CSci 8980
Mobile Cloud Computing

Cloud Crowd Sourcing
Medusa: A Programming Framework for Crowd-Sensing Applications

Moo-Ryong Ra, Bin Liu
Tom La Porta, and Ramesh Govindan
Crowd-Sensing

Crowds + Sensing

- Camera/Mic.
- GPS/WPS
- Accelerometer
- Compass/Gyro
- Proximity

Motivation → Problem → Design → Evaluation → Conclusion
Example: Documenting Social Events

Motivation

Problem

Design

Evaluation

Conclusion
Other Crowd-Sensing Tasks

Auditioning

Television producer wants potential actors to submit clips of their acting.

Collaborative Learning

Mobile app developer wants samples of activities to train classifier.

Forensic Analysis

Security personnel wants to find pictures of attack taken by mall visitors.
The Problem

Recruiting Workers
Providing Incentives
Coordinating worker actions
Gathering contributions

Stage 1
Stage 2
Stage 3
Stage 4

Automation using a high-level programming system
Contributions

- Describe crowd-sensing requirements
- Devise design principles
- Design and implement Medusa, a programming system for crowd-sensing
- Evaluate using qualitatively different tasks
## Crowd-Sensing Requirements

### Expressivity
- Worker Mediation
- Access Stored Data
- In-network Processing
- Extensibility
- Timeliness Support
- Incentives

### Runtime
- Privacy
- Concurrent Tasks
- Unsynchronized Execution
- Anonymity
- Robustness
- Sandboxing
- Secure Communication
- Resource Usage Controls
Runtime Design Principles

Partitioned Services

Cloud Runtime
- Task Management
- Worker Recruiting
- Incentive Management

Client Runtime
- Sensing
- In-network Processing

Opt-in Data Transfer

Cloud (Server)

Mobile Devices

Dumb Smartphone

Motivation → Problem → Design → Evaluation → Conclusion
Medusa Design

MedScript
Programming Language

Medusa Runtime
Expressivity of *MedScript* Language

XML-based extensible language
Task Execution Example

- **User**
  - Medusa
  - Alice
  - Bob
  - Charlie

- **Task Initiation**
  - Create 'Recruit'
  - Sign-up to 'Recruit'
  - Take Video 'TakeVideo'

- **Unsynchronized Execution**
  - Bob
  - Upload Video 'UploadVideo'
  - Taking Video
  - Uploading Video

- **Cloud-Initiated Stage Execution**
  - Initiate 'TakeVideo'
  - Initiate 'UploadVideo'
  - Optimized Privacy

- **Got the Video**
  - Sending the Video to Alice

- **Opt-in Privacy**

**Time Flow**

**Design**
Medusa Runtime Design

Data Repository

Stage Library

Interpreted MedScript Program

Task Tracker

Worker Manager

C2DM

Amazon Mechanical Turk

Cloud (Server)

Mobile Devices

Data Transfer

Stage Tracker

MedBox

Upload Manager

Sensor Manager

Transform Manager

Storage Manager

Bob

Charlie

Motivation ➔ Problem ➔ Design ➔ Evaluation ➔ Conclusion
Medusa Prototype

Implementation

- Python + LAMP
- Java Servlet + Tomcat
- Android
- 10 crowd-sensing tasks

Evaluation

- Expressivity
- Concurrent Execution
- Performance
- Robustness
# Expressivity

<table>
<thead>
<tr>
<th>Application</th>
<th>LOC</th>
<th>Sensors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video Documentation</td>
<td>86</td>
<td>Camera</td>
</tr>
<tr>
<td>Collaborative Learning</td>
<td>62</td>
<td>Accel., Mic.</td>
</tr>
<tr>
<td>Road Monitoring</td>
<td>90</td>
<td>Accel.</td>
</tr>
<tr>
<td>Citizen Journalist</td>
<td>45</td>
<td>GPS</td>
</tr>
<tr>
<td>Party Thermometer</td>
<td>62</td>
<td>GPS, Mic.</td>
</tr>
<tr>
<td>WiFi/Bluetooth Scanner</td>
<td>45</td>
<td>Net. Sensors</td>
</tr>
</tbody>
</table>

Fewer than 100 LOC for tasks.
Concurrent Execution

Each task has a different number of stages. Workers may start each stage at different times. Each stage has a different execution time.

- Workers I, II, III, and IV have tasks (1) to (10).
- Command Arrival via SMS is at 22:30 and 22:40.
- Worker subscribed to the Task is at 22:15.

Time Flow

Motivation
Problem
Design
Evaluation
Conclusion
Summary

Medusa: A High-Level Programming Framework for Crowd-Sensing Applications

Q: can cloud execute any stages? Not clear why not.

Code: http://code.google.com/p/medusa-crowd-sensing/

Demo Video: http://youtu.be/jL1dGA21ciA
System Performance

Medusa runtime has very small overheads.

C2DM gives much less latency