PI: Oliver Kennedy, Co-PIs: Geoffrey Challen and Lukasz Ziarek (Univ. of Buffalo, Dept. of Comp. Sci. and Eng.)
Type: CI-P; CISE Core Division: IIS; Keywords: databases, smartphones, benchmarking

1 — Infrastructure Description

The world's 2 billion smartphones and 4 million apps have become a large part of most people's computing experiences. Most apps need to persist structured data, a task frequently performed using an *embedded database* such as SQLite. These are heavily used, with Android smartphones generating an average of more than two SQLite queries per second [48] per phone. Unsurprisingly, mobile app usage of embedded databases is quite different from the workloads experienced by database servers supporting websites or big data applications. For example, while database servers are frequently tested and tuned for continuous high-throughput query processing, embedded databases experience lower-volume but bursty workloads produced by interactive use. As another example, enterprise database servers are frequently provisioned to have exclusive access to an entire machine, while apps using embedded databases compete for shared system resources with other apps and may be affected by system-wide policies that attempt to conserve limited energy on battery-constrained mobile devices. So while the fundamental challenges experienced by mobile apps using embedded databases — minimizing energy consumption, latency, and disk utilization — are familiar ground for database researchers, the specific tradeoffs produced in this domain of *pocket-scale data* are far less well understood.

To date, there have been some initial explorations of small-, personal-, or pocket-scale data management, both in academia and by industry:

- Arnab Nandi's group at Ohio State is exploring issues of query specification, database responsiveness, and database performance on smartphones and tablets.
- Saarland University's Janiform Document project explores interactive manuscripts that include embedded, query-able research data and visualizations.
- Oracle, SAP Labs, Facebook, LMDB, SQLite, and MongoDB are all actively engaged in research on and development of embedded database software.
- The Presto group at Ohio State is exploring responsiveness issues in Android caused by data-flow limitations.
- The DAS Lab at Harvard's work on adaptive data management considers the challenges of specializing databases for small data.
- The POCKETDATA project at the University at Buffalo is exploring how smartphone apps interact with embedded-databases.

There is clearly interest in data management challenges that arise in small-scale data management. Unfortunately, unlike the largely homogeneous workloads and platforms that are standard in research on classical enterprise databases, this new POCKETDATA setting is far more diverse. Data access patterns are extremely bursty and can vary wildly by user, time of day, mix of installed apps, network accessibility, and many other factors. Platform properties such as RAM, persistent storage, CPU performance, and network bandwidth also exhibit extreme variations across phones, sometimes by multiple orders of magnitude. Resource availability can also vary; Some users keep their phones constantly charged, while others go multiple days without plugging their phones in.

The heterogeneity of the POCKETDATA setting makes it challenging for researchers to understand the tradeoffs and requirements of the setting. This lack of clear high-level goals, in turn, makes it difficult to clearly identify successful research contributions. Unfortunately, pinning down specific goals first requires a concerted effort to gather (and analyze) traces of data usage patterns from real-world settings, creating a high barrier to entry for new researchers. Lacking the resources necessary to better understand and adapt to the POCKETDATA scale, research efforts in the area are presently limited.

Target Community Research on mobile devices and the more general space of the internet of things (IoT) is cross cutting, intersecting communities that work on data management systems, real-time and embed-

ded devices, programming languages, and operating and mobile systems. We believe research involving POCKETDATA also lies at the intersection of these communities. Specialized database systems for embedded devices are re-emerging as an interesting topic in the database community. As embedded processors become more capable, with larger amounts of main memory available (e.g. Intel's Edison platform), there is a growing push from the embedded systems and the real-time communities to explore larger software capabilities, including database systems and query processing systems in embedded hardware deployments. The programming language community is exploring domain specific languages for specialized query processing. The mobile community is continually exploring how to push the envelope on smartphone based computing, whether via power aware mechanisms, or through more adaptive systems. Many of these solutions use mobile databases as their fundamental computation engine (e.g. the '`maybe`' system developed at UB) or must consider the performance characteristics of database systems (e.g. power modeling).

This proposal aims to create a community research infrastructure around our POCKETDATA toolchain to enable a myriad of research activities for the above mentioned communities. Additionally, in this planning grant, we will explore the precise needs of these communities to ensure an infrastructure that has broad applicability. We will reach out to researchers in closely related areas including Internet of Things, Adaptive Data Management, Sensor Networks, and help them to explore how POCKETDATA can help to improve their research. As part of these outreach efforts, we will provide resources that will simultaneously support researcher's existing projects, while also helping to enable new projects with a focus on POCKETDATA.

During this planning grant we will focus our efforts in three key areas:

1. **Growth of the Mobile Embedded Database community**: We have established an initial community of interested CISE researchers for POCKETDATA from both academia and industry. We believe that this community shows that there is sufficient interest within CISE to pursue our proposed POCKETDATA infrastructure. However, for long term success we would like to expand this community to ensure that the infrastructure meets the needs of the broader community and not just a specific research niche.
2. **Expansion to IoT**: Our preliminary efforts have focused on questions relating to POCKETDATA in the mobile domain, specifically Android. Although a POCKETDATA infrastructure based solely in Android is valuable, we believe a more comprehensive infrastructure must take into account recent developments in IoT. There are similarities between how mobile applications leverage embedded databases and how proposed IoT applications would use embedded databases, specifically in the areas of personal health care devices that aggregate and summarize a user's personal data and smart city deployments where small devices process data before sending *relevant* data for more centralized big data analytics. We propose to expand and modify our POCKETDATA infrastructure to meet the needs of IoT community.
3. **Workshops and Tutorials**: To facilitate engagement with the broader CISE community and to develop an initial community for POCKETDATA, we proposed to run workshops and tutorials co-located with major conferences in the database, systems, real-time systems, and programming language communities. Our budget includes funding for travel to such conferences to host workshops and tutorials. This will also enable the PIs to receive valuable feedback on the needs of the community in designing and building out the POCKETDATA infrastructure.

A successful planning grant will enable us to proceed with the development of a full POCKETDATA infrastructure proposal. Concretely, the following three resources will be developed as part of the full infrastructure proposal:

1. **Datasets and Traces**: We will provide the POCKETDATA community with the data and statistics necessary to understand and emulate embedded database usage on smartphones and similar devices. Our preliminary study [48] tracked usage patterns for SQLite on 11 smartphones "in-the-wild" for a period of 1 month. The study clearly illustrates the uniqueness of these usage patterns and we have already made our source data and the results of our analysis freely available. We presently have further data on approximately 170 phones for a period of several months. If funded, we would be able to provide researchers with larger, more extensive traces of smartphone embedded database usage, as well as more comprehensive analyses summarizing those datasets.
2. **Standards and Benchmarks**: We will create a toolkit to establish a set of standards for evaluating research efforts on POCKETDATA for both Android and IoT. The POCKETDATA setting requires unique

metrics that can be difficult to reliably measure on the Android platform. The toolkit will include instrumentation for Android that will make it easier for researchers to measure performance through rarely used metrics like availability of idle time, thread scheduling, power consumption, and other measures that can be hard to gather reliably on the Android platform like CPU and memory usage for specific libraries. Second, to standardize comparisons across different research efforts, the toolkit will include a benchmark suite. This benchmark will established clearly defined metrics for evaluating data management solutions. Moreover, by making it extensible, the benchmark will act a clearinghouse for app behaviors discovered in the wild and changing database requirements.

3. **Visualization**: We will create a data visualization tool and associated queries to help researchers understand and navigate the data. The raw traces we plan to offer researchers are very large. Moreover, the rich structure and variability of SQL queries generated by smartphone apps does not admit traditional indexing strategies often used for analytics. By providing database-driven tools that aid in the analysis and visualization of the resulting queries, we will enable researchers to more quickly and accurately explore relevant characteristics of real-world POCKETDATA workloads.

Qualifications of the PIs: The PIs bring cross-cutting expertise from three of the main communities our proposed POCKETDATA infrastructure would impact. All three have a record of successful collaboration [48, 14], and PIs Kennedy and Ziarek have been working together for the last three and a half years on adaptive indexing [52, 3, 53]. PI Kennedy's expertise covers databases, incremental computation [5, 47, 58], uncertain data management [51, 45, 94], online aggregation [46, 50], and compiler design [47, 5, 58]. PI Ziarek's expertise covers programming languages [96, 95], real-time systems [9, 91], virtual machines [75, 76], and compiler design [84, 97]. PI Challen's expertise comprises smartphone systems, including networking [81, 82, 80], architectural [8], energy management [62, 61] and security [25] aspects.

1.1 — Datasets

In a preliminary study [48], we instrumented Android smartphones being used as the primary device of 11 UB students, faculty and staff for a period of one month. The SQLite embedded database included as part of the Android platform was modified to log a trace of all SQL statements executed, along with metadata such as the number of rows returned, time taken, and the application process that issued the statement. To protect participant privacy, our instrumentation removed as much personally-identifying information as possible and recorded prepared statement arguments only as hash values. With participant permission, we have made these traces publicly available [48].

We conducted a preliminary analysis to summarize these traces, the key parts of which we summarize here to provide a sense of the type of information that we will make available to the POCKETDATA community. We captured approximately 45 million statements executed by SQLite over the 1 month period. As might be expected, `SELECT` forms almost three quarters of the workload by volume. `UPSERT` statements (*i.e.*, `INSERT OR REPLACE`) form a similarly substantial 16% of the workload — more than simple `INSERT` and `UPDATE` statements combined. Also of note is a surprising level of complexity in `DELETE` statements, many of which rely on nested sub-queries when determining which records to delete.

Of the 45 million queries analyzed, 33.47 million were read-only `SELECT` queries. Figure 1 shows the distribution of `SELECT` queries by number of tables accessed by the query, as well as the maximum level of query nesting. Nesting includes from-nesting (*e.g.*, `SELECT ... FROM (SELECT ...)`), as well as expression-nesting (*e.g.*, `SELECT ... WHERE EXISTS (SELECT ...)`). Even at this coarse-grained view of query complexity, the read-only portion of the embedded workload distinguishes itself from existing TPC benchmarks. Like TPC-C [22], the vast majority of the workload involves simple, small requests for data that touch a small number of tables. 29.15 million, or about 87% of the `SELECT` queries were simple select-project-join queries. Of those, 28.72 million or about 86% of all queries were simple single-table scans or look-ups. In these queries, which form the bulk of SQLite's read workload, the query engine exists simply to provide an iterator over the relationally structured data it is being used to store. Conversely, the workload also has a tail that consists of complex, TPC-H-like [24] queries. Several hundred thousand queries involve at least 2 levels of nesting, and over a hundred thousand queries access 5 or more tables. As an extreme example, our trace includes 10 similar `SELECT` queries issued by the Google Play Games Service, each of which accesses up to 8 distinct tables to combine and summarize developer-provided game state, user preferences, device profile meta-data, and historical game-play results from the user.

