Mobile Offloading

Matti Kemppainen
kemppi@cs.hut.fi
Agenda

1. Problem scope
2. Overview of mobile computation offloading
3. Appearance in application development
4. Challenges
5. Some first-hand experiences
6. What next?
7. Summary
Problem Description
Slide by Prof. Jukka K. Nurminen

Energy Consumption

Mobile Phone Usage
Web
Social media
Email
Multimedia
Navigation
New Services

Battery Capacity


• less new services
• more frequent battery charging
• physically larger battery
• more energy-efficient components
• a breakthrough in battery cell technology
• a more clever way to utilize the available power
Mobile Computation Offloading
Transfer of Execution of Computation Outside The Mobile Device

Phone
Client
Local
Mobile Node

results = state + metadata

Server
Remote
Cloud
Augmenter
Computation Node

Surrogates

computation = state + metadata
Some Application Examples

Primary functionalities
• speech, video processing…

Background tasks
• web crawling, photo analysis…

Hardware augmentation
• speedup with more resources, specialized resources…

Multiple execution paths
• artificial intelligence, different analysis methods…

Chun & Maniatis, 2009
Augmented Smartphone Applications Through Clone Cloud Execution
Gaining Benefit
End User’s Perspective

Offloading is beneficial, if

the related overhead costs are less than 
the cost of computation done locally.

Kumar & Lu, 2010

Cloud Computing for Mobile Users: Can Offloading Computation Save Energy?
Motivation...

- Saving Energy
- Enhancing Reliability
- Enabling Performance
- Exploiting Context
- Easiness for Application Developers

...Constraints

- Monetary Cost
- Security and Trust
- Code Migratability, Limits of Automation
Key Features of Offloading Frameworks

Migration Support
• no need for application-specific networking protocols

Offloading as an Alternative
• Remote execution is an opportunistic alternative, not a must.
• Offloading is an optimization method, not a requirement.

Dynamic Decisionmaking
• Environmental conditions may have an effect on the execution location.
Offloading Framework Architectures
Levels of Offloading

Feature
• idea: implement features and use them through an interface
• example: a typical network-enabled mobile application

Method
• idea: execute resource-hungry methods remotely
• example: AI analysis of game logic

System
• idea: clone the runtime environment (or the relevant parts)
• example: everything that might run on a system
Feature Offloading
Architecture

```
interface RemoteService {
   /* tasks required by the framework */
}
```

offload semantically coherent parts of the application

Cuckoo (Android)
Vrije Universiteit, Amsterdam

Requirements: Standard Dalvik VM and Android software stack
1. Developer defines an interface (in AIDL) for the part of the application that is subject to offloading.
2. Building system generates the needed implementation stubs and proxies.
3. Developer implements the features. Local and remote implementations may differ.
Elastic Application Architecture
Feature Offloading

Towards an Elastic Application Model for Augmenting Computing Capabilities of Mobile Platforms
Zhang et al., 2011
**Method Offloading**

**Architecture**

offload method calls including needed data

**Example: MAUI (.NET)**

Duke, U. Mass. Amherst, UCLA, Microsoft Research

**Requirements:** Standard .NET software stack

1. Developer annotates the desired methods as *remoteable*.
2. Framework *considers* offloading of the remoteable methods. It may also choose to invoke a method locally.

```csharp
class MyClass {
    @remoteable
    void myMethod() {
        // implementation
    }
}
```
Image Offloading

Architecture

offload bytecode, program image or even volume image

CloneCloud (Android)
Intel Labs, Berkeley

Requirements: A custom version of Dalvik VM
1. Developer lets the underlying system make partitioning and offloading decisions.
# Architecture Comparison

<table>
<thead>
<tr>
<th>Abstraction Level</th>
<th>Developer Workload</th>
<th>Level of Automation</th>
<th>Need for Platform Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature</td>
<td>+++ high</td>
<td>+ medium</td>
<td>- not needed</td>
</tr>
<tr>
<td>Method</td>
<td>+ medium</td>
<td>++ medium</td>
<td>? depends</td>
</tr>
<tr>
<td>System</td>
<td>- low</td>
<td>+++ high</td>
<td>+++ necessary</td>
</tr>
</tbody>
</table>
Migration of Process State

Data transfer costly
• Transfer as little as possible

Serializability
• Data needs to be transferrable (e.g. hardware driver cannot be offloaded)
• Class inheritance may pose considerable problems

Complexity of Automation
• What is the needed dataset?

```c
struct DataSet {
    void *data;
    struct DataSet *nextItem;
};
struct ResultSet* do_the_trick(struct DataSet *data);
```
Decisionmaking

