Аникевич Ю.В., Бахмуров А.Г., Чайчиц Д.А.

Разработка и реализация платформы для сбора и обработки физиологических данных о человеке

Введение

С каждым днем все известнее становится такое направление в здравоохранении, как (IoMT, Internet of Medical Things). Данная формулировка обозначает концепцию сети, которая позволяет объединять устройства и приборы, способные отслеживать состояние человеческого организма, окружающую его среду и влиять на профилактический, лечебный и реабилитационный процессы.

На сегодняшний день проблема мониторинга состояния человека вне медицинских учреждений чрезвычайно актуальна. Данное направление существует с целью получения детальной информации об организме человека, на основании которой можно принимать обоснованные решения, предотвращающие возникновение и развитие заболеваний. Современная концепция мониторинга предлагает проводить непрерывный контроль за состоянием человека, и, с помощью оценки диагностических показателей, выявлять случаи отклонения их от нормы.

Персональные мониторные устройства способны выявить нарушения в работе систем организма на более ранней стадии, а также могут послужить дополнительным источником информации для специалистов. Также, такая система позволяет оперативно предупредить критическое состояние пациента, что даст возможность вовремя провести необходимые мероприятия для его стабилизации.

1. Анализ предметной области и актуальность работы

К настоящему времени сформирован обширный рынок портативных носимых устройств и систем для регистрации физиологических параметров человека (например, умные часы, фитнес-браслеты). Для большинства из них производитель предлагает свои проприетарные решения для обработки и хранения собираемых данных. Также известны приложения, например, Apple Health, Google Fit, которые работают с датчиками разных производителей, но также являются

проприетарными и предусмартривают хранение данных на серверах производителя. Следовательно, из них невозможно получить необработанные данные, и персональные данные пользователей хранятся “на стороне”. Актуально создание платформы сбора и обработки медицинской телеметрии с открытым исходным кодом, так как оно позволит:

- реализовывать собственные методы обработки телеметрии, как в исследовательских целях, так и для обучения студентов, например, по специальности “медицинская кибернетика”;
- устранить необходимость хранения персональных данных на зарубежных серверах;
- устранить зависимость от ошибок сторонних разработчиков; так, например, была описана ошибка в Google Fit;
- создать задел для отечественного промышленного средства сбора и обработки медицинской телеметрии.

Однако в большинстве своем они не являются open-source проектами, и данные из них получить либо невозможно, либо они выдаются уже в обработанном виде. Таким образом, создание открытой платформы сбора и обработки физиологических данных является актуальной задачей на сегодняшний день.

2. Архитектура платформы

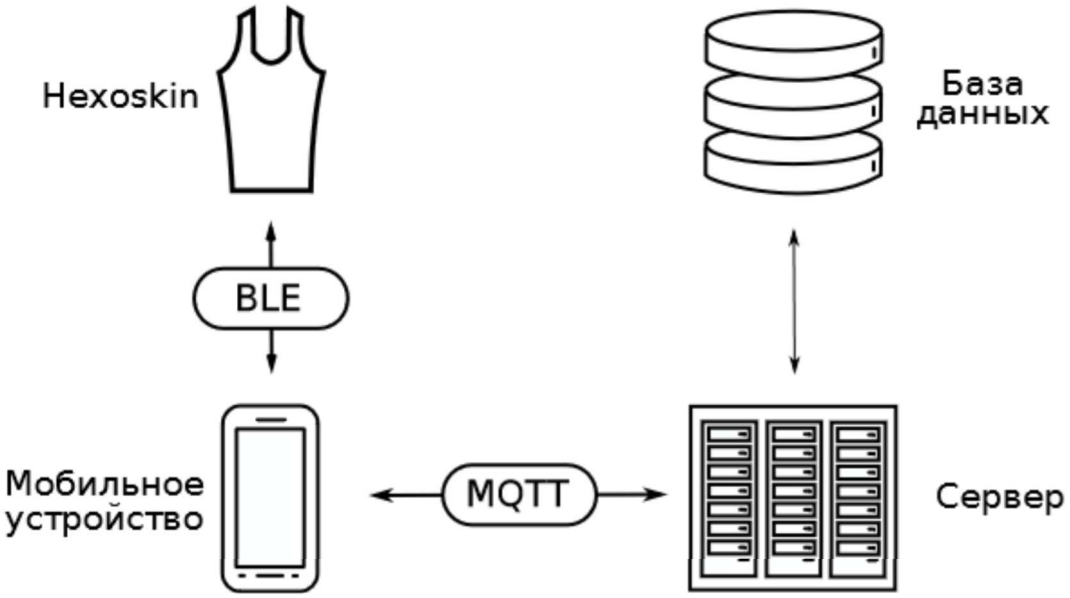

Рис. 1: Архитектура платформы

Архитектура платформы показана на рисунке 1:

