University of Missouri

TigerAware Microservices: A Modern Backend for Improved Platform Scalability and Consistency

Connor Rowland

Department of Electrical Engineering and Computer Science

Prof. Yi Shang, Advisor

Contents

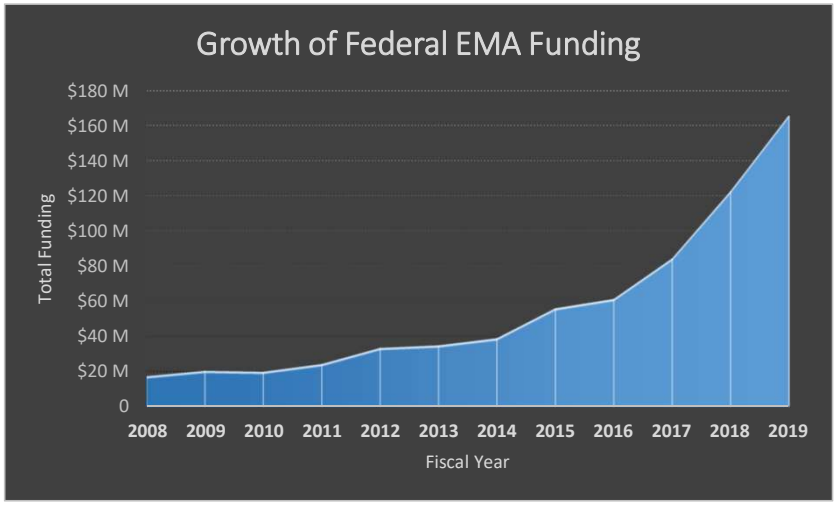
- Introduction
- Background and Related Works
- System Improvements
- Conclusion

Contents

- Introduction
  - Motivation
  - Existing Platform
  - Proposed Improvements
- Background and Related Works
- System Improvements
- Conclusion

Ecological Momentary Assessment

- Popular in psychology and medicine
- Repeated sampling in real-time
- Increased popularity due to proliferation of smartphones
  - 81% of all Americans own smartphones
  - 96% of young adults 18-29
- Smartphones allow for easy EMA data collection



Challenges in EMA

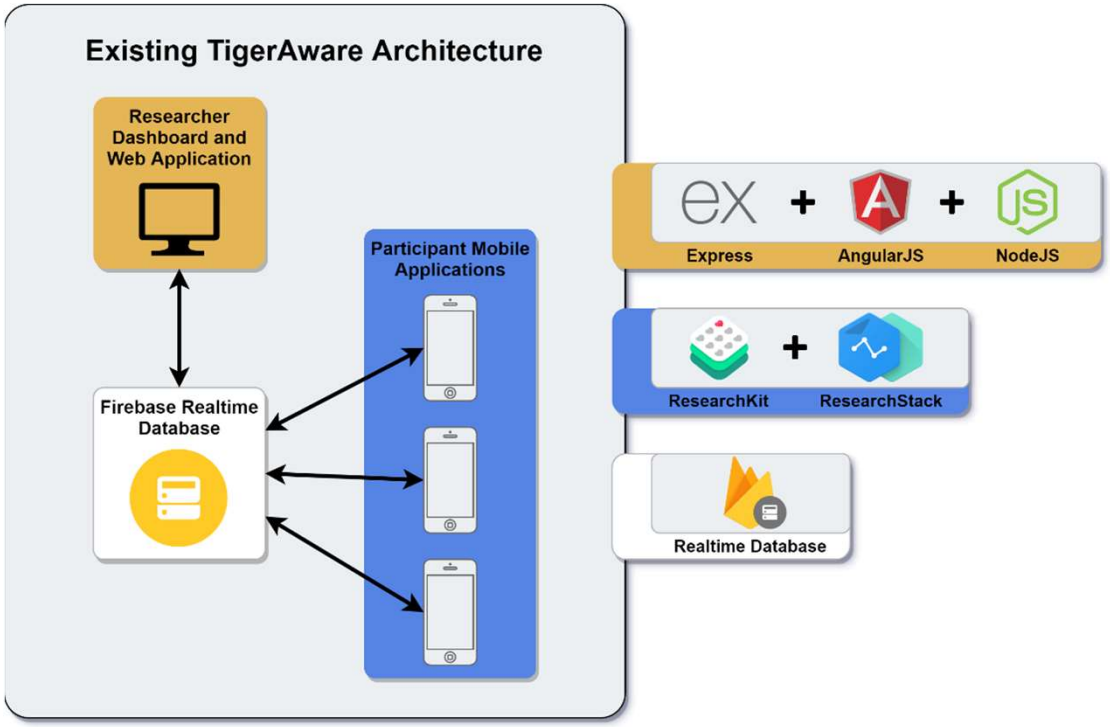
- Hard to develop application for a single EMA study
  - Expensive to contract development work
  - Long time frame
- Wide range of study needs
  - Need flexible, extensible platform

TigerAware

- Create an EMA platform extensible enough for a wide range of studies
- Built-in common features
  - Question types
  - Notification structure
  - Study administration
- Modular design to easily add or extend functionality for specific studies

