Mobile Cloud Computing
Architectures – Algorithms - Applications

Pradipta De
The State University of New York, Korea (SUNY Korea)
StonyBrook University

pradipta.de@sunykorea.ac.kr

Parts of the research material in this presentation was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the "IT Consilience Creative Program" (NIPA-2013-H0203-13-1001) supervised by the NIPA (National IT Industry Promotion Agency)
Service Models

Cloud Computing as Gartner Sees It

- **SaaS**
  - Google Apps, Salesforce.com, Netsuite, Lotus, WebFilings, Zoho, Yahoo!Mail, Hotmail, ...

- **PaaS**
  - Google App Engine, Force.com, Windows Azure, LongJump, Rollbase, Amazon Elastic Beanstalk, VMware CloudFoundry, ...

- **IaaS**
  - Amazon EC2, Rackspace, VMware, Joyent, Google Cloud Storage, ....

Source: Gartner AADI Summit Dec 2009
## Mobile Backend as a Service

<table>
<thead>
<tr>
<th>What</th>
<th>Provides <strong>mobile application developers a way to connect</strong> their application to backend cloud storage and processing</th>
</tr>
</thead>
</table>
| Why  | • Abstract away complexities of launching and managing own infrastructure  
• Focus more on **front-end development** instead of backend functions |
| When | • Multiple Apps, Multiple Backends, Multiple Developers  
• Multiple Mobile Platforms, Multiple Integration, Multiple 3rd Party Systems & Tools |
| How  | • Meaningful **resources for app development acceleration** – 3rd party API, Device SDK’s, Enterprise Connectors, Social integration, Cloud storage |
Examples of MBaaS

• Apps need common services which can be shared among apps instead of custom development

• Few companies
  – AnyPresence
  – Appcelerator
  – FeedHenry
  – Kinvey
  – Parse
Augmenting Mobiles with Cloud Computing

- Amazon Silk browser
  - Split browser

- Apple Siri
  - Speech recognition in cloud

- Apple iCloud
  - Unlimited storage and sync capabilities

- Image recognition apps on smartphones useful in developing augmented reality apps on mobile devices
  - Augmented reality app using Google Glass
What is missing?

• Battery capacity on smartphones is limited
  – Applications are not designed with the objective of optimal power consumption

• Smartphone processors are not fast \( \rightarrow \) time to compute can be high \( \rightarrow \) bad user experience

• How can we use Cloud Computing to overcome these limitations?
Part-I

Mobile Cloud Computing Architectures
Mobile Cloud Computing is a framework to augment a resource constrained mobile device to execute parts of the program on cloud based servers.

**Pros:**
- Saves battery power
- Makes execution faster

**Cons:**
- Must send the program states (data) to the cloud server ➔ consumes battery
- Network latency can lead to execution delay

Typical MCC workflow

- Device
- Program
- Network

MCC key components

• Profiler
  – The profiler monitors application execution to collect data about the time to execute, power consumption, network traffic

• Solver
  – The solver has the task of selecting which parts of an app runs on mobile and cloud

• Synchronizer
  – The task of synchronizer modules is to collect results of split execution and combine, and make the execution details transparent to the user
Key challenges?

• MCC requires dynamic partitioning of an application to optimize
  – Energy saving
  – Execution time

• Requires a software (middleware) that decides at app launch which parts of the application must execute on the mobile device, and which parts must execute on cloud
  – This is a classic optimization problem
MAUI server is the cloud component.

The framework has the necessary software modules required in the workflow.

- MAUI enables the programmer to produce an initial partition of the program
  - Programmer marks each method as “remoteable” or not
  - Native methods cannot be remoteable
- MAUI framework uses the annotation to decide whether a method should be executed on cloud server to save energy and time to execute
Smartphone processors are ARM based, cloud servers are x86 ➔ How to run same program code on different architectures

- Uses Microsoft .NET Common Language Runtime
- MAUI apps are written in C#
- MAUI server has copies of the executable
  - Only program states must be sent to server to execute a method
- CloneCloud system does not require the developer to annotate the methods as remoteable ➔ it can work on unmodified applications (or binaries)
- CloneCloud transforms a single machine execution into a distributed execution optimized for various factors (network connection, processing speeds, application computing patterns)

