Dual Scheme Phone: A User Assisted Mechanism to Effectively Run Sensor Analytics Applications on Smartphones

Swarnava Dey, Arijit Mukherjee, Pubali Datta, Himadri S Paul
Innovation Labs Kolkata
Tata Consultancy Services
{swarnava.dey, mukherjee.arijit, pubali.datta, himadrisekhar.paul}@tcs.com

Anupam Basu
Computer Sc. & Engg.
IIT Kharagpur
anupam@cse.iitkgp.ernet.in

Categories and Subject Descriptors: H.4 [Information Systems Applications]: Miscellaneous; D.2.8 [Software Engineering]: Metrics—*complexity measures, performance measures*

General Terms: Theory

Keywords: smartphone, resource prediction, machine learning, usage pattern

ABSTRACT

In recent times, many human-centric applications are being developed to leverage the diverse range of sensors present in Android smartphones. Smartphones are also being used as edge network gateways for fusing data from multiple sensors. These applications often run as background services and usually obtain and process sensor data. However, smartphones are traditionally optimized for prioritizing foreground and interactive applications and the user remains unaware about the resource-starved background applications, some of which may be important to him. In this work we present the argument in favour of optimizing the smartphones for background services too with a certain degree of user intervention. We exploit the user specific unique usage pattern of smartphones and show that machine learning techniques can be applied to predict CPU usage of applications. Based on the system-wide and app-specific predicted values of CPU usage, our system detects if any preferred background service is resource-starved and alerts the user accordingly. We show with an example sensor analytics application, that the accuracy of data collection and analysis of that application increase significantly by applying our scheme. We further show that irrespective of variations in the accuracy of prediction due to users, handsets and algorithms used, reasonable period of forecast about future resource usage is possible using historical data about the applications under execution and different context parameters including time of day, day of week, network cell, location, accelerometer data.

MOBIQUITOUS 2014, December 02-05, London, Great Britain

DOI 10.4108/icst.mobiquitous.2014.258081

1. INTRODUCTION

During the last few years, smartphones and the in-phone sensors are being used to develop diverse range of applications (apps) and some of these even solve real time and mission critical problems including monitoring fire-fighting, heart rate. Smartphones are also used to fuse data from external sensors in Fog Computing [1] and IoT [2]. These apps often run as background services and usually obtain and process sensor data. The primary apps running in smartphones are however games, browsers, social networking apps, or related to communication, productivity etc. [3]. As people traditionally gauge the performance of a smartphone based on the responsiveness of the user interface and foreground apps, smartphones are optimised for prioritizing foreground and interactive applications. However, in our opinion, to maximize the benefit of running the new breed of sensor analytic applications in smartphones, user's idea of priority should override some of the system defaults. In this work we try to accommodate the resource needs of such background services perceived important by the user, remaining within the realms of currently existing smartphone scheduling policies. We propose a scheme and present a prototype implementation that exploits the user specific unique phone usage patterns and applies machine learning based time series forecasting techniques to predict CPU usage based on different context parameters including time of day, day of week, network cell, location, accelerometer data and apps under execution. Based on the predicted system-wide and app-specific CPU usage, our system detects potential resource starvation for preferred background services and alerts the user. This spares the user selected background processes from being resource starved. Effectively, the proposed implementation serves as a wrapper on Android to implement a different optimisation policy that favours the interactive and foreground processes, but can accommodate the need of a selected background service, when specified by the user. We show that MobiDriveScore [4](a sensor analytics app that logs GPS and accelerometer data from phone and analyses the same to model driving behaviour and road condition) experience lesser measurement and analysis lag when our scheme is used.

2. DESIGN AND IMPLEMENTATION

Earlier research efforts including Falaki et. al. [5] etc. shows that unique usage patterns exist for mobile phone usage on a per user basis. With this idea of *user specific patterns*, we try to predict future CPU load, based on the apps under execution. Our usage prediction system can be specified as follows: Let $J = \{j_1 , j_2, . . . , j_n \}$ be the set of apps , $J\prime_\tau \in J$ be the set jobs that use relatively high memory during the interval τ, j_{bg} be the user preferred background sensor analytics service, $C_\tau = \{c_1 , c_2, . . . , c_k \}$ be the set contextual parameters including time, day, location, network cell etc. during interval τ ; For a time interval τ, the output of the data collection will be the following: $O^{cpu}_\tau = \{$ *low*, *average*, *high* $\}$ depending on the combination of apps in $J\prime_\tau$, where the levels are obtained by running unsupervised clustering on app wise CPU usage. A set of instances of $J\prime_\tau, \hat{J} = \{J\prime_{\tau_1}, J\prime_{\tau_n}, ... J\prime_{\tau_p}\}$ may be used as features to classify the output O^{cpu}_τ. $\hat{J}$ in conjunction with the context parameters C_τ may be used as historical data for prediction of system resources for an arbitrary interval τ_n.
If the predicted output from such a system, $O^{cpu}_{\tau_n}$, is *high* and for the same τ_n CPU demand of j_{bg} is also *high*, we raise an alert for the user. As given in Fig. 1, the alert raised by our

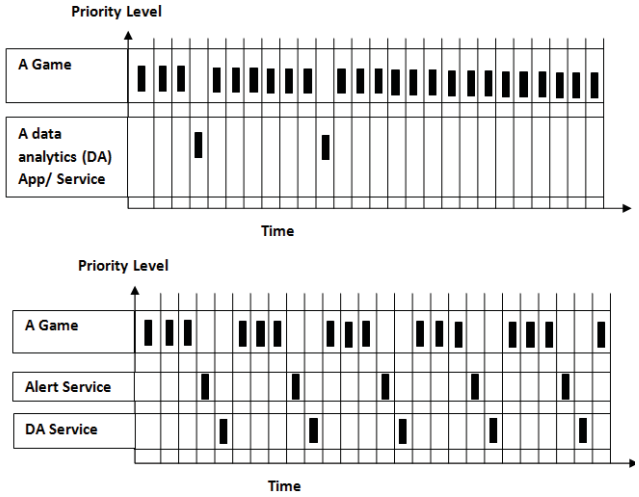


Figure 1: Alert service to notify resource starvation of user's preferred app

system will also allow the user's preferred app to get access to CPU, as the current foreground activity will be paused.

We implement the scheme in Fig. 2, where user intervenes with the scheduling from userspace by pausing some foreground app, to facilitate uninterrupted running of preferred apps.

To build usage model, we log app wise and overall CPU usage, context parameters from users. We implement an adaptive lightweight data collection scheme given in algorithm 1, that prolongs the data collection interval based on average app usage time reported by Böhmer et. al. in [3].

We observe that CPU usage of the apps that vary at different periods, follow a pattern. Sometimes CPU usage percentage of these apps fall in a lower bracket and sometimes in a higher bracket. However, the high or low bracket of CPU usage continues for certain time. During this time period, the per-app CPU usage remains quite uniform. This

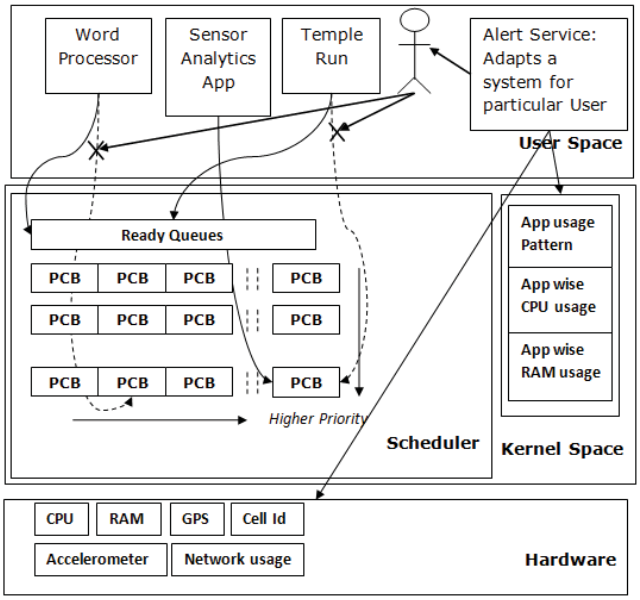
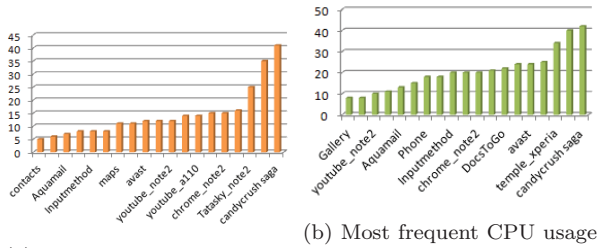


Figure 2: Alert service for raising execution priority of preferred app



(a) Mean Value - CPU usage (b) Most frequent CPU usage brackets

Figure 3: Determining the top CPU consuming apps

measure in Fig. 3b is different from per app average CPU usage in Fig. 3a and helps in the selection of the top CPU consuming apps during a time period.

We use Android APIs and Linux `/proc` filesystem to get system load. We apply k-means clustering on the idle CPU usage values, to reach three clusters *low*, *average*, *high*. We apply standard machine learning algorithms after removing temporal ordering of individual feature instances, using Weka tool API [6]. Weka achieves this by encoding the time dependency via additional input fields, added as lagged variables in each row. We adjust the lag variables, try different base learners for comparing prediction errors for selecting the base learners for building offline usage models.

3. EVALUATION

We observe that SVM with polynomial kernel and Random Forest gives better accuracy with respect to Mean Absolute Error (MAE), Root Mean Square Error (RMSE) and Mean Square Error (MSE). We present a sample MAE comparison between SVM with polynomial kernel and Random forest and comparison of SVM based prediction between datasets of two users in Fig. 4. We observe that with a history data of less than 24 hours, the prediction errors are high. Better forecasting happens once we have data for more than a day, i.e., the diurnal pattern is captured in the model. Beyond

Algorithm 1: Algorithm for data collection

```
1  Algorithm: CollectData:AppTable,CurrApp,LastApp, lastSkip
2  begin
3      foreach i e AppTable do
4          key ← getKey(i);
5          if key matches CurrApp then
6              value[] ← getValue(key);
7              avgRunTime ← value[1];
8              lastRunTime ← value[2];
9              if CurrApp matches LastApp then
10                 timer ←
                   avgRunTime − lastSkip − lastRunTime;
11                 LastRunTime ← lastRunTime + lastSkip;
12             else
13                 timer ← avgRunTime − lastSkip/2
                   avgRunTime ←
                   (avgRunTime + lastRunTime)/2;
14                 lastRunTime ← lastSkip/2;
15             if timer >= 25 then
16                 timer ← 25;
17                 lastSkip ← timer;
18                 value[2] ← lastRunTime;
19                 setKey(AppTable);
20     return (timer);
```

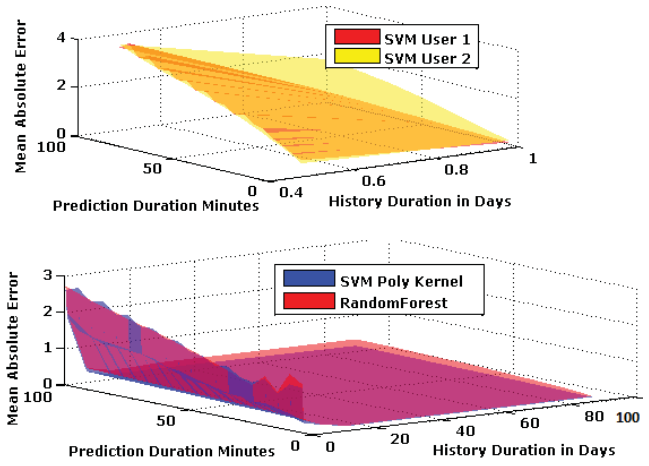
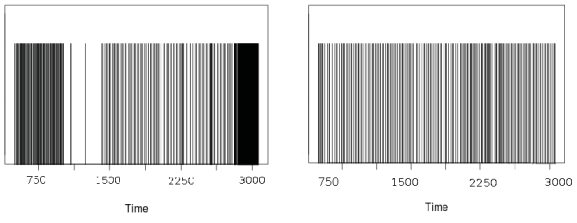


Figure 4: Comparison of different time series forecasting algorithms and on our datasets from different users

the weekly patterns, however, there is not much noticeable change in prediction accuracy. To demonstrate the useful-



(a) Without our scheme (b) Applying our scheme

Figure 5: Scheduling frequency of the MobiDriveScore test app

ness of our proposal, we have created a prototype app running on an Android smartphone which flashes alert when the chosen background app *MobiDriveScore* is starved of system resources. Fig. 5b and Fig. 5a shows that *MobiDriveScore* app was scheduled periodically and got access to CPU more often during a session of heavy movie watching and gaming, by application of our scheme. Discussing the results of *MobiDriveScore* with its developers, we understood that inaccuracy in data collection and processing may not affect the driving pattern recognition for a long drive but will surely alter the road condition model created by that app.

4. CONCLUSIONS

The issue of low priority sensor analytics service not getting access due the default scheduling policy of Android Operating System and producing inaccurate results can be solved by implementing the service as a part of a foreground app. What if the user has one preferred app among several foreground apps running and needs to ensure that the preferred one gets access to resources when needed? To achieve the above goal we predicted systemwide and per-app CPU usage based on apps under execution and context data. We devised a low overhead data collection scheme, analysed in depth to find the CPU usage brackets of apps and applied standard machine learning based time-series analysis techniques to forecast heavy CPU load scenarios. If a preferred app is CPU starved in those periods, our system provided alert to the user to pause the then current activity. We showed that a sensor data analytics app is benefitted by applying our scheme and overcame the lag in data collection and analysis in smartphones under active usage. This work is still in its preliminary stages and we plan to integrate this into OS scheduling policy so that the smartphone OS provides options for accepting user's idea of priority along with system defaults.

References

[1] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proc. of the 1st Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, (New York, NY, USA), pp. 13–16, ACM, 2012.

[2] A. Mukherjee, H. Paul, S. Dey, and A. Banerjee, "Angels for distributed analytics in iot," in *Proc. of the 2014 IEEE World Forum on Internet of Things (WF-IoT)*, WF-IOT '14, pp. 565 – 570, IEEE, 2014.

[3] M. Böhmer, B. Hecht, J. Schöning, A. Krüger, and G. Bauer, "Falling asleep with angry birds, facebook and kindle: A large scale study on mobile application usage," in *Proc of the 13th Int Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '11, (New York, NY, USA), pp. 47–56, ACM, 2011.

[4] T. Chakravarty, A. Ghose, C. Bhaumik, and A. Chowdhury, "Mobidrivescore: A system for mobile sensor based driving analysis: A risk assessment model for improving one's driving," in *Sensing Technology (ICST), 2013 Seventh Int Conference on*, pp. 338–344, Dec 2013.

[5] H. Falaki, R. Mahajan, S. Kandula, D. Lymberopoulos, R. Govindan, and D. Estrin, "Diversity in smartphone usage," in *Proc of the 8th Int Conference on Mobile Systems, Applications, and Services*, MobiSys '10, (New York, NY, USA), pp. 179–194, ACM, 2010.

[6] "Time Series Analysis and Forecasting with Weka." `http://wiki.pentaho.com/display/DATAMINING/Time+Series+Analysis+and+Forecasting+with+Weka`.