Existing TigerAware Architecture

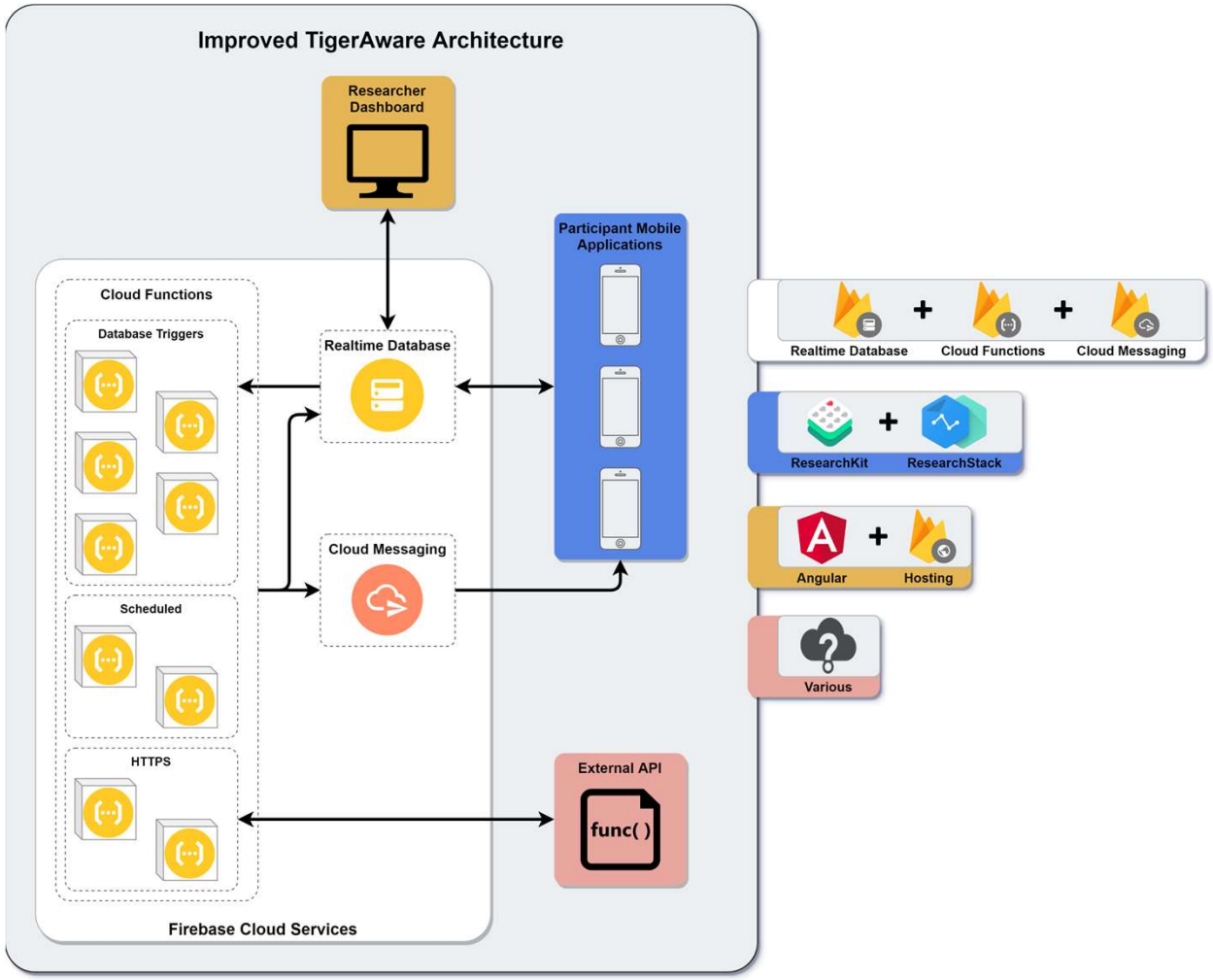
- Native Mobile Applications
  - ResearchKit (iOS)
  - ResearchStack (Android)
- Web Dashboard
  - FEAN stack
- Firebase Realtime Database



Proposed Changes

- Change web dashboard to lightweight, easily hostable setup
  - Migrate to Angular
  - Utilize Firebase Hosting
- Microservices
  - Move Express/NodeJS business logic to microservices
  - Utilize Firebase Cloud Functions
- Cloud Messaging
  - Implement hybrid notification scheme

Improved Architecture



Contents

- Introduction
- Background and Related Works
  - MUDICL and TigerAware
  - Microservices
- System Improvements
- Conclusion

Early EMA Studies

S. Ravi, "Development of a Wireless Body Area Sensing System for Alcohol Craving Study," University of Missouri, Department of Computer Science, 2013.

- Created a mobile application for alcohol craving EMA study
- Set groundwork for participant-facing mobile applications

Enhancing EMA Capabilities

D. P. Srivatsav, "MTD: Mood Toolkit Dashboard," University of Missouri, Department of Computer Science, 2017.

- Created integrated dashboard for researcher engagement
- Early version of TigerAware architecture

All-in-One EMA Functionality

W. Morrison, L. Guerdan, J. Kanugo, T. Trull and Y. Shang, "TigerAware: An Innovative Mobile Survey and Sensor Data Collection and Analytics System," in *Third International Conference on Data Science in Cyberspace*, Guangzhou, China, 2018.

- Seminal paper on the TigerAware platform

Other TigerAware Works

J. Kanugo, "TigerAware Dashboard: An Improved Survey Generation and Response Visualization Dashboard," University of Missouri, Department of Computer Science, 2018.

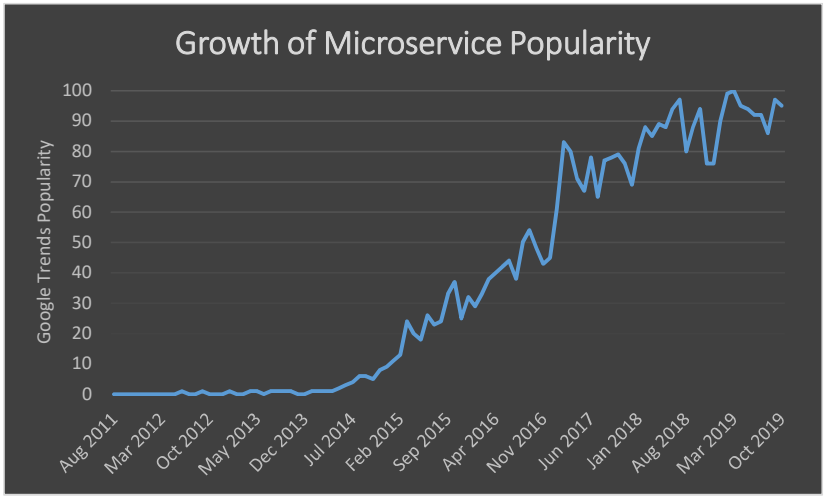
- Previous TigerAware dashboard implementation

W. Xia, "TigerAware Android: An Improved Survey and Notification System," University of Missouri, Computer Science, 2019.

