Ubiquitous and Mobile Computing
CS 525M: Fast App Launching for Mobile Devices Using Predictive User Context

XIAOCHEN HUANG

Computer Science Dept.
Worcester Polytechnic Institute (WPI)
AUTHORS

Tingxin Yan  David Chu  Deepak Ganesan  Aman Kansal  Jie Liu

University of Massachusetts Amherst, Microsoft Research
{yan, gamesman}@cs.umass.edu, {davidchu, kansal, jie.liu}@microsoft.com
Mobile Apps … Loading Slowly

Measured on Samsung Focus Windows Phone

Average loading time >12s
Slow network Content Fetching

Email

News

Social

Loading time > 10s in 3G

Measured on iPhone 3GS
## Two approaches …

<table>
<thead>
<tr>
<th>Cache apps in memory</th>
<th>Push notification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decrease launch time</td>
<td>Address stale content</td>
</tr>
</tbody>
</table>

- Demand large amounts of memory, overwhelming memory real estate for other apps
- Content may become stale by the time the user interacts with the app

- The energy cost of push communication can be prohibitively high
**FALCON**

- **What is Prelaunch?**
  
  *Prelaunch*: Schedule an app to run before the user launches it

- **Challenges and Approaches**
  
  **Which app to prelaunch?**
  
  Context clues indicate *predictable* app usage patterns

  Data insights shed light on informative context

  **At what cost and benefit?**
  
  Cost: energy, memory
  Benefit: latency reduction

  Problem formulation efficiently provides optimal solution

  **Requiring what systems support?**

  FALCON’s scheduling and memory management complement kernel’s

  Windows Phone OS mod prototype implementation
BACKGROUND AND PROBLEM SCOPE

Slow launch times & Brief usage durations

- Examine the thesis that mobile interactions are inherently brief
- Slow app launch is a substantial drag on use experience
FALCON Architecture & Prototype

Context Source Manager
- location
- Apps Used
- Time
- ...

Launch Predictor
- Feature Extractors
- Decision Engine
- Model Trainer
- Proc Tracker

Server-based Model Trainer

Inference Cost-Benefit Analysis

Dispatcher

Kernel Memory Manager

Context
- Trigger
- Location
- burst
Personalized Features

- **Triggers Context**

  **Session**
  
  ![](image)

  **Triggers**
  - Email
  - News
  - Games

  **Followers**

  **Most popular triggers overall**

  ![Graph showing SMS, Email, Social, Phone, Browser]

  **Trigger Context**: Given a trigger app, identify most likely follower apps

  [interpretation]
  
  *Interruptions lead us to habitual routines*
Personalized Features

• Triggers Context
  
  • Problem
    
    The best triggers are different for different applications
    
    Significant variability across users

  • Dynamic triggers
    
    Be calculated on a prelaunch-candidate basis as the set of top-k triggers most likely to lead to the launch candidate as a follower
Personalized Features

• Location Context

[Graph showing app usage probabilities for HOME, WORK, and SHOP]

App usage is correlated with location

[interpretation] What we are doing depends on Where we are
Personalized Features

- Location Context

Productivity Apps @ Work

Social Apps @ Outdoors
Personalized Features

- Burst Context

App usage likelihood changes throughout the day as well
Decision Engine

- Step 1: Likelihood Estimation
  - Context
    - Trigger, Location, Burst
  - App Likelihood Estimation
    - Conditional probability on <Trigger, Location, Burst>
  - Likelihood Estimates used as ranking metric...
Decision Engine

• Step2: Cost-Benefit Analysis

Likelihood Estimates

Potential Benefit: Latency reduction

Potential Benefit: Energy wasted

Optimization

• Maximize Latency Reduction
• Subject to energy budget

• Knapsack Problem
• Top-K Greedy approximation
Implementation

- An OS services
- New application event upcall
- An ambient display widget
Evaluation: Micro-benchmarks

- Three critical parameters
  - the time taken for launching an app
  - the amount of energy consumed during app launch
  - the memory consumed by the app at launch time
Evaluation: Benefits of Individual Features

• Session Triggers and Followers
  • Triggers need to be both user- and app-specific
  • Make the case for Dynamic personalized triggers

(a) Top 5 trigger across all users
(b) Top 5 triggers for a specific user
(c) Top 5 triggers for “Angry Birds”
Evaluation: Benefits of Individual Features

• Temporal Bursts

  • An effective feature for improving prelaunch accuracy
  • Three categories: all applications, Games, and First-party apps
  • A discriminating feature that can improve prelaunching performance, especially for games
Evaluation: Benefits of Individual Features

- Location Clusters
  - Three categories: all applications, Games, and First-party apps
  - Also a discriminating feature, particularly for games
Evaluation: Combining Features

- Benefits of location + temporal features
  - Combining the two features gives better performance
Evaluation: Combining Features

• Benefits of dynamic triggers
  • The performance of dynamic triggers for each location is stable for different users

![Graph for User 1](image1.png)

![Graph for User 2](image2.png)

a. User 1  
b. User 2
Evaluation: Evaluation of cost-benefit learner

• Performance of Prefetching
  • Look at the benefit for apps that require fetching new content during the launch process
Evaluation: Evaluation of cost-benefit learner

- Performance of Preloading
  - Look at how app loading time can be improved by CBL learner, compared with LRU caching
Evaluation: Evaluation of cost-benefit learner

- Overall benefits
  - X-axis represents the energy budget provided to CBL
  - Y-axis represents the benefit of loading time which includes both app loading time and content fetching time
Evaluation: Bootstrapping FALCON

- Look at how fast the cost-benefit learner can learn from history of app usage to make accurate prelaunch decisions.
- The performance of FALCON grows as the training data size increases.

(a) Precision

(b) Recall

Precision and recall of bootstrapping
Evaluation: System Overhead

- The resource consumption profile as follows

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary Size</td>
<td>129 KB</td>
</tr>
<tr>
<td>Memory (stable state)</td>
<td>1840 KB</td>
</tr>
<tr>
<td>Processor utilization (stable state)</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>Processor utilization per prediction</td>
<td>&lt;3%</td>
</tr>
<tr>
<td>Energy cost per prediction</td>
<td>&lt; 3 $\mu$Ah</td>
</tr>
</tbody>
</table>

- Do not account for the periodic geolocation sampling costs

- Online feature extraction
  
  (a) Perform location clustering cost much on a phone, therefore need to be done in the cloud
  
  (b) Implement a light-weight online burst window detection algorithm
  
  (c) Online burst detection as a small performance loss
DISCUSSION AND CONCLUSION

• Contribution
  • From extensive data analysis, design spatial and temporal features that are highly indicative of mobile app access patterns
  • Design a cost-benefit learning algorithm
  • Prototype FALCON on a Windows Phone

• Future work
  • Eliminating reliance on external servers or cloud services for model training in FALCON
  • How users’ expectations will change as the OS predictively prelaunches apps on their behalf
THE END
THANK YOU!