1. Физиологические показатели пользователя считываются с помощью датчиков умной одежды Hexoskin, после чего отправляются на мобильный телефон на операционной системе Android по протоколу Bluetooth Low Energy (BLE).

2. Мобильный телефон принимает считанные показатели и отправляет их на сервер по протоколу MQTT.

3. Сервер записывает полученные данные в базу данных, откуда их можно взять для медицинской обработки.

3. Обоснование выбора компонентов

В качестве носимого устройства выступает умный жилет Hexoskin. Это устройство позволяет считывать с человека следующие показатели:

- частоту сердцебиения;
- вариабельность сердечного ритма;
- число шагов;

- частота дыхания;
- VO2 max;
- объем легких;
- уровень активности;
- число шагов в минуту(каденция);

Выбор обусловлен как широкими возможностями жилета по сбору данных, так и широким его применением для медицинских исследований. Жилет подключается к смартфону посредством протокола Bluetooth Low Energy.

3.1 Выбор СУБД для хранения телеметрии на сервере

Данные, получаемые с носимого устройства, представляют собой временные ряды. Под временным рядом понимается совокупность значений какого-либо показателя, собранного в различные моменты времени. Измерения, составляющие временной ряд, упорядочены по временной шкале, что показывает историю изменения данных во времени и может быть очень полезным для медицинской диагностики.

Для работы с временными рядами существуют специализированные СУБД, которые более удобны для пользователя, и обеспечивают большую скорость записи и более высокую производительность запросов, несмотря на большой объем данных, которые они организуют. В некоторых случаях СУБД временных рядов выполняют те же функции, что и обычные СУБД, однако попытка использовать реляционную базу данных для данных временных рядов приведет к снижению производительности.

Запросы к базе данных временных рядов аналогичны запросам в других типах баз данных, но вместо поиска по значениям пользователи могут осуществлять поиск по прошедшему периоду времени, диапазону дат или конкретному моменту времени, когда произошло событие. В рамках данной работы для хранения телеметрии на сервере были рассмотрены системы управления базами данных, оптимизированные для хранения и обработки временных рядов.

Сценарий использования БД и обусловленные им требования к СУБД

Для выбора СУБД рассматривались такие критерии, как открытость исходного кода, долговременное хранение, масштабируемость

и наименьшая гранулярность.

Под масштабируемостью понимается возможность увеличить объем хранилища или производительность за счет добавления дополнительных узлов.

Долговременное хранение необходимо для больших объемов данных, поскольку не все СУБД могут хранить все значения в полном разрешении в течение длительного периода времени. Старые данные могут удаляться или храниться в агрегированном состоянии (например, хранение среднего значения за минуту вместо значений для каждой миллисекунды).

Гранулярность описывает наименьшее возможное расстояние между двумя временными метками. При вставке данных в базу данных степень детализации входных данных может быть выше, чем степень детализации хранилища, для которой СУБД гарантирует безопасное хранение. Некоторые СУБД принимают данные с меньшей степенью детализации, чем они могут хранить, что приводит к агрегированию или потере данных.

База данных представляет собой набор таблиц, где каждая таблица отведена под конкретного пользователя. В таблицы будет производиться запись семи измерений, в соответствии параметрами, получаемыми с датчиков Hexoskin.

Согласно информации о датчиках Hexoskin, самая высокая частота получения данных имеет значение 256 Гц. Строго говоря, для выбора СУБД важно, чтобы она имела возможность сохранять данные с разрешением в 4 мс. Предполагаем, что гранулярность принимает значения 1 мс, 10 мс, 100 мс и т.д. Таким образом, необходимая гранулярность имеет значение 1 мс.