- Improvements to TigerAware mobile applications

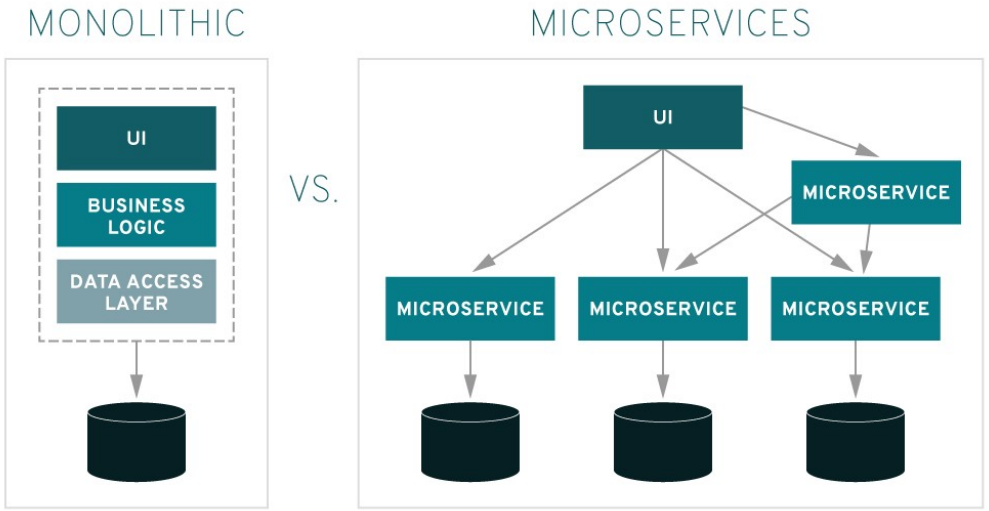
What are Microservices?

- Pattern for system backend
- Utilizes numerous small, standalone services for business logic
- Massive spike in popularity recently
  - Amazon, Uber, Netflix



Why Microservices?

- System Resiliency
  - Failures are easy to identify
- System Scalability
  - Individually load balanced
- Ease of Development
  - Don’t require platform-wide knowledge

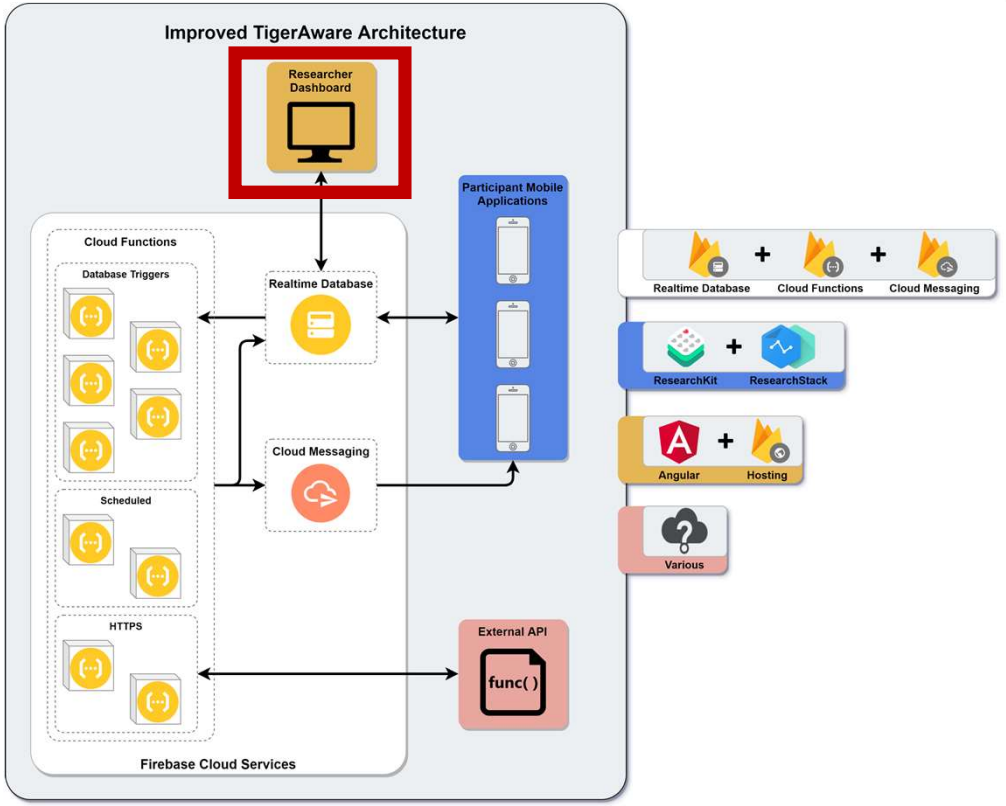


Contents

- Introduction
- Background and Related Works
- System Improvements
  - Dashboard Changes
  - Microservice Implementation
  - Cloud Messaging
  - Server Schedule Creation
- Conclusion

Contents

- Introduction
- Background and Related Works
- System Improvements
  - Dashboard Changes
  - Microservice Implementation
  - Cloud Messaging
  - Server Schedule Creation
- Conclusion



Angular Frontend Framework

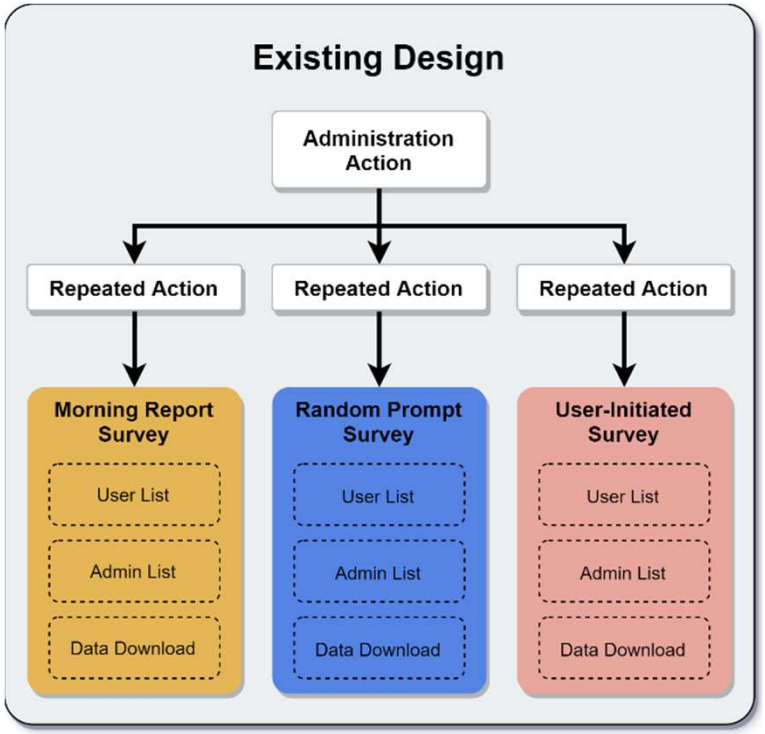
- Successor to original AngularJS framework
- Released in late 2016
- Improved Performance
  - One-way change detection for bindings through Observables
  - Up to 5-10 *times* faster than old framework
- Typescript Support
  - Superset of standard JavaScript
  - Statically-typed, compiled

