Integrating Mobile Applications with Cloud Computing Resources

[Middleware’09] Calling the Cloud: Enabling mobile phones as interfaces to cloud applications

Ioana Giurgiu, Oriana Riva, Dejan Juric, Ivan Krivulev, Gustavo Alonso

Middleware 2019, December 12th, UC Davis
Agenda

• About the team

• What is mobile cloud computing?

• Calling the cloud: Enabling mobile phones as interfaces to cloud applications

• Alternative application models in mobile cloud computing

• Adoption in industry
About the team

ML / AI Senior Researcher
IBM Research | Zurich

Head of Connectivity
SIX GROUP

Principal Researcher
Microsoft Research

Developer
ELCA

Prof. & Lead of Systems Group
ETH Zürich
What is mobile cloud computing?
Calling the Cloud – Trends since the 2000s

Cloud computing

Easier connectivity

Energy efficiency

Mobile devices

Complex applications

Mobile users want fast and spontaneous interactions
Calling the cloud - Traditionally...

• Applications run on the mobile device
  • Compatibility issues
  • Resource limitations

• Mobile devices are thin clients
  – No device customization
  – Latency cost

No flexibility
Calling the cloud - Application distribution
Calling the cloud - Application distribution
Calling the cloud - Application distribution

Problem #1
Focuses on the mobile perspective

Goals
- Application performance
- Device customization
- Battery life extension
- Easier application development
Calling the cloud - Application distribution

Problem #1
Focuses on the mobile perspective

Goals
- Application performance
- Device customization
- Battery life extension
- Easier application development
Calling the cloud - Application distribution

Problem #1
Focuses on the **mobile** perspective

**Goals**
- Application performance
- Device customization
- Battery life extension
- Easier application development

Problem #2
Focuses on the **cloud** perspective

**Goals**
- Meet demands and constraints
- Efficient resource allocation
- High acceptance rate
- Good resource balancing
Calling the cloud - Application distribution

Problem #1
Focuses on the **mobile** perspective

**Goals**
- Application performance
- Device customization
- Battery life extension
- Easier application development

Problem #2
Focuses on the **cloud** perspective

**Goals**
- Meet demands and constraints
- Efficient resource allocation
- High acceptance rate
- Good resource balancing
Calling the cloud

• Assumes modular applications

• **Dynamic & flexible software deployment**
  • On-the-fly migration, installation and updates of application modules

  Modular techniques ➔ OSGi, R-OSGi [Rellermeyer’07, Rellermeyer’08]

• **(Optimal) distribution of application modules**
  • Reduce interaction time and extend battery life
  • Adapt to dynamically changing factors (CPU, inputs, bandwidth)

  Objective-driven algorithms

• Adapt to different execution environments

  Performance estimation model with passive measurements
OSGi & R-OSGi

- Open standard maintained by the OSGi alliance

- OSGi is
  - A programming model to develop Java apps from modular units (bundles) decoupled through service interfaces
  - A runtime framework for controlling the life cycle of bundles
  - Local to a single system

[Rellermeyer’07]
OSGi & R-OSGi (cont.)

• R-OSGi allows for distributed OSGi => transparently distribute an OSGi app at service boundaries by using proxies

• R-OSGi uses
  • Dynamic proxy generation (allows cross-network invocation of services)
  • Distributed service registry
  • Mapping network & remote failures to local module hot-plug events to ensure transparent distribution
  • Type injection to resolve distributed type system dependencies
Calling the cloud – Architecture

Diagram showing the architecture with components such as Migrator, Optimizer, Cache, Application graph generator, Device profiler, Application profiler, and Application profiler.
Calling the cloud – Application representation

- Modular design
  - Separate functionalities into layered modules
  - Functionalities accessible through services
  - Reduce dependencies per module
  - No cyclic dependencies

module ‘2’ {
  dependencies: ['4', '5']
  type: [nIO, movable]
  CPU: 1068ms
  size: 17kB }
wire 'w24': ('2', '4', 50KB)
wire 'w25': ('2', '5', 700KB)

network: ('WiFi', 6Mbps, 2Mbps)
Calling the cloud - Optimization

- **Constraints**
  - MAX(upload data), MAX(code size)

- **Objective** → Minimize interaction time

\[
\begin{align*}
\min OP &= \min \left( \sum_{i=1}^{k} \frac{c_i}{B} + t_{is} \times k + t_p \times r + \right. \\
& \quad \left. \sum_{i=1}^{k} t_i + \sum_{j=1}^{l} t_j + \sum_{i=1}^{k} \sum_{j=1}^{l} \frac{(in_{ij} + out_{ji})}{B} \right)
\end{align*}
\]

such that: \[ \sum_{i=1}^{k} c_i \leq C_{MAX} \text{ and } \sum_{i=1}^{k} \sum_{j=1}^{l} in_{ij} \leq D_{MAX} \]

**ALL** → global optima, expensive

**K-Step** → local optima, cheaper with lower K

Considers code migration time, module installation time and data transfer time
Calling the cloud - Steady-state behavior

- Up to 75% (45%) improvement compared to ALL and UI
- Different optimal distributions with small instances
Calling the cloud – Adaptation to CPU load

• Similar observations for network bandwidth
  • More computation is moved to the device when bandwidth drops
Calling the cloud – Resource overhead

• Installation time: 12-14s
  • Can be improved with optimized OSGi implementations for resource-constrained devices [https://www.osgi.org/business/markets-and-solutions/mobile/]

• Less than 200kB code on the mobile device
• Memory footprint < 7MB
• Opportunistic profiling for network

• Cheap profiling
  • Additional 2-3kB of profiling code per application module
  • Performance degradation < 8%
Calling the cloud – Recap

• **Our approach** distributes complex applications between the mobile device and the cloud
  • Uses modular techniques for flexible code deployment (OSGi)
  • Algorithms for code and data offloading
  • Reduces user interaction time and power consumption
  • Dynamically adapts to changing factors and user inputs
Alternative application models – Augmented execution

- **CloneCloud** [Chun’09]
  - Offload execution from the phone to the cloud where one or more cloned replicas of the phone’s software is running
  - No proxies or code partitioning are required

- Assumes all apps and data are always replicated into the cloud
Alternative application models – Augmented execution (cont.)

- **Cloudlets** [Satyanarayanan’09]
  - Computation is offloaded to cloudlets (trusted, resource-rich cluster well connected to nearby mobile devices)
  - Based on dynamic VM synthesis (small VM overlays are applied to the base VM to resume execution from the state in which it was suspended)
  - Mobile acts like a thin client
  - VM synthesis takes between 60 and 90 seconds

- **Tango** [Gordon’15]
  - Based on flip-flop replication (app is executed on both sides) and tries to determine which replica will be faster, by allowing it to lead execution and display outputs to the user
  - Needs to ensure synchronization at all times
Alternative application models – Elastic modularization

- **MAUI** [Cuervo’10]
  - Maximize battery life by offloading at method-level
  - Methods that can be offloaded need to be annotated during development phase
  - Data is assumed to be replicated in the cloud
  - Offloading at method level can be counter-productive, as it may incur higher network latency costs

- **Weblets** [Zhang’09, Zhang’10]
  - Apps are partitioned into weblets (elastic components that can be executed transparently in different clouds)
  - Dynamic configuration of where weblets execute
  - Decoupled from a particular programming language
Alternative application models – Elastic modularization (cont.)

- **Cuckoo** [Kemp’12]
  - Runs as temporary clone of the phone in the cloud to avoid needing full synchronization
  - Optimizes for intermittent network connection
  - Differentiates between local and remote code implementations

- **ThinkAir** [Kosta’12]
  - Parallel execution of code, via the Divide-and-Conquer approach between multiple VMs (dynamically creating, resuming and creating VMs on a per-need basis)
  - Allows distribution in the cloud

- **Comet** [Gordon’12]
  - Uses distributed shared memory instead of RPC to provide thread migration and full multi-threading support
  - Assumes data is present in the cloud

- Other notable work: Odessa [Ra’11], Wishbone [Newton’09], Spectra [Flinn’12], Tactics [Balan’03], [Yigitbasi’11]
Alternative application models – Application mobility

• **ISR** [Satyanarayanan’05]
  • Moving application state between hosts during execution, by using VMs and distributed FS
  • Each VM encapsulates a distinct execution and user customization state
  • State is transported via the distributed FS
  • *Scalable only when using one platform type (else latency is too high)*

• **JADE** [David’07]
  • Migrate individual applications across heterogeneous hardware
  • Supports adaptation
  • *Limited to Java applications*
Alternative application models – Ad-hoc mobile clouds

• Mobiles expose their computing resources to other mobiles via ad-hoc clouds

• **Hyrax** [Marinelli’09], [Huerta-Canepa’10]
  • Enable high horizontal scaling of the available ad-hoc mobile nodes
  • Use Hadoop on the mobile device to distribute tasks and storage
  • How to enable Hadoop on a mobile is challenging
  • High communication overheads
When ML comes into play...

• ML and AI have been used recently to make better code offloading decisions

• **Workload prediction** [Flores’17] => 60-70% accuracy in predicting mobile performance $k$ steps ahead
  - Regression models
  - Deep learning models

• **Code execution prediction** [Nawrocki’17, Nawrocki’18, Flores’13] => 40-50% reduction in task execution
  - Reinforcement learning
  - Fuzzy logic decision models

• **Code offloading scheduling** [Eom’13] => 87.5% accuracy in making correct scheduling decisions
  - Decisions trees
  - Bayesian networks

• ... and many more
Adoption in industry

• **Project Hawaii** [Microsoft]
  • Develop cloud-enhanced mobile applications that access a set of cloud services and Windows Azure for computation and data storage
  • Provided as an SDK
  • Services included:
    • **Social Mobile Sharing (SMASH) Service** => client-server system that enables rapid prototyping and development of solutions for social computing scenarios
    • **Path Prediction Service** => enables a mobile application to predict a user’s destination based on current route data
    • **Key-Value Store Service** => key-value store for mobile applications state
    • **Translator Service** => enables a mobile application to translate text from one language to another and to obtain an audio stream that renders a string in a spoken language
    • **Relay Service** => provides an endpoint naming scheme and buffering for sent messages
    • **Optical Character Recognition (OCR) Service** => extracts text from images
    • **Speech-to-Text Service**
  • **AppInsight** => provides users with detailed insights into how their application is being used in the wild – who the users are, when they use the app, and how they use it
Adoption in industry (cont.)

- **Google Lens** [Google]
  - Google’s engine for seeing, understanding, and augmenting the real world
  - It lives in the camera viewfinder of Google-powered software like Assistant and within the native camera of top-tier Android smartphones
  - Uses the expansive knowledge base of Search to surface actionable info like purchase links for products and Wikipedia descriptions of famous landmarks
  - The goal is to give users context about their environments and any and all objects within those environments
  - Computation is done on distributed VMs in the Google Cloud
Adoption in industry (cont.)

- **Deep learning on mobile devices @ Apple**
  - BNNS (Basic Neural Network Subroutines) are part of the **Accelerate** framework
    - Collection of math functions that take full advantage of the CPU’s fast vector instructions
  - MPSCNNs are part of the **Metal Performance Shaders** framework
    - Library of optimized compute kernels that run on the GPU instead of on the CPU

- Training is done in the cloud
- Inference happens on any resource constrained iOS device
Adoption in industry (cont.)

- **Image sharing and processing**
  - Storage capacity and processing power are reduced on the mobile device, by uploading and processing photos to the cloud as they are taken
  - *Flickr* [flickr.com], *ShoZu* [acquired by CriticalPath], even *Facebook*

- **Music search services**
  - 10s audio is sent from the device to the cloud to be matched against a centralized database
  - *Shazam* [shazam.com], *Soundhound* [soundhound.com]

- **Augmented reality**
  - Use compass, GPS, accelerometer and device camera to identify a user’s location and field of view by offloading the heavy computation to one or more VMs
  - *Layar* [acquired by Blippar]
Conclusions

- Mobile cloud computing has been active since the early 2000s
- **Our approach has been one of the very first proposed**
  - Module-level partitioning
- Various application models have been proposed
  - **Elastic partitioning / modularization**
  - Augmented remote execution
  - Application mobility
  - Ad-hoc mobile clouds

- **Elastic app modularization** has arguably been the most popular approach to enable code offloading
  - Picked up by major players
  - Implemented in popular applications
Thank you! Questions?

igi@zurich.ibm.com
References

- [Rellermeyer’07] R-OSGi: distributed applications through software modularization
- [Rellermeyer’08] AlfredO: An Architecture for Flexible Interaction with Electronic Devices
- [Cuervo’10] MAUI: Making Smartphones Last Longer with Code Offload
- [Chun’09] Augmented Smartphone Applications through Clone Cloud Execution
- [Satyanarayanan’09] The Case for VM-based Cloudlets in Mobile Computing
- [Zhang’10] Towards an Elastic Application Model for Augmenting Computing Capabilities of Mobile Platforms
- [Kemp’12] Cuckoo: A Computation Offloading Framework for Smartphones
- [Kosta’12] ThinkAir: Dynamic Resource Allocation and Parallel Execution in the Cloud for Mobile Code Offloading
- [Gordon’12] Comet: Code Offload by Migrating Execution Transparently
- [Satyanarayanan’05] Towards Seamless Mobility on Pervasive Hardware
- [David’07] Supporting Adaptive Application Mobility
- [Marinelli’09] Hyrax: Cloud Computing on Mobile Devices using MapReduce
- [Huerta-Canepa’10] A Virtual Cloud Computing Provider for Mobile Devices
References (cont.)

- [Nawrocki’18] Adaptable mobile cloud computing environment with code transfer based on machine learning
- [Flores’17] Mobile Code Offloading: Should it be a Local Decision or Global Inference?
- [Ra’11] Odessa: Enabling Interactive Perception Applications on Mobile Devices
- [Newton’09] Wishbone: Profile-based Partitioning for Sensornet Applications
- [Flinn’12] Balancing Performance, Energy, and Quality in Pervasive Computing
- [Balan’03] Tactics-Based Remote Execution for Mobile Computing
- [Yigitbasi’11] Incremental Placement of Interactive Perception Applications