Необходимость в долговременном хранении обусловлена возможностью хранить данные мониторинга за длительный промежуток времени, что позволяет получить более полную картину о состоянии человека.

Высоким приоритетом при выборе обладает такой критерий, как время выполнения запросов, то есть важно, чтобы чтение из базы данных происходило достаточно быстро. Более низким – время записи в базу данных и степень сжатия записанных данных на диске.

Выбор СУБД для обзора и результаты обзора

Для обзора были взяты СУБД временных рядов, которые занимают первые десять позиций в рейтинге DB-Engines СУБД временных рядов. DB-Engines [10] - это независимый веб-сайт, который ранжирует системы управления базами данных на основе популярности в поисковых системах, упоминаний в социальных сетях, количества

предложений о работе и частоты технических обсуждений. Также в список обозреваемых СУБД отдельно была включена ClickHouse, разрабатываемая компанией Яндекс, в виду хороших результатов, показанных в статье [9], где сравнивается производительность СУБД ClickHouse, InfluxDB и TimescaleDB. Таким образом, в список обозреваемых систем управления базами данных были включены СУБД InfluxDB, Kdb+, Prometheus, Graphite, RRDtool, Druid, OpenTSDB, TimescaleDB, FaunaDB, KairosDB, ClickHouse. Одним из критериев выбора СУБД является открытость исходного кода, поэтому далее такие решения, как Kdb+ и FaunaDB не рассматривались. Результаты сравнения по оставшимся критериям приведены в таблице 1. По результатам сравнения, всем перечисленным критериям удовлетворяют InfluxDB, ClickHouse и Druid.

	Масштабируемость	Долговременное хранение	Наименьшая гранулярность для хранения	Наименьшая гранулярность для безопасного хранения
InfluxDB	+	+	1 мс	1 мс
Prometheus	+	-	1 мс	1 мс
Graphite	+	+	1000 мс	1000 мс
RRDtool	-	-	1000 мс	1000 мс
Druid	+	+	1 мс	1 мс
OpenTSDB	+	-	1 мс	>1 мс (1000 мс)
TimescaleDB	-	+	1 мс	1 мс
KairosDB	+	-	1 мс	1 мс
ClickHouse	+	+	1 мс	1 мс

Таблица 1: Сравнение СУБД

Из статьи [14] можно сделать вывод, что Druid больше подходит для сценариев, когда имеется большой кластер с большим количеством таблиц и данных. Таблицы и наборы данных периодически появляются в кластере, анализируются и удаляются из него в отличие от ClickHouse, где таблицы и данные находятся в кластере перманентно. Поэтому Druid был исключен из дальнейшего сравнения.

Экспериментальное сравнение СУБД

Для сравнения по производительности СУБД InfluxDB v1.8.0 и ClickHouse v19.3.3 были выбраны следующие критерии: время загрузки данных, степень сжатия данных на диске и время выполнения запросов.

InfluxDB [8] - это нереляционная система управления базами данных временных рядов, разработанная компанией InfluxData, которая имеет открытый исходный код с дополнительными компонентами с закрытыми исходными ко- дами. Она написана на языке программирования Go и оптимизирована для обработки временных рядов. Поддерживает SQL-подобный язык запросов. Время является самой важной концепцией в InfluxDB. Столбец времени включен в каждую базу данных InfluxDB и содержит дискретные временные метки, которые связаны с конкретными данными.

ClickHouse [7] – это аналитическая колоночная база данных с открытым исходным кодом, разработанная компанией Яндекс для OLAP сценариев работы. Язык запросов ClickHouse представляет собой диалект SQL.

Колоночные базы данных хранят записи в блоках, сгруппированных по столбцам, а не по строкам. Поскольку такая БД не загружает данные для столбцов, которых нет в запросе, она тратит меньше времени на чтение данных при выполнении запросов. Потому колоночные базы данных могут вычислять и возвращать результаты для определенных рабочих нагрузок, таких как OLAP, намного быстрее, чем традиционные строчные системы.

