HOT Topics in Computer Science (HOT-T-CS)

Mobile Cloud Computing Applications

Pradipta De
pradipta.de@sunykorea.ac.kr
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
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

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

- 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 \( \rightarrow \) related to frame rate

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
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 \(\Rightarrow\) blur detection possible using on-board sensors at low energy cost
  – If there is similarity across frames, then do not send the frames across \(\Rightarrow\) 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 \(\Rightarrow\) 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
### Result on Twitter App

<table>
<thead>
<tr>
<th>Measurement</th>
<th>Wi-Fi (avg/stdev)</th>
<th>3G (avg/stdev)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total energy (measured)</td>
<td>2.67/0.59 J</td>
<td>3.92/1.97 J</td>
</tr>
<tr>
<td>Execution time for refresh to complete (measured)</td>
<td>2.69/0.59 s</td>
<td>3.86/2.02 s</td>
</tr>
<tr>
<td>Total traffic size (measured)</td>
<td>7.7/0.8 kB</td>
<td>6.0/2.1 kB</td>
</tr>
<tr>
<td>Energy used for network transmissions (estimated using model)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Offloaded</td>
<td>25% less/0.3 J</td>
<td>18% more/2.1 J</td>
</tr>
<tr>
<td>Original</td>
<td>6% more/0.5 s</td>
<td>33% more/2.1 s</td>
</tr>
<tr>
<td>Offloaded</td>
<td>17% less/0.5 kB</td>
<td>31% less/1.8 kB</td>
</tr>
<tr>
<td>Original</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Offloaded</td>
<td>46% less/0.04 J</td>
<td>33% more/2.0 J</td>
</tr>
</tbody>
</table>

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