**Prior Analyses**

Developer’s Decisions

Application Profiling
- CPU usage, memory consumption
- network usage
- disk I/O

Use-case Profiling

**Runtime Analyses**

User’s decisions

Environment Profiling
- hardware resources, network availability…

Action Monitoring
- feedback-driven controlling of offloading process
Optimization Problem

Constraints:
- memory
- CPU
- min. latency

Goals:
- min. monetary cost
- max. performance
- min. energy consumption
- max. security
- min. data exchange
- min. interaction time

Inputs:
- battery level
- network parameters
- device profiling
- application profiling
- code movability
- user preference

Output:
- execution configuration
  - start local
  - start remote
  - migration

Kovachev et al, 2011
Mobile Cloud Computing: A Comparison of Application Models
Infrastructure
Where to Offload?

Runtime environment for the migrated code
• different implementations or a common software stack?

Cloud services
• virtualization as a way to providing a suitable environment

Networking performance
• surrogates closer to the clients

Existing resources
• private clouds, PCs, specialized processors, other network devices in the local environment
Networking

Mobility means wirelessness
- Sparse connectivity
- Multitude and heterogeneity of network stacks
- Energy consumption of antenna amplifier
- Long RTTs, packet loss

Some Resolutions
- Network stack abstractions
- Traffic shaping
- Route selection (a.k.a. data offloading)
Other Considerations

**Code Transfer**
- application caching at surrogate
- application libraries

**Data Transfer Optimizations**
- transfer deltas
- delay tolerance of data

**Service Discovery**
- mainstream users don’t want to configure IP addresses

**Trust And Security**
- how to make offloading trustable?
ThinkAir Offloading Framework
Deutsche Telekom (modifications in Aalto)

Method-level offloading framework
• runs on default Dalvik VM, no modifications needed
• modifications necessary to application code

Client-server networking paradigm
• target application acts as an offloading client
• surrogate is a server application, runs inside an unmodified Dalvik VM
• automatic application code transfer

Kosta et al., 2011
Unleashing the Power of Mobile Cloud Computing using ThinkAir
What about mainstream applications?

Existing promises
- MAUI: 45% energy savings for Chess AI
- CloneCloud: 20x speedup and energy savings for a large image search
- MACS (2012): more than 20x speedup in face recognition from a video

Biased measurements?
- Tailored application sets in previous literature

Effect of communication energy consumption
- Could offloading be utilized for traffic shaping?
Experiences And Results

Communication Offloading with ThinkAir

We offloaded successfully!

- ThinkAir handled the necessary procedures for execution migration
- Some advantage with WLAN

...but...

- We did modify the application for custom serialization
- 3G RTT nullified the advantages
- Debugging is hard: Errors may even be unnoticed

Saarinen at al., 2012

Can offloading save energy for popular apps?
Method Migratability

Definition of migratability (simplified)
• Method does not access physical resources of the mobile device.

Real-life migratability
• 15% of all methods in 16 different open-source applications

*Application Developer must be an active part of offloading process.*

Saarinen at al., 2012
Can offloading save energy for popular apps?
Some Cool Ideas

Popular services brought nearby
• e.g. many subscribers for a newspaper on an airplane → a clever proxy that retrieves personalized content

Collaborative services
• Many users with a common goal in vicinity → ad-hoc collaboration for reaching the goal

Universal application execution
• one application with two different interfaces for supporting many terminals (e.g. desktop computer, mobile phone)
• “transfer” of live process with help of offloading
Ongoing Research (late 2011)

Cuckoo: dynamic decision-making
• MACS at RWTH Aachen (2012)

MAUI: state transfer optimization

CloneCloud
• Hardware accesses
• Advanced concurrency
• Trust

A few days ago: TransOS
• an operating system in the cloud?
Summary
Theoretical Aspects

Mobile Computation Offloading: transfer of computation outside the mobile device
• related terminology is emerging while research continues

MCO differs from traditional distributed computing
• opportunistic operation
• low-quality networking environments

Offloading brings many potential benefits
• energy saving, performance, reliability, ease for the software developers, better exploitation of contextual information...

Offloading has also many other opportunities
• business opportunities, collaborative local services, universal application execution...
Summary

Current State of Art

- Today’s frameworks deal with the essentials.
- There is no publicly available offloading framework.
- Current frameworks seem to be more or less for academic purposes.

The big question:

What kind of mainstream application would benefit from offloading?
Thank you!
Any questions or comments?

Matti Kemppainen
kemppi@cs.hut.fi