ЕКОЛОГІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

УДК 004.052.42

Туркин И.Б., Вдовитченко А.В.
Национальный аэрокосмический университет им. Н.Е. Жуковского "ХАИ"

ОПТИМИЗАЦИЯ КОДА ПОЧТОВОГО ПРОГРАММНОГО ПРИЛОЖЕНИЯ НА ПЛАТФОРМЕ APPLE IOS ПО КРИТЕРИЮ МИНИМУМА ЭНЕРГОПОТРЕБЛЕНИЯ

На основе анализа рынка компьютерных устройств показано, что его рост сейчас определяется прежде всего увеличением продаж мобильных устройств. Общей тенденцией является расширение перечня функций и возможностей смартфона или планшета, но при этом его интенсивное использование возможно только в течение нескольких часов, поэтому одним из наиболее важных показателей качества мобильного устройства является продолжительность его работы в автономном режиме. Выполненный анализ возможностей увеличения продолжительности работы мобильного устройства в автономном режиме показал, что разработчики прикладного программного обеспечения не уделяют должного внимания оптимизации кода по критерию минимума энергопотребления. В результате проведенного эксперимента по оптимизации кода конкретного приложения доказана эффективность такой оптимизации.

На основі аналізу ринку комп'ютерних пристроїв показано, що його зростання зараз визначається насамперед збільшенням продажів мобільних пристроїв. Загальною тенденцією є розширення переліку функцій і можливостей смартфона або планшета, але при цьому його інтенсивне використання можливе тільки протягом декількох годин, тому одним з найбільш важливих показників якості мобільного пристрою є тривалість його роботи в автономному режимі. Виконаний аналіз можливостей збільшення тривалості роботи мобільного пристрою в автономному режимі показав, що розробники прикладного програмного забезпечення не приділяють належної уваги оптимізації коду за критерієм мінімуму енергоспоживання. У результаті проведеного експерименту з оптимізації коду конкретного додатка доведена ефективність такої оптимізації .

Based on market analysis of computer devices is shown that its growth is now determined primarily by increased sales of mobile devices. The general trend is to extend the list of features and capabilities of the smart phone or tablet , but its intensive use is possible only for a few hours , so one of the most important indicators of the quality of the mobile device is the length of his work in stand-alone mode. Performed analysis of the possibilities of increasing the length of the mobile device when offline showed that application software developers do not pay enough attention to code optimization criterion of minimum power consumption. As a result of the experiment on the application-specific code optimization proved the effectiveness of this optimization..

Ключевые слова: *зеленое программное обеспечение, оптимизация кода, критерий энергопотребления, стандарт ACPI, энергетическое программное обеспечение*

Введение

Мобильные операционные системы (ОС) быстро набирают популярность, что доказывают последние исследования рынка мобильных устройств. Темп роста продаж компьютеров (настольных и портативных) перестал увеличиваться и держался на одном уровне в 2012-2013 годы, а мобильные устройства в свою очередь демонстрировали значительный рост продаж, включая, естественно, и планшетные компьютеры.

По состоянию на первый квартал 2013 года согласно данных аналитического агентства International Data Corporation (IDC) [1], объем продаж смартфонов с установленной операционной системой Apple iOS составляет более 1/3 всего рынка продаж мобильных устройств.

iOS (до 24 июня 2010 года – iPhone OS) – мобильная операционная система, разрабатываемая и выпускаемая американской компанией Apple. Была выпущена в 2007 году, первоначально – для iPhone и iPod touch, позже

– для таких устройств, как iPad и Apple TV. В отличие от Windows Phone и Google Android, выпускается только для устройств, производимых фирмой Apple [2].

iOS разработана на основе Mac OS X и использует POSIX-совместимый набор основных компонентов Darwin – открытой POSIX-совместимой операционной система, выпущенной Apple Inc. в 2000 году. Система совмещает код, написанный самой Apple, с полученным от NeXTSTEP (первый выпуск – 1989 г.), FreeBSD (1993 г.) и прочих свободных проектов. Система Darwin представляет собой набор основных компонентов, используемых в Mac OS X и Apple iOS. Она совместима с третьей версией спецификации единой UNIX (SUSv3) и POSIX-приложениями и утилитами.