CloneCloud uses static analysis of the code, and partitions at the thread level
COMET: Code Offload by Migrating Execution Transparently

- Works on unmodified applications (no source code required)
- Allows threads to migrate between machines depending on workload
- It implements a Distributed Shared Memory (DSM) model for the runtime engine
  - DSM allows transparent movement of threads across machines
  - In computer architecture, **distributed shared memory** (DSM) is a form of **memory** architecture where the (physically separate) memories can be addressed as one (logically **shared**) address space

Requires only program binaries
Execute multi-threaded programs correctly
Improve speed of computation

Further improvements to data traffic during migration is also possible by sending only the parts of the heap that has been modified
Alternative Architectures

• Micro-cloud for offloading
  – Form a transient cloud using mobile devices in the vicinity

• Edge Cloud for offloading
  – Use the routers and/or other nearby servers to act as the compute resource

• Fog Computing (Mobile Fog)
  – Use ubiquitous sensor devices (Internet of Things) to act as a platform for unlimited computing power
Assumption is that a mobile device can only connect to other devices in the vicinity. Computation offloading can be performed among a set of mobile devices → Mobile Device Cloud.

Goal is to maximize the lifetime of the collection of the mobile devices.

- Ported MapReduce framework to execute on Mobile Device Cloud
- Has been shown to be useful for other latency sensitive applications
Edge Cloud or Cloudlet

Goal is to reduce the latency in reaching the cloud servers
Use servers that are closer to the mobile devices  ⇒  use cloudlet

A cloudlet is a new architectural element that arises from the convergence of mobile computing and cloud computing. It represents the middle tier of a 3-tier hierarchy: mobile device --- cloudlet --- cloud

Use remote cloud

Use cloudlet
Fog Computing

- **Fog computing** is an architecture that uses one or a collaborative multitude of end-user clients or near-user edge devices to carry out a substantial amount of storage, communication, control, configuration, measurement and management.

- Fog computing is a non-trivial extension of the cloud computing
The service oriented framework, like MBaaS, is focused mainly on application developers, less on user experience.

MCC focuses more on user experience:
- Lower battery consumption
- Faster application execution

MCC architectures design the middleware to partition an application execution transparently between mobile device and cloud servers.

Alternative architectures focus on efficient use of computing resources in the user’s environment.
Part-II

Mobile Cloud Computing
Algorithms
Key Problem to Solve

• At its core, MCC framework must solve how to **partition a program for execution on heterogeneous computing resources**

• This is a classic “Task Partitioning Problem”

• Widely studied in processor resource scheduling as “job scheduling problem”
  – But in MCC the assumptions change and often makes it more challenging to solve
Task Partitioning Problem in MCC

Input:
• A call graph representing an application’s method call sequence
• Attributes for each node in the graph denotes (a) energy consumed to execute the method on the mobile device, (b) energy consumed to transfer the program states to a remote server

Output:
• Partition the methods into two sets – one set marks the methods to execute on the mobile device, and the second set marks the methods to execute on cloud

Goals and Constraints:
• The energy consumed must be minimized
• There is a limit on the execution time of the application
• Other constraints could be – some methods must be executed on mobile device, total monetary cost, etc.
Directed Acyclic Graph represents an application Call Graph

Directed Acyclic Graph

- Highlighted nodes must be executed on the mobile device \( \Rightarrow \) called native tasks \((v1, v4, v9)\)
- Edges represent the sequence of execution
- Any non-highlighted node can be executed either locally on the mobile device or on cloud

Integer Linear Program to solve the Task Partitioning Problem (as used in MAUI)

\[
\begin{align*}
\text{maximize} & \quad \sum_{v \in V} I_v \times E^l_v - \sum_{(u,v) \in E} |I_u - I_v| \times C_{u,v} \\
\text{such that:} & \quad \sum_{v \in V} ((1 - I_v) \times T_v^l) + (I_v \times T_v^r)) \\
& \quad + \sum_{(u,v) \in E} (|I_u - I_v| \times B_{u,v}) \leq L \\
\text{and} & \quad I_v \leq r_v, \forall v \in V
\end{align*}
\]

