САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Математико-механический факультет
Кафедра системного программирования

Афонина Ольга Андреевна

Энергопрофилирование многопоточных Android приложений

Курсовая работа

Научный руководитель :
ст. преп. С. Ю. Сартасов

Санкт-Петербург, 2019 г.

Содержание

Введение

Различные мобильные устрайства такие, как смартфоны или планшеты, стали неотъемлемой частью повседневной жизни в современном обществе. Так, уже в 2016 году число пользователей смартфонов превосходило 2 миллиарда людей, а к 2019 превысило 3 миллиарда и продолжает расти [1]. И большинство пользователей беспокоит время работы этих устройств. На скорость разряда батареи влияет множество факторов, среди которых: сложность устройства запускаемых приложений, параллельное исполнение программ, внешние факторы и износ батареи вследствие срока эксплуатации устройства. Проблему быстрого разряда аккумулятора можно решать путем улучшения аппаратной части, а именно: при помощи физического увеличения емкости батареи или увеличения энергоэффективности работы компонент девайсов (процессора, работы с памятью, периферийных устройств). Однако, несмотря на то, что стали распространены литиумные батареи с большей вместимостью, появилась новая архитектура CPU (big.LITTLE2), положительно сказавшиеся на времени работы устройств, было выявлено, что улучшения аккумулятора и технологий в области аппаратных средств имеют предел[2]. По этой причине разработчикам следует уделять пристальное внимание улучшению энергоэффективности работы самих приложений. Для выявления "узких" мест приложения разработчики используют профилирование — сбор характеристик его работы. Но измерить потребление ресурсов в течение какого-то времени и связать с протекающими в это время операциями — не самая тривиальная задача. Стоит также учитывать, что на сложность профилирования оказывает влияние и многопоточность — разделение

задач, реализуемых приложением, на цельные блоки, которые могут одновременно исполняться и работать с информацией. Поэтому разработчикам неоходима утилита, которая поможет отследить энергоэффективность программного кода, имеющего возможность многопоточного исполнения.

1 Цели и задачи

Целью данной работы является вычисление энергопотребления Android приложения. Для достижения этой цели были поставлены следующие задачи:

- Провести обзор методологий измерений
- Провести обзор существующих решений
- Провести обзор инструментов
- Выбрать методологию и используемые инструменты
- Получить данные, позволяющие вычислить потребляемую энергию

2 Обзор предметной области

2.1 Методологии измерения

Можно выделить два основных подхода к оценке энергопотребления программного обеспечения: прямое и косвенное измерение мощности.

Первый из них заключается в получении прямых замеров тока, напряжения или мощности с помощью внешнего устройства (мультиметра) или встроенных в устройство датчиков. Стоит учитывать, что при использовании внешнего мультиметра необходимо синхронизировать часы смартфона и этого устройства до начала проведения замеров.

Второй метод основывается на использовании модели энергопотребления, содержащей некоторые коэффициенты потребления мощности определенных инструкций или компонентов, находящихся в определенном состоянии. Такой подход позволяет оценить энергопотребление тестового прогона приложения без запуска на реальном устройстве. Однако одного коэффициента для каждой инструкции недостаточно для точной оценки в многопоточном случае, что значительно усложняет вычисление модели.

2.2 Существующие решения

Для поиска существующих решений использовался сервис “Google Scholar”. Из множества работ было выделено несколько подходящих решений, в которых используются разные подходы к замеру энергии.

Для подробного обзора было выделено несколько характерных решений с применением разных подходов и наиболее удач-

ной реализацией.

2.2.1 GreenDroid

GreenDroid[3] — инструмент для замера и анализа энергопотребления устройств Android. Он использует модель Power Tutor[4] — приложение для мобильных устройств на базе Android, отображающее энергию, потребляемую основными компонентами устройства (процессор, экран, GPS, сеть) и различных приложений. Вычисление энергопотроебления осуществляется на основе модели, построенной для смартфонов HTC G1, HTC G2 и Nexus One, что влечет за собой неточность получаемых данных на других моделях устройств. Кроме того, последнее обновление приложения было в 2011 году, поэтому оно может не учитывать появившиеся в новых версиях Android ограничения.

2.2.2 PETrA

PETrA[5] — инструмент для профилирования энергии устройств версии Android 5.0 и выше, разработанный в 2017 году. Для определения потребления энергии, используется инструменты Batterystats[6] (для определения состояния каждого компонента в конкретный момент времени) и Systrace[7] (для анализа изменения частоты процессора) и файл power_profile.xml.

Существенным минусом данного инструмента является отсутствие открытого исходного кода, а также каких-либо данных о последующей разработке. Это свидетельствует о том, что данный инструмент так и не вышел за пределы исследовательской работы.

2.2.3 ANEPROF

В ANEPROF[8] для профилирования энергии используется подход, основанный на прямых замерах с помощью тестового стенда PXA270 с Android 2.0. Для получения данных стенд подключается к считывателю данных NI DAQ. Для извлечения нужной информации и исключения не относящейся к непосредственно вызову методов (например, управление кучей и сборка мусора) используется инструмент DVMTI. Данные замеров энергии и трассы программы коррелируются между собой после синхронизации по времени, после чего генерируется отчет, показывающий энергопотребление методов в джоулях и процентах.