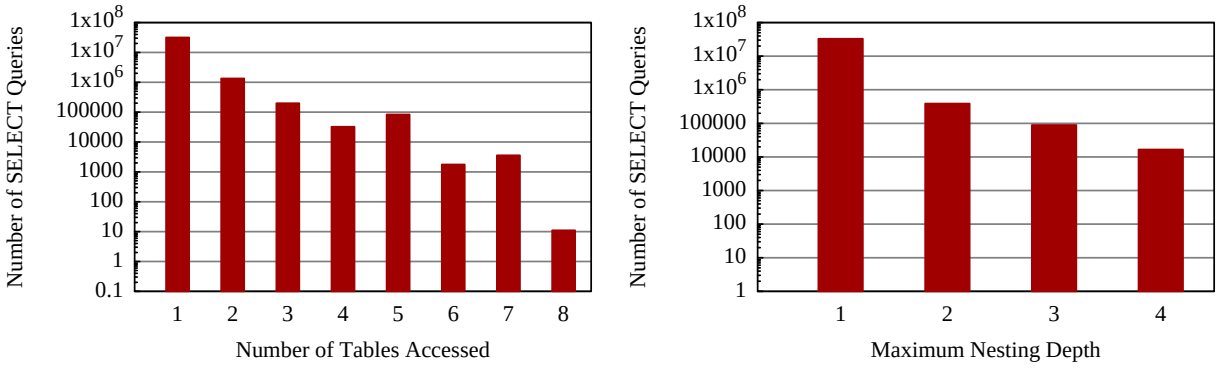


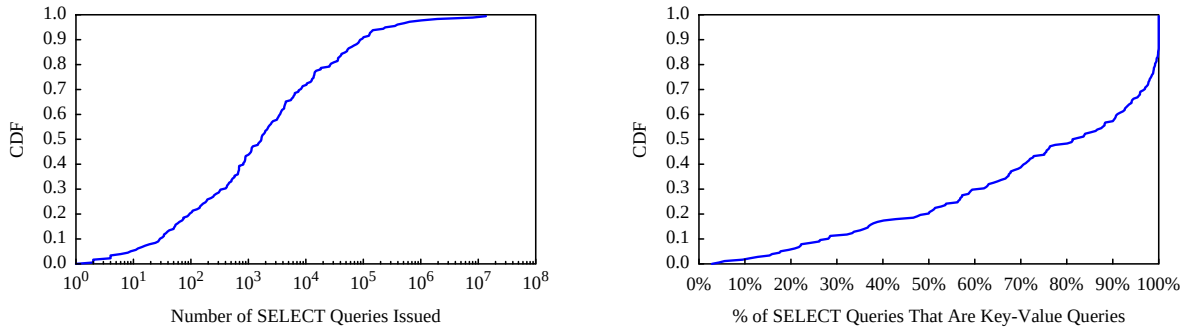
Figure 1: `SELECT` **queries by (a) number of tables accessed and (b) maximum nesting depth.**

	Join Width					
Where Clauses	1	2	3	4	6	**Total**
0	1,085,154					**1,085,154**
1	26,932,632	9,105				**26,941,737**
2	1,806,843	279,811	5,970			**2,092,624**
3	384,406	80,183	29,101	1		**493,691**
4	115,107	70,891	10,696	939		**197,633**
5	28,347	15,061	1,162	17	11	**44,598**
6	212	524	591	471	3	**1,801**
7	349	22,574	333	1,048	8	**24,312**
8	35	18			6	**59**
9		541	2,564	4		**3,109**
10	159					**159**
11	545					**545**
Total	**30,353,789**	**478,708**	**50,417**	**2,480**	**28**	**30,885,422**

Figure 2: **Number of simple look-up queries subdivided by join width (number of tables) and number of conjunctive terms in the** `WHERE` **clause.**

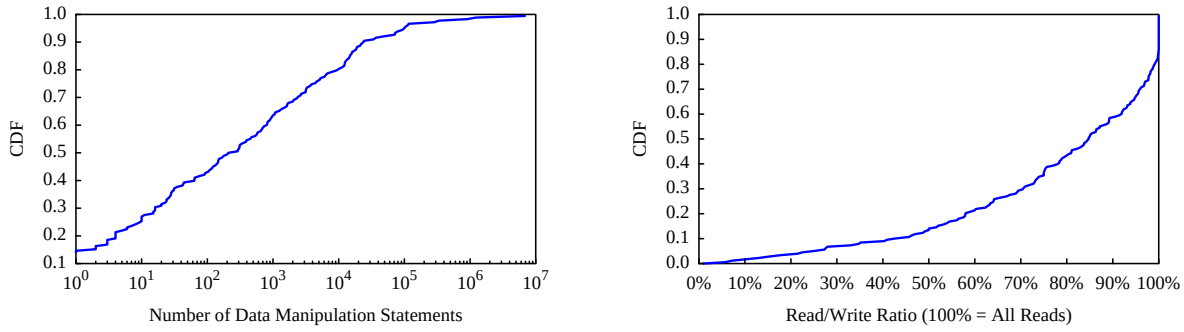
The majority of `SELECT` queries we encountered fall into a class of *simple look-up* queries, defined as any `SELECT` query that consists exclusively of selections, projections, joins, limit, and order by clauses, and which does not contain any nested sub-queries or unions. Figure 2 shows queries of this class, broken down by the number of tables involved in the query (Join Width) and the complexity of the where clause, as measured in number of conjunctive terms (Where Clauses). For example, consider a query of the form: `SELECT R.A FROM R, S WHERE R.B = S.B AND S.C = 10` This query would have a join width of 2 (`R`, `S`) and 2 conjunctive terms (`R.B = S.B` and `S.C = 10`). The first column of this table indicates queries to a single relation. Just over 1 million queries were full table scans (0 where clauses), and just under 27 million queries involved only a single conjunctive term. This latter class constitutes the bulk of the simple query workload, at just over 87% of the simple look-up queries. Single-clause queries appear to be the norm.

Over the course of the one-month trace we observed 179 distinct apps, varying from built-in Android applications such as *Gmail* or *YouTube*, to video players such as *VLC*, to games such as *3 Kingdoms*. Figure 3a shows the cumulative distribution of apps sorted by the number of queries that the app performs. The results are extremely skewed, with the top 10% of apps each posing more than 100 thousand queries over the one month trace. The most query-intensive system service, *Media Storage* was responsible for 13.57 million queries or just shy of 40 queries per minute per phone. The most query-intensive user-facing app was *Google+*, which performed 1.94 million queries over the course of the month or 5 queries per minute. At the other end of the spectrum, the bottom 10% of apps posed as few as 30 queries over the entire month.



(a) (b)

Figure 3: **Breakdown of `SELECT` queries by app. (a) Cumulative distribution of applications by the number of `SELECT` queries issued (note the logarithmic scale). (b) Cumulative distribution of applications by the percent of the app's `SELECT` queries that are key value queries (full table scans or exact key look-ups).**



(a) (b)

Figure 4: **App-level write behavior. (a) Cumulative distribution of applications by number of data manipulation statements performed (note the logarithmic scale). (b) Cumulative distribution of applications by read/write ratio.**

We noted above that a large proportion of `SELECT` queries were exact look-ups; Indeed many applications running on the device are using SQLite as a simple key-value store. As seen in Figure 3b, for 24 apps (13.4%), we observed *only* queries that would have been supported by a trivial key-value API for the full span of the month-long trace.

Figure 4a illustrates app-level write workloads, sorting applications by the number of `INSERT`, `UPSERT`, `UPDATE`, and `DELETE` operations that could be attributed to each. The CDF is almost perfectly exponential, suggesting that the number of write statements performed by any given app follows a long-tailed distribution, a feature to be considered in the design of a pocket data benchmark.

Figure 4b breaks apps down by their read/write ratio. Surprisingly, 25 apps (14% of the apps seen) did not perform a single write over the course of the entire trace. Manual examination of these apps suggested two possible explanations. Several apps have reason to store state that is updated only infrequently. For example, *JuiceSSH* or *Key Chain* appear to use SQLite as a credential store. A second, far more interesting class of apps includes apps like *Google Play Newsstand*, *Eventbrite*, *Wifi Analyzer*, and *TuneIn Radio Pro*, all of which have components that query data stored in the cloud. We suspect that the cloud data is being encapsulated into a pre-constructed SQLite database and being pushed to, or downloaded by the client applications. This type of behavior might be compared to a bulk ETL process or log shipment in a server-class database workload, except that here, the database has already been constructed. Pre-caching through database encapsulation is a unique feature of embedded databases, and one that is already being used in a substantial number of apps.

