PocketData: Establishing Empathy with an App

Gokhan Kul and Gourab Mitra

University at Buffalo, 338 Davis Hall, Buffalo, NY 14260

ABSTRACT

Mobile databases are the statutory backbones of many applications on smartphones. Their performance very much depends on the performance of the underlying databases. However, these databases and the querying engines in the applications are usually uncontrolled, not properly designed and not tuned for optimal performance. We take the initiative to analyze mobile database logs to investigate the interaction between the application and the database to model the application characteristics. Although various techniques have already been produced for database log exploration, they target enterprise environments where the data is accessed from many different machines and by many different users. On Android phones, on the other hand, the database is exclusive to one application, which means, understanding the log can lead to understanding the application, hence, improving the performance considerably. In this paper, we first describe the PocketData dataset while pointing out the details that we exploit. We then propose a clustering scheme where we analyze the query logs to identify and group the SQL queries with similar interests together. Finally, we elaborate on using these clusters to model common behaviors and interesting patterns. We show experimentally that the clustering scheme is able to categorize queries with similar interests together.

Keywords: Benchmark, Database, Workload, Mobile Systems

1. INTRODUCTION

Smartphones we so naturally carry and use today do not have a long history. The smartphones as we know them today started to get used worldwide by average people around 2007. Of course, there were earlier representatives of smartphones, but most of them did not reach to the mass crowds due to their price and lack of network coverage. This fundamental shift in technology pushed phone-makers into adopting their technology into the new trends as fast as possible, to be able to stay in the market. Once world leaders Nokia, Ericsson and Blackberry rapidly lost their market shares due to failing to adapt to the new trends. However, these developments led to a chaotic environment, where creating clear and consistent standards got disregarded.

One of the areas affected by this environment was how the data is stored on these devices. The data can be structured or unstructured, and the data storage methodologies got adopted from computers which had a lot more processing power and available memory. Although the processing power and memory barriers are fading away with the current technology in the smartphones, the applications still depend on either file-based storages like JSON and CSV or embedded SQL database systems like SQLite.[1, 2]

We argue that many apps could benefit from understanding how the user uses that specific app. For example, Bob, an *Android photographer*, may be using Instagram to post a lot of pictures for the brands, hotels and touristic places. Alice, on the other hand, may be using Instagram for browsing photos from the users she follows, and may not have the habit of posting too many photos. The workload these two people create on the local mobile database is different and should be addressed accordingly. We can utilize this usage characteristics information to (1) increase performance for various workloads, (2) find out bugs and unnecessary database calls in the apps, (3) give more accurate recommendations to the user, and (4) explore the data flow improvement opportunities within the app.

Further author information:
Gokhan Kul: E-mail: gokhanku@buffalo.edu, Telephone: 1 716 243 1729
Gourab Mitra: E-mail: gourabmi@buffalo.edu, Telephone: 1 716 816 9096

This paper represents a step towards creating empathy between the app database and the developers using query logs. Concretely, in this paper we: (1) motivate for a mobile database benchmark, (2) utilize query similarity metrics to find similar queries by structural similarity, (3) analyze the vectors and similarity matrix of the query load to create clusters of structurally similar queries, and (4) introduce techniques for exploring repeating usage patterns and interesting usage characteristics. We finally experimentally demonstrate that our methods are able to infer usage characteristics of users.

This paper is organized as follows. We first describe the motivation and give background reasoning in Section 2. Then, we introduce a sample dataset for workload characterization and explain methods we utilize in Section 3. In Section 4, we evaluate the accuracy and performance of our proposed techniques. Finally, we conclude in Section 5 and identify the steps needed to deploy our methods into practice in Section 6.

2. BACKGROUND AND MOTIVATION

The lack of standards and the need for a better understanding of mobile storage systems can easily be seen by surveying through standardized and well-known mobile database system benchmarks, which in fact non-existant,[3] while traditional database systems have a few dependable benchmarks.[4,5] Also, traditional database systems are usually managed by professional database administrators who tune-up the databases according to changing workloads while smartphone databases work with predetermined indexes and are not subject to tuning up depending on the workload they are experiencing. Although there were some efforts to measure the performance of SQLite and Android devices under different workloads,[6] these benchmarks do not specify the bottlenecks, how and where the tune ups should be performed or they do not provide any information specific to the app performance.

While the most visible parameter of data is its volume, it is not the only characteristic that matters. In fact, there are four key characteristics about data:[7]

- Volume. The most intuitive characteristic about data is the amount of data itself. It fact, it is the sheer amount of data that we generate and process these days that calls for a better approach to data management. It is one of the driving forces behind this work.
- Velocity. Velocity refers to the idea of the amount of data flowing through an interface in unit time.
- Variety. Traditional data formats were relatively well defined by a data schema and changed slowly. In contrast, modern data formats change at a dizzying rate. This is referred to as variety in data.
- Value. The value of data varies significatly. The challenge in drawing insights from data is identifying what is valuable and then transforming and extracting the data for analysis.

Database servers and web applications experience a workload that is not typical in smartphones. Majority of these servers form the backbone of an application. In most cases, they store the business data to support OLTP and OLAP operations. The data volumes may range from medium to high amounts for systems with a large concurrent user base. The data velocity also grows in proportion to the number of concurrent users.

The usage pattern of databases in smartphones differs significatly from the above mentioned ideas. Most modern day smartphones rely on some kind of a web service to help a mobile application deliver the desired functionality to the user. This introduces various new application design cnsiderations. Mobile users must be able to work without a network connection due to poor or non-existent connection. In that case, a mobile database serves as a cache to hold recently accessed data and transactions so that they are not lost due to connection failure. In many cases, users might not expect live data during connection failures; only recently modified data. Update of recent changes and downloading of live data can be deffered until connection is restored.

Mobile computing devices tend to have slower CPUs and limited battery life. The luxury of having a cluster of powerful computers to deploy a database is just not there. Also, the fact that battery power is scarce drives the case furthur to achieve high resource utilzation.

It is a common practice among smartphone users to have multiple devices. Most smartphones have an authentication system which is powered by an email account which is accessed in other devices too. In most cases, the user has atleast one more device or a web service that needs to sync with the smartphone. This leads to occasional synchronization activities that occur between different devices and a centralized data store. Often, this activity happens in background so that the user is not blocked from using other functions on a smartphone.

The PocketData dataset[3] provides handset-based data directly collected from smartphones for multiple users. User sessions in context of smartphones might not be similar to a session on other more traditional computinf devices like PCs. Typically, an end user would use their smartphones in multiple small intervals of time through a day. These 'bursts of activity' can be referred to as session. In context of a single mobile application, the user would access multiple logical transactions in these bursts of activity. Intuitively, a user session is quite straightforward to understand but its technical aspects require defining. Some smartphone usage studies define a session as the time period where the smartphone's screen is active.[8] Smartphone usage is dominated by usage of the applications that the smartphone has to offer. Thus, the idea of a smartphone usage session can be reduced to an application usage session. This is relevant to us because we aim to study the interaction with the smartphone database through understanding a single application.

3. METHODOLOGY

We propose a heuristic to analyze query logs and find out interesting patterns. This process has three steps:

1. Figuring out the activities of interest with respect to the application
2. Clustering the queries
   (a) Extracting features
   (b) Query comparison
   (c) Clustering with different strategies
3. Detecting patterns in user activity
   (a) Appoint an label to each cluster
   (b) Identify which cluster a new-coming query belongs to
   (c) Identify patterns with different strategies

The strategies for applying these steps are given in this section.

3.1 Dataset

As a sample dataset, we use PocketData[3] dataset. This dataset includes one month's trace of SQLite activity on 11 PhoneLab[9] smartphones running the Android smartphone platform.

This dataset consists of various information about the usage patterns across a wide variety of apps. Each line in the log has:

- Device ID: Unique identifier for each device
- UNIX timestamp: Milliseconds since 1970
- Ordering: Timestamp and request order
- Date and time: Human readable timestamp
- Process ID: Standard UNIX process ID
- Thread ID: Standard UNIX thread ID
- Log level: Verbose (V), Debug (D), Info (I), Warning (W), Error (E)

- Tag: Source of log information, "SQLite-Query-PhoneLab"
- JSON object that holds various information about the event that is logged

information.

Note that the app ID is not included in the log, because different apps can have the same process and thread IDs in different times. Our strategy to get the log lines for our app of interest is to search for the app name in JSON object parsing from the beginning of the file. When we first encounter the app of interest, we use process and thread IDs to identify the events related to that app until we encounter a different app name in the JSON object.

3.2 Clustering

In the clustering process, we first filter the activities belonging to the app of our interest without distinguishing which user the activity belongs to. Then, we create clusters using all the activities belonging to that specific app. The workflow is illustrated in Figure 1.

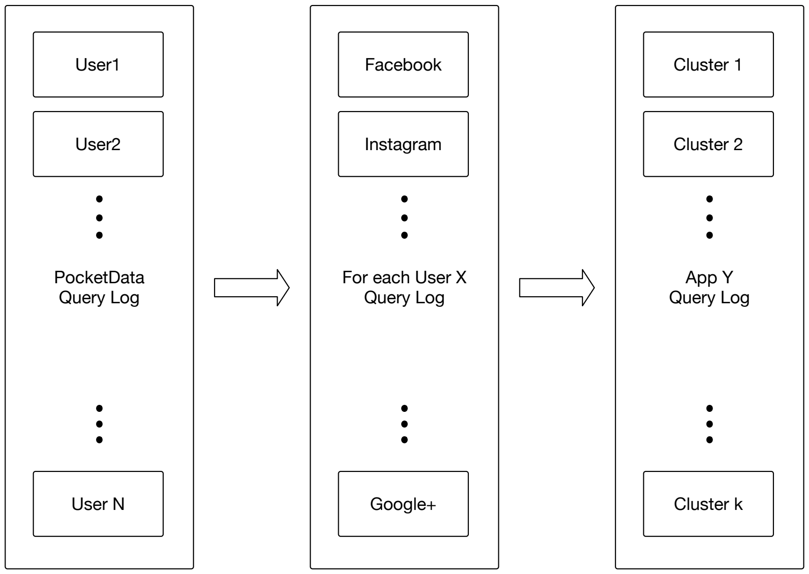

Figure 1: The workflow for clustering process

To achieve this we need to be able to extract features out of SQL queries, compare them and compute their similarity. Extracting features from a SQL query, can be done in many ways. Let's consider the following queries:

```
Q1: SELECT username FROM user WHERE rank = "admin"

Q2: SELECT rank, count(*) FROM user
    WHERE rank <> "admin" GROUP BY rank
```

These two queries share many attributes and seem to be working on similar concepts. Usually, what we consider important in a query can roughly be listed as:

- Selection
- Joins

- Group-By
- Projection
- Order-By

One of the main topics in Ettu[10] project is to investigate how SQL queries can be compared effectively and accurately. In their research,[11] the authors survey the literature for the methodologies used in this field and elaborate on three available methods.[12–14]

Aouiche *et al.*[12] utilize a pairwise similarity metric between two SQL queries in order to optimize view selection in data warehouses by creating feature vectors out of the *selection*, *joins* and *group by* items in the query while not considering the appearance count. They use Hamming Distance to calculate a similarity value between these two vectors.

Aligon *et al.*[13] present a survey on a great range of approaches which seek to put forward a similarity function to compare the similarity of OLAP sessions. Inspired by their findings, they propose their own query similarity metric which considers *projection*, *group by*, *selection* and *join* items for queries issued on OLAP datacubes. Their method creates separate sets for each of these components and appoint an importance weighting to each set. When comparing two SQL queries, each of these three sets get compared to the other query's corresponding set, hence, by using Jaccard Coefficient, they get a similarity score for each set. Finally, they compute an overall similarity score by the average of these three scores.

Makiyama *et al.*[14] put forward the most similar work we are working on. They perform query log analysis with a motivation of analyzing the workload on the system, and they provide a set of experiments on Sloan Digital Sky Survey (SDSS) dataset. They extract the terms in *selection, joins, projection, from, group by and order by* items separately, and create the query vector out of their appearance frequency for each query in the dataset. They compute the pairwise similarity of queries with cosine similarity.

When we inspect closely, we can easily see that Aouiche method is a naive version of the other two methods. In our experiments, we apply both Aligon and Makiyama methods, since they have competitive performance as shown in Ettu's log summarization study.[11] Makiyama method can be used to perform k-means clustering since it provides the feature vectors, whereas Aligon method can be used to apply hierarchical clustering since it is used to create a distance matrix. To clarify the ambiguity between distance and similarity terms, we define distance as follows:

$$distance = 1 - similarity$$

where the similarity is the score we get from the methods explained above.

We consider two possible clustering approaches for use in our system: k-means and hierarchical clustering.[15] K-means outputs a set of queries for k clusters. On the other hand, hierarchical clustering outputs a dendrogram – a tree structure which shows how each query can be grouped together. In addition, a dendrogram is a convenient way to visualize the relationship between queries and how each query is grouped in the clustering process. We will explore the possibility of using both of these methods.

3.3 Pattern Matching

Soikkeli *et al.*[8] say that even considering the time between launch and close of an application is not a reliable notion of an application usage session. Applications running in the foreground are visible to the user. Applications which are running in background are not visible to user even though he might have launched them before. An individual session now consists of two parameters : a start time and an end time. Two user sessions can be back to back or might have an idle time in between them. User sessions for a single application can now be modelled as a time-wise closely-spaced series of queries issued to the smartphone database. A threshold value T is defined for the idle time between two transactions. If the idle time is less than or equal compared to T, the transaction belongs belong to the same user session.

The above mentioned approach would enable us to identify a time window which can be applied to the transactions in the PocketData dataset. This time window would contain a chronologically ordered subset of queries issued to the smartphone database. We pick a simple and easy-to-apply measure, Longest Common Sunsequence,[16] to study multiple sessions for the same user.

After the clustering operation for every user in the dataset, we would annotate each query with the cluster it belongs to. Each cluster is a logical group of queries with similar intents. The cluster would be represented by an integer ID. At this stage, we need to find repeating patterns for a single user. It is achieved by identifying the common subsequences of query ids across sessions for a user. The top k longest common subsequences across multiple sessions of a particular user become candidates of interest now. They are now vetted against other top k longest common subsequences of other users in the dataset. We can apply cross-correlation to indentify the most interesting and prevalent pattern among all top k subsequences of all users in the dataset.

We use a ranking formulae to order the all the candidate subsequences among top k subsequences for each of the n users in the dataset.

$$\rho_{S_i,U_j} = \eta\, r^2$$

where

$$S_i\ is\ the\ i^{st}\ longest\ subsequence\ for\ a\ user$$
$$U_j\ is\ the\ j^{th}\ user\ in\ the\ dataset$$
$$\rho_{S_i,U_j}\ is\ the\ rank\ of\ subsequence\ S_i\ of\ User\ U_i\ when\ vetted\ against\ all\ S_a\ and\ U_b\ \forall\ a\ in\ (1,k),\ b\ in\ (1,n)$$
$$r^2\ is\ square\ of\ Coefficient\ of\ correlation\ r\quad 0 \le r^2 \le 1$$

We order the candidate subsequences in non-increasing order of ρ and choose the topmost. Now that this subsequence is obtained, it can be used to construct a feature vector to characterize the workload of the application

4. EXPERIMENTS

4.1 Environment

In our experiments regarding clustering, we used an Apple Macbook Pro with macOS Sierra operating system, 2.7 GHz Intel Core i5 processor, 8GB RAM, Java 1.8 SE Runtime Environment and R v3.3.2.

In our experiments regarding pattern matching, we used a Lenovo Thinkpad with Windows 10, 2.3 GHz Intel Core i5 processor, 8GB RAM, Java 1.8 SE Runtime Environment and R v3.3.1.

We will provide timing results of our experiments in the next phase of our project.

4.2 Clustering

For our experiments, we selected Facebook to be our example app. For visual purposes, we clustered the activities of only one user in Figure 2. For this specific user's case, there are 84273 rows of activities in the log. There are 8795 parsable select queries, however, there are only 59 unique queries among them. Keep in mind that PocketData dataset is an anonymized dataset where most of the constant values are replaced with "?", which reduces the number of distinct queries greatly. The dendrogram we created using Makiyama method and hierarchical clustering can be seen in Figure 2.

As random examples, here are some the queries that are put in the same clusters:

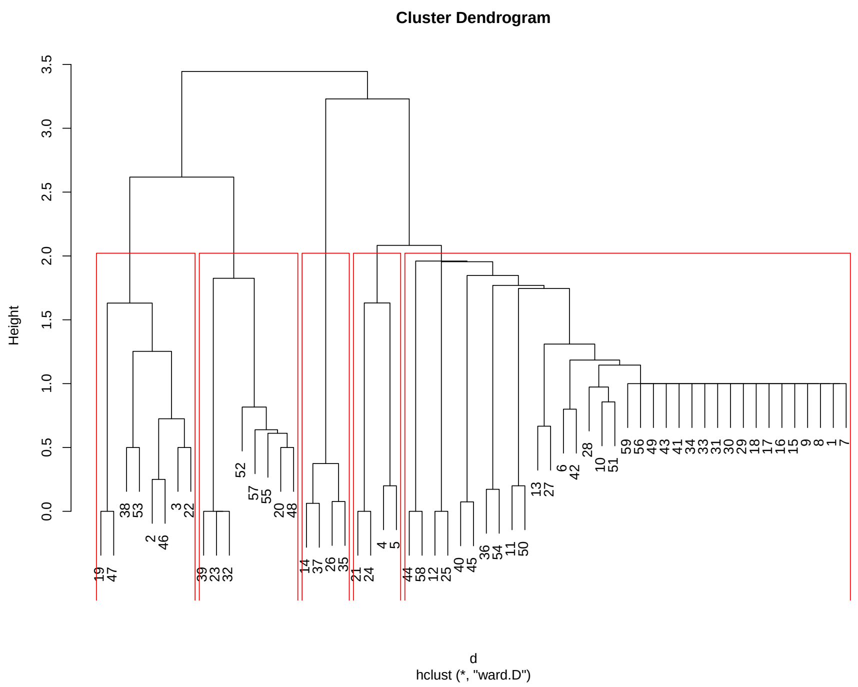

Figure 2: Dendrogram of Facebook usage for a user

```
Cluster 1:
19 -> SELECT seen_state, updated, cursor, cache_id, dashing, icon_uri, photo_uri,
profile_picture_uri, summary_graphql_text_with_entities,
short_summary_graphql_text_with_entities, notif_id, star_rating
FROM gql_notifications
WHERE (recipient_id=?) ORDER BY updated DESC LIMIT 4

38 -> SELECT * FROM gql_notifications WHERE (notif_id=?) ORDER BY updated DESC

3 -> SELECT cursor FROM gql_notifications
WHERE (recipient_id=?) ORDER BY updated DESC LIMIT 1

Cluster 4:

4 -> SELECT timestamp, data, fetchreason FROM cache WHERE cachekey= ?

21 -> SELECT value, timestamp FROM cache
WHERE (name='MFacewebVersion:MRootVersion') ORDER BY name DESC
```

4.3 Pattern matching

We will be presenting results from the pattern matching operations in the following week.

5. CONCLUSION

The focus of this paper is to identify common behaviors and interesting patterns in user activities on mobile databases with the hypothesis that making use of this information can open up a lot of opportunities for mobile apps. To achieve this, we analyze PocketData dataset which consist of SQL queries posed by different applications for 11 users for a month. We utilize different query similarity methods to identify important features out of these queries, form feature vectors out of them, and cluster the similar queries together by their feature vectors with hierarchical and k-means clustering methods. Finally, we discuss on how we can make use of this clusters; we appoint an integer label to each cluster, and whenever there is an incoming query from a user, we identify which cluster the new query belongs to. This labels sequentially create a stream of integers for that specific user, in which we explore interesting and repeating patterns.

6. FUTURE WORK

This paper represents the first steps for developing a *small data* benchmarking tool for apps running on mobile devices. We plan several extensions as future work.

First, our efforts, until now, focused on how users access data instead of their complete utilization of database system capabilities: inserts, updates, and deletions along with select statements. This will require us to modify upon the current query comparison methods since their specifications do not support them. We will continue to expand the scope of our analysis through understanding more statement types and their effects on the query load.

Second, extracting meaningful patterns from user data is central to a reliable characterization of the smartphone database workload. This calls for more robust similarity measures which take into the account the considerations and constraints that are imposed by the problem domain.

Finally, we will validate our clustering results by selecting sample query loads and manually categorizing them to compare them with the automated clustering result. This step will be the measure of the sanity of our system since the following steps are exploratory analysis provided by this step.

ACKNOWLEDGMENTS

We thank Dr. Oliver Kennedy and Dr. Lukasz Ziarek for their support and guidance for our project. We also would like to thank the CSE662 students for their helpful comments.

REFERENCES

[1] Wei, J., [*Android database programming*], Packt Publishing Ltd (2012).

[2] Allen, G. and Owens, M., [*The definitive guide to SQLite*], Springer (2010).

[3] Kennedy, O., Ajay, J., Challen, G., and Ziarek, L., "Pocket data: The need for tpc-mobile," in [*Technology Conference on Performance Evaluation and Benchmarking*], 8–25, Springer (2015).

[4] Transaction Processing Performance Council, "TPC-H benchmark specification," *Published at http://www.tpc.org/tpch/* (2008).

[5] Transaction Processing Performance Council, "TPC-C benchmark specification, v5.11," *Published at http://www.tpc.org/tpcc/* (2010).

[6] Kim, J.-M. and Kim, J.-S., "Androbench: Benchmarking the storage performance of android-based mobile devices," in [*Frontiers in Computer Education*], 667–674, Springer (2012).

[7] Dijcks, J.-P., "White paper: Big data for the enterprise," Tech. Rep. 519135, Oracle Corporation, 500 Oracle Parkway Redwood Shores, CA 94065 U.S.A. (June 2013).

[8] Soikkeli, T., Karikoski, J., and Hammainen, H., "Diversity and end user context in smartphone usage sessions," in [*2011 Fifth International Conference on Next Generation Mobile Applications, Services and Technologies*], 7–12, IEEE (2011).

[9] Shi, J., Santos, E., and Challen, G., "Why and how to use phonelab," *GetMobile: Mobile Comp. and Comm.* **19**, 32–38 (Mar. 2016).

[10] Kul, G., Luong, D., Xie, T., Coonan, P., Chandola, V., Kennedy, O., and Upadhyaya, S., "Ettu: Analyzing query intents in corporate databases," in [*Proceedings of the 25th International Conference Companion on World Wide Web*], 463–466, International World Wide Web Conferences Steering Committee (2016).

[11] Kul, G., Luong, D., Xie, T., Chandola, V., Kennedy, O., and Upadhyaya, S., "Towards effective log summarization," *Published at http://odin.cse.buffalo.edu/papers/2017/EDBT-SummarizingSQL-submitted.pdf* (2016).

[12] Aouiche, K., Jouve, P.-E., and Darmont, J., "Clustering-based materialized view selection in data warehouses," in [*ADBIS*], (2006).

[13] Aligon, J., Golfarelli, M., Marcel, P., Rizzi, S., and Turricchia, E., "Similarity measures for OLAP sessions," *Knowledge and information systems* (2014).

[14] Makiyama, V. H., Raddick, M. J., and Santos, R. D., "Text mining applied to SQL queries: A case study for the SDSS SkyServer," in [*SIMBig*], (2015).

[15] Xu, R., Wunsch, D., et al., "Survey of clustering algorithms," *Neural Networks, IEEE Transactions on* **16**(3), 645–678 (2005).

[16] Hirschberg, D. S., "Algorithms for the longest common subsequence problem," *Journal of the ACM (JACM)* **24**(4), 664–675 (1977).