- A 0-1 integer linear program, where 
  \( I_v = 0 \) if method executed locally,
  \( = 1 \) if method executed remotely
- \( E \) is the energy cost to execute method \( v \) locally
- \( C_{u,v} \) is the cost of data transfer
- \( T \) is the time to execute the method
- \( B \) is the time to transfer program state
Use of the Formulation

• Static Partitioning
  – When an application is launched, invoke an ILP solver which will tell where each method should be executed
  – There are also heuristics to find solutions faster

• Dynamic or Adaptive Partitioning
  – For a long running program, the environmental conditions can vary
  – Depending on the input, the energy consumption of a method can vary
  – Should we adapt the partition as the program executes?
Scenarios of Variations

• User moves while using smartphone ➔ network conditions can change
• Another app starts executing ➔ residual energy to execute application changes
• Another user starts using cloud service ➔ load on the cloud server increases slowing down the app
• Connectivity to network is lost ➔ how to handle failures
• Support for variety of mobile devices ➔ energy and time to execute profiles will change
Sources of Variation and QoE

- Energy Consumption
- Completion Time
- Security
- Monetary Cost
Mobile Ecosystem

Sources of Variation

Applications
- Concurrency
- Multi-tasking
- Real-time Constraints

Network
- Channel Error
- Multiple Interfaces
- Bandwidth Variation
- Intermittent Connectivity
- Latency Variation
- User Mobility

Execution Platform
- Mobile Processor
- Coprocessor
- Energy/Time Profile
- Cloud Architecture
Sources of Variation in Operating Environment

- **App Concurrency:**
  - Smartphone apps are concurrent – task-level (independent tasks) or data-level (streaming apps)

- **Simultaneous apps:**
  - Presence of foreground and background apps

- **Real-time constraints:**
  - Imposes constraints on task completion

- **Network variations:**
  - Bandwidth: signal quality varies leading to changing bandwidth
  - Connectivity: cannot have guaranteed connectivity
  - Latency: The routes to the server can change, as well as, congestion on the path
  - User mobility: Can lead to change in network conditions
  - Multiple interfaces: Choice of interface can change communication properties
Sources of Variation in Operating Environment

• Heterogeneous Operating Platforms
  – Mobile Processors: Smartphones can have 2 to 8 processors, can implement dynamic voltage and frequency scaling
  – Coprocessors: GPUs are present in many smartphones ➔ does it help to use the GPUs
  – Energy and Time Profile: Hardware components vary in characteristics ➔ individual profiles are required
  – Cloud Architecture: Different hierarchy of cloud architecture is possible
How to evaluate MCC performance

• Energy Consumption
  – Must reduce energy usage and extend battery life

• Time to Completion
  – Should not take longer to finish the application compared to local execution

• Monetary Cost
  – Cost of network usage and server usage must be optimized

• Security
  – As offloading transfers data to the servers, ensure confidentiality and privacy of data ➔ how to identify methods which process confidential data
<table>
<thead>
<tr>
<th>Offloading solutions in MCC systems</th>
<th>Year</th>
<th>Application</th>
<th>Network</th>
<th>Execution Platform</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Concurrency</td>
<td>Multi-tasking</td>
<td>Real-time Constraints</td>
</tr>
<tr>
<td>Gu et al. [18]</td>
<td>2004</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>MAUI [1]</td>
<td>2010</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Misco [19]</td>
<td>2010</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Odessa [20]</td>
<td>2010</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ECOS [21]</td>
<td>2012</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Serendipity [22]</td>
<td>2012</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Yang et al. [23]</td>
<td>2012</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Eom et al. [26]</td>
<td>2013</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ENDA [27]</td>
<td>2013</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TDM [28]</td>
<td>2013</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>COSMOS [29]</td>
<td>2014</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Gao et al. [31]</td>
<td>2014</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Hermes [32]</td>
<td>2015</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
## Comparative View