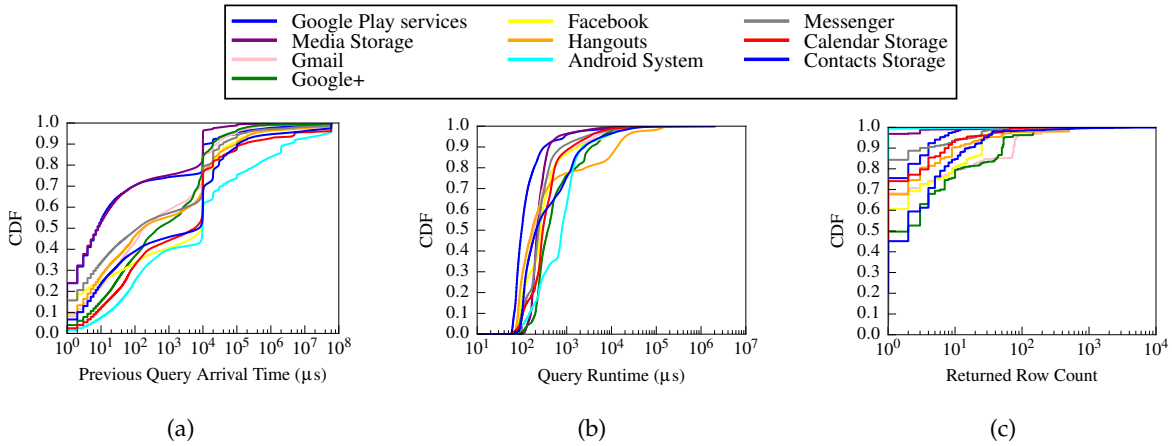


Figure 5: **Per-App Summary Statistics for Android SQLite Queries. Distributions of (a) inter-query arrival times, (b) query runtimes, and (c) rows returned per query.**

Figure 5 shows query interarrival times, runtimes, and returned row counts for ten of the most active SQLite clients. As seen in Figure 5a, a 0.01Hz periodicity in arrival times is common to all applications, suggesting filesystem locking as a culprit. Two of the most prolific SQLite clients, *Google Play services* and *Media Storage* appear to be very bursty: 70% of all statements for these applications are issued within 0.1ms of the previous statement. Also interesting is the curve for queries issued by the *Android System* itself. The interarrival time CDF appears to be almost precisely logarithmic for rates above 10μs, but has a notable lack of interarrival times in the 1ms to 10ms range. This could suggest caching effects, with the cache expiring after 1ms.

As seen in Figure 5b, most apps hold to the average runtime of 100μs, with several notable exceptions. Over 50% of the *Android System*'s statements take on the order of 1ms. Just under 20% of *Hangouts* statements take 10ms, suggesting an update-heavy workload. Also, *Contacts Storage* has a heavier-duty workload, with 30% of statements taking between 100μs and 1ms. Figure 5c shows that the *Android System* and *Media Storage* issue almost exclusively single-row lookup queries. The remaining apps issue a large number of single-row queries — Even *Contacts Storage* has a workload consisting of 45% single-row reads — the number of rows returned in general varies much more widely. Many of these apps' user interfaces have both a list and a search view that show multiple records at a time, suggesting that these views are backed directly by SQLite. Although all apps have long tails, two apps in particular: *Gmail* and *Google+* are notable for regularly issuing queries that return on the order of hundreds of rows.

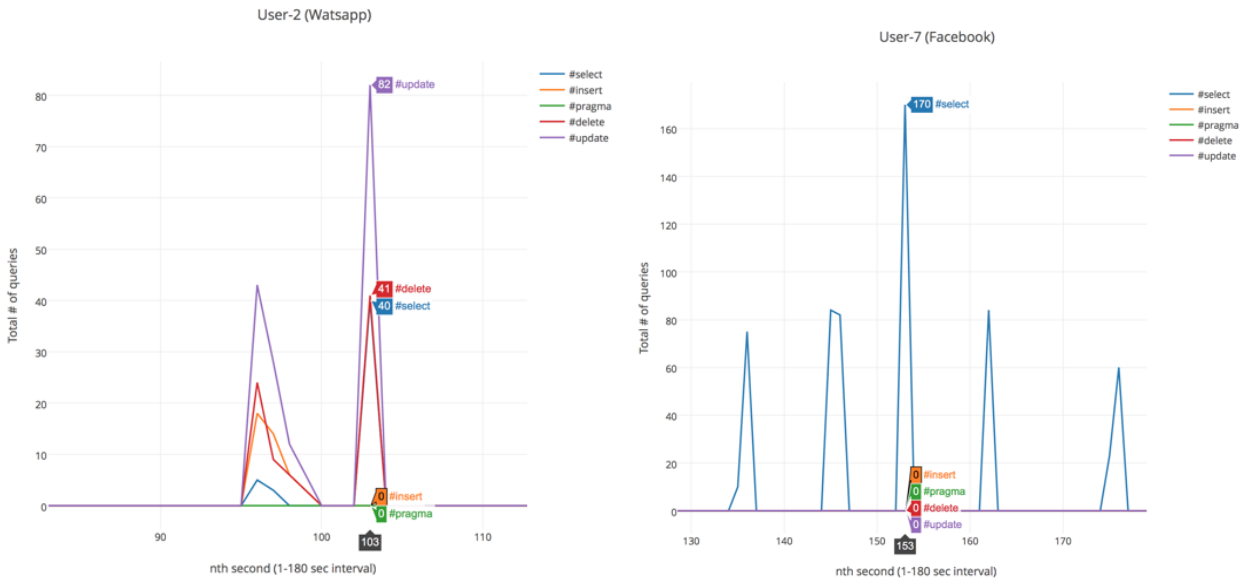
Figure 6 shows variations in query burstiness across multiple apps and users[1]. Two features immediately emerge from this data. First, POCKETDATA workloads are extremely bursty; The default steady state is completely idle, with infrequent bursts of hundreds of operations per second. Second, the nature of these bursts varies significantly by the calling app; In this trace Facebook generates a read-only workload, while Whatsapp produces two bursts each with a distinct mix of updates, inserts, deletes, and selects.

We will freely release aggregate metrics about database usage patterns in embedded smartphone databases. We also plan to make our source traces available to researchers with approval from their institution's IRB. By doing this, we will enable other researchers to begin exploring the bottlenecks in and practical limitations of existing embedded databases, as well as in abstraction layers like object-relational mappers. Better understanding the space will help to identify new research challenges, and help to encourage researchers to join the POCKETDATA community.

1.2 — Instrumentation

We will provide an instrumentation toolkit for the POCKETDATA community. The goal of this toolkit is twofold: (1) Gathering usage traces and metrics from phones deployed in real-world settings, and (2)

[1]The PIs have already incorporated material from this proposal into their coursework. Figure 6 is from a student report [77] from UB's CSE-662, jointly instructed by PIs Kennedy and Ziarek. The student group performed an app-centric analysis of the query traces.



(a) (b)

Figure 6: **Variations in bursty data access patterns [77] for WhatsApp (a) and Facebook (b).**

Reliably measuring system performance on simulated and replayed PocketData workloads. There are several challenges unique to the PocketData setting that make instrumenting smartphone embedded databases difficult. The simplest of these is that smartphones rely on specialized operating systems, hardware, and virtualization that can make it difficult to deploy existing measurement tools designed for desktops. Many of these tools can be ported and we will endeavor to supplement existing community efforts in doing so. There are also several more subtle challenges specific to instrumenting PocketData.

A key challenge is the types of bottlenecks that PocketData workloads encounter. Typical metrics for enterprise benchmarks include throughput at saturation, joules per unit of throughput, and throughput vs latency curves. These metrics makes sense for evaluating systems where large numbers of users make scalability and average and peak throughputs the dominant concern. However, PocketData workloads are generated by individual users and apps acting on their behalf. In this setting, scalability and throughput are not as important as latency, power consumption, and memory use. This subtle distinction makes reliable benchmarking far more difficult; It is not reasonable to throw an infinite load at the system and simply measure how fast the load is processed.

A naive strategy might be to issue a normal, bursty, steady state load similar to the loads we observed in practice [48, 77] and measure user-facing properties like latency to first response and time to completion. However, this approach fails to consider important factors not directly visible to the user, including CPU use, memory requirements, and overall power consumed. Measuring several of these properties involves instrumenting components of the OS. For example, when a CPU is spending time idling, reliably attributing CPU cycles to an Android app requires a specially instrumented scheduler.

A further concern that makes PocketData instrumentation difficult is that embedded databases are typically stored in self-contained files. Replicating an embedded database can be as simple as initiating a file transfer over HTTP. We observed many apps exploiting this feature in practice [48]. File transfers bypass the normal embedded database library, so fully capturing an app's interactions with an embedded database requires jointly instrumenting other aspects of the OS, including the filesystem and network layers.

As part of the proposed work, we will develop an instrumentation toolkit that can reliably track an app's embedded database activities and measure all facets of an embedded database's performance.

1.3 — Benchmark

The second component of our research toolkit is a modular benchmarking suite, which will serve three roles for the POCKETDATA community. First, a benchmark will foster research on embedded databases by creating a realistic standard for evaluation, allowing for fair comparisons across competing research efforts. Second, by providing a precise set of metrics to optimize for, a benchmark will serve to guide the research community's efforts towards pertinent real-world challenges faced by smartphone applications.

Finally, smartphones and smartphone applications are still in their infancy, and their data management needs are evolving very quickly. To remain useful, the benchmark will require a concerted research effort to track changes in app usage behaviors and bottlenecks. We will develop a modular benchmark along the lines of PolePosition [1], driven by modules that capture the semantics and behavior of a class of applications. Based on the metrics gathering efforts discussed above, we will lead an effort to continually monitor app's data usage patterns for changes, as well as for changes in how phone users engage with data-driven apps. As new patterns are discovered by the POCKETDATA community, we will maintain *a repository of modules describing these behaviors*.

Our initial benchmark design is organized into two tiers: Application and User. The Application tier will consist of a set of app *modules*; Each module will simulate the query behavior of one smartphone app (or class of apps), mirroring workloads that we see in our query traces. Our goal will be to understand the types of data that the app stores (based on inspection of the app's schema and query patterns), as well as the app's approach to accessing and modifying its data. Ideally, we will be able to link individual queries to triggering events (user interactions, timers firing, display updates, etc...), allowing us to model the app's query patterns in terms of the distribution of these events. Although we hope to automate this process eventually, our initial approach will be to focus on one app at a time. This will not only help us to better understand the space, but also to generate realistic datasets by being able to analyze the specific app's schema and updates/inserts. The Application tier of the full benchmark will consist of a representative cover of the 179 apps that we encountered in our preliminary analysis, as well as apps that we encounter in subsequent data gathering efforts. The User tier will simulate the complete phone environment. Statistics for single user include of a cluster of app modules, patterns of charging behavior (when is the phone plugged in?), network access (when is the phone on the internet, and with what quality?), and other behavioral traits that impact app data access patterns. To simulate users, we will use standard clustering techniques on our trace data first to create canonical user profiles, and then to identify natural variation around those profiles.