В iOS есть четыре слоя абстрагирования: Core OS, Core Services, Media Layer и Cocoa Touch. Для текущей версии операционной системы (iOS 7.1.1) выделяется 1,4÷2 Гб флеш-памяти устройства для системного раздела и примерно 800 Мб свободного места (варьируется в зависимости от модели). По состоянию на конец мая 2013 года магазин приложений App Store содержит более 900 тыс. приложений для iOS, которые все вместе были загружены более **50 миллиардов** раз [3].

iOS является примером закрытой мобильной платформы, доступ к системным функциям ограничен – специфика языка Objective C (надмножество языка Си), который не допускает скрытых методов и реализаций. Плюсами iOS являются: качество приложений, распространяемых через стандартный магазин, и отсутствие вредоносного ПО. Минус – закрытость самой платформы, поскольку многие возможности ОС не доступны из-за отсутствия доступа к внутренним API.

В итоге, многие из современных мобильных устройств на основе iOS не уступают по вычислительной мощности стационарным компьютерам, но одновременно с этим обладают такими свойствами, как портативность и переносимость. Но эти плюсы влекут за собой ряд довольно значительных проблем: вывод излишков тепла, образуемого при работе таких компонентов как процессор, видеокарта и др., малый срок работы устройства без подключения к сети электропитания. Обе эти проблемы имеют два взаимодополняющих пути решения – это применение новых технологий для создания аппаратных компонент (процессора, видеокарты, аккумулятора и др.) и внедрение новых подходов к разработке энергосберегающего программного обеспечения (ПО) для данных компонентов.

1. Анализ проблемы оптимизация кода по критерию минимума энергопотребления

Характеризуя сегодняшнее состояние проблемы оптимизация кода конкретного приложения по критерию минимума энергопотребления всего мобильного устройства, в любом случае потребуется рассмотреть следующие аспекты:

– аппаратное обеспечение управления энергопотреблением;

– поддержка аппаратуры со стороны ОС;

– наличие и возможности доступных инструментальных средств – энергетических профилировщиков кода прикладного ПО.

1. **Аппаратное обеспечение.** Благодаря своей высокой удельной и объемной плотности энергии, в портативных устройствах широко используются Li-ионные и Li-полимерные аккумуляторные батареи (АКБ). Одной из самых сложных задач при разработке является обеспечение безопасности системы с питанием от АКБ.

Микросхемы памяти с уникальным идентификационным номером батареи, например, bq2022A от Texas Instruments (TI) или микросхема защиты на базе SHA-1 типа bq26100 от TI, дают возможность подтвердить, получена ли АКБ от авторизованного поставщика и, соответственно, безопасна ли она [4].

Температура батареи является критическим параметром, определяющим безопасность батареи. Повышенная температура эксплуатации ускоряет деградацию АКБ и может вызвать выход из строя и взрыв Li-ионной АКБ. Этот тип батарей достаточно опасен из-за использования очень агрессивного активного материала. Быстрое повышение температуры батареи возможно в случае перезаряда большим током или при наличии внутреннего короткого замыкания. Во время перезаряда Li-ионной АКБ активный металлический литий осаждается на аноде. Этот материал значительно повышает опасность взрыва при реагировании с различными материалами, включая электролит и материалы катода. Например, смесь лития с включениями углерода реагирует с водой и выделяемый при этом водород может воспламениться от тепла, выделяемого при прохождении реакции. Материал катода, например $LiCoO_2$, вступает в реакцию с электролитом, когда температура превышает порог 175 °C. С другой стороны, заряд АКБ при низких температурах также сокращает срок службы АКБ, поскольку ионы лития могут осаждаться на аноде и превращаться в металлический литий, который легко реагирует с электролитом. Ионы лития постоянно исчезают и более не могут участвовать в процессе хранения энергии.

