Collecting and analyzing smartphone sensor data for health

Justin A. Drake
Kelly Gaither
jdrake@tacc.utexas.edu
kelly@tacc.utexas.edu
Texas Advanced Computing Center
Austin, TX, USA

Karl W. Schulz
Radek Bukowski
karl@utexas.edu
radek.bukowski@austin.utexas.edu
Dell Medical School
Austin, TX, USA

ABSTRACT

Evidence suggests that signatures of health and disease, or digital biomarkers, exist within the heterogeneous, temporally-dense data gathered from smartphone sensors and wearable devices that can be leveraged for medical applications. Modern smartphones contain a collection of energy-efficient sensors capable of capturing the device's movement, orientation, and location as well characteristics of its external environment (e.g. ambient temperature, sound, pressure). When paired with peripheral wearable devices like smart watches, smartphones can also facilitate the collection/aggregation of important vital signs like heart rate and oxygen saturation. Here we discuss our recent experiences with deploying an open-source, cloud-native framework to monitor and collect smartphone sensor data from a cohort of pregnant women over a period of one year. We highlight two open-source integrations into the pipeline we found particularly useful: 1) a dashboard–built with Grafana and backed by Graphite–to monitor and manage production server loads and data collection metrics across the study cohort and 2) a back-end storage solution with InfluxDB, a multi-tenant time series database and data exploration ecosystem, to support biomarker discovery efforts of a multidisciplinary research team.

CCS CONCEPTS

• **Information systems → Information integration**; • **Applied computing → Life and medical sciences**; • **Networks → Cloud computing**.

KEYWORDS

smartphone, sensors, digital biomarker, open-source, time-series

ACM Reference Format:
Justin A. Drake, Kelly Gaither, Karl W. Schulz, and Radek Bukowski. 2021. Collecting and analyzing smartphone sensor data for health. In *Practice and Experience in Advanced Research Computing (PEARC '21), July 18–22, 2021, Boston, MA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3437359.3465599

PEARC '21, July 18–22, 2021, Boston, MA, USA

ACM ISBN 978-1-4503-8292-2/21/07…$15.00
https://doi.org/10.1145/3437359.3465599

1 INTRODUCTION

Although nascent, the use of smartphones as a passive-sensing modality and research tool for health applications continues to grow due to the array of embedded sensors (e.g. accelerometer, gyroscope, GPS, etc.) and the ubiquity of ownership [10, 17]. Evidence for the clinical utility of the data generated by smartphone sensors continues to mount as it has been used, for example, to passively monitor cognitive function, mental health, cardiovascular activity [14], and post-surgery recovery [15], among other applications. Data collection infrastructure and methods to extract clinically meaningful signatures (i.e. digital biomarkers) from such data are active areas of research. *Additionally, the need to monitor such studies in near-real time and support data exploration and larger machine-learning, post-processing analysis by multidisciplinary research teams on traditional high performance computing (HPC) systems poses unique infrastructure challenges.* Here we discuss our recent experiences deploying a cloud-native framework to longitudinally collect smartphone sensor data from a cohort of pregnant women over the duration of their pregnancy (approximately one year). We discuss extensions to the pipeline that support data collection/management and downstream efforts to uncover biomarkers associated with the progression of (un)healthy pregnancies. The resulting infrastructure is a blend of commercial cloud and academic HPC resources.

2 DATA COLLECTION INFRASTRUCTURE

A number of solutions exist to collect data from smartphone sensors. Common open-source platforms used for health research include Beiwe (beiwe.org) [16], AWARE (awareframework.com) [11], and the MD2K Software Platform (md2k.org) [12], among others. While implementations and data sampling strategies differ, the frameworks all include a front-end smartphone application to collect/transfer raw sensor data from individual devices and a web server responsible for receiving/organizing data from a fleet of these registered devices. All must operate under the limitations imposed by Apple and Android SDKs in terms of how and to what extent various sensors are exposed to developers.

Because it is open-source, supports both Android and Apple operating systems, employs industry standard encryption, and deploys natively on Amazon Web Services (AWS), we elected to use the Beiwe platform depicted in Figure 1a and detailed in Torous et al. [16]. The Beiwe web server runs on a `t2.medium` EC2 instance and hosts a study manager portal in which participant IDs are generated and study settings, like data sampling schedules, are set. These settings are initialized within the Beiwe application running on a participant's smartphone following device registration.