It is reasonable to ask why a specialized POCKETDATA database benchmark is necessary. To answer this question, we surveyed a variety of existing benchmarks. Benchmarks such as AndroBench [54] and MobiGen [6] target mobile devices and mobile software. Although AndroBench does include a component for simulating the filesystem access patterns of SQLite, neither of these benchmarks explicitly generates the structured data access patterns necessary to evaluate a complete data management system. Previous research efforts [41] have used mobility traces generated by MobiGen, fed to a virtual machine running common apps such as Facebook to generate semi-realistic traces of embedded database access patterns. Although POCKETDATA follows a more principled approach based on real-world traces, the metrics we release could be used to validate and standardize data generation tricks of this sort.

A substantial fraction of data management tasks in smartphone embedded databases involve data manipulations. Such tasks are often well modeled by Online Transaction Processing (OLAP) query benchmarks such as TPC-C [22] or YCSB [21]. However, OLAP benchmarks fall short in two ways. First, OLAP benchmark query patterns are typically designed to stress concurrency bottlenecks in the systems being tested. Conversely, mobile SQLite databases are isolated into specialized app-specific silos. The most intensive database user in our preliminary study, *Google Play services* had 14.8 million statements attributed to it, just under half of which were writes. This equates to about one write every 3 seconds, which is substantial from a power management and latency perspective, but which is unlikely to create a concurrency bottleneck. Second, many OLAP benchmarks focus on comparatively simple queries. A notable portion of the workload we observed in our preliminary study can indeed be described as simple: 13% of the applications we observed had a read workload that consisted exclusively of key/value queries, and over half of the applications we observed had a workload that consisted of at least 80% key/value queries.

However, the trace also exhibited a long tail of extremely complex queries. A small, but significant number of queries we observed in our preliminary study include multiple levels of query nesting, wide joins, and extensive use of aggregation. As such, they more closely resemble analytics (OLAP) workload benchmarks such as TPC-H [24], The Star-Schema Benchmark [73], and TPC-DS [23]. This mix of OLAP and OLTP workloads is unique to POCKETDATA applications, which are individually simple, but can have a huge impact when taken in aggregate across all apps on a phone. We do however note that the presence of analytics queries in pocket data management is likely to increase further, as interest grows in smartphones as a platform for personal sensing [12, 55, 59] and analytics [85, 67, 83]

The most conceptually similar benchmark is the PolePosition [1] embedded database benchmark. PolePosition simulates the behavior of specific data structure abstractions that need to be backed by a data management system. Because data structures are defined using higher-level operational semantics rather than through a fixed database API, databases are allowed to specialize benchmarks to specific access patterns that the database may be optimized for. The fundamental goals of PolePosition and the POCKETDATA benchmark are similar, but POCKETDATA will operate at a higher level of abstraction, capturing the behavior of entire apps, as well as users that engage with those apps.

1.4 — Visualization

Even the short, month-long query trace with only 11 users on which our preliminary study was based included over 45 million SQL statements. As the experiment is scaled up, analyzing these query traces will become increasingly difficult. Compounding the issue, the comparatively high complexity of many of the the queries makes it difficult to flatten the SQL parse trees into a simple relational format for analysis. Our preliminary analysis required repeated iterations of our feature extraction process: We would define a procedure for extracting interesting features of a SQL statement's parse tree, construct a visualization from the extracted feature, and then identify a new feature of interest.

As part of the proposed work, we will release tools for analyzing query logs that streamline this iterative process, by making it easy define new feature extractors. As feature extraction is an embarrassingly parallel task, simple optimizations like caching, parallelism, and incremental computation [47] can be used to make these tools extremely efficient[2]. Source code for all visualizations that we release as part of our summary metrics will be released to the public to further encourage community participation in POCKETDATA.

1.5 — Workshop

We will build an initial POCKETDATA community and facilitate engagement with the broader CISE community through outreach efforts including attending poster and demo sessions and hosting workshops and tutorials co-located with major conferences in databases (VLDB, SIGMOD, ICDE), mobile and real-time systems (MobiSys, OSDI, RTSS, RTAS), and programming languages (POPL, PLDI, OOPSLA). Poster sessions provide an ideal opportunity to meet researchers in related areas, to advertise the resources we plan to offer, and gather feedback about the needs of potential POCKETDATA community members.

Tutorials offer a more formal, extended opportunity to introduce members of broader communities in CISE to technologies related to POCKETDATA, and to train them in the technology's use. As a side effect, a tutorial offers an opportunity to advertise the proposed resources as tools for conducting research in these areas. Potential tutorial subjects pertinent to POCKETDATA include embedded databases, data management on smartphones, object-relational mappers, and smartphone development.

Our final outreach effort will be a POCKETDATA workshop. This will offer a forum for researchers exploring opportunities in small-scale data management to come together and offer each other feedback. Simultaneously, a workshop creates an incentive for researchers in adjacent areas like IoT, mobile systems, and adaptive data management to carefully venture into the POCKETDATA space. In the interest of having broad appeal, our proposed POCKETDATA workshop will target several related spaces, including small-data management [27], gestural query interfaces [67], adaptive databases [37], and embedded databases [10], among others.

We have budgeted for each project participant to attend at least one conference over the one-year planning

[2] As a comment on the utility of specialized tools for log analysis, we return to the CSE-662 project involving analyzing POCKETDATA logs. The four students began with a naive analysis tool (written by the students in Java) that took multiple hours to complete one iteration of the analytics cycle. By the end of the course, they had optimized the tool to run in under 10 seconds [77].

period, allowing for the project to connect with multiple communities. If funded, we will continue these efforts for the span of the infrastructure grant.

2 — New Research Opportunities

In this section we present a few concrete projects that would benefit from POCKETDATA and describe how the proposed infrastructure will enable research for the PIs and the broader CISE community.

2.1 — Adaptive Data Management

Selecting the correct physical structure for a database under a given workload is an extremely challenging [18, 16, 17, 4] part of database management. The index selection problem becomes even harder when workload characteristics fluctuate rapidly or are not known in advance. There is currently substantial interest in a breed of self-adjusting, adaptive index structures [37, 40] that address dynamic index selection by facilitating *incremental, online* changes to the index. Examples of adaptive indexes include Cracker Indexes [39, 38, 35], Adaptive Merge Trees [34, 32], SMIX [88], H2O [7], and Just-in-Time Data Structures [53]. Adaptive indexes automatically optimize their physical representation in response to incoming queries, reusing work used to answer the query to also improve subsequent queries. Given enough time, a stable workload, and queries that touch all data objects, an adaptive index eventually converges to a data representation similar to that of a static index.

Infrastructure Justification: Although there have been several efforts [33, 79] to develop benchmarks for adaptive indexes, these benchmarks rely on purely synthetic data and unit-tests rather than real-world scenarios. This is in part because typical enterprise workloads rarely exhibit the type of drastic shifts that adaptive indexes target. As a result most data management benchmarks evaluate systems under stable, steady-state workloads. By contrast, POCKETDATA workloads often show extreme variation in both application demands and resource availability. As a trivial example, an app might demand low-latency, low-power access to data when a user is actively using the phone, while admitting high-latency high-power organizational tasks when the phone is plugged in [14].

Community Interest: *Stratos Idreos*'s DAS lab at Harvard will use the POCKETDATA metrics and benchmark workloads to evaluate their work on adaptive data systems.

> *I think work on adaptive data systems could benefit. I assume Pocket Data will capture diverse workloads (from various apps) and so this would be a perfect environment to test adaptive data systems. I have a new project on easy to design systems out of modules that can be synthesized. The input is workloads. Perhaps PocketData can provide a testing framework for such work for designing data systems for mobile environments.*
> -Stratos Idreos (Harvard)

2.2 — Small-Data Analytics and Personal Internet of Things

The prevalence of tablet and smartphone computing devices makes them an ideal analytics front-end. Apps such as Zillow, Google Earth, and MapMyRun provide specialized front-ends for data exploration. Apple's iTunes Store has an "Apps for Healthcare Professionals" section with dozens of apps for visualizing and exploring patient statistics. These apps are part of a growing number of small-data analytics and personalized IoT applications that present new and intriguing opportunities for research. For example, smartphone and tablet touch-based interfaces require a significant redesign of the way users pose queries [66, 67, 42, 30]. Embedded databases create opportunities for more detailed, interactive academic manuscripts [27, 28] that help to ensure reproducible results. The relatively limited compute and memory resources available on tablets and smartphones also demand new techniques for rapidly building visualizations of medium sized databases [43, 83, 86, 72].

Infrastructure Justification: Small-data analytics efforts are presently siloed, with most research efforts targeting entire software stacks, from the user interface front-end to the back-end database. The standard evaluation tools offered by the POCKETDATA benchmark would help to decouple the research challenges involved in small-data analytics and allow a broader community of researchers to contribute. For example, an embedded database benchmark simulating a visual query interface workload would serve as a standard for evaluating novel algorithms, indexes, and data management tools.

Community Interest: *Arnab Nandi* from Ohio State has offered to contribute traces of human interactions

with his tools for gestural query specification to the POCKETDATA effort. *Jens Dittrich* of Saarland University is interested in connections between PocketData and his work on Janiform Documents [28].

2.3 — Data-Driven Apps

Virtually all access to embedded databases on smartphones occurs through SQL statements that have been procedurally generated by apps — Smartphone users do not manually write SQL queries. In this respect, data-driven smartphone apps are similar to data-driven enterprise applications. However, enterprise software is typically supported by experienced database administrators who can carefully fine-tune the database to efficiently support the application. This is not the case for smartphone apps, which instead rely on compiler tools and software libraries to efficiently mediate access to persistent data. Consequently, POCKETDATA offers new research opportunities at the interface between imperative programming languages like C, C#, or Java, and back-end data management tools. Forms of inline SQL like LinQ [11, 63] have existed for nearly a decade, but are not frequently used in the development of smartphone apps. Instead, app developers frequently rely on higher level primitives including object-relational mappers [64] (ORMs) like Hibernate [87] to mediate access to the database. Unfortunately, at present, most ORMs are implemented as libraries and lack the ability to introspect the invoking program. This creates an impedance mismatch between the available information and SQL's declarative syntax, forcing ORMs to misuse SQL, or to rely on optional hints provided by the app developer to provide efficient data access. In our preliminary exploration [48], we found significant anti-patterns emerging in queries to SQLite. Examples include the use of expensive `UPSERT` operations when `UPDATE`s would be sufficient, the use of multiple `SELECT` queries to dereference foreign-keys instead of using an outer-join query, and the use of separate read-then-write queries rather than in-place updates. Several research efforts, including StatusQuo [20], Sloth [19], and Truffle/Graal [89] have addressed similar problems in enterprise data-driven applications and could find new challenges in the POCKETDATA space. Other research efforts explore data-flow in smartphones for performance optimization [92, 93, 78] and correctness [90], and would benefit from more detailed tools for introspection and measurement.