Для контроля температуры секций и защиты от перегрева Li-ионной АКБ обычно используются термисторы. Например, заряд Li-ионной батареи не допускается, если температура секции ниже 0 или выше 45 °C; также не допускается разряд, если температура секции выше 65 °C.

Во многих портативных медиа-системах имеется возможность эксплуатировать систему при одновременном заряде глубоко разряженной батареи с целью предоставления конечному пользователю возможности вести телефонный разговор или играть независимо от состояния АКБ при наличии адаптера.

Для заряда глубоко разряженной батареи, напряжение секции которой упало ниже 3,0 В, используется небольшой ток предварительного заряда. Системная нагрузка использует некоторую часть этого тока, делая эффективный зарядный ток ещё меньше. При этом не только возрастает время заряда батареи, но также происходит ложное срабатывание таймера предварительного заряда, поскольку напряжение батареи не может подняться до 3,0 В за время предварительного заряда. Возможна даже ситуация, когда системный ток превышает ток предварительного заряда, то есть батарея вместо заряда продолжает разряжаться.

Эти проблемы являются результатом взаимодействия между зарядным устройством и системой и могут быть исключены путём питания системы и зарядки батареи от независимых источников питания. Данная технология известна как управление потоками мощности (PPM). Когда ток, требуемый системе и зарядному устройству, больше входного тока, и напряжение на системной шине падает до значения предварительно заданного порога PPM, зарядный ток снижается до такой степени, что суммарный ток, необходимый системе и зарядному устройству, становится равным максимальному току, обеспечиваемому адаптером.

Это позволяет максимально использовать питание, обеспечиваемое адаптером или USB. Большинство системных нагрузок имеют высокую динамику с большими выбросами тока. Поскольку средняя мощность системы значительно ниже её пиковой мощности, адаптер будет иметь завышенный запас по мощности, если его номинальную мощность выбрать исходя из пиковой мощности системы и зарядного устройства. PPM позволяет разработчику использовать адаптеры переменного тока с меньшей номинальной мощностью, то есть менее затратные.

Индикация уровня заряда аккумуляторной батареи развивалась от простых предупреждающих устройств до более сложного использования информации на системном уровне, например программного отключения для предотвращения потери данных. Ошибка в определении ёмкости для конечного пользователя эквивалентна снижению полезного времени работы устройства. Так, например, использование индикатора ёмкости с погрешностью в 10 % эквивалентно использованию АКБ с ёмкостью на 10 % меньше, или системы преобразования энергии с КПД на 10 % меньше.

Широко применяемый в переносных устройствах типа мобильных телефонов метод контроля уровня заряда АКБ, основанный на измерении напряжения, имеет недостатки, связанные с изменением сопротивления АКБ во времени.

Альтернативным является метод подсчёта кулонов, при котором производится непрерывное интегрирование тока АКБ для расчёта потребляемого заряда и уровня оставшегося заряда, когда, заранее зная полную, можно вычислить остаточную ёмкость. Недостатком этого метода является то, что трудно смоделировать саморазряд батареи, поскольку он зависит от изменения свойств АКБ в результате старения и температуры. Без проведения периодических калибровок по полному циклу ошибка измерения накапливается во времени. Ни один из этих алгоритмов не учитывает изменений сопротивления АКБ, поэтому для предотвращения неожиданных отключений, разработчик должен зарезервировать больше ёмкости и неиспользованной энергии путём заблаговременного отключения системы.

Запатентованная корпорацией TI технология Impedance Track™ представляет собой уникальный и значительно более точный метод определения остаточной ёмкости АКБ по сравнению с упоминавшимися выше алгоритмами. При этом обе технологии используются для преодоления влияния старения, саморазряда и изменений температуры. В технологии Impedance Track™ реализуется алгоритм динамического моделирования для определения и отслеживания характеристик АКБ путём измерения и последующего отслеживания изменений полного сопротивления и ёмкости в процессе фактического использования АКБ. При помощи этой технологии практически в реальном времени обеспечивается информация о времени работы батареи, максимальной рабочей температуре, числе циклов, максимальном напряжении секции и максимальных зарядном и разрядном токах. Это самообучающийся механизм учитывает химическую ёмкость в отсутствие нагрузки (Q_{MAX}) и влияние старения на изменения сопротивления АКБ. Компенсация для

нагрузки и температуры точно моделируется с помощью информации о полном сопротивлении секции АКБ. При использовании этого алгоритма не требуется периодического определения ёмкости по полному циклу.