<table>
<thead>
<tr>
<th>Offloading solutions in MCC systems</th>
<th>Year</th>
<th>Application</th>
<th>Network</th>
<th>Execution Platform</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Concurrency</td>
<td>Multi-tasking</td>
<td>Real-time Constraints</td>
</tr>
<tr>
<td>Simulation-based Studies</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fesehaye et al. [33]</td>
<td>2012</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>MapCloud [34]</td>
<td>2012</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Balakrishnan and Tham [35]</td>
<td>2013</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CARMS [36]</td>
<td>2013</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Chen et al. [37]</td>
<td>2013</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Lin et al. [38]</td>
<td>2013</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>MuSIC [39]</td>
<td>2013</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Mobile fog [40]</td>
<td>2013</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Foreseer [41]</td>
<td>2014</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Chen et al. [15]</td>
<td>2014</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Lin et al. [42]</td>
<td>2014</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Impact Analysis of the Parameters

• In real implementation, it is difficult to create controlled experiments to study the impact of individual parameters

• Perform a simulation based analysis, but model all the parameters into the simulation model, using different constraints
  – Precedence: maintaining the DAG ordering
  – Concurrency: degree of parallelism
  – Execution time: time budget
  – Deadline: real time restrictions
  – Energy budget: total energy to be saved
Impact of parallelism

With increasing number of threads, both execution time and energy consumed increases.

But, Execution time variation is much lower as the time to migrate data is overlapped with execution.
Impact of Cloud Processor Speed

- If bandwidth is low, then there is little impact of a faster cloud server
  - Low network bandwidth leads to longer use of network card, and higher energy consumption

- If the bandwidth is high, then the benefit of faster cloud server is evident
At lower bandwidths, RTT delay has relatively low impact on execution time.

At high bandwidth, the RTT is higher than the actual transmission time, while it is opposite for low bandwidth ➔ so at low bandwidth, there is relatively low impact of RTT.
Open Questions

• How can one design a practical and usable MCC framework
  – System as well as partitioning algorithm

• Is there a scalable algorithm for partitioning
  – Optimization formulations are NP-hard
  – Heuristics fail to give any performance guarantee

• If all parameters cannot be considered, which are the most relevant parameters to consider in the design of MCC systems
Part-III

Mobile Cloud Computing
Applications
Types of Applications

• Computation Intensive
  – Speech Translation, Computer Vision

• Streaming apps (or data parallel applications)
  – Continuous Sensing and Processing
    • Augmented reality on video streams

• Communication Intensive
  – Social network apps, like Twitter
  – Apps with high push notifications

• Gaming Apps
What did MAUI show?

• MAUI system was evaluated using
  – Face recognition application \(\rightarrow\) computation intensive
  – Video Game \(\rightarrow\) latency sensitive
  – Chess game \(\rightarrow\) computation + latency

• Speech translation \(\rightarrow\) demonstrated how to overcome resource limits on smartphone
One order of magnitude improvement
On 3G, the energy consumption is higher than that over WiFi

Improvement over WiFi
On 3G, there is no energy gain

The gains are diminishing
On 3G, there is no energy gain
[2] MAUI Results: Execution Time Savings

**ONE RUN FACE RECOGNITION**
- Reduced processing time from 19 seconds to less than 2 seconds

**400 FRAMES of VIDEO GAME**
- Offloads one method “Move Complex”

**30 MOVE CHESS GAME**
- Offloads two methods “Select Piece” and “Move Piece”
- Overhead of “Select Piece” is high
CPU and memory utilization of running a translator application on a PC
Peak memory consumption was 110 MB ➔ impossible to run on a smartphone with 32 MB smartphone RAM)

Using MAUI, the memory limitations can be overcome
CloneCloud: Test Applications

• Virus Scanning
  – Scans the content of the phone file system and matches against a library of 1000 signatures, one file at a time

• Image Search
  – Finds all faces in images stored in the phones using a face-detection library

• Privacy preserving targeted advertising
  – Uses behavioral tracking across websites to infer user’s preferences on the smartphone (protects user’s privacy) ➔ keyword matching problem
Mean Execution Time for each application

[Image of bar graphs showing execution time for different applications and data sizes.]
Mean Energy Consumption for each application
Interactive Perceptual Applications