Сервер тестового стенда имел следующую конфигурацию:

- Ubuntu 18.04.3
- Intel(R) Xeon(R) CPU E5-2697 v4 @ 2.30GHz
- 3,9 Gb RAM
- L2 cache: 4096K
- L3 cache: 16384K

Для генерации тестового набора данных была воспроизведена непрерывная запись данных с самой высокой частотой из всех датчиков Hexoskin. Временной промежуток генерации составил около двух недель. В конечном итоге, набор данных состоит из 398.458.880 строк и содержит 16 показателей. Общий объем тестовых данных составил 55 Гигабайт.

Для сравнения выбранных СУБД по времени выполнения запросов было использовано 8 типов запросов, характерных для работы

с временными рядами. Предположим, что $v_1, v2, ..., vn$ – значения, соответствующие датчикам $1, 2, ..., n$. Также в нижеприведенных запросах func() представляет собой функцию агрегации, такую как avg, min и т.д.

1. Точечный запрос
   SELECT v1 ... FROM data WHERE time = ?
2. Запрос временного диапазона
   SELECT v1 ... FROM data WHERE time >? AND time <?
3. Запрос с фильтром значений
   SELECT v1 ... FROM data WHERE v op ? (op: <,>,=)
4. Запрос временного диапазона с фильтром значений
   SELECT v1 ... FROM data WHERE time > ? AND time < ? AND v1 op ? (op: <,>,=)
5. Запрос с лимитом строк
   SELECT v1 ... FROM data LIMIT ?
6. Запрос с функцией агрегации
   SELECT func(v) FROM data
7. Запрос временного диапазона с функцией агрегации
   SELECT func(v) FROM data WHERE time > ? AND time < ?
8. Агрегация + group by
   SELECT func(v) FROM data GROUP BY time(?)

Результаты сравнения СУБД

Загрузка данных

Для загрузки тестового набора данных в базы данных был использован интерфейс командной строки и утилита time для измерения времени загрузки. Загрузка данных в ClickHouse заняла 712 секунд, а в InfluxDB – 2367 секунд, что в 3,3 медленнее, чем время загрузки данных в ClickHouse.

Степень сжатия данных

Пространство, занимаемое тестовым набором данных в случае ClickHouse, составляет 4.7 Гб, а в случае InfluxDB - 3.1 Гб. Таким образом, коэффициент сжатия равен 1:12 для ClickHouse и 1:18 для InfluxDB.

Время выполнения запросов

Для измерения производительности выполнения запросов было использовано 8 типов запросов, приведенных выше. Каждый запрос

был выполнен 5 раз, после чего для него было вычислено среднее значение. Среднее время ответа на запросы для ClickHouse составляет около 3,5 секунд, а для InfluxDB – около 107 секунд. В среднем, время выполнения запроса в ClickHouse в 77 раз быстрее, чем в InfluxDB. По двум из трех критериев ClickHouse показал лучшие результаты. Время загрузки тестового набора в ClickHouse заняло в 3,3 раза меньше времени, чем в InfluxDB и также выполнение тестового набора запросов в среднем вышло в 77 раз быстрее, чем в InfluxDB. Таким образом, по результатам сравнения для использования в реализации была выбрана СУБД ClickHouse.

3.2 Выбор инструмента для промежуточного хранения данных на мобильном устройстве

В связи с возможными разрывами в подключении к сети Интернет немедленная отправка данных на сервер после получения их с жилета не является хорошей стратегией, так как велик риск потери данных. Поэтому после того, как данные с носимого устройства будут получены, необходимо их сохранять до момента отправки. В ОС Android на выбор доступны следующие механизмы хранения данных: внутреннее хранилище, внешнее хранилище, SharedPreferences и база данных. Сравним эти механизмы.

Внутреннее хранилище

Каждое приложение Android имеет свой собственный внутренний каталог хранения, взаимодействующий с ним, в котором приложение может хранить текстовые и двоичные файлы. Файлы внутри этого каталога недоступны для пользователя или других приложений, установленных на устройстве пользователя. Они также автоматически удаляются, когда пользователь удаляет приложение. Таким образом, это обычный текстовый файл, в который можно записать данные и откуда их потом можно прочесть. Для чтения конкретного значения придется искать его в файле вручную. Объем вмещаемой информации будет ограничен объемом свободной памяти во внутреннем хранилище смартфона.