Чтобы технология Impedance Track™ работала, постоянно должна поддерживаться база табличных данных о сопротивлении АКБ как функции уровня разряда (DOD) и температуры. В энергонезависимой памяти измерителя программируется несколько порогов по току для определения заряда, разряда и «времени восстановления», представляющего собой интервал времени, в течение которого напряжение АКБ стабилизируется после прекращения зарядки или разряда.

2. Стандарт управления питанием и конфигурацией ACPI (Advanced Configuration и Power Management) представляет собой открытый промышленный стандарт, который определяет общий интерфейс для обнаружения аппаратного обеспечения, управления питанием и конфигурации материнской платы и устройств [5].

Наиболее известной частью стандарта ACPI является управление питанием, имеющее два значительных усовершенствования по сравнению с предшествующими стандартами.

Во-первых, концепция ACPI в отличие от существовавшей ранее модели APM (Advanced Power Manager) передаёт управление питанием операционной системе (OS). Таким образом, ОС получает практически полный контроль над энергопотреблением.

Во-вторых, ACPI – это предоставление на серверах и настольных компьютерах таких возможностей по управлению питанием, которые до того были доступны только на портативных компьютерах. Например, система может быть переведена в состояние чрезвычайно низкого энергопотребления, в котором питание подается лишь на оперативную память, но при этом прерывания некоторых устройств (часы реального времени, клавиатура, модем и т. д.) могут достаточно быстро перевести систему из такого состояния в нормальный рабочий режим (то есть «пробудить» систему).

Помимо требований к программному интерфейсу ACPI также требует специальной поддержки от аппаратного обеспечения. Таким образом, поддержку ACPI должны иметь ОС, чипсет материнской платы и даже центральный процессор.

На данный момент последней версией спецификации ACPI является версия 5.0 от 6 декабря 2011 года. Задачей ACPI является обеспечение взаимодействия между операционной системой, оборудованием и BIOS материнских плат [6]. Стандартный ACPI был разработан специально для реализации OnNow идеологии. OnNow – это идеология компьютерного устройства, которое готово возобновиться и быстро приступить к работе в любое время. ACPI определяет понятие «глобальное состояние системы». Каждое состояние системы характеризуется уровнем производительности и потребления энергии. Глобальные состояния системы определяются по следующим критериям (таблица 1).

Состояние G1 (Sleep) имеет 4 подсостояния, которые отличаются уровнем потребления энергии, а также способом хранения и восстановления системного контекста:

– S1, состояние сна – характеризуется низким временем ожидания перехода из состояния сна в рабочее состояние и наоборот. В этом состоянии сохраняется весь системный контекст (и центрального процессора, и чипсета), и аппаратные средства поддерживают весь системный контекст;

– S2, состояния сна – характеризуется по-прежнему малым, но больше чем в S1 временем ожидания перехода из состояния сна в рабочее состояние и наоборот. Это состояние подобно S1, за исключением того, что содержимое кэша центрального процессора и системы потеряно (ОС отвечает за поддержание содержания основного кэша и контекста центрального процессора). Контроль ОС начинается с вектора сброса процессора, после события пробуждения;

Таблица 1

Состояния системы согласно ACPI

Название состояния	Описание состояния
G0 (S0) (Рабочее)	Выполнение стандартных операций. Максимальная производительность, при обычном энергопотреблении.
G1 (приостановлено, спящее, гибридный сон)	Пониженное энергопотребление, с возможностью быстрого восстановления рабочего состояния, без потери данных.
G2 (S5) (программное выключение)	Выключение операционной системы при незначительном потреблении энергоресурсов, возможность более быстрого запуска по сравнению с «холодным стартом» состояния G3.
G3 (Выключенное)	Полное отключение, не потребляет энергии.