• Applications that use camera and other high data rate sensors on smartphones for continuous sensing
  – Continuous face or object detection
  – Human machine interfaces
  – Interactive augmented reality experience

• Key requirements
  – Quick response
  – Continuous processing of high fidelity sensors
  – Compute intensive processing (ML, Comp. Vision)
  – Algorithm performance highly dependent on data

Conceptually similar to data parallel streaming applications
Overview of ODESSA system

The idea is to exploit **offloading** and **degree of parallelism**

Three techniques to improve performance

- Offloading: move computationally intensive stages to server
- Pipelining: allow different stages to process different frames
- Increase data-parallelism: split frames into multiple sub-frames

<table>
<thead>
<tr>
<th>Application</th>
<th># of Stages</th>
<th>Avg. Makespan &amp; Frame Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Face Recognition</td>
<td>9</td>
<td>2.09 s, 2.50 fps</td>
</tr>
<tr>
<td>Object and Pose Recognition</td>
<td>15</td>
<td>15.8 s, 0.09 fps</td>
</tr>
<tr>
<td>Gesture Recognition</td>
<td>17</td>
<td>2.54 s, 0.42 fps</td>
</tr>
</tbody>
</table>

- Makespan is the time taken to execute all stages of a dataflow graph for a single frame
- Throughput is the rate at which frames are processed ➔ related to frame rate
Large image transfer is very sensitive to even a small amount of delay and loss leading to significant performance loss. Even a loss-less 802.11g link cannot support more than 3 fps.

Lesson: Simply offloading compute intensive task does not help.
Strawman I: Offload-All strategy where only video source stage and display runs locally, and a single instance of all other stages are offloaded to server

Strawman II: Domain-specific partitioning, where knowledge about application and input from developer is used to identify compute intensive stages in application graph

Both are static partitioning techniques
Offload Shaping

• The idea that sometimes it is valuable to perform additional cheap computation, not part of the original pipeline, on the mobile device in order to modify the offloading workload

• Scenarios
  – Object detection on a continuous video stream: If the video frame is blurry due to motion, then it is not useful to send it for processing
    • Detect blurry frame and discard ➔ blur detection possible using on-board sensors at low energy cost
  – If there is similarity across frames, then do not send the frames across ➔ the result of CV algo will not change significantly
  – If one wants to detect the Coke can in a scene, then filtering for Red color can indicate if a coke can may be there ➔ application context is exploited
Communication Intensive Apps

• Most popular apps involve intensive communication that consumes a significant part of the energy
  – Does offloading help in saving energy for such apps

• Key insight:
  – Reduce network traffic that is handled by mobile device ➔ offload methods that handle communication with a server
  – Optimize traffic patterns ➔ aggregate traffic across applications or within an application
Used opensource Twitter app – AndTweet
Offloaded the communication intensive methods

It is non-trivial to identify the cases where communication offloading will definitely benefit

Extension of Communication offloading idea
- Mobile browsing
  - How to split the page load over cellular network to minimize energy consumption on the mobile device?
  - Can you proritize/serialize content download while browsing
Several commercial providers – OnLive, AMD Game Servers
Cloud Gaming

• Higher than 100 ms RTT makes interactive game play experience suffer

• Problem:
  – Rendering and transmission from server takes long since frame size can be large
  – Game input always comes from mobile device end

• Solution
  – Speculative Execution: Produce speculative rendered frames of future possible outcomes, and deliver them to client one RTT ahead of time

• Tested on Doom3 and another action-based role playing game
Results

Improved Game Play significantly
Epilogue

Open Issues
Take-Away Questions

• Infrastructure-Platform
  – How will MCC model evolve with the proliferation of wearable devices, 5G network standards?

• Middleware
  – How to adapt task scheduling in a more diverse environment with many heterogeneous devices?

• Applications
  – How does MCC apply to different types of applications?
  – What novel applications can it enable in the wearables and IoT space?
For the slides, please visit: http://www.mosys.cs.sunykorea.ac.kr

Parts of the research material in this presentation was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the "IT Consilience Creative Program" (NIPA-2013-H0203-13-1001) supervised by the NIPA (National IT Industry Promotion Agency)


References


References