Внешнее хранилище

Поскольку внутреннее хранилище устройств Android обычно фиксировано и часто довольно ограничено, некоторые устройства Android поддерживают внешние носители данных, такие как съем-

ные microSD-карты. Это все еще будет файл, но в который можно будет записать гораздо больший объем данных.

SharedPreferences

Средства SharedPreferences позволяют приложениям сохранять настройки или другие данные в виде пар ключ-значение. Эти данные сохраняются даже когда пользователь закрывает приложение. Android хранит SharedPreferences в виде XML файла. SharedPreferences зависят от приложения, то есть сохраненные данные будут удалены из Android устройства после удаления приложения или после очистки данных приложения. Это также является файлом, но поиск и запись в нем потребуют меньше времени.

База данных

Если использовать базу данных, нужно быть уверенным, что каждое устройство будет её поддерживать. В этом случае в качестве единственного кандидата на выбор остается СУБД SQLite, так как она входит в состав ОС Android и уже есть у каждого в смартфоне. Размер индексируемой памяти для этой СУБД равен 140 Тб, а реальный размер будет ограничен размером внутренней памяти.

Исходя из полученных результатов оптимальным вариантом является использование СУБД SQLite. Данные, которые будут в ней храниться структурированы, что позволит производить предобработку показаний с помощью SQL запросов в будущем. Кроме того, при принудительном завершении приложения (или выключении смартфона), возможна потеря данных при работе с файлом, а СУБД позволяет восстановить данные при перезапуске благодаря журналу.

4. Описание работы мобильного приложения

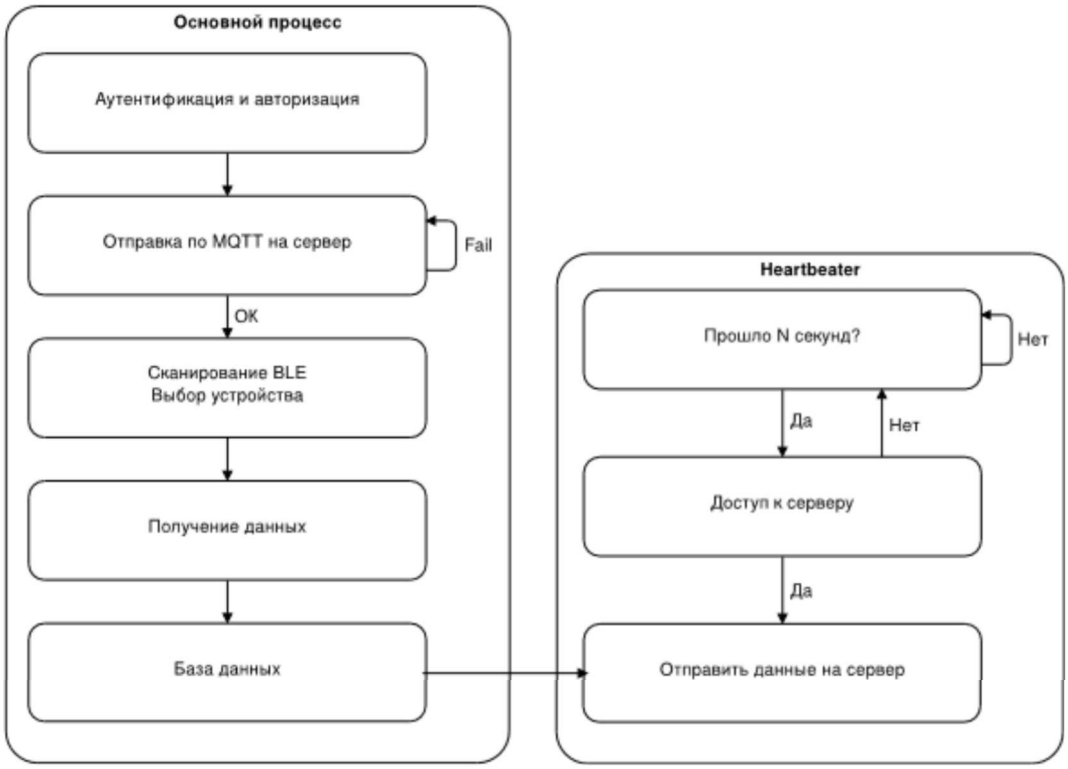

Рис. 2: Схема работы мобильного приложения

Схема работы мобильного приложения показана на рисунке 2. После запуска приложения перед пользователем открывается окно входа в систему. Получив логин и пароль, приложение использует их для создания соединения по MQTTS, после чего в случае ответа от сервера о правильности данных, брокер отправляет подтверждение устройству. В случае ошибки пользователю будет предложено ввести данные заново. После успешного входа приложение переходит в работы с носимым устройством.

Пользователь имеет возможность выбирать, к какому из доступных носимых устройств подключаться. После выбора такового приложение устанавливает соединение и начинает приём данных. Параллельно с этим начинает жизнь второй процесс - Heartbeater. Из название следует его роль: периодически с некоторым интервалом он оживает и проверяет есть ли соединение с сервером и возможность принятия данных. В случае наличия он отправляет содержимое базы данных и очищает последнюю. Если же нет возможности что-то отправить, то процесс снова засыпает до следующего “удара”.

Заключение

После реаализации базовой функциональности системы, проведения серии тестов получилось наладить передачу данных с умного устройства на сервер с промежуточным хранением в базе данных на смартфоне. В ходе экспериментального исследование было выяснено, что расход заряда батареи при использовании радиоинтерфейса телефона увеличивается при росте частоты отправки данных на сервер.

Также в рамках настоящей работы был проведён обзор СУБД временных рядов, на основании которого было проведено дальнейшнее сравнение по производительности двух выбранных СУБД, в результате которого была выбрана СУБД для использования в реализации.

В ближайшей перспективе продолжения работ - экспериментальное исследование программной реализации, измерение экономии заряда аккумулятора. В более отдалённой перспективе - изучение возможности масштабирования решения с помощью облачной платформы (в частности, запуска нескольких экземпляров сервера) для поддержки большого числа пользователей.

Благодарности

Авторы благодарят декана медико-биологического факультета РНИМУ им. Н.И. Пирогова Е. Б. Прохорчука, зам. декана А.А. Лагунина и научного сотрудника ФИАН Д.В. Малахова за предложение по совместной работе и материальное обеспечение - предоставление жилетки Hexoskin. Также авторы выражают благодарность сотруднику компании Яндекс А.А. Любицкому за технические консультации по выбору и использованию ПО.

Литература

1. Balasubramanian N., Balasubramanian A., Venkataramani A. *Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications* //Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement. – 2009.

2. AndroidDevelopersCommunity `https://developer.android.com`

3. Hexoskin SDK https://bitbucket.org/carre/hexoskin-smart-demo/src/master/hexoskin-smart-android

4. EclispeMQTTAndroid https://www.eclipse.org/paho

5. Хранение данных в Android https://code.tutsplus.com/ru/tutorials/android-from-scratch-how-to-store-application-data-locally-cms-26853

6. Authentication with Username and Password - MQTT Security Fundamentals https://www.hivemq.com/blog/mqtt-security-fundamentals-authentication-username-password

7. ClickHouse Documentation. https://clickhouse.tech/docs/ru/

8. InfluxDB Documentation. https://docs.influxdata.com/influxdb/

9. ClickHouse Crushing Time Series.

   https://www.altinity.com/blog/clickhouse-for-time-series

10. DB-Engines https://db-engines.com/en/

11. Rui Liu, Jun Yuan, Benchmark Time Series Database with IoTDB-Benchmark for IoT Scenarios, 2019 https://arxiv.org/pdf/1901.08304.pdf

12. Bader A., Kopp O., Falkenthal O., Survey and Comparison of Open Source Time Series Databases, 2017 https://www.researchgate.net/publication/315838456_Survey_and_Comparison_of_Open_Source_Time_Series_Databases

13. Naqvi S.N. Yfantidou S., Time Series Databases and InfluxDB, 2017 https://cs.ulb.ac.be/public/_media/teaching/influxdb_2017.pdf

14. Comparison of the Open Source OLAP Systems for Big Data: ClickHouse, Druid, and Pinot

    https://leventov.medium.com/comparison-of-the-open-source-olap-systems-for-big-data-clickhouse-druid-and-pinot-8e04-2a5ed1c7