– S3, состояние сна – характеризуется низким временем ожидания перехода из состояния сна в рабочее состояние и наоборот, но больше чем в S2. При этом весь системный контекст потерян, кроме системной памяти. Центральный процессор, кэши и контекст чипсета потеряны также. Аппаратные средства поддерживают контекст памяти и восстанавливают некоторую конфигурацию центрального процессора и конфигурацию кэша L2. Контроль начинается с вектора сброса процессора после события пробуждения;

– S4 («Спящий режим» (Hibernation) в Windows, «Safe Sleep» в Mac OS X, также известен как «Suspend to disk») – в этом состоянии всё содержимое оперативной памяти сохраняется в энергонезависимой памяти, такой как жёсткий диск: состояние операционной системы, всех приложений, открытых документов и т. д. Это означает, что после возвращения из S4, пользователь может возобновить работу с места, где она была прекращена, аналогично режиму S3. Различие между S4 и S3, кроме дополнительного времени на перемещение содержимого оперативной памяти на диск и назад, – в том, что перебои с питанием компьютера в S3 приведут к потере всех данных в оперативной памяти, включая все не сохранённые документы, в то время как компьютер в S4 этому не подвержен. S4 весьма отличается от других состояний S и сильнее S1-S3 напоминает G2 Soft Off и G3 Mechanical Off. Система, находящаяся в S4, может быть также переведена в G3 Mechanical Off (Механическое выключение) и все ещё оставаться S4, сохраняя информацию о состоянии так, что можно восстановить операционное состояние после подачи питания;

В ACPI есть отдельные состояния мощности для центрального процессора (CPU):

– состояние C0 – рабочее состояние, когда процессор выполняет все инструкции;

– состояние C1 характеризуется самым низким временем ожидания процессора. Время ожидания аппаратных средств в этом состоянии должно быть достаточно низким, поэтому операционное программное обеспечение не анализирует готовность аппаратного средства, принимая решение о его использовании. Кроме перевода процессора в более низкое состояние потребления энергии, это состояние не имеет никаких других эффектов, видимых программным обеспечением;

– состояние C2 обеспечивает улучшенное сбережение электроэнергии, по сравнению с C1. Время ожидания аппаратных средств в этом состоянии определяется системным программируемым оборудованием ACPI и операционным программным обеспечением. Кроме перевода процессора в нерабочее состояние, это состояние не имеет никаких других видимых программным обеспечением эффектов;

– состояние C3 предлагает улучшенное сбережение энергии, по сравнению с C1 и C2. Худшее время ожидания аппаратных средств в этом состоянии должно учитываться системным программируемым оборудованием ACPI и операционным программным обеспечением, которое может использовать текущую информацию, чтобы сделать правильный выбор между состояниями C2, либо C3. В состоянии C3 кэши процессора поддерживаются включенным, но игнорируются любые запросы к ним. Операционное программное обеспечение должно обеспечить корректное сохранение кэшей.

Для различных устройств, например, модемов, мониторов, сетевых карт, видеокарт также поддерживаются состояния энергосбережения ACPI:

– D3 (выключенное). Устройство полностью обесточено. Контекст устройства потерян, таким образом, программное обеспечение OS повторно инициализирует устройство, приводя его в действие;

– D1 и D2 – промежуточные состояния, активность определяется устройством;

– D0 – полностью оперативное состояние, устройство включено.

3. Инструментальные средства энергетического профилирования ПО. Аппаратных средств измерения абсолютных показателей потребления энергии у процессоров Intel долгое время просто не было. Точнее, такие средства были реализованы как отдельные устройства, подключаемые к материнской плате (или, например, как Watts Up, через USB [7]), что достаточно неудобно, а порой почти нереализуемо (например, если нет физического доступа к машине).

