device applications through ambient resource execution to make its applications better

---

Cyber foraging (Balan et al. 2002)

Abundant computation, lots of faster connectivity
Approach: Dynamic Splitting of Apps

• Client-Server model?
  – Split is made manually, early
  – Split persists even when hardware, networks, costs, workload, urgency change

• Dynamically shift computation between mobile device and cloud
What Does This Look Like?

Originally

Slow device, Cloud available

100%

10%
What Does This Look Like?

Originally

Fast device, Cloud available

Phone

Application

100%

MID

Application

80%
Talk Outline

• Introduction
• CloneCloud design
  – Clones
  – Partitioning
  – Migration
• Evaluation
• Conclusion
CloneCloud v1 Strategy

• Partition apps written for a mobile device automatically

• Seamlessly execute locally & remotely

• Optimize high-level objective (time, energy) by adaptation

• Exploit application-level VMs
The Before-After Picture

Phone

UI

W

Clone

Phone

UI

W

Cloud

W'

Clone

Clone

Clone
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

Construct cost models

Static analysis

Dynamic profiling

Optimization solver

Optimal partitioning points

Partitioning constraints

Pick best choice for an objective

Profiling inputs

Construct cost models

Partitioning constraints

Optimization solver

Pick best choice for an objective
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

- **Node Manager**
- **Partitions**

**Mobile Device**
- Application
  - Migrator
  - App-VM

**Clone**
- Clone
  - Migrator
  - App-VM

(1) (2) (3) (4)
Migration: Suspend-Transmit-Resume

**At migration point**
- Suspend thread
- Capture context

**At reintegration point**
- Merge migrated context
- Resume thread

**Clone**
- Launch process
- Patch migrated context
- Resume thread
- Suspend thread
- Capture context
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>

GC'ed

New objects
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)

Execution time (seconds)

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

Execution times:
- 1 image: 1x
- 10 images: 20x
- 100 images: 20x
Energy Consumption Comparison
(Image Search)

Energy consumption (J)

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

1 image: 1x
10 images: 20x
100 images: 20x
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
Related Work

- MAUI
- Partitioning
- Migration
- Remote execution
Conclusion

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

Lots of excitement in mobile cloud computing