Computing While Charging: Building a Distributed Computing Infrastructure Using Smartphones

Arslan et al.

Presented by John Rula
Motivation

• Unused computation in smartphones
• Wasted cycles while we sleep (and the phone charges)
• Energy efficiency of smartphones
Stated Contributions

- Profiling charging behaviors
- Task scheduling on smartphones
- Migration of tasks across phones
- Automation of task executions
- Preserving user experience
- Implementation and experimentation
Related Work

• Smartphone testbeds
  • Seattle, CrowdLab

• Resource constrained scheduling
  • Condor

• Grid computing
  • Map-reduce, etc…
Feasibility

• Adequate computing power?
• Idle time for computation?
• Stable communication link?
Feasibility cont.

(a) Charging Duration

Day
Night

Charging Duration (Minutes)

CDF

0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

0 150 300 450 600 750

Stability of the wireless network:

Our study suggests that the charging behaviors of users are typical at users’ homes, we conclude that bandwidth stability is not an issue. Cellular links can also be utilized as appropriate.

Low-density wirelessly to smartphones. This makes task completion times sensitive to the bandwidth variability across smartphones. To validate this in practice, we design a simple experiment where we have central server (a regular PC) that interacts with 6 smartphones and the supporting server will typically be via WiFi links where we run an experiment to test the bandwidth. Since we only schedule jobs when a phone is on charge (typically at night, and offer an opportunity for half a day fluctuates when phones are charged again at night.

The server has 600 files to copy to the phones. To do this in parallel, we design a simple experiment where we have file processing machines are inter-connected via Ethernet. The server sends the file to one of the idle phones, which then processes a portion of the files. For each file, the typical cycle is the following: for each file, the server finds the largest idle phone. The server then sets the size of the file to be processed by the phones (each phone finds the largest idle phone). The file is then queued. Since all the phones are initially idle, the first 6 files are processed in parallel without any queuing delays. The server then sends the file to one of the idle phones, which then processes a portion of the file. For each file, the typical cycle is the following: the server finds the largest idle phone. The server then sets the size of the file to be processed by the phones (each phone finds the largest idle phone). The file is then queued. Since all the phones are initially idle, the first 6 files are processed in parallel without any queuing delays.

Figure 2: The median charging interval at night is around 7 hours.

Figure 4: WiFi network stability.
Feasibility cont.

![Graph showing Bandwidth (Kbps) vs. Device Number]
Benefits

• Energy Savings
  • $74.5/year for server, $1.33/year for smartphone
  • $73.16 savings, a 56X reduction!

• Example Applications?
  • Render movie scenes
  • Business analytics
  • Log file analysis
Design and Architecture

• Model similar to map-reduce
  • Distribute executable and data
• “breakable” and “atomic” tasks
• Predictable task execution times
  • $E_j \cdot b_i + L_j \cdot (b_i + c_{ij})$
• Automating Task Execution
  • Android service and “reflection”
• Maintain user expectations and experience
  • MIMD (sleep jobs to nearly match original charging profile)
Predicting Task Times

Network Bandwidth Matters!

Completion Time (ms)

CDF

4 Phones
6 Phones

0 1000 2000 3000 4000 5000

0 0.2 0.4 0.6 0.8 1

Savings in infrastructure costs:

Predicting Task Times

In this section, we describe the design of CWC. We first define the task model and its implications on the overall performance and cost. We then proceed to analyze the task execution time, taking into account factors such as network bandwidth, data transfer, and computational power.
Predicting Task Times

\[ E_j \times b_i + L_j \times (b_i + c_{ij}) \]
Task Scheduling

• Bin packing
  • Want smallest bin height across all bins
  • Greedy: Pack bins with largest jobs (fewest partitions)
  • Binary search on upper bound to get lowest cost

• Handling Failures
  • Mentions timeouts for rescheduling
  • Partial failure recovery and restart
  • No specifics
Evaluation??

• 18 Android phones
• Single run evaluation → Case study at best
• How is data packaged and transferred
• No comparison
• What can this DO??
• Sold on idea of power saving, but no comparison to any workstation it is supposed to be replacing
• Vague on all fronts of implementation and evaluation
• Infrastructure trivial
Benchmarking the Scheduler:

Evaluation?? Cont.

<table>
<thead>
<tr>
<th>Phone Index</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
</tr>
</tbody>
</table>
Evaluation?? Cont.

![Diagram of task re-execution]

- **Phone Index**: 0, 1, 6, 7, 8, 13, 14
- **Time (sec)**: 0, 300, 600, 900, 1200, 1500

**Task re-execution**

Phone 1 is re-scheduled to phones 7, 13, 14.

Phone 6 is re-scheduled to phones 0, 7, 8, 14.

Overall, re-scheduling tasks from failed phones improves efficiency.
Discussion

- General lack of specifics
- Evaluation
- Can smartphones really replace traditional servers?
- Would the power savings really be 56X?