Этот инструмент не имеет открытого исходного кода и был разработан еще в 2011 году, поэтому непонятно, как он будет работать на современных устройствах с более новыми версиями ОС Android. Кроме того, может возникнуть проблема синхронизации часов, если данные о потреблении энергии получаются на одном девайсе, а трасса программы — на другом.

2.2.4 Battery Historian

Battery Historian[9] — интсрумент, предоставляющий информацию о расходе батареи и энергии устройства и приложений для Android версии 5.0 и выше. Вычисления производятся на основе данных bug report. Данный инструмент не предоставляет данных о потреблении конкретных методов приложения, что затрудняет оптимизацию энергоэффективности.

3 Инструменты

3.1 Для прямых замеров

В данном разделе рассматриваются инструменты для внутренних замеров, поскольку использование внешних устройств (например, мультиметра) сильно ограничит круг потенциальных пользователей в силу отсутствия у них нужных приспособлений.

3.1.1 dmesg

dmesg — команда, выводящая буфер сообщений ядра операционной системы, содержащая данные о мгновенном токе и напряжении. Время, отображаемое в сообщениях, записано в секундах с момента загрузки, поэтому для сопоставления данных об энергопотреблении с работой приложения, требуется перевести его во время часов устройства. Это можно сделать путем соотнесения общих для dmesg и logcat сообщений. Кроме проблемы вычисления времени, возникает еще одна — на устройствах версии 8.0 и выше для получения данного буфера требуются права суперпользователя, которыми обладает не каждый разработчик, так как они влекут за собой риск непреднамеренно изменить системные файлы, что приведет к неисправной работе смартфона. К тому же частота сообщений об изменении значений тока и напряжения слишком мала для отслеживания изменений потребления энергии в рамках методов исполняемой программы.

3.1.2 Файл current_now

Системный файл /sys/class/power_supply/battery/current_now хранит данные о мгновенном токе, однако для доступа к нему

на новых версиях Android требуются права суперпользователя.

3.1.3 BatteryManager

BatteryManager[10] — класс, предоставляющий методы для запроса данных батареи устройства. С его помощью можно получить информацию о мгновенном токе и напряжении. Однако для сопоставления полученных значений с методами приложения требуется непрерывно отслеживать состояние батареи, что увеличивает нагрузку на нее и "зашумляет" полученные результаты. К тому же частота изменения значений переменных может оказаться недостаточно высокой.

3.2 Для основанных на модели

3.2.1 Power profile

Файл power_profile.xml[11] — файл, предоставляемый производителем устройства и содержащий коэффициенты для вычисления потребляемой мощности компонентами устройства (ядрами процессора, модулями GPS, Wi-Fi, Bluetooth и т.д), находящихся в различных состояниях.

3.2.2 Settings.System

Класс Settings.System[12] содержит данные о яркости экрана устройства, лежащей в диапазоне от 0 до 255. Поскольку в Android нет события, оповещающего об изменении яркости экрана, нужно непрерывно отслеживать его состояние, что увеличивает нагрузку на батарею, кроме того в режиме автоматической настройки яркости или при выключенном экране значение константы SCREEN_BRIGHTNESS не изменяется.

3.2.3 Dumpsys display

С помощью команды dumpsys display[13] можно получить данные о текущей яркости экрана. При выключенном экране отображается 0, также при автонастройке яркости отображается реальное значение (а не установленное пользователем вручную). Проблема использования данного инструмента заключается в необходимости запуска команды внутри инструментовки кода для каждого метода, что требует прав суперпользователя на устройстве или разрешения permission.DUMP[14], которое не предоставляется третьесторонним приложениям.

3.2.4 Batterystats

Batterystats[6] — инструмент для получения статистических данных об использовании батареи, с его помощью можно получить данные об изменении яркости и включении, выключении экрана. Значения яркости отображаются в относительных единицах dark, dim. medium, light и bright.

3.2.5 WifiManager

WifiManager[15] позволяет получить информацию о состоянии wifi. Однако, начиная с API 29 (Android 10) класс NetworkInfo[16] объявлен устаревшим и не рекомендуется для использования.

3.2.6 Dumpsys wifi

Команда dumpsys wifi выводит данные об изменении состояния wifi и передаче, получении данных с отметками времени устройства. При запуске на устройствах с разными версиями Android (6.0, 7.0, 7.1, 8.0 и 9) было выявлено, что запись этих событий отличается и что для каждого смартфона необходимо

анализировать получаемый файл отдельно.

3.2.7 cpu frequency

/sys/devices/system/cpu/cpuN/cpufreq/stats/time_in_state – системный файл, содержащий время проведенное на каждой возможной частоте для каждого ядра. Эти данные можно получать для каждого метода, а затем вычислить для них время и энергию, потребленную ядрами.

3.3 Выбранные инструменты

В ходе обзора инструментов выбор был сделан в пользу подхода, основанного на модели с использованием данных power_profile.xml. Для получения данных об активности компонент выбраны Batterystats и файл, с данными о частоте cpu. А для запуска команды Batterystats используется Android Debug Bridge (adb)[17] — универсальный инструмент командной строки, позволяющий взаимодействовать с мобильным устройством и предоставляющий доступ к оболочке Unix, которую можно использовать для запуска различных команд на устройстве. adb входит в пакет Android SDK Platform-Tools, который можно загрузить с помощью SDK Manager.

4 Реализация

Итак, для энергопрофилирования был выбран подход, основанный на power_profile.xml. Данные о частоте cpu, необходимые для вычисления потребления энергии, извлекаются из файла /sys/devices/system/cpu/cpuN/cpufreq/stats/time_in_state для каждого метода на этапе инструментовки. С его помощью для каждого ядра вычисляется время, проведенное на той или иной частоте.

Для получения информации об изменении состояния экрана (включения, выключения или изменения яркости) можно использовать инструмент Batterystats. Время изменений записывается в нем отметками относительно времени сброса журнала истории, поэтому, во-первых, перед началом профилирования требуется сбросить этот журнал, во-вторых, возникает необходимость пересчета времени для сопоставления с методами приложения. В журнале истории Batterystats время сброса записывается в формате "yyyy-MM-dd-HH-mm-ss", в котором не записываются миллисекунды, с точностью до которых записываются отметки событий истории батареи. Поэтому нужно использовать Batterystats с опцией -c, что позволит получить время сброса журнала в миллисекундах и вычислить, когда было записано то или иное событие. На данный момент обработка этих данных не включена в профилятор, поскольку их еще нужно соотнести с методами исполняемого приложения для вычисления энергопотребления.

Заключение

К концу весеннего семестра были выполнены следующие задачи:

- Произведен обзор методологий измерения
- Произведен обзор существующих решений
- Произведен обзор инструментов
- Выбрана методология и используемые решения
- Получены данные о частоте ядер, позволяющие на их основе для каждого метода вычислить энергопотребления

В дальнейшем планируется добавить вычисление энергопотребления других компонентов таких, как экран, wifi, bluetooth, gps.

Список литературы

[1] Statista Inc. Number of smartphone users worldwide from 2014 to 2020 (in billions). https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/, 2019. Дата обращения 03.05.2020.

[2] Jason Flinn and M. Satyanarayanan. Energy-aware Adaptation for Mobile Applications. *SIGOPS Oper. Syst. Rev.*, 34(2):13–14, April 2000.

[3] Marco Couto, Jácome Cunha, João Fernandes, Rui Pereira, and Joao Saraiva. GreenDroid: A tool for analysing power consumption in the android ecosystem. pages 73–78, 11 2015.

[4] PowerTutor. http://ziyang.eecs.umich.edu/projects/powertutor/index.html. Дата обращения 20.12.2019.

[5] Dario Di Nucci, Fabio Palomba, Antonio Prota, Annibale Panichella, Andy Zaidman, and Andrea Lucia. PETrA: A Software-Based Tool for Estimating the Energy Profile of Android Applications. 05 2017.

[6] Batterystats. https://developer.android.com/studio/command-line/dumpsys#battery. Дата обращения 03.05.2020.

[7] Systrace. https://developer.android.com/topic/performance/tracing/command-line. Дата обращения 03.05.2020.

[8] Yi-Fan Chung, Chun-Yu Lin, and Chung-Ta King. ANEPROF: Energy Profiling for Android Java Virtual Machine and

Applications. *Proceedings of the International Conference on Parallel and Distributed Systems - ICPADS*, pages 372–379, 12 2011.

[9] Battery Historian. `https://developer.android.com/topic/performance/power/battery-historian`. Дата обращения 03.05.2020.

[10] BatteryManager. `https://developer.android.com/reference/android/os/BatteryManager`. Дата обращения 03.05.2020.

[11] Power profile. `https://source.android.com/devices/tech/power/values#values`. Дата обращения 03.05.2020.

[12] Settings.System. `https://developer.android.com/reference/android/provider/Settings.System#SCREEN_BRIGHTNESS`. Дата обращения 03.05.2020.

[13] Dumpsys display. `https://developer.android.com/studio/command-line/dumpsys`. Дата обращения 03.05.2020.

[14] Permission DUMP. `https://developer.android.com/reference/android/Manifest.permission#DUMP`. Дата обращения 03.05.2020.

[15] WifiManager. `https://developer.android.com/reference/kotlin/android/net/wifi/WifiManager`. Дата обращения 03.05.2020.

[16] NetworkInfo. `https://developer.android.com/reference/android/net/NetworkInfo`. Дата обращения 03.05.2020.

[17] Android Debug Bridge. `https://developer.android.com/studio/command-line/adb`. Дата обращения 20.12.2019.

[18] Google I/O 2019. Empowering developers to build the best experiences on Android + Play. https://android-developers.googleblog.com/2019/05/google-io-2019-empowering-developers-to-build-experiences-on-Android-Play.html. Дата обращения 20.12.2019.