Infrastructure Justification: Research on data-driven app development requires a detailed understanding of application requirements, and programming language research needs real-world workloads to demonstrate its viability. The metrics that we propose to gather and the benchmark suite we propose to develop are critical for driving research in this space.

Community Interest: *Nasko Rountev* of Ohio State will use POCKETDATA as part his work on data-flow analysis debug of GUI responsiveness issues and as part of his LeakDroid project.

2.4 — Database-App Coupling

Smartphone apps are integrated with the data management tools they use to a far greater degree than enterprise applications. Embedded databases are libraries that operate within the app's memory space, and not external tools. Apps generate virtually all queries procedurally, making it possible to specify their data management requirements extremely precisely at compile time. Moreover, access to data often occurs through higher-level primitives like ORMs. In short, although embedded databases are in principle capable of emulating stand-alone database engines, in practice they are used more as toolkits of data management building blocks. The tight coupling between app and database promises to offer numerous avenues for workload-driven database optimization. A leader in this area is BerkeleyDB. Although BerkeleyDB does provide a SQL emulation front-end, its core functionality is to provide simple database building blocks like primary and secondary indexing, foreign-key consistency primitives, and transactional access to data. Similar efforts are taking place across multiple industrial research labs and startup companies, as numerous organizations have begun to invest into embedded databases. Corporate investment in embedded databases includes MongoDB's WiredTiger [29], SAP's SqlAnywhere [10], and Facebook's RocksDB, as well as open-source efforts including the H2 Database [65] and SQLite [74]. The tight coupling between database and the invoking application also admits possibilities for more aggressive database specialization. Database compilers like DBToaster [47, 58, 5], HyPer/LLVM [71], and Legorithmics [57, 56] aggressively compile and optimize database engines that are uniquely specialized for a specific application's query and update workload, as well as its underlying hardware. As already noted above, many of these statistics are available at compile time, making the POCKETDATA setting an ideal candidate for deploying these applications.

Infrastructure Justification: Realistic evaluation of embedded databases and database compilers requires

realistic workloads. Moreover, smartphones are one of the most prolific examples of embedded databases deployed in the wild. Given the variation in smartphone apps' data management requirements, even limited data releases by a single app developer will not be representative. The metrics we will gather, and the benchmark we are proposing will be key to helping researchers evaluate new embedded database tools.

Community Interest: *Christoph Koch*'s DATA lab at EPFL is interested in using the POCKETDATA benchmark to evaluate their work on database compilers. *Ashok Joshi* and *Michael Brey* of Oracle are interested in participating in the POCKETDATA community to advance research on embedded databases.

> *I got some feedback from one of my colleagues on this topic. Yes, the real-world traces of embedded data usage would be useful; so would the benchmarking toolkit.* -Ashok Joshi (Senior Director at Oracle)

> *Within Oracle, we are always looking at how the industry both consumer and enterprise is using data in mobile applications. Things like db size, access patterns, single/multi user (multiple apps accessing same db), speed of access required, record size/structure etc. are all important to understand. We are also very interested in the movement of data from the device to some backend repository.* -Michael Brey (Oracle)

2.5 — Enabled Research For the PIs

The PIs have a joint research project aimed at exposing *uncertainty* in mobile computing [14]. The project focuses on exposing new language primitives to the programmer to specify multiple implementations of system functionality allowing the system to pick which implementation to use at runtime. This allows the system to specialize software to a given hardware platform and more importantly to a given set of external considerations (e.g. network connectivity, available sensors, etc.). Our proposed infrastructure will enable us to study two key aspects of uncertainty: (1) Almost all mobile applications store user data and configuration parameters in mobile databasesand access to this data can have a profound impact on the behavior of an application. POCKETDATA will allow us to more readily study this aspect of mobile uncertainty; (2) The infrastructure powering our runtime system for exposing uncertainty is built around a mobile database that stores possible choices the software system can make. POCKETDATA will allow us to optimize this database to reduce choice latency.

PIs Kennedy and Ziarek have a joint research project, Just-in-Time Data Structures (JITDs), focusing on adaptive indexing [53]. The project explores the use of standardized, composable data structure building blocks to dynamically assemble indexes that adapt to rapidly changing workload requirements. The level of variation in load and resource availability that occurs in POCKETDATA workloads creates an ideal use-case for JITDs. As noted above, our proposed infrastructure will provide us with a benchmark workload that will help us to evaluate adaptive indexes under real-world conditions, rather than through purely synthetic workloads.

PI Kennedy is part of a collaborative research project with *Shambhu Upadhyaya* (UB), *Varun Chandola* (UB), *Hung Ngo* (UB), and *Long Nguyen* (UMich) that explores techniques for identifying insider attacks on databases (NSF-CNS-1409551). Although the threat of insider attacks on mobile devices may be minimal, the specific methodology behind the work involves summarizing query logs by creating clusters of queries with similar "intent." The approach is showing promise for summarizing query logs from a corporate (banking) setting. Having query logs from other settings like POCKETDATA would show that the approach can be generalized to domains other than Insider Threat detection (for example to the design of index selection tools). If successful, these efforts could also contribute back to the POCKETDATA project, as a tool for quickly summarizing and clustering query logs would help to build out the visualization and benchmark design components of the proposed infrastructure.

3 — Community Involvement

The PIs have already reached out to the database and mobile systems communities for feedback on the current infrastructure, providing the PIs with an initial community and a preliminary source of feedback on design, APIs, and features. A detailed description can be found above, in Section 2. In summary, there is interest from researchers working on embedded databases, small-scale data management, personal sensing, query interfaces, and several closely related areas. The POCKETDATA benchmark will serve as a focal point for the community's involvement by providing the community with a way to explore, discuss,

and disseminate new data management use cases, and by offering a standard way to evaluate systems on those use cases.

Preliminary work on characterizing differences between POCKETDATA and traditional benchmarking infrastructures was presented at the TPC-TC symposium, which has allowed the PIs to solicit industrial feedback. The PIs are currently in first stage discussions with researchers from VMware, Cisco, Google, Samsung, and Oracle regarding TPC involvement in the POCKETDATA benchmark. From this starting point the PIs will also broaden their target communities to included researchers from programming languages as well as real-time and embedded systems.

The PIs believe that expanding the pervue of POCKETDATA to also include IoT will broaden the utility of the proposed infrastructure. IoT has recently renewed interest in databases systems that are specialized for IoT (stream databases, in-network query processors) and/or are capable of running on embedded devices (*e.g.*, TinyDB [60]). As embedded processors become more capable, with larger amounts of main memory available (e.g. Intel's Edison platform), there is a growing push from the embedded and also from the real-time communities to explore including databases and query processing in small scale embedded systems. The PIs believe that emerging research in smart cities and personalized medical devices that aggregate and processes biometric data would benefit from POCKETDATA. Domain specific languages (DSLs) are becoming more pervasive as mechanisms to both amplify programmer effort for specialized systems and to greatly improve performance in time, space, and even energy consumption. The PIs believe that POCKETDATA will be of interested to both database and programming language researchers in the IoT space.

> *I think synchronizing device data with server data is a very common occurrence in this space. As a simple example, you should be able to synchronize your 'contacts' database on your cell phone with a server repository. Recently, Mike Brey, Raghu Nambiar and I proposed a "strawman" IoT benchmark [44] — I think extending your work to include large-scale data synchronization would be worth considering.*
> -Ashok Joshi (Senior Director; Oracle NoSQL Database, Berkeley DB, Database Mobile Server)

Evidence of Support The PIs have conducted several informal surveys and participated in many discussions with members of the CISE community to better understand the community's needs. Table 7 summarizes our current efforts toward building a community and the community's support for our proposed infrastructure.

researcher	affiliation	research area	enabled research
Stratos Idreos	*Harvard*	Databases	Adaptive indexes
Arnab Nandi	*Ohio State*	Databases/HCI	Interactive analytics
Nasko Routnev	*Ohio State*	Programming Languages	Mobile data flow analysis
Christoph Koch	*EPFL*	Databases/Theory	Database compilers
Ashok Joshi	*Oracle*	Databases/IoT	IoT performance
Michael Brey	*Oracle*	Databases/Mobile Systems	Embedded DB performance
Meikel Poess	*Oracle*	Databases	Performance measurement
Raghunath Nambiar	*Cisco*	Databases	Performance measurement
Reza Taheri	*VMWare*	Databases	Performance measurement
Jens Dittrich	*Saarland University*	Databases/Mobile Systems	Small-data analytics
Sharad Agarwal	*Microsoft Research*	Mobile Systems/Sensing	Mobile systems performance

Figure 7: **Existing Community Interest in** POCKETDATA

4 — Planning Activities

Our planning process will consist of a development effort and an outreach effort. First and foremost, the centerpiece of our development side community-building efforts is the POCKETDATA benchmark. In addition to acting as a standard for evaluating research efforts that overcome bottlenecks and limitations of existing technology, the *modular* benchmark will serve as a hub for the community to discuss and describe these limitations. Under the guidance of the PIs, the graduate student supported by this proposal will be responsible for developing a preliminary prototype benchmark. The first version of this benchmark will

stress bottlenecks identified in our preliminary study [48] by simulating the behavior of a small number of smartphone apps. Using data and query logs derived from our preliminary study, we hope to have version one of the benchmark ready within 4-6 months. The benchmark will be released and advertised over community mailing lists like DBWorld [2]. By this point, we expect to have expanded the POCKETDATA community through our outreach efforts. After releasing the benchmark we will hold a 3 month community feedback process, allowing us to release version 2 of the benchmark based on community feedback before the end of the planning period. Additionally we will pursue feedback from the IoT community to understand how POCKETDATA can be extended to meet the IoT community's needs. We envision that these needs will vary depending on the aspect of IoT a given community is interested in (e.g. language runtime design vs. embedded databases). To avoid creating an infrastructure only suited to the needs of a particular niche, we will solicit feedback from many sources.

In addition to the PocketData community, we will leverage interest from the Transaction Processing Council (TPC) in developing an embedded database benchmark. The TPC represents one of the most prominent names in database benchmarking, and is responsible for benchmarks like TPC-C [22], TPC-H [24], and TPC-DS [23] that are touchstones for evaluating research in databases. After presenting our preliminary work at the TPC's annual symposium colocated with VLDB 2015, *Raghunath Nambiar* (Cisco), *Reza Taheri* (VMWare), and *Meikel Poess* (Oracle) of the TPC expressed interest in helping us to develop POCKETDATA as an eventual TPC benchmark. The PIs hope to also participate in TPC discussions on IoT concerns. The TPC discussions will provide the PIs will both industry and academic perspectives on both embedded databases as well as IoT. The PIs hope to leverage this information in the design of the proposed POCKETDATA infrastructure. Although all PIs will be involved in communications with the TPC and its members, PI Kennedy will act as a lead point of contact.

To continue building our current community and to expand it to include IoT researchers, the PIs expect to travel to top conferences in a variety of fields. Our outreach efforts will begin with poster sessions, tutorials, and demos presented at prominent database conferences. One candidate is ICDE 2017, which takes place early in the planning period. PI Kennedy will coordinate efforts to perform a demonstration at a database conference to incite discussion and interest in POCKETDATA from the database community. PI Ziarek will coordinate efforts for a demonstration or poster presentation initially targeting SPLASH 2016 to reach out to the PL community, and PI Challen will coordinate efforts for a demonstration or poster presentation initially targeting MobiSys 2017 to reach out to the mobile systems community. At these conferences the PIs will network with researchers who work on IoT as well. In addition, there are many new conferences focusing on IoT that are emerging. The PIs expect to attend IoTA, IoTDI, and WF-IoT. Towards the end of the first year of the proposal, the PIs will begin to develop a tutorial on embedded databases and plan for a POCKETDATA workshop.

The PIs will submit a **CI-NEW** proposal for POCKETDATA in Fall of 2017, approximately 15 months after the start of the planning proposal.

5 — Broader Impacts of the Proposed Work

With 2 billion smartphones in the world and more being added every day, mobile platforms together form the most pervasive distributed system on the planet. People are increasingly relying on smartphones to manage their lives, from contacts and todo lists, to their health, their homes, and the contents of their wallets. This proliferation of data-driven smartphone apps is causing a need for more, faster, more user-friendly, and more power-aware techniques for managing data on smartphones and embedded devices.

To meet the challenges of this new frontier in data management, it is critical that we begin understand how smartphone apps store and retrieve structured state and establish standards for evaluating potential advances based on this understanding. Our proposal lays the groundwork for research on pocket-scale data management. We have interest from the Transaction Processing Council for our proposed benchmark, and even before the planning stage, several members of the database, systems, and programming language communities have expressed interest in the resources we propose to offer.

In addition to supporting research in a critical area, this proposal will support one graduate student during the planning phase and up to two graduate students in later phases, contributing to between one and two PhD theses. We anticipate that the proposed work may also lead to one or two MS theses, and if funded, plan to apply for an REU supplement for this proposal. The resources created by this proposal will also

be integrated into courses taught by the PIs, a process that has already started: PIs Kennedy and Ziarek recently co-taught a project-oriented course entitled "CSE-662: Languages and Runtimes for Big Data." The course included material related to POCKETDATA research, and three of the seven groups in the course worked on projects based on POCKETDATA and the Internet of Things.

6 — Results from Prior NSF Support

PI Kennedy has been awarded "TWC: Medium: Collaborative: Data is Social: Exploiting Data Relationships to Detect Insider Attacks", starting on 10/01/2014 and ending on 09/30/2018. The award number is CNS-1409551 and the amount is $959,999 with a duration of four years, including an REU supplement.

Results Related to Intellectual Merit: The project has collected a trace of all queries performed on all database servers at M&T bank for a period of one week. With the help of M&T security staff, this trace was anonymized and cleared for export to UB hardware. The project team recently submitted a short paper to a WWW 2016 workshop describing our approach to intent-based query clustering, and is working on a full version of the paper for publication within a month.

Results Related to Broader Impact: The project's use of M&T query data as a test case provides feedback about the reliability of our tool, and ensures that it can be applied to real-world problems.

Evidence of Research Products: Project Co-PIs Upadhyaya and Chandola presented our preliminary work at a SaTC PI meeting in January 2015. PI Upadhyaya and team member Gokhan Kul presented a preliminary paper on insider attack ontologies at the USENIX MIST workshop.

PI Ziarek is a PI on two current NSF awards. The one most related to the proposed project is "II-EN: Collaborative Research: Positioning MLton for Next-Generation Programming Languages Research" starting on 07/30/2014 and ending on 07/29/2017. The award number is CNS-1405614 and the amount is $381,640.00 with a duration of three years.

Results Related to Intellectual Merit: The development team has currently merged the Multi-MLton GC structure with that of the main line MLton branch and is currently undergoing extensive testing and performance tuning. The restructured GC will allow MLton users to seamlessly migrate their existing code base to execute on multi-core machines and leverage multiple concurrent and parallel GC configurations. This award has resulted in a publication titled "Adding Real-time Capabilities to a SML Compiler" in DPRTCPS'15.

Results Related to Broader Impact: The development team has produced a new version of the MLton hacker's guide available at: `https://github.com/UBMLtonGroup/HackersGuide`. The substantially updated hacker's guide provides new MLton contributors detailed knowledge on the compiler and runtime structures, focusing on the new additions of the parallel runtime.

Evidence of Research Products: The development team has all source code available in a public git hub repository `https://github.com/UBMLtonGroup`. The changes are slated to be merged into the mainline MLton branch by the end of the year.

PI Challen is a PI on three current NSF awards. The one most related to the proposed project is "PhoneLab: A Programmable Participatory Smartphone Testbed", which started 06/01/2012 and ends 05/30/2016. The award number is CNS-1205656 and the amount is $1,322,510 with a duration of three years.

Results Related to Intellectual Merit: PHONELAB opened its platform sources to external researchers in February 2015, and we are actively collaborating with several groups on experiments.

Results Related Broader Impact: PHONELAB software and datasets are available to external researchers subject to human subjects review.

Evidence of Research Products: PHONELAB has produced 13 internal [69, 15, 68, 26, 80, 25, 70, 61, 13, 49, 81, 62, 82] and 2 external [31, 36] publications. Several more recent works using PHONELAB are under submission.

REFERENCES

[1] PolePosition: The open source database benchmark. http://www.polepos.org.

[2] ACM SIGMOD. Dbworld. https://research.cs.wisc.edu/dbworld/.

[3] Sumit Agarwal, Daniel Bellinger, Oliver Kennedy, Ankur Upadhyay, and Lukasz Ziarek. Monadic logs for collaborative web applications. In *WebDB*. ACM, 2013.

[4] Sanjay Agrawal, Surajit Chaudhuri, and Vivek R. Narasayya. Automated selection of materialized views and indexes in sql databases. In *Proceedings of the 26th International Conference on Very Large Data Bases*, VLDB '00, pages 496–505, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. `http://dl.acm.org/citation.cfm?id=645926.671701`

[5] Yanif Ahmad, Oliver Kennedy, Christoph Koch, and Milos Nikolic. DBToaster: Higher-order delta processing for dynamic, frequently fresh views. *PVLDB*, 2012.

[6] Sabbir Ahmed. MobiGen: a mobility generator for environment aware mobility model. http://arrow.monash.edu.au/hdl/1959.1/109933, 2009.

[7] Ioannis Alagiannis, Stratos Idreos, and Anastasia Ailamaki. H2o: A hands-free adaptive store. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Snowbird, Utah, 2014.

[8] Rizwana Begum, Guru Prasad Srinivasa, David Werner, Jerry Ajay, Geoffrey Challen, and Mark Hempstead. Energy-performance trade-offs on energy-constrained devices with multi-component dvfs. In *Proc. of the 2015 IEEE International Symposium on Workload Characterization (IISWC'15)*, October 2015.
`https://blue.cse.buffalo.edu/papers/iiswc2015-agility`

[9] Ethan Blanton and Lukasz Ziarek. Non-blocking inter-partition communication with wait-free pair transactions. In *Proceedings of the 11th International Workshop on Java Technologies for Real-time and Embedded Systems*, JTRES '13, pages 58–67, New York, NY, USA, 2013. ACM.
`http://doi.acm.org/10.1145/2512989.2512994`

[10] I.T. Bowman, P. Bumbulis, D. Farrar, A.K. Goel, B. Lucier, A. Nica, G.N. Paulley, J. Smirnios, and M. Young-Lai. Sql anywhere: A holistic approach to database self-management. In *Data Engineering Workshop, 2007 IEEE 23rd International Conference on*, pages 414–423, April 2007.

[11] Don Box and Anders Hejlsberg. LinQ: .NET language-integrated query. *MSDN Developer Centre*, 89, 2007.

[12] A.T. Campbell, S.B. Eisenman, N.D. Lane, E. Miluzzo, R.A. Peterson, Hong Lu, Xiao Zheng, M. Musolesi, K. Fodor, and Gahng-Seop Ahn. The rise of people-centric sensing. *Internet Computing, IEEE*, 12(4):12–21, July 2008.

[13] Geoffrey Challen, Jerry Ajay, Nick DiRienz, Oliver Kennedy, Anudipa Maiti, Anandatirtha Nandugudi, Sriram Shantharam, Jinghao Shi, Guru Prasad Srinivasa, and Lukasz Ziarek. `maybe` we should enable more uncertain mobile app programming. In *Proc. of the 16th Workshop on Hot Topics in Mobile Computing Systems and Applications (HotMobile'15)*, February 2015.
`https://blue.cse.buffalo.edu/papers/hotmobile2015-maybe/`

[14] Geoffrey Challen, Jerry Antony Ajay, Nick DiRienzo, Oliver Kennedy, Anudipa Maiti, Anandatirtha Nandugudi, Sriram Shantharam, Jinghao Shi, Guru Prasad Srinivasa, and Lukasz Ziarek. maybe we should enable more uncertain mobile app programming. In *HotMobile*, pages 105–110, 2015.
`http://doi.acm.org/10.1145/2699343.2699361`

[15] Geoffrey Challen, Scott Haseley, Anudipa Maiti, Anandatirtha Nandugudi, Guru Prasad, Mukta Puri, and Junfei Wang. The mote is dead. long live the discarded smartphone! In *Proc. of the 15th Workshop on Mobile Systems and Applications (HotMobile'14)*, February 2014.
`https://blue.cse.buffalo.edu/papers/hotmobile2014-sustainability`

[16] Surajit Chaudhuri and Vivek Narasayya. AutoAdmin "what-if" index analysis utility. In *SIGMOD*, 1998.
`http://doi.acm.org/10.1145/276304.276337`

[17] Surajit Chaudhuri and Vivek Narasayya. Self-tuning database systems: A decade of progress. In *VLDB*, 2007.
`http://dl.acm.org/citation.cfm?id=1325851.1325856`

[18] Surajit Chaudhuri and Vivek R. Narasayya. An efficient cost-driven index selection tool for microsoft sql server. In *PVLDB*, pages 146–155, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers

Inc.
http://dl.acm.org/citation.cfm?id=645923.673646

[19] Alvin Cheung, Samuel Madden, and Armando Solar-Lezama. Sloth: Being lazy is a virtue (when issuing database queries). In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 931–942, New York, NY, USA, 2014. ACM.
http://doi.acm.org/10.1145/2588555.2593672

[20] Alvin Cheung, Samuel Madden, Armando Solar-Lezama, Owen Arden, and Andrew C. Myers. StatusQuo: making familiar abstractions perform using program analysis. In *Conference on Innovative Data Systems Research (CIDR)*, January 2013.

[21] Brian F. Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, and Russell Sears. Benchmarking cloud serving systems with YCSB. In *SOCC*, New York, NY, USA, 2010. ACM.
http://doi.acm.org/10.1145/1807128.1807152

[22] Transaction Processing Performance Council. TPC-C specification. http://www.tpc.org/tpcc/.

[23] Transaction Processing Performance Council. TPC-DS specification. http://www.tpc.org/tpcds/.

[24] Transaction Processing Performance Council. TPC-H specification. http://www.tpc.org/tpch/.

[25] Nick DiRienzo and Geoffrey Challen. Controlling smartphone user privacy via objective-driven context mocking. In *Proc. of the 6th International Conference on Mobile Computing, Applications and Services (MobiCASE'14)*, November 2014.
https://blue.cse.buffalo.edu/papers/mobicase2014-pocketmocker

[26] Nick DiRienzo and Geoffrey Challen. Should smartphone users mock apps? In *Proc. of the 6th ACM HotPlanet Workshop (HotPlanet'14)*, October 2014.
https://blue.cse.buffalo.edu/papers/hotplanet2014-pocketmocker

[27] Jens Dittrich. The Case for Small Data Management. In *CIDR*, 2015.

[28] Jens Dittrich and Patrick Bender. Janiform intra-document analytics for reproducible research. *Proc. VLDB Endow.*, 8(12):1972–1975, August 2015.

[29] ShakuntalaGupta Edward and Navin Sabharwal. Mongodb limitations. In *Practical MongoDB*, pages 227–232. Apress, 2015.

[30] Aren Memet Erkan and Euclid Keramopoulos. Egql: A gesture query language. In *Proceedings of the 19th Panhellenic Conference on Informatics*, PCI '15, pages 348–353, New York, NY, USA, 2015. ACM.
http://doi.acm.org/10.1145/2801948.2802006

[31] Zhaoyu Gao, Arun Venkataramani, James F. Kurose, and Simon Heimlicher. Towards a quantitative comparison of location-independent network architectures. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, pages 259–270, New York, NY, USA, 2014. ACM.
http://doi.acm.org/10.1145/2619239.2626333

[32] Goetz Graefe, Felix Halim, Stratos Idreos, Harumi Kuno, and Stefan Manegold. Concurrency control for adaptive indexing. *PVLDB*, 2012.

[33] Goetz Graefe, Stratos Idreos, Harumi Kuno, and Stefan Manegold. Benchmarking adaptive indexing. In *TPC-TC*, 2011.
http://dl.acm.org/citation.cfm?id=1946050.1946063

[34] Goetz Graefe and Harumi Kuno. Self-selecting, self-tuning, incrementally optimized indexes. In *EDBT*, 2010.
http://doi.acm.org/10.1145/1739041.1739087

[35] Felix Halim, Stratos Idreos, Panagiotis Karras, and Roland H. C. Yap. Stochastic database cracking: Towards robust adaptive indexing in main-memory column-stores. *PVLDB*, 2012.

[36] Marian Harbach, Alexander De Luca, and Serge Egelman. The anatomy of smartphone unlocking: A field study of android lock screens. In *Proceedings of the 2016 ACM Conference on Human Factors in Computing Systems (CHI'2016)*, 2016.

[37] Stratos Idreos, Martin L Kersten, and Stefan Manegold. Database cracking. In *CIDR*, 2007.

[38] Stratos Idreos, Martin L. Kersten, and Stefan Manegold. Updating a cracked database. In *SIGMOD*, 2007.
http://doi.acm.org/10.1145/1247480.1247527

[39] Stratos Idreos, Stefan Manegold, and Goetz Graefe. Adaptive indexing in modern database kernels. In *EDBT*, 2012.
http://doi.acm.org/10.1145/2247596.2247667

[40] Stratos Idreos, Stefan Manegold, Harumi Kuno, and Goetz Graefe. Merging what's cracked, cracking what's merged: Adaptive indexing in main-memory column-stores. *PVLDB*, 2011.
[41] Sooman Jeong, Kisung Lee, Seongjin Lee, Seoungbum Son, and Youjip Won. I/O stack optimization for smartphones. In *USENIX ATC*, pages 309–320, Berkeley, CA, USA, 2013. USENIX Association. `http://dl.acm.org/citation.cfm?id=2535461.2535499`
[42] Lilong Jiang, Michael Mandel, and Arnab Nandi. Gesturequery: A multitouch database query interface. *Proc. VLDB Endow.*, 6(12):1342–1345, August 2013.
[43] Lilong Jiang and Arnab Nandi. SnapToQuery: Providing interactive feedback during exploratory query specification. In *VLDB*, volume 8, pages 1250–1261. VLDB Endowment, July 2015.
[44] Ashok Joshi, Raghunath Nambiar, and Michael Brey. Benchmarking internet of things solutions. In Tilmann Rabl, Kai Sachs, Meikel Poess, Chaitanya Baru, and Hans-Arno Jacobson, editors, *Big Data Benchmarking*, volume 8991 of *Lecture Notes in Computer Science*, pages 29–36. Springer International Publishing, 2015.
[45] O. Kennedy and C. Koch. PIP: A database system for great and small expectations. In *ICDE*, pages 157–168, 2010.
[46] O. Kennedy, C. Koch, and A. Demers. Dynamic approaches to in-network aggregation. In *ICDE*, pages 1331–1334, 2009.
[47] Oliver Kennedy, Yanif Ahmad, and Christoph Koch. DBToaster: Agile views for a dynamic data management system. In *CIDR*, pages 284–295, 2011.
[48] Oliver Kennedy, Jerry Ajay, Geoffrey Challen, and Lukasz Ziarek. Pocket Data: The need for TPC-MOBILE. In *TPC Technology Conference on Performance Evaluation & Benchmarking*, 2015.
[49] Oliver Kennedy, Jerry Ajay, Geoffrey Challen, and Lukasz Ziarek. Pocket data: The need for tpc-mobile. In *Proc. of the 7th Technology Conference on Performance Evaluation and Benchmarking (TPCTC'15)*, August 2015.
`https://blue.cse.buffalo.edu/papers/tpctc2015-pocketdata/`
[50] Oliver Kennedy, Steve Lee, Charles Loboz, Slawek Smyl, and Suman Nath. Fuzzy prophet: Parameter exploration in uncertain enterprise scenarios. In *SIGMOD*, pages 1303–1306, 2011.
`http://doi.acm.org/10.1145/1989323.1989482`
[51] Oliver Kennedy and Suman Nath. Jigsaw: Efficient optimization over uncertain enterprise data. In *SIGMOD*, pages 829–840, 2011.
`http://doi.acm.org/10.1145/1989323.1989410`
[52] Oliver Kennedy and Lukasz Ziarek. BarQL: Collaborating through change. Technical report, CORR, arXiv:1303.4471, 2013.
[53] Oliver Kennedy and Lukasz Ziarek. Just-in-time data structures. In *CIDR*, 2015.
[54] Je-Min Kim and Jin-Soo Kim. AndroBench: Benchmarking the storage performance of Android-based mobile devices. In Sabo Sambath and Egui Zhu, editors, *Frontiers in Computer Education*, volume 133 of *Advances in Intelligent and Soft Computing*, pages 667–674. Springer Berlin Heidelberg, 2012.
[55] Predrag Klasnja, Sunny Consolvo, David W McDonald, James A Landay, and Wanda Pratt. Using mobile & personal sensing technologies to support health behavior change in everyday life: lessons learned. In *AMIA*, 2009.
[56] Yannis Klonatos, Christoph Koch, Tiark Rompf, and Hassan Chafi. Building efficient query engines in a high-level language. *Proc. VLDB Endow.*, 7(10):853–864, June 2014.
[57] Yannis Klonatos, Andres Nötzli, Andrej Spielmann, Christoph Koch, and Victor Kuncak. Automatic synthesis of out-of-core algorithms. In *SIGMOD*, 2013.
`http://doi.acm.org/10.1145/2463676.2465334`
[58] Christoph Koch, Yanif Ahmad, Oliver Andrzej Kennedy, Milos Nikolic, Andres Nötzli, Daniel Lupei, and Amir Shaikhha. DBToaster: Higher-order delta processing for dynamic, frequently fresh views. *VLDBJ*, 2013 (to appear).
[59] S.C.K. Lam, Kai Lap Wong, Kwok On Wong, Wenxiu Wong, and Wai Ho Mow. A smartphone-centric platform for personal health monitoring using wireless wearable biosensors. In *ICICS*, Dec 2009.
[60] Samuel R. Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. TinyDB: An acquisitional query processing system for sensor networks. *ACM TODS*, 30(1):122–173, March 2005.
[61] Anudipa Maiti and Geoffrey Challen. The missing numerator: Toward a value measure for smartphone apps. In *Proc. of the 15th Workshop on Hot Topics in Mobile Computing Systems and Applications*

(HotMobile'15), February 2015.
https://blue.cse.buffalo.edu/papers/hotmobile2015-numerator

[62] Anudipa Maiti, Yihong Chen, and Geoffrey Challen. Jouler: A policy framework enabling effective and flexible smartphone energy management. In *Proc. of the 7th International Conference on Mobile Computing, Applications and Services (MobiCASE'15)*, November 2015.
https://blue.cse.buffalo.edu/papers/mobicase2015-jouler/

[63] Erik Meijer, Brian Beckman, and Gavin Bierman. Linq: Reconciling object, relations and XML in the .NET framework: Reconciling object, relations and xml in the .net framework. In *SIGMOD*, pages 706–706, New York, NY, USA, 2006. ACM.

[64] Sergey Melnik, Atul Adya, and Philip A. Bernstein. Compiling mappings to bridge applications and databases. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, SIGMOD '07, pages 461–472, New York, NY, USA, 2007. ACM.
http://doi.acm.org/10.1145/1247480.1247532

[65] Thomas Mueller. H2 database engine, 2006.

[66] Arnab Nandi. Querying without keyboards. In *CIDR*, 2013.

[67] Arnab Nandi, Lilong Jiang, and Michael Mandel. Gestural query specification. *Proc. VLDB Endow.*, 7(4):289–300, December 2013.

[68] Anandatirtha Nandugudi, Taeyeon Ki, Carl Nuessle, and Geoffrey Challen. Pocketparker: Pocket-sourcing parking lot availability. In *Proc. of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp'14)*, September 2014.
https://blue.cse.buffalo.edu/papers/ubicomp2014-pocketparker/

[69] Anandatirtha Nandugudi, Anudipa Maiti, Taeyeon Ki, Fatih Bulut, Murat Demirbas, Tevfik Kosar, Chunming Qiao, Steven Y. Ko, and Geoffrey Challen. PhoneLab: A large programmable smartphone testbed (invited). In *Proc. of the 1st International Workshop on Sensing and Big Data Mining (SenseMine 2013)*, November 2013.
https://blue.cse.buffalo.edu/papers/sensemine2013-phonelab/

[70] Anandatirtha Nandugudi, Carl Nuessle, Geoffrey Challen, Emiliano Miluzzo, and Yih-Farn Chen. The pocketlocker personal cloud storage system. In *Proc. of the 6th International Conference on Mobile Computing, Applications and Services (MobiCASE'14)*, November 2014.
https://blue.cse.buffalo.edu/papers/mobicase2014-pocketlocker/

[71] Thomas Neumann. Efficiently compiling efficient query plans for modern hardware. *VLDBJ*, 4(9):539–550, June 2011.

[72] Sadegh Nobari, Farhan Tauheed, Thomas Heinis, Panagiotis Karras, Stéphane Bressan, and Anastasia Ailamaki. Touch: In-memory spatial join by hierarchical data-oriented partitioning. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 701–712, New York, NY, USA, 2013. ACM.
http://doi.acm.org/10.1145/2463676.2463700

[73] Patrick O'Neil, Elizabeth O'Neil, Xuedong Chen, and Stephen Revilak. The star schema benchmark and augmented fact table indexing. In Raghunath Nambiar and Meikel Poess, editors, *Performance Evaluation and Benchmarking*, volume 5895 of *Lecture Notes in Computer Science*, pages 237–252. Springer Berlin Heidelberg, 2009.

[74] Mike Owens and Grant Allen. *SQLite*. Springer, 2010.

[75] Filip Pizlo, Lukasz Ziarek, Ethan Blanton, Petr Maj, and Jan Vitek. High-level programming of embedded hard real-time devices. In *Proceedings of the 5th European Conference on Computer Systems*, EuroSys '10, pages 69–82, New York, NY, USA, 2010. ACM.
http://doi.acm.org/10.1145/1755913.1755922

[76] Filip Pizlo, Lukasz Ziarek, Petr Maj, Antony L. Hosking, Ethan Blanton, and Jan Vitek. Schism: Fragmentation-tolerant real-time garbage collection. In *Proceedings of the 2010 ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '10, pages 146–159, New York, NY, USA, 2010. ACM.
http://doi.acm.org/10.1145/1806596.1806615

[77] Naveen Kumar Ramamurthy, Sankara Vadivel Dhandapani, Saravanan Adaikkalavan, and Sathish Kumar Deivasigamani. Pocketdata benchmark (team tbd). University at Buffalo CSE-662 Final Report, Dec 2015.

[78] Atanas Rountev and Dacong Yan. Static reference analysis for GUI objects in Android software. In *International Symposium on Code Generation and Optimization*, pages 143–153, 2014.

[79] Felix Martin Schuhknecht, Alekh Jindal, and Jens Dittrich. The uncracked pieces in database cracking. *Proceedings of the Very Large Data Bases Endowment (PVLDB)*, 7(2), 2013.

[80] Jinghao Shi, Zhangyu Guan, Chunming Qiao, Tommaso Melodia, Dimitrios Koutsonikolas, and Geoffrey Challen. Crowdsourcing access network spectrum allocation using smartphones. In *Proc. of the 13th ACM Workshop on Hot Topics in Networks (HotNets'14)*, October 2014.
`https://blue.cse.buffalo.edu/papers/hotnets2014-pocketsniffer`

[81] Jinghao Shi, Liwen Gui, Dimitrios Koutsonikolas, Chunming Qiao, and Geoffrey Challen. A little sharing goes a long way: The case for reciprocal wifi sharing. In *Proc. of the 2nd Workshop on Hot Topics in Wireless (HotWireless'15)*, September 2015.
`https://blue.cse.buffalo.edu/papers/hotwireless2015-sharing`

[82] Jinghao Shi, Lei Meng, Aaron Striegel, Chunming Qiao, Dimitrios Koutsonikolas, and Geoffrey Challen. A walk on the client side: Monitoring enterprise wifi networks using smartphone channel scans. In *Proc. of the 2016 IEEE International Conference on Computer Communications (INFOCOM'16)*, May 2016.
`https://blue.cse.buffalo.edu/papers/infocom2015-scans/`

[83] Manish Singh, Arnab Nandi, and H. V. Jagadish. Skimmer: Rapid scrolling of relational query results. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, pages 181–192, New York, NY, USA, 2012. ACM.
`http://doi.acm.org/10.1145/2213836.2213858`

[84] KC Sivaramakrishnan, Lukasz Ziarek, and Suresh Jagannathan. Eliminating read barriers through procrastination and cleanliness. In *Proceedings of the 2012 International Symposium on Memory Management*, ISMM '12, pages 49–60, New York, NY, USA, 2012. ACM.
`http://doi.acm.org/10.1145/2258996.2259005`

[85] Alexandros Stougiannis, Farhan Tauheed, Mirjana Pavlovic, Thomas Heinis, and Anastasia Ailamaki. Data-driven Neuroscience: Enabling Breakthroughs Via Innovative Data Management. In *Proceedings of International Conference on Management of Data (SIGMOD '13)*, 2013.

[86] F. Tauheed, L. Biveinis, T. Heinis, F. Schurmann, H. Markram, and A. Ailamaki. Accelerating range queries for brain simulations. In *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*, pages 941–952, April 2012.

[87] The Hibernate Team. Hibernate orm. http://hibernate.org/orm/.

[88] Hannes Voigt, Thomas Kissinger, and Wolfgang Lehner. SMIX: Self- managing indexes for dynamic workloads. In *SSDBM*, 2013.
`http://doi.acm.org/10.1145/2484838.2484862`

[89] Christian Wimmer and Thomas Würthinger. Truffle: A self-optimizing runtime system. In *Proceedings of the 3rd Annual Conference on Systems, Programming, and Applications: Software for Humanity*, SPLASH '12, pages 13–14, New York, NY, USA, 2012. ACM.
`http://doi.acm.org/10.1145/2384716.2384723`

[90] Dacong Yan, Guoqing Xu, Shengqian Yang, and Atanas Rountev. LeakChecker: Practical static memory leak detection for managed languages. In *International Symposium on Code Generation and Optimization*, pages 87–97, 2014.

[91] Yin Yan, Sree Harsha Konduri, Amit Kulkarni, Varun Anand, Steven Y. Ko, and Lukasz Ziarek. Rtdroid: A design for real-time android. In *Proceedings of the 11th International Workshop on Java Technologies for Real-time and Embedded Systems*, JTRES '13, pages 98–107, New York, NY, USA, 2013. ACM.
`http://doi.acm.org/10.1145/2512989.2512990`

[92] Shengqian Yang. *Static Analyses of GUI Behavior in Android Applications*. PhD thesis, Ohio State University, September 2015.

[93] Shengqian Yang, Dacong Yan, Haowei Wu, Yan Wang, and Atanas Rountev. Static control-flow analysis of user-driven callbacks in Android applications. In *International Conference on Software Engineering*, pages 89–99, 2015.

[94] Ying Yang, Niccolò Meneghetti, Ronny Fehling, Zhen Hua Liu, and Oliver Kennedy. Lenses: An on-demand approach to etl. *Proc. VLDB Endow.*, 8(12):1578–1589, August 2015.

[95] Lukasz Ziarek and Suresh Jagannathan. Lightweight checkpointing for concurrent ml. *J. Funct. Program.*, 20(2):137–173, March 2010.

[96] Lukasz Ziarek, KC Sivaramakrishnan, and Suresh Jagannathan. Composable asynchronous events. In *Proceedings of the 32Nd ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '11, pages 628–639, New York, NY, USA, 2011. ACM.
http://doi.acm.org/10.1145/1993498.1993572

[97] Lukasz Ziarek, Stephen Weeks, and Suresh Jagannathan. Flattening tuples in an ssa intermediate representation. *Higher Order Symbol. Comput.*, 21(3):333–358, September 2008.