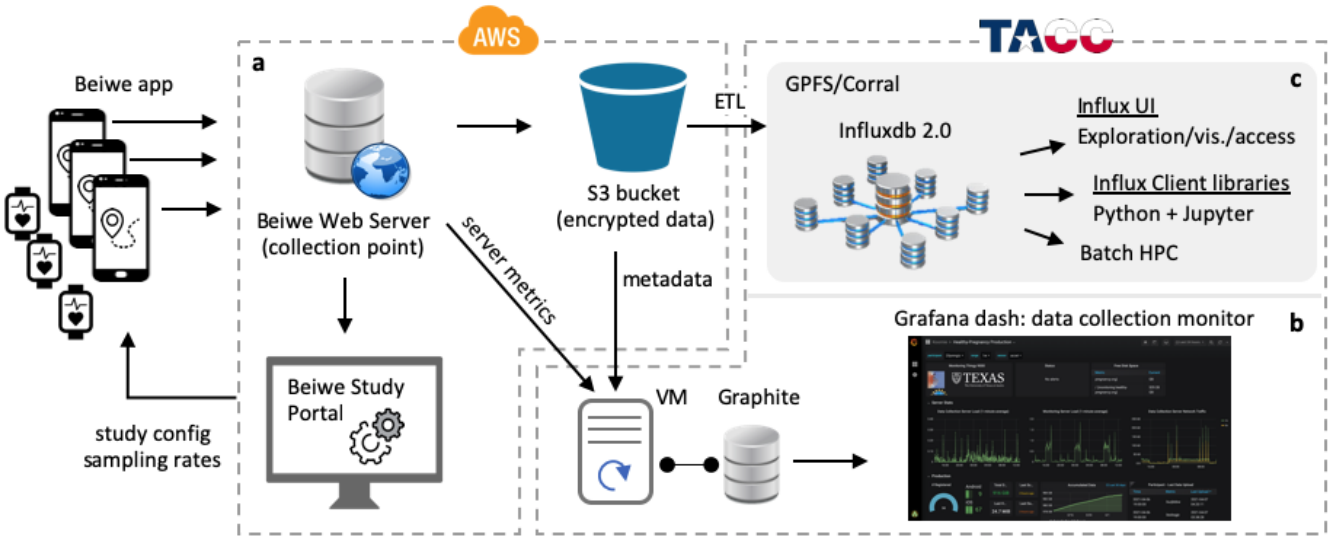

Figure 1: Infrastructure to collect, monitor, store, and analyze smartphone sensor data streams. (a) Beiwe open-source data collection framework includes a front-end smartphone application to sample from embedded, web server, and S3 bucket to store encrypted study data. (b) Data collection monitoring solution tracks EC2 server load/data ingress and data accumulation in S3. (c) Back-end storage and data analysis/visualization solution with InfluxDB.

For our work, data was collected from the following physical and virtual sensors: accelerometer, GPS, power/screen state, call logs, gyroscope (iOS), heart rate if available (iOS watch), step count and distance traveled (iOS watch + phone), and WiFi logs (Android). Collection from data streams occurs at varying time schedules. For example, data is collected in 10 sec on/off cycles from low-power accelerometers and is intended to be used for activity/context recognition over the course of an individual's day. However, GPS data is collected according to a 1 minute on/10 minute off schedule to avoid rapidly draining battery life. The Beiwe application organizes data by source and stores them in CSV files which are periodically uploaded over WiFi to the web server. The server ultimately routes data to an S3 bucket where the raw data remains encrypted.

3 REAL-TIME DATA ACQUISITION

Data drop-out was one of the first major challenges we identified with using consumer smartphones as a health research tool. A number of causes lead to data not being collected and/or uploaded to the Beiwe web server (see Kiang et al. [13] for detailed discussion). For example, a participant may disable location services, accidentally close the app from the task manager, allow their phone to run out of battery, or delete the app because they did not wish to participate in the study further. Additionally, the operating system may limit access to sensors for performance-related issues. Underlying decisions to limit such access remains proprietary and cannot be anticipated. This latter scenario often results in short-lived, transient data drop-out that is often uncovered in downstream analyses. Based on our preliminary findings, user actions tend to result in longer periods of missing data. Given the number of participants and large number of sensors we track, it became necessary to create a monitoring/management tool to rapidly and automatically identify instances of significant data drop-out in real-time and troubleshoot the underlying cause (e.g. so that a research coordinator could contact participants to re-open the app or re-evaluate their desire to participate in the study). Our study protocol does not allow anyone other than the research coordinator to have direct access to the raw sensor data for participants until after they complete the study. Consequently, real-time monitoring strategies regarding participant data fidelity must leverage file metadata only.

