Mobile Cloud Computing

Mobile Cloud Computing (MCC) is the combination of cloud computing, mobile computing and wireless networks to bring rich computational resources to mobile users, network operators, as well as cloud computing providers. (Wikipedia)

Empowering mobile devices - in particular smartphones and tablets - with the capabilities of stationary resources residing in giant data centers or offloading high computation tasks from mobile devices to data centers.
Motivation

- By 2022 a single device to manage all.
- All in one smart, powerful, and versatile.
- Ultimate fusion of phones, tablets and personal computers.
- So how would mobile cloud computing be different from now?????
mClouds- A Vision

• A mobile cloud computing architecture that runs resource intensive applications on collections of cooperating mobile devices.

• Do it locally if you can.

• What’s in the offer:
  • Take advantage of the improving hardware capabilities.
  • Lower bandwidth usage. Is it really catch??Some statistics:
    • Just 1% of all users now consume half of the entire downlink data.
    • Wireless data traffic of AT&T network went from 0.1 PB in 2006 to 27.1 PB in 2011- a 27000% increase in 5 years.
    • And this will grow even more.
mCloud Design
MCloud Processing

• mDevs – Mobile device members of mCloud.
• mTasks – Processing tasks for mDevs.
• Two types of processing:
  • Individual Processing- Tasks can be accomplished by single devices.
  • Distributed processing- Split heavy tasks into independent smaller subtasks and execute on multiple devices.
    • Master mDev.
    • Slave mDev.
    • When resources not available use backend cloud.
MCloud Management

• Challenges:
  • Dynamic nature.
  • Resource constraints.
  • When to offload to the cloud.

• Solution: Sub Dividing management tasks:
  • Resource Discovery – Master broadcasts solicitation messages.
  • Formation – Slaves respond to master’s call with their unique Id’s.
  • Maintenance – Run resource discovery multiple times or send release messages on specific events.
  • Release – mDev sends a release message anytime.
Incentives

• Why someone should join mClouds on cost of network usage and battery drainage?

• To study the pricing model few terms:
  • $u$ – amount user willing to pay for a service. ($10$)
  • $t$ – time to complete the task with mCloud. ($10$)
  • $m$ – number of slave mDevs ($5$)
  • $w$ – waiting per second ($0.25$)
  • $s$ – average payment to slave mDevs by master ($0.25$)
  • $q$ – cellular usage by master ($1.25$)

Useful if:

$$u - w * t - m * s - q > 0$$
$$10 - 0.25 * 10 - 5 * 0.25 - 1.25 > 0$$
$$10 - 2.5 - 1.25 - 1.25 > 0$$
$$5 > 0$$
Incentive model Implications

• If incentives are provided by carriers – intended to reduce congestion in their networks – may significantly boost the adoption of mClouds.
• In this case $s$ (average payment to slave mDevs by master) becomes 0 and previous cost equation reduces to $U - w*t - q > 0$

This is even greater savings.

• Conclusion: slave mDev will participate if:
  • $s > b + v$ (b is cost of battery drain and v is cellular data cost).
Positive Points

• Can take advantage of local processing to cut the cost of sending data to cloud via cellular network or Wifi.

• Network congestion can drastically be reduced.

• Local computation power can be efficiently utilized.

• A good incentive model can produce a service model which can benefit users and manage cost cuttings.

• Can complement other approaches of edge computing like offloading computation to cloud and Cloudlets.
Negative Points

• A vision paper, no implementation provided.
• A lot of implementation details missing like how to decide how to offload the tasks to slave mDev.
• If cellular data used as a communication medium between mDevs then battery consumption will be high.
• Security and trust mechanisms needs to be touched.
• Author assumes that hardware will grow rapidly but haven’t touched any future prospects of battery power.
• Incentive model touched in a very crude way.
Discussion
Is this idea feasible

• Computation point of view
  • PocketSphinx — A speech recognition engine implementation for mobile devices without backend support.
  • Siri needs internet, Cortana can run without internet.

• Win-Win for both users and network carrier.

• What about battery cost? Will this scale?

• Who will provide this service? Will the user or the mobile carrier?
Few Other Points

• What type of devices should take part in this model?
• How scalable this model can be?
• What about security and trust?
• Performance issues when task offloaded to the cloud.
Questions
STRATUS : Assembling Virtual Platforms from Device Clouds

Center for Experimental Research in Computer Systems
Georgia Institute of Technology

Presented By: Anoop Shukla
Motivation

• A Use case: You are watching a video and as soon you enter your house you can offload this task to a PC at your home which can process the video and can even display the video on some other display device like large screen TV, saving your mobile power.

• In a nutshell how can we freely combine different devices and their capabilities to form a virtual platform to cater the needs of end user?
Stratus is a framework that dynamically constructs, synthesizes virtual platform from sets of network enabled devices, device clouds present in the end user environment. The framework uses a hypervisor on each device to achieve this design.
Design Principles and Assumptions
• Hypervisor-level functionality
  • Discovery: Runtime device discovery and inclusion, exclusion.
  • Management: Monitor the current virtual platform and its management.
  • Automation: Run discovery and management based on user policies.

• Assumption: All devices in device cloud are trustworthy and network delays are negligible as the intra cloud network bandwidth of device cloud are pretty high.
STRATUS Components
1. **Stratus Master:** Responsible include
   - Manage virtual platform and Stratus Nodes lifecycle.
   - Build virtualized platforms based on user policies.

2. **Stratus Node:** Contributes features to each Virtual Platform.

3. **Features:** Building block of a Virtual Platform, is a virtualized resource. E.g. a processor or a display.

4. **Virtual Platform:** A set of features coordinated by the Stratus Master.

5. **Management Channel:** Connects Stratus Nodes and Master. Authors have used improved implementation of M-Channels.

6. **Event Forwarding:** Intra Virtual Platform event management.

7. **User Policies:** Describes desired features with certain quantifiable attributes.
Figure 2. Relation between Stratus Components
Lifecycle of a Virtual Platform

1. Joining a Device Cloud
   • Device registers itself and its features to the Stratus Master.
   • Receives set of available features from Stratus Master.

2. Creation Of Virtual Platform
   • Policy stating the requirement sent to Stratus Master.
   • If features are present Virtual Platform is created.

3. Termination of Virtual Platform
   • Normal Termination
   • Abnormal Termination

4. Leaving a Device Cloud
   • Implicit
   • Explicit
Evaluation
Desktop #1 acts as the STRATUS Master.

<table>
<thead>
<tr>
<th>Devices</th>
<th>Desktop #1 (DT#1)</th>
<th>Desktop #2 (DT#2)</th>
<th>Mobile Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>Quad Core</td>
<td>Dual Core</td>
<td>Atom Z530</td>
</tr>
<tr>
<td>Memory</td>
<td>6GB</td>
<td>2GB</td>
<td>768MB</td>
</tr>
<tr>
<td>Storage</td>
<td>650GB</td>
<td>200GB</td>
<td>16GB</td>
</tr>
<tr>
<td>Network</td>
<td>1Gbit</td>
<td>1Gbit</td>
<td>100Mbit</td>
</tr>
</tbody>
</table>
**Table II. Participants of the Device Cloud**

<table>
<thead>
<tr>
<th>Devices</th>
<th>Desktop #1 (DT#1)</th>
<th>Desktop #2 (DT#2)</th>
<th>Mobile Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>Quad Core</td>
<td>Dual Core</td>
<td>Atom Z530</td>
</tr>
<tr>
<td>Memory</td>
<td>6GB</td>
<td>2GB</td>
<td>768MB</td>
</tr>
<tr>
<td>Storage</td>
<td>650GB</td>
<td>200GB</td>
<td>16GB</td>
</tr>
<tr>
<td>Network</td>
<td>1Gbit</td>
<td>1Gbit</td>
<td>100Mbit</td>
</tr>
</tbody>
</table>

Desktop #1 acts as the STRATUS Master.
### TABLE III. PERFORMANCE COMPARISON: VP VS. NETBOOK

<table>
<thead>
<tr>
<th>Performance</th>
<th>Netbook</th>
<th>Virtual Platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decompressing Time (Sec)</td>
<td>4845</td>
<td>383</td>
</tr>
<tr>
<td>Transcoding Time (Sec)</td>
<td>12312</td>
<td>1386</td>
</tr>
</tbody>
</table>

### TABLE IV. POWER CONSUMPTION COMPARISON - SINGLE VP

<table>
<thead>
<tr>
<th>Policy</th>
<th>Idle</th>
<th>Power Efficiency</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power Consumption (watt)</td>
<td>177.2</td>
<td>178.3</td>
<td>183.4</td>
</tr>
<tr>
<td>Drop Frames/Total (drop-rate%)</td>
<td>-</td>
<td>0/34631 (0%)</td>
<td>0/34631 (0%)</td>
</tr>
</tbody>
</table>

### TABLE V. POWER CONSUMPTION COMPARISON - 5 VPs

<table>
<thead>
<tr>
<th>Policy</th>
<th>Power Efficiency</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power Consumption (W)</td>
<td>201.5</td>
<td>225.2</td>
</tr>
<tr>
<td>Drop Frames/Total (drop-rate%)</td>
<td>9/34631 (~0%)</td>
<td>3/34631 (~0%)</td>
</tr>
</tbody>
</table>
### TABLE VI. DISTRIBUTION OF VPs BASED ON DIFFERENT POLICIES

<table>
<thead>
<tr>
<th>Policy</th>
<th>Power Efficiency</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>DT#1</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>DT#2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Netbook</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

### TABLE VII. COMPARISON: BATTERY LASTING TIME

<table>
<thead>
<tr>
<th>Environment</th>
<th>Netbook</th>
<th>Battery Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Battery Lasting Time (min)</td>
<td>92.8</td>
<td>124</td>
</tr>
<tr>
<td>Power Consumption (W)</td>
<td>16.2</td>
<td>13.3</td>
</tr>
</tbody>
</table>
Evaluation Implications

Experimental results for instance, show a 92% reduction of time, a 11.7% reduction of power consumption, and a 15% battery life enhancement when operating on a Stratus virtual platform vs. on single devices.

Also Virtual Platform construction costs should be included.

<table>
<thead>
<tr>
<th>TABLE IX.</th>
<th>CONSTRUCTION OVERHEAD (LARGER NUMBERS OF NODES)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numbers of Nodes</td>
<td>2</td>
</tr>
<tr>
<td>Delays (us)</td>
<td>5084</td>
</tr>
</tbody>
</table>
Positives

• Exploits the difference properties and capabilities of distributed devices to efficiently provide desired functionality.
• Provides improved experiences to end users.
• A single device cloud within which users can utilize all aggregated resources.
• Elastic virtual platforms.
• Driven by user specified policies.
• Privacy and security issues addressed in way that local computations need not be offloaded to cloud.
• Availability.
Negatives

• Design architecture in detail missing.
• How master node is selected?
• No point mentioned in case of abnormal termination of Virtual Platform.
• Fault-Tolerance mechanisms not covered in case master node fails.
• Doesn't’t include cloud computing in case resources are in scarcity.
• All devices need to have Xen hypervisor.
• Scalable?
• Dynamic reconfiguration of virtual platform missing.
Discussion

• How scalable is this system?
• Can we use containers in place of heavy hypervisor layer?
• Any improvement in the architecture?
Questions