Ubiquitous and Mobile Computing

CS 525M:

Mobile MapReduce: Minimizing Response Time of Computing Intensive Mobile Applications

Vijay Sukhadeve

Computer Science Dept.

Worcester Polytechnic Institute (WPI)
Introduction/motivation

- overall users’ response time due to network problems
- minimize users’ response time
- outsourcing to nearby residential computers vs public clouds
- to build Mobile MapReduce (MMR)
- MMR to leverage the best computing resources to conduct computation.
- Apps: text searching, face detection and image processing
Smartphone Constraints

- Processing power Lagging behind due to size and weight
- Limited battery power, additional consumption due to sensors
To show

- Outsourcing to appropriate resources more advantageous
- Speed up computing – use parallel processing techniques
Idea

- Based on Original MR framework design and implement MMR
- Scheduling Model – Dynamically leverages best computing resources – residential computers vs clouds
- Results: outperforms on-device computing
  Response time: 15 times improvement, battery consumption: 20 times improvement
Path

- leveraging residential computers and MapReduce
- design of MMR
- mobile MapReduce implementation
- evaluation results
- Some related work
Nearby Computers vs. Public Clouds

Experiment: find a string in a text file: response time is longer than if the job is outsourced to nearby residential computers because of the impact of network latency.

<table>
<thead>
<tr>
<th>File Size (KB)</th>
<th>Android</th>
<th>Amazon EC2</th>
<th>Residential Computers</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.0481</td>
<td>0.0146</td>
<td>0.0459</td>
</tr>
<tr>
<td>100</td>
<td>0.425</td>
<td>0.096</td>
<td>0.4245</td>
</tr>
<tr>
<td>200</td>
<td>0.424</td>
<td>0.971</td>
<td>1.300</td>
</tr>
<tr>
<td>400</td>
<td>0.465</td>
<td>1.600</td>
<td>1.300</td>
</tr>
<tr>
<td>750</td>
<td>0.480</td>
<td>3.400</td>
<td>3.300</td>
</tr>
<tr>
<td>1000</td>
<td>0.503</td>
<td>4.500</td>
<td>6.600</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>File Size (KB)</th>
<th>Android</th>
<th>Amazon EC2</th>
<th>Residential Computers</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.117</td>
<td>0.098</td>
<td>0.122</td>
</tr>
<tr>
<td>100</td>
<td>0.332</td>
<td>0.984</td>
<td>1.995</td>
</tr>
<tr>
<td>200</td>
<td>0.886</td>
<td>1.815</td>
<td>4.099</td>
</tr>
<tr>
<td>400</td>
<td>1.327</td>
<td>3.603</td>
<td>8.235</td>
</tr>
<tr>
<td>750</td>
<td>2.467</td>
<td>6.637</td>
<td>15.366</td>
</tr>
<tr>
<td>1000</td>
<td>3.092</td>
<td>8.823</td>
<td>20.583</td>
</tr>
</tbody>
</table>

Table 1. Response Time (Sec)

Table 2. Energy Consumption (J)
energy consumed on the mobile device
bandwidth consumption has to be taken into consideration outsourcing to nearby residential computers is faster
Deciding on what will be the **key** and what will be the **value** is the developer’s responsibility.
Reasons for Modification

- Map and Reduce nodes are connected to each other, which is not always possible in our mobile computing environment.
- HDFS in the original MapReduce contains the data prior to the job submission and computation, which is less likely to be practical in our mobile computing environment.
- Data size in our mobile computing is relatively small.
Architecture of MMR

Resource Overlay – Users residential computers plus public cloud
MMR then submits the job to an appropriate set of computers
MMR Workflow

Mobile Device – Master Node

Residential computer – worker node and may work as Map and reduce
Mobile MapReduce Implementation

- Dynamic Mobility Property of MMR: mobile users without persistent connections and the master node does not have any knowledge about the neighboring worker nodes.
- Non-Distributive File System of MMR: selected nearby residential computers do not have a copy of the input file until the file is transferred there.
- Handling Isolated Worker Nodes
  - In our framework, the Map node sends the list of <Key, Value> pairs to the master node who eventually forwards to the Reducers.
- Node Failure
- As the input data is partitioned into small independent chunks, the failure of any worker causes only re-execution of that portion of data.
Mobile MapReduce Implementation

- Handling Isolated Worker Nodes
- In our framework, the Map node sends the list of `<Key, Value>` pairs to the master node who eventually forwards to the Reducers.
- Node Failure
- As the input data is partitioned into small independent chunks, the failure of any worker causes only re-execution of that portion of data.
Preliminary Evaluation
3 user model and they have identical residential computers and mobile devices

- Text Search
- Face Detection
- Image Sub Pattern Search
Experimental Results

repeat each experiment five times and present the average of the results

- Text Search
  
a) outsourcing to EC2 results in the worst performance in terms of both the response time to the user and the amount of energy consumed

b) computation is parallelized among multiple machines shows that the response time and energy consumption first decrease with the increase of parallelization level

(a) Response Time and Energy Consumption
(b) Response Time and Energy Consumption
References

7. iPhone Heart Monitor Tracks Your Heartbeat Unless You Are Dead. gizmodo.com/5056167/.