Но уже процессоры Pentium содержали в себе модуль измерения производительности (Performance Monitoring Unit), который давал возможность собирать статистику по количеству аппаратных событий, происходящих в процессоре (например, количество исполненных инструкций, промахов кешей и т.д.). На базе этой статистики долгое время и производились оценки энергопотребления. Происходило это следующим образом: строилось некоторое

предположение о корреляции между измеряемыми метриками и энергопотреблением системы (энергетическая модель), и далее, когда профайлер с заложенной моделью попадал на целевую машину, запускался процесс калибровки, связывающий оригинальные метрики и реальное энергопотребление. Далее по полученным коэффициентам можно оценивать энергопотребление.

Такой подход не может гарантировать безошибочный результат, и поэтому в архитектуру Sandy Bridge был заложен RAPL интерфейс (Running Average Power Limit), после чего стало возможно получить информацию о количестве энергии, потребленной системой с момента запуска. Естественно, после этого обсуждать профилировщики, работающие на базе счетчиков производительности, кажется бессмысленным, но это не совсем так, поскольку вполне можно все еще встретить машину, не оснащенную процессором на архитектуре Sandy Bridge и старше, и RAPL позволяет оценивать только энергопотребление CPU, чего в некоторых случаях может быть недостаточно.

Intel Power Gadget [8] – профилировщик, предоставляющий статистику по энергопотреблению выбранного пользователем приложения на машинах с CPU на базе Sandy Bridge и новее. Предоставляет также интерфейс Power Gadget API, позволяющий пользовательским приложениям получать метрики энергопотребления. Не может связывать статистику с исходным кодом.

XCode 5 [9]. В состав IDE от Apple был включен мощный инструментарий по анализу энергопотребления приложения. Он предоставляет агрегированные метрики, которые могут оказаться даже более информативными, чем абсолютные значения потребления энергии. Среди них можно выделить WakesPerSecond – количество “пробуждений” CPU в связи с активностью приложения (например, из-за таймеров и ряда других событий). В дополнение к этой метрике данный инструмент выстраивает CPU-usage трассу, на которой выделен CPU Wake Overhead. Такие трассы строятся отдельно для каждого из потоков, что упрощает анализ приложения. Инструмент от Apple не связывает напрямую измеряемые метрики и код приложения, но имея на руках информацию о проблемной тенденции, в большинстве случаев можно детализировать информацию при помощи других инструментов (например, причину частых пробуждений можно искать при помощи утилиты timerfires, а неоправданно высокий уровень утилизации CPU – профилировщиком Instruments).

Ниже рассмотрены только 2 утилиты измерения энергопотребления для Apple iOS (таблица 2).

Таблица 2

Основные характеристики энергетических профилировщиков для Apple iOS

Профилировщик	ОС	Измерение энерготребления CPU	Привязка к исходному коду
Intel Power Gadget	Linux, OS X, Windows,	Прямые измерения через RAPL	Нет
XCode 5 Power Profiler	OS X	Сбор агрегированных метрик от ОС	Раздельная статистика по потокам.

Цель и задачи работы

На текущий момент большинство разработчиков мало внимания уделяют оптимизации ПО по критерию энергопотребления, хотя основные принципы известны:

– распараллеливание задачи может снизить потребление энергии процессора до 40 % в мультимедиа приложении [1];

– уменьшение количества системных вызовов, например, использование функции API Windows «EnterCriticalSection()» для синхронизации передачи данных между потоками в пространстве пользователя вместо «WaitForSingleObject()», выполняющийся в пространстве ядра экономит более 60 % энергии [10];

– объединение сеансов ввода-вывода в пакеты экономит 30-60 % энергии для твердотельных накопителей;

– минимизации использования оперативной памяти, которая достигается за счет исключения ненужных преобразований формата графики, кэширования часто используемых структур данных, ограничения количество перемещений данных между пространством ядра и пространством пользователя и т.д [11].