A variety of open-source tools can be used to design a comprehensive monitoring platform for longitudinal studies. Our approach is illustrated in Figure 1b and consists of the following software and system elements:

- **collectd** (Linux performance statistics collection daemon)
- **graphite/carbon** (time series storage and aggregation)
- **boto3** (Python SDK for AWS)
- **Grafana** (dashboard monitoring platform)
- **AWS S3 bucket** (raw storage for participant sensor data)
- **Linux virtual machine (VM)** (dedicated monitoring/query server)

The first aim of our monitoring solution is focused on watching server loads and data ingress on the Beiwe web server collection point. This server is critical to the execution of the study. It is the central server that all participant's smartphones communicate with to upload locally cached data. In particular, we sought to ensure that the chosen EC2 instance type for this server was sufficient to respond to the demands of our study as more participants/phones are added over time. For server tracking, we leverage the *collectd* [2] daemon available from the underlying Linux distro (Ubuntu 16.04.6) to track and gather performance statistics from the EC2 instance hosting the Beiwe web server. In particular, we track performance metrics for standard 1/5/15 minute cpu load averages, memory

usage, disk usage, and receive/transmit network packets. We configure this daemon to send packets over tcp to a separate monitoring server hosted on a Linux VM at TACC. This monitoring VM runs the *carbon-cache* daemon, a component of graphite [4] which listens for time-series data and dumps to local disk.

Our second focus was to track individual sensor data as it was uploaded. As highlighted in Figure 1a, the Beiwe software used for the current study delivers encrypted sensor data organized by participant into a top-level AWS S3 bucket. Files are stored on a per-sensor basis (8 potential sensors for Android, 13 potential sensors for iOS) in CSV format. Individual samples are aggregated into small chunks with hundreds to thousands of objects uploaded to S3 each day per study participant. Consequently, the number of stored objects to track scales very quickly. For example, after one year in our study, we have accumulated over 15.3 million object stores. To detect potential data outages for participants, we created a standalone S3 bucket monitoring tool in Python that leverages the *boto3* library [1]. For convenience, this utility is executed on the same Linux VM that is allocated to serve our Grafana dashboard [3] and is automated via crontab execution at 8-hour intervals. The general approach for this utility is to scan all objects within the S3 bucket to track how much sensor data is being accumulated per participant as a function of time. Fortunately, Beiwe includes a Unix epoch timestamp in the object filename (e.g. `1576542542834.csv`), so we can use this to discern sensor timestamps without accessing the raw contents directly. Object timestamps and their associated file sizes are cached so that new study participants and sensor uploads can be uniquely identified during subsequent executions of the bucket scanning utility.

To integrate into a unified monitoring dashboard, this utility uses the *graphyte* Python library [5] to send newly detected files and their statistics (timestamp/filesize) to the carbon metrics server described previously. Once ingested, we can use Grafana to create dynamic snapshots tracking overall study statistics and individual participant history to inform the study research coordinator. One such example from the current study is highlighted in Figure 2. This figure shows the time history of data coming from the accelerometer of an anonymized participant (`op1lbeg2`) over the course of a recent 3 month period. We leverage these types of plots to track each study participant and troubleshoot those for which we have not received data within the last seven days.

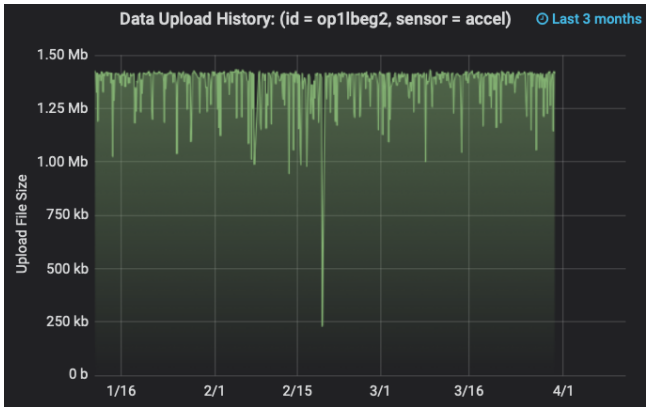

Figure 2: Derived monitoring example for accelerometer data upload using Grafana dashboard.

4 TOWARDS A DIGITAL BIOMARKER DISCOVERY LAB

We have recently deployed, and are currently testing, an open-source solution built around the InfluxDB v2.0 time series platform [9] to effectively store heterogeneous, temporally-dense smartphone data and support the discovery of digital biomarkers associated with maternal and infant birth outcomes. Current components and features of this pipeline (Figure 1c) leverage the scientific computing ecosystem at TACC and are detailed in the following:

Extract-transform-load: After a participant has left the study, their encrypted raw data is downloaded from the AWS S3 bucket to a secure, HIPAA-compliant partition on Corral, a GPFS file system at TACC. While the data is not inherently considered protected health information, we selected this partition because of the sensitive nature of GPS location data and the fact that abstracted medical record data will be included in the future. The initial data collection phase involving roughly 60 participants generated millions of individual CSV files. For performance reasons and the fact that Corral is a shared resource with limited inodes, raw data was aggregated, ordered by calendar date and grouped by sensor stream to reduce the underlying file count by several orders of magnitude. The various data streams differ significantly in terms of their density and are a direct result of sampling schedules. For example, for a single participant with an iOS device, 116M accelerometer samples were collected and only 96K power/screen state records over the same 10 month period.

The Influx CLI was used to transform the aggregated CSV files into *line protocol* format [8] in which each line represents a data point with a measurement (data stream), field set, tag set, and timestamp. This required for each data stream the creation of annotated CSV headers that specified the mapping of column names and data types to field and tag sets as well as the date-time column. Although the Influx CLI supports writing directly to an InfluxDB server, we elected to use open-source utility scripts, published on the InfluxDB Github and provided as a benchmarking suite [7], that support multi-threaded, batch uploads and performance profiling. We modified the scripts, written in Go, to be compatible with InfluxDB v2.0. From a preliminary scaling study, average ingest rates of 334,474.0, 718,327.5, 878,267.8, 853,420.3, and 868227.3 values/sec were achieved with 2, 4, 6, 8, and 10 cores, respectively, on a single Stampede2 Skylake node for a single participant's aggregated accelerometer data. Note: for administrative and security purposes, data upload to an InfluxDB instance running on a Corral data server was performed on a Stampede2 compute node via an ssh tunnel through the login nodes, likely affecting write performance.

InfluxDB configuration: InfluxDB is optimized for writing and querying high-density time series data. An influxdb instance was deployed on a Corral server node with the majority of default settings preserved. To accommodate high write-rates of data that span a long period of time, we found it necessary to increase the storage-cache-max-memory-size to 500MB (influxdb setting) and

double the number of files (~2000) that can be concurrently opened by altering the appropriate *ulimit* Linux setting.

Data exploration and visualization: For this use case, downstream analyses involve collaboration amongst a diverse, multidisciplinary team. Based on our initial experiences, the InfluxDB environment appears well suited to provide a shared platform to coordinate and enhance these efforts, especially as it pertains to prototyping methods to uncover digital biomarkers from smartphone data. The Data Explorer (Figure 3), part of the InfluxDB UI, provides an interface to graphically compose and execute complex queries of the data and visualize the results, thus removing the requirement that users be proficient programmers. We have found this very useful as we can interrogate and explore the data in real-time as a team. Figure 3 highlights a particular use case in which we were interested in identifying patterns of missing GPS data in two pilot study participants. The plot shows the number of GPS records (y-axis) in one hour, non-overlapping windows over a 1.5 month period (x-axis). A significant drop in GPS data is readily identified for the participant represented by the yellow trace as well as a periodic decrease in samples for both participants, warranting further investigation. The underlying Flux query was composed with the UI and follows:

```
from(bucket: "beiwe-tacc-pilot")
|> range(start: 2019-09-11, stop: 2019-11-02)
|> filter(fn:(r) => r["id"]=="p1" or r["id"]=="p2")
|> filter(fn:(r) => r["_measurement"]=="gps")
|> filter(fn:(r) => r["_field"]=="latitude")
|> aggregateWindow(every: 1h, fn:count)
```

Flux is an open-source, standalone scripting and query language, packaged as part of the InfluxDB platform. There are currently 11 client libraries under active development that integrate with InfluxDB v2.0 API. For prototyping biomarker discovery methods, we have begun to use the *influxdb-client-python* client library [6] to query and aggregate data, the results of which will be processed and analyzed with forecasting and machine learning Python packages within Jupyter notebooks. These methods will then be used to analyze and process data at scale via batch HPC workflows.

Figure 3: InfluxDB Data Explorer used to assess patterns in GPS sampling/drop-out patterns in two participants (green and yellow). Number of records are displayed on the y-axis and time on the x-axis

5 CONCLUSION

While this work is motivated by our current research aimed at discovering digital biomarkers of pregnancy, the resulting infrastructure is domain-agnostic and could be used to monitor/investigate other health conditions or perhaps the onset and population-spread of infectious diseases. Monitoring data collection at scale across many devices is critical for our work, and we highlighted a number of open-source tools that can be combined effectively to derive scalable monitoring and analysis solutions applicable to similar multi-sensor studies. Extracting and fusing features from multiple smartphone data streams is critical for us to discover digital biomarkers associated with birth outcomes. Although still in testing, we believe the InfluxDB platform is well-suited for these purposes to temporally-align and query heterogeneous, dense time series data as it is trivial and efficient with the UI and client libraries.

ACKNOWLEDGMENTS

The authors acknowledge the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing HPC resources that have contributed to the research results reported within this paper. This work is also supported in part by the National Science Foundation under Grant No. 1838901.

REFERENCES

[1] 2021. *Boto3 - The AWS SDK for Python.* https://github.com/boto/boto3
[2] 2021. *collectd – The system statistics collection daemon.* https://collectd.org
[3] 2021. *Grafana.* https://github.com/grafana/grafana
[4] 2021. *Graphite enterprise-scale monitoring tool.* https://graphite.readthedocs.io/en/stable/index.html
[5] 2021. *graphyte Python library.* https://github.com/benhoyt/graphyte
[6] 2021. *InfluxDB 2.0 Python Client Library.* https://github.com/influxdata/influxdb-client-python
[7] 2021. *influxdb-comparisons.* https://github.com/influxdata/influxdb-comparisons
[8] 2021. *InfluxDB Line Protocol.* https://docs.influxdata.com/influxdb/v2.0/reference/syntax/line-protocol/
[9] 2021. *InfluxDB v2.0.* https://docs.influxdata.com/influxdb/v2.0/
[10] Victor P. Cornet and Richard J. Holden. 2018. Systematic review of smartphone-based passive sensing for health and wellbeing. *Journal of Biomedical Informatics* 77 (Jan. 2018), 120–132. https://doi.org/10.1016/j.jbi.2017.12.008
[11] Denzil Ferreira, Vassilis Kostakos, and Anind K. Dey. 2015. AWARE: Mobile Context Instrumentation Framework. *Frontiers in ICT* 2 (2015). https://doi.org/10.3389/fict.2015.00006
[12] Syed Monowar Hossain et al. 2017. mCerebrum: A Mobile Sensing Software Platform for Development and Validation of Digital Biomarkers and Interventions. *Proceedings of the International Conference on Embedded Networked Sensor Systems. International Conference on Embedded Networked Sensor Systems* 2017 (Nov. 2017). https://doi.org/10.1145/3131672.3131694
[13] Mathew V. Kiang et al. 2021. Sociodemographic Characteristics of Missing Data in Digital Phenotyping. *medRxiv* (Jan. 2021), 2020.12.29.20249002. https://doi.org/10.1101/2020.12.29.20249002 Publisher: Cold Spring Harbor Laboratory Press.
[14] Sumit Majumder and M. Jamal Deen. 2019. Smartphone Sensors for Health Monitoring and Diagnosis. *Sensors (Basel, Switzerland)* 19, 9 (May 2019). https://doi.org/10.3390/s19092164
[15] Nikhil Panda et al. 2020. Using Smartphones to Capture Novel Recovery Metrics After Cancer Surgery. *JAMA Surgery* 155, 2 (Feb. 2020), 123–129. https://doi.org/10.1001/jamasurg.2019.4702
[16] John Torous, Mathew V Kiang, Jeanette Lorme, and Jukka-Pekka Onnela. 2016. New Tools for New Research in Psychiatry: A Scalable and Customizable Platform to Empower Data Driven Smartphone Research. *JMIR Mental Health* 3, 2 (May 2016). https://doi.org/10.2196/mental.5165
[17] Alina Trifan, Maryse Oliveira, and José Luís Oliveira. 2019. Passive Sensing of Health Outcomes Through Smartphones: Systematic Review of Current Solutions and Possible Limitations. *JMIR mHealth and uHealth* 7, 8 (Aug. 2019). https://doi.org/10.2196/12649