Firebase Hosting

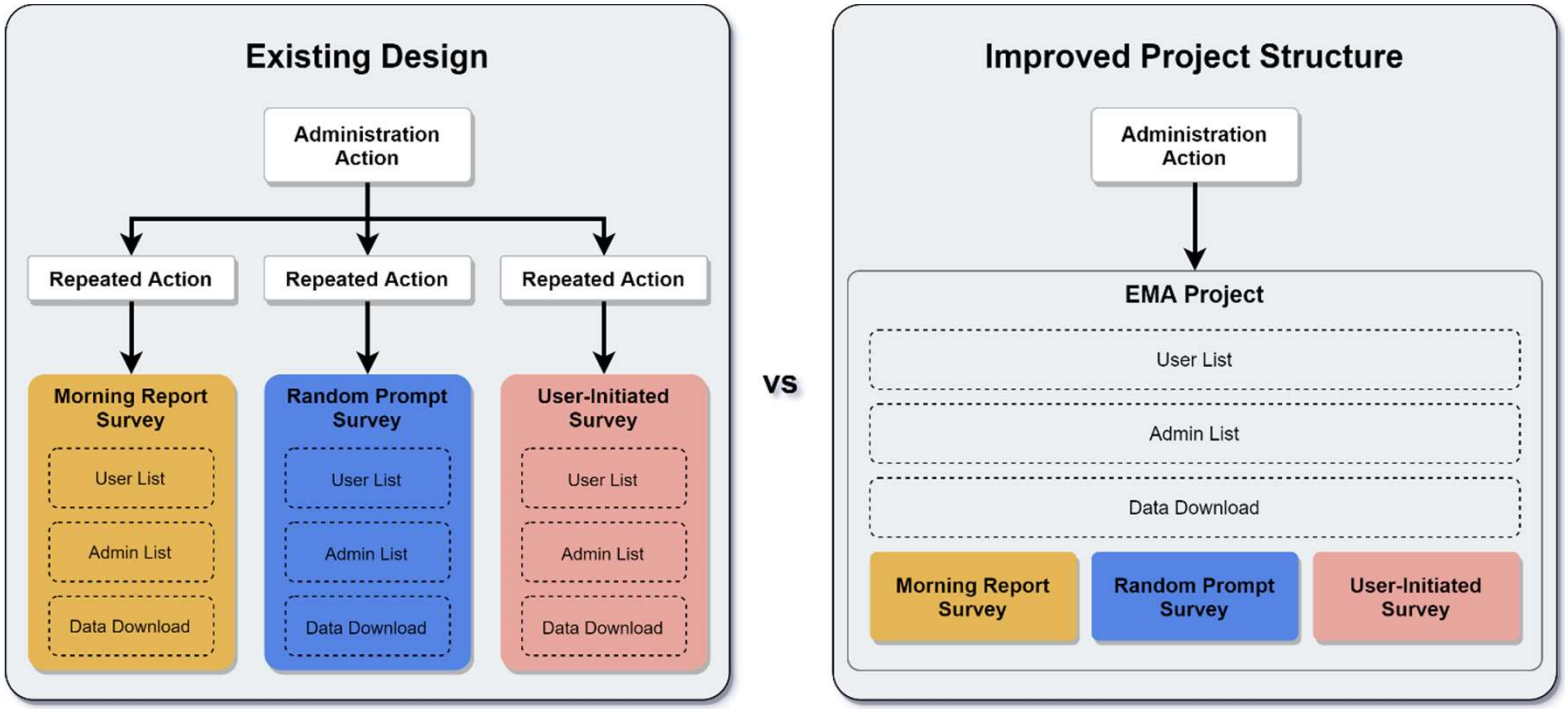
- Existing dashboards hosted on Heroku
  - Automated deployments difficult for many apps
  - Expensive – current TigerAware deployments $3,000+ annually on standard dynos
- Firebase Hosting
  - Easy to manage many applications and deployments
  - Generous free tier

Improved Project Structure

- Many protocols desire multiple surveys throughout the day
- Existing architecture requires each to be managed individually
- Data not shared between surveys
  - User lists
  - Admin lists
  - Data downloads



Improved Project Structure



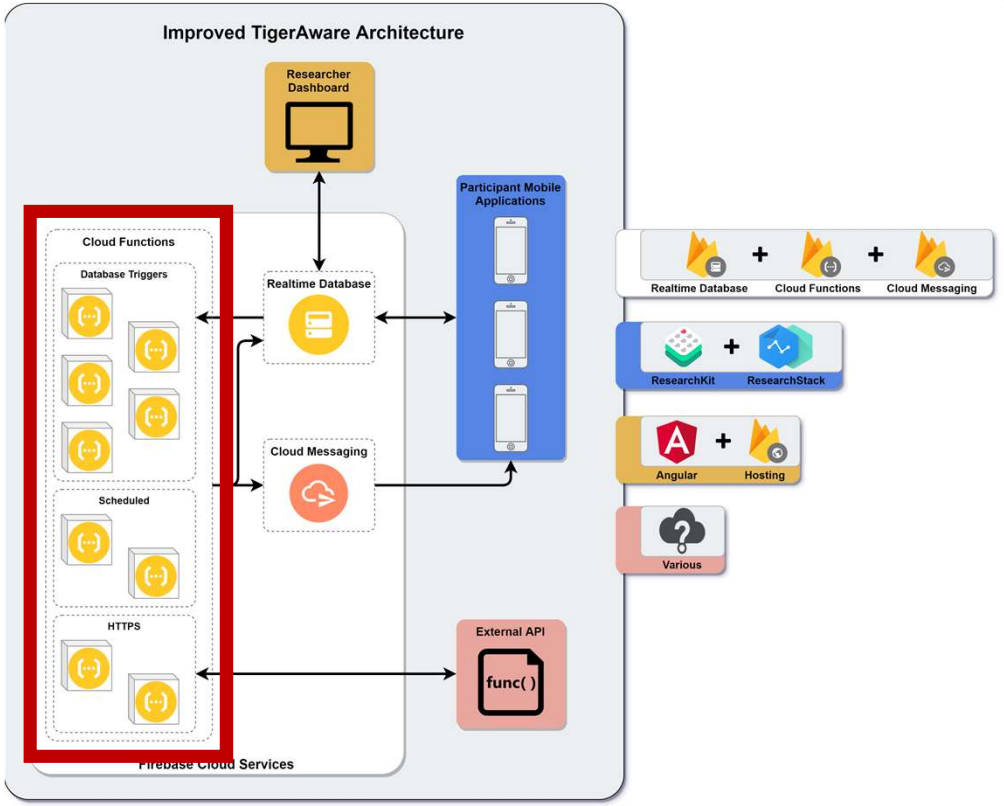
Dashboard Performance Comparison

	Login Page Load Time	Overview Page Load Time	Relative Performance Increase
Existing TigerAware Dashboard	8.92 sec	8.83 sec	-
Improved TigerAware Dashboard	1.80 sec	2.39 sec	**423.6%**

- Improvements provided by Angular, hosting, and improved querying
- Experiment with cleared cache and throttled connection

Contents

- Introduction
- Background and Related Works
- System Improvements
  - Dashboard Changes
  - Microservice Implementation
  - Cloud Messaging
  - Server Schedule Creation
- Conclusion

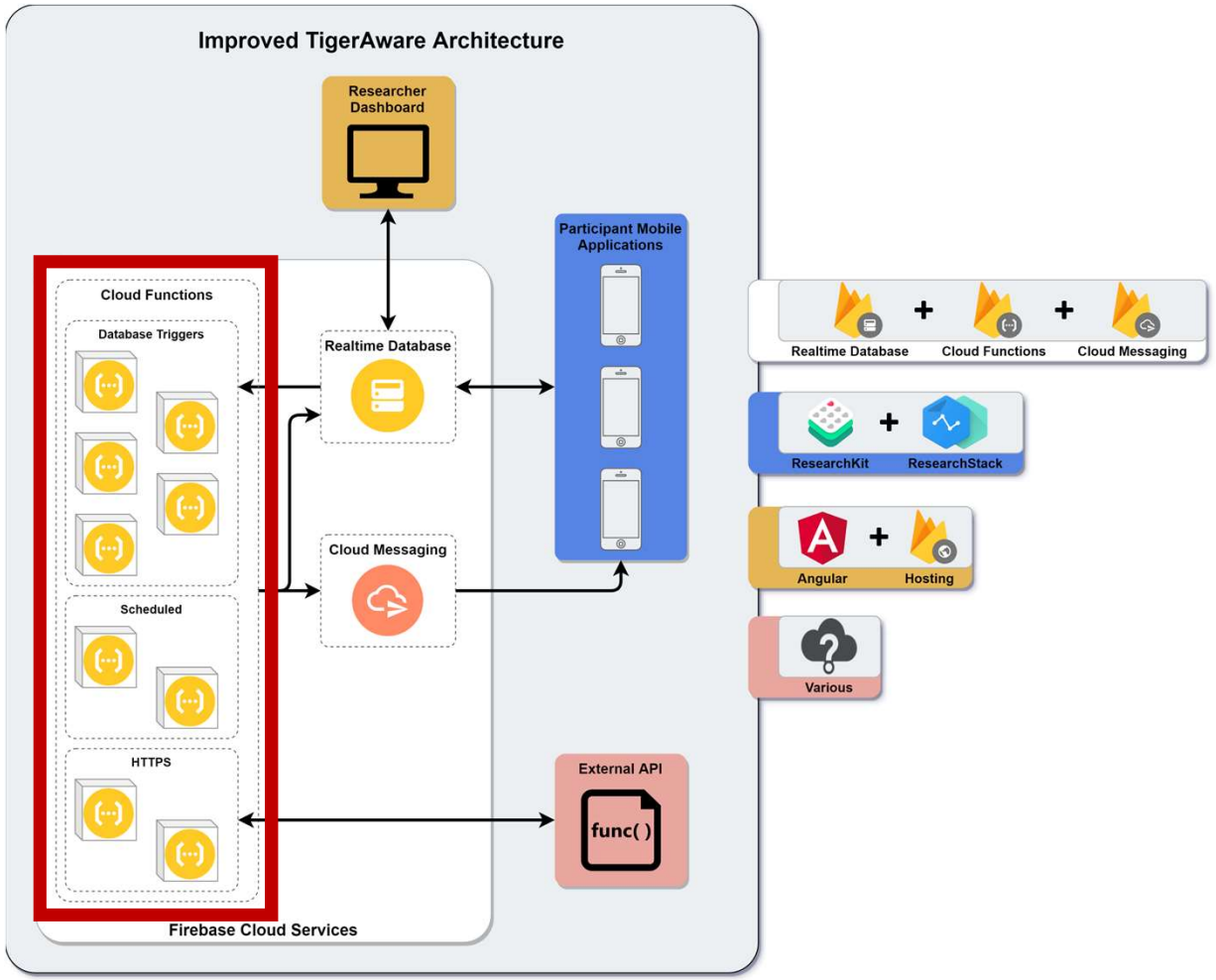


Firebase Cloud Functions

- Allow remote invocation of code in the cloud
- Automatically scale to meet demand
- Very cheap
  - 125k invocations free each month
- Automatically triggered from database changes
- Shared authorization environment
  - Identical functions easily deployed to different systems

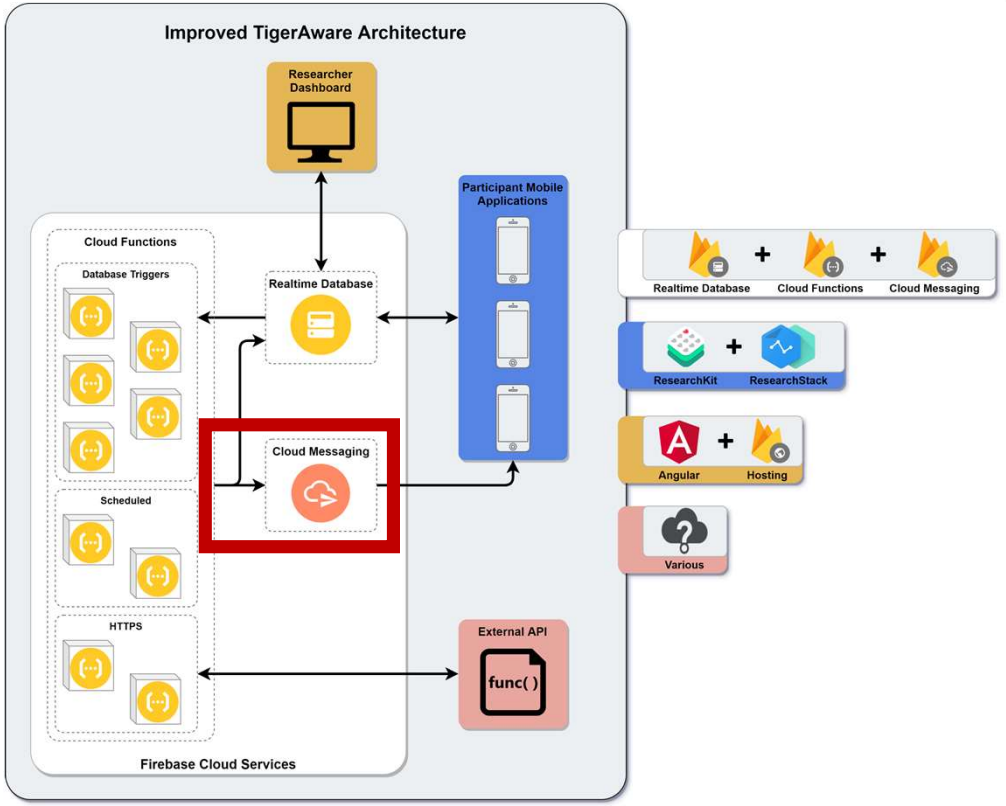
TigerAware Microservices

- Firebase Write
- Firebase Create
- Firebase Update
- Firebase Delete
- Scheduled
- HTTPS



Contents

- Introduction
- Background and Related Works
- System Improvements
  - Dashboard Changes
  - Microservice Implementation
  - Cloud Messaging
  - Server Schedule Creation
- Conclusion



EMA Notifications

- Notifications serve as backbone of EMA studies
- Participants receive notifications to know when to interact with the protocol
  - Scheduled reminders
  - Random prompts
- TigerAware needs consistent, reliable notification delivery

Local-Only Notification Delivery

+ High rate of deliverability
+ Support fully-offline participation
– Scheduling done on-device
– iOS maximum 64 notifications per app
  - Not uncommon for protocol to have 15 or more notifications per day
  - TigerAware limits to 30 notifications per day
  - Notifications stop after 2 days

Remote-Only Notification Delivery

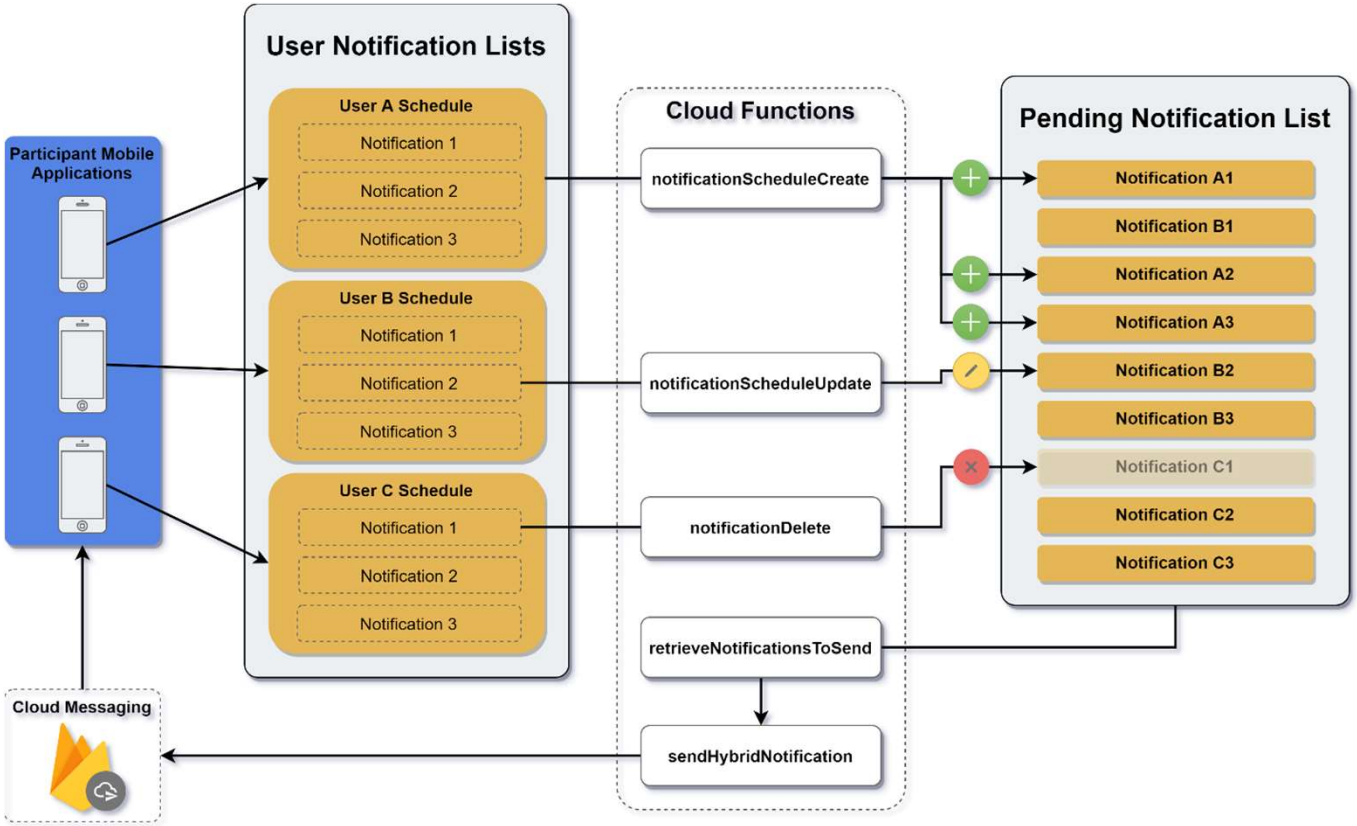
+ Easy to control and modify
+ Increased transparency
+ Avoids 2-day limit
– Requires consistent strong network connection
  - EMA protocols require immediate delivery (no retry)
– Device-specific delivery factors
  - Battery saving
  - Low priority notifications
  - Application limits

Hybrid Notification Delivery

- Combine the strengths of local and remote delivery
- Schedule as many notifications as possible on-device
  - Maybe more than 2 days
- Remaining notifications delivered remotely

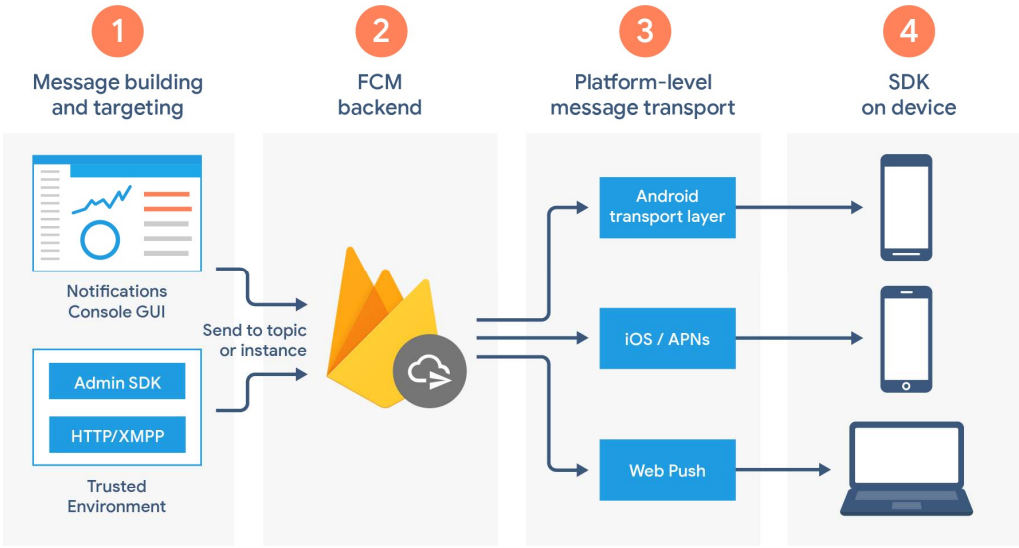
	High Deliverability (first 2 days)	Supports Offline Participants	Avoids 2-Day Notification Limit	Allows Off-Device Scheduling
Local-Only Notifications	✓	✓	✗	✗
Remote-Only Notifications	✗	✗	✓	✓
Hybrid Notifications	✓	✓	✓	✓

Cloud Messaging Microservices



Firebase Cloud Messaging

- Delivers cross-platform
- Allows volatile notifications
- Easy to send using FCM token
- Free



Contents

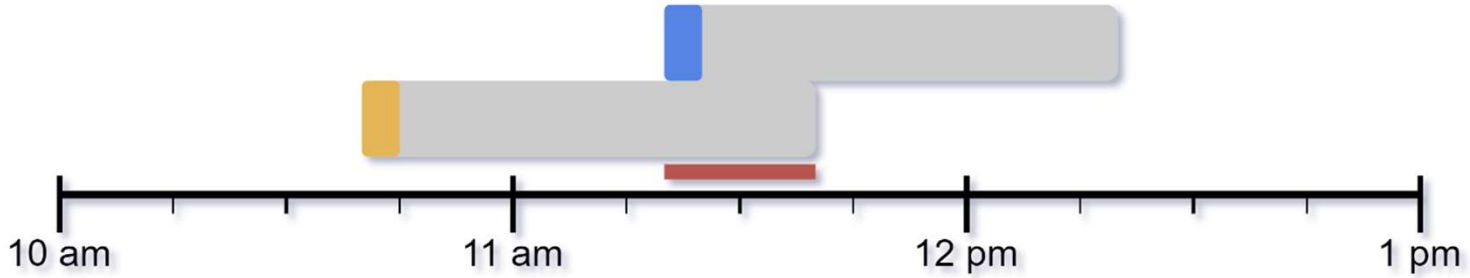
- Introduction
- Background and Related Works
- System Improvements
  - Dashboard Changes
  - Microservice Implementation
  - Cloud Messaging
  - Server Schedule Creation
- Conclusion

Benefits of Scheduling in Microservice

- Unifies scheduling between iOS and Android
- New changes only need to be implemented once
- Improved testability
  - Speeds up deployment of new features
  - Schedule testing no longer observational

Compliance Duration

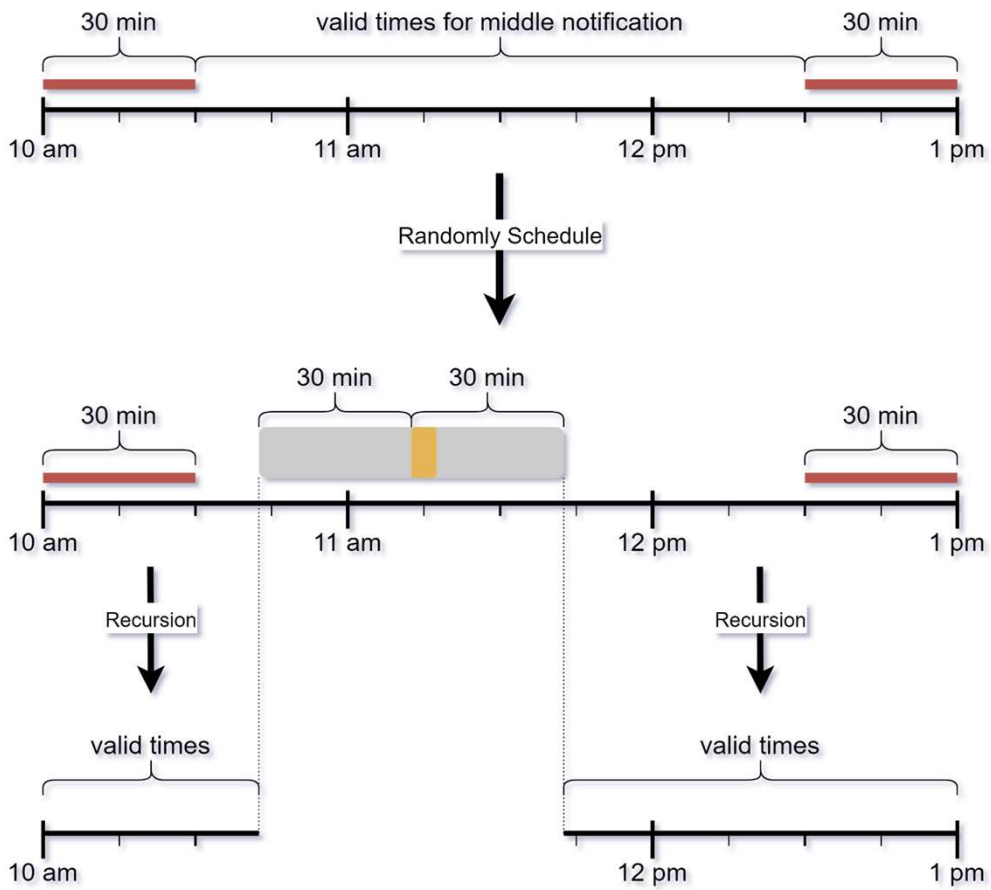
- Period of time users can respond to surveys
- Notifications should not have overlapping compliance periods
  - Ambiguous which notification a participant is responding to
  - Can lead to missed notification



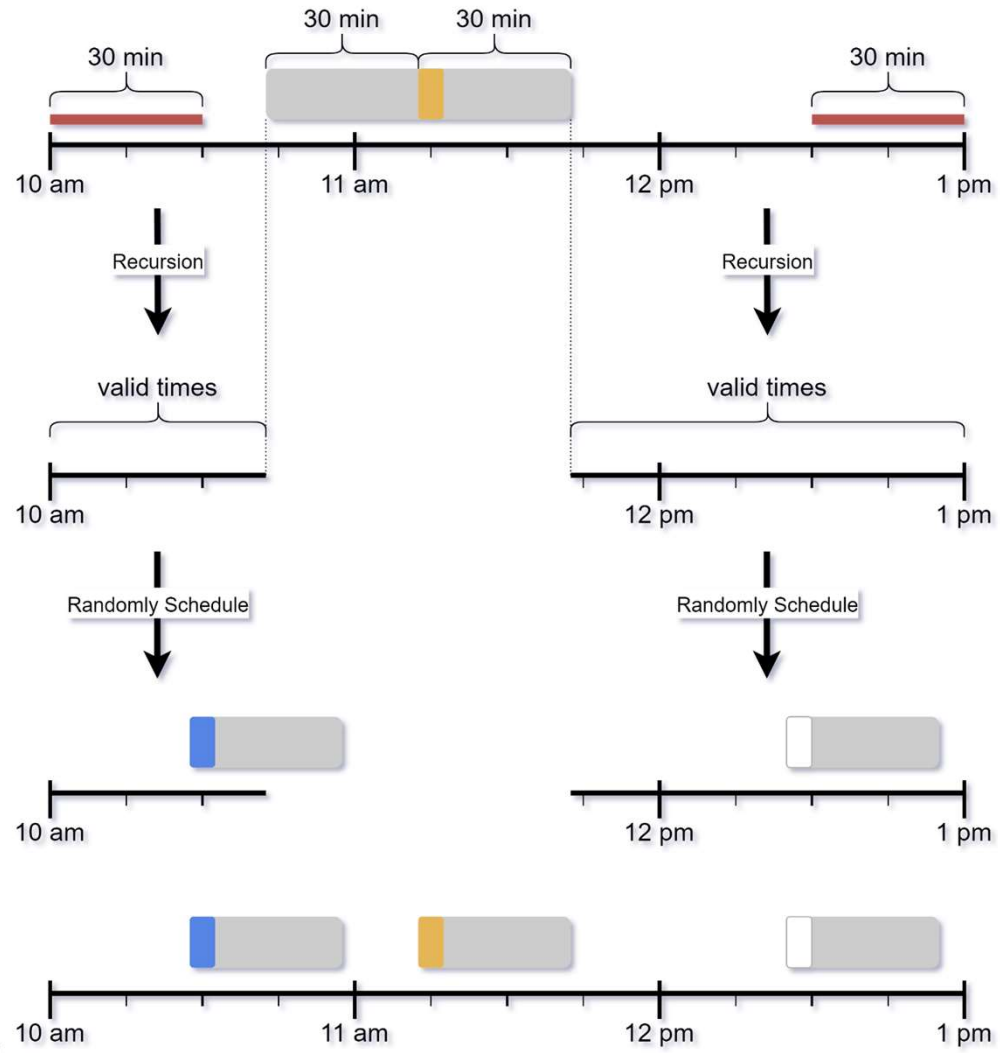
Random Scheduling Algorithm

- Select middle notification
- Temporarily schedule all preceding and succeeding notifications
  - Schedule as early and late as possible, respectively
  - Separate each notifications by a *c*-minute gap, where *c* is compliance duration
- Randomly schedule notification in remaining valid times
- Recursively schedule disjoint windows before and after notification

Random Notification Scheduling



Random Notification Scheduling



Contents

- Introduction
- Background and Related Works
- System Improvements
- Conclusion

Conclusion

- Greatly improved dashboard performance
- Created scalable, flexible microservice backend
- Improved user engagement through cloud messaging
- Implemented consistent notification scheduling in the cloud

Future Work

- Question-based survey creation
  - Lowers learning curve for researchers new to the platform
- Action-based intervention
  - Further extend the ways researchers can interact with participants
- True continuous integration pipeline for TigerAware microservices
  - Fully-integrated testing for all functions


Thank You!
University of Missouri