Цель настоящей работы – экспериментально проверить эффективность энергосберегающих решений в архитектуре и коде конкретного приложения с учетом трудоемкости внесения модификаций и возможного отрицательного воздействия на его usability свойства.

Результаты исследований

1. Общая характеристика приложения

DDMailSender предназначен для поддержки очереди и последовательной отправки данных (текстовой информации и графических файлов) на электронную почту. Так же имеется функция сохранения в базу данных для возможности возобновления отправки, если приложение завершило работы и функция хранения истории удачных и не удачных отправок для их повторной отправки. Хотя программный продукт является тестовым (написанным для целей эксперимента), но задачи, которые он решает, встречаются в реальных проектах и являются достаточно сложными, и это делает его кандидатом для внедрения оптимизаций по энергопотреблению.

Ниже перечислены основные алгоритмы.

1. Алгоритм добавления данных на отправку в очередь должен обеспечить:

– отсутствие влияния на быстродействие интерфейса пользователя;

– поддержание истории отправок;

– эффективную работа с ресурсами для предотвращения закрытия приложения из-за нехватки оперативной памяти;

– возобновляемость работы после экстренного завершения приложения;

– работу в фоновом режиме, если приложение свернуто;

– поддержание последовательной и контролируемой отправки.

2. Алгоритм сохранения данных в базу данных должен поддерживать отдельную очередь операций для управления обращениями к базе данных (как средство по работе с базой данных выбран Фреймворк CoreData, основным недостатком, которого является специфичная работа в многопоточных приложения).

3. Алгоритм обновления истории отправок. Для исключения заполнения всего свободного места на диске, история отправок имеет конечный размер и при каждом обновлении истории происходит проверка количества записей и удаление наиболее старых.

2. Схема эксперимента и его результаты

Для измерений энергопотребления используется Xcode instruments energy diagnostics (см. табл. 2). Измерения проводятся на iPad3 с версией операционной системы iOS6.0.

Потребляемая мощность определяется профилировщиком по внутренней математической модели, результат вычисляется в среднем один раз в секунду по измеренным значениям системных счетчиков. Результат представлен в условных единицах EUL (Energy Usage Level). Пример работы Xcode instruments energy diagnostics можно увидеть на рис. 1. На графике скачки энергопотребления соответствует сворачиванию и разворачиванию меню настроек, а также работе Wi-Fi.

Каждое измерение энергопотребления включало действия:

– при работающем профилировщике запустить приложение;

– остановить очередь отправки;

– добавить 5 одинаковых картинок в очередь;

– продолжить работу очереди и измерить время начала отправки;

– по окончании отправки измерить время и проверить содержание файла с результатами измерений, сохраненного на диске;

– закрыть приложение.

Такое измерение было выполнено для каждого из следующих проектных решений почтового приложения:

– базовый вариант, в построении которого аспект энергосбережения не рассматривался;

– увеличение количества потоков, участвующих в отправке данных, при этом увеличится загрузка процессора, но отправка выполнится быстрее, обеспечив экономию энергию;

– минимизация дисковых операций путем сохранение данных до их окончательной отправки только в памяти;

– потоковые операции с файлами в процессе отправки данных. Энергия экономится за счет минимизации использования оперативной памяти, картинка загружается по частям по мере отправки.

