MobiProg: an adaptive programming system for cloud-enabled smartphone applications


Embedded Networks Laboratory, Department of Computer Science, USC
Department of Computer Science, UCLA

http://enl.usc.edu/projects/mobiprog/

Cloud-Enabled Mobile Computing

Smartphones:
- Personal & Ubiquitous
- Always Connected
- Application Centric
- Limited Computation / Storage
- Limited Connectivity
- Power Constrained
  - Typical Applications:
    - Personal Information Management
    - Telephony & Messaging
    - Browsing & E-Mail
    - Media Player

Cloud Computing:
- Scalable & Reliable
- Virtually Limitless Computation
- Virtually Limitless Storage
- High-Speed connectivity
- Large Data Sets
  - Typical Applications:
    - E-Commerce
    - Web & Data Hosting
    - Software-as-a-Service
    - Scientific Computing
    - Data Mining & Analytics

Cloud-Enabled Mobile Applications:
- Power of the Cloud, on the Phone
  - Typical Applications:
    - Compute Intensive
      - Face & Voice Recognition
    - Artificial Intelligence
    - Gaming
    - Crowd-Sourcing
    - Pothole Reporting
    - Traffic Monitoring
    - Social Networking
    - Context Aware Messaging

The Problem

Development Challenges:
- Heterogeneous Platforms
- Complex Distributed Systems
- Short Time-to-Market

Systems Challenges:
- Arbitrary Partitioning is Intractable
- Useful Partitioning is Application Specific
- Partitioning Strategy Depends on Context
  - Resource Usage & Availability

User Challenges:
- Optimize What?
  - Battery / Responsiveness
  - Quality / Cost
  - Performance Constraints / Bounds?

The MobiProg Approach

Applications:
- Component Graph
  - Components
    - Reusable Unit of Functionality
    - Runs in Phone / Cloud
    - Message Passing Between Components
    - Pipe Component A's Output into B's Input

Compile-Time System:
- Use Static Analysis & Developer Input
- Generate Application Componentization & Constraints

Run-Time System:
- Profile Resource Usage
- Capture Resource Availability
- Dynamically Migrate Components
  - Optimize User Utility
  - Maintain Constraints

Example: Translation Application

Main Components:
- Speech Recognition
- Natural Language Translation
- Text-to-Speech

Audio Capture Native Audio
Speech Recognition Native Text
Translation

Audio Playback Translated Audio
Text-to-Speech Translated Text

It works!
Ça marche!
Es funktioniert!
Funziona!
¡Funciona!