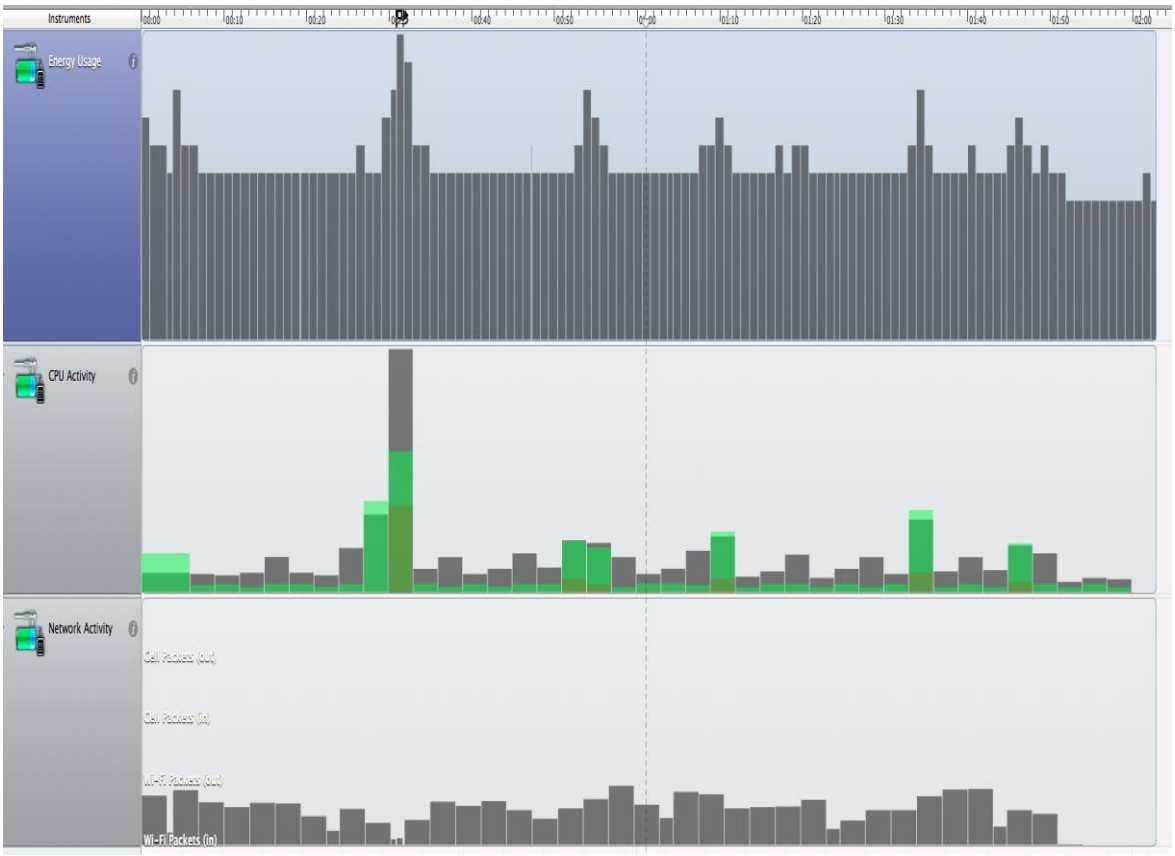

Рис. 1 Энергопотребление, загрузка процессора и активность сети устройства без запущенных

Основные результаты приведены в таблице 3.

Таблица 3
Сравнение с базовым вариантом альтернативных энергосберегающих проектных решений

Параметр	**Увеличение количества потоков**	**Минимизация дисковых операций**	**Потоковое чтение**
Дополнительные трудозатраты в реализации, час.	1	12	40
Экономия энергии относительно базового приложения, %	38	9	17
Количество модифицированных классов	1	3	4
Оценка usability	+	-	+/-

Оценка usability субъективна и основана на том, что

– при увеличенном количестве потоков ощутимо ускорилась скорость отправки данных, изменилось поведение пользовательского интерфейса, поскольку не всегда первый элемент в очереди успевает отправиться первым, но это никак не влияет на общее впечатление от приложения;

– при сохранение данных до их окончательной отправки только в памяти возможно аварийное завершение приложения при больших объемах данных;

– при потоковых операциях появляется возможность работы с очень большими файлами, но в общем случае увеличивается нагрузка на жесткий диск.

Выводы

На сегодня одной из основных тенденций развития вычислительной техники является стремительное расширение рынка мобильных устройств, для которых вопрос энергетически эффективного использования ресурсов особенно актуален. Любому из нас доводилось в лучшем случае просто огорчиться из-за пропавшей связи, внезапно обнаружив разряженную аккумуляторную батарею в смартфоне или планшете. В худшем случае, последствия могли быть намного печальнее, что для бизнес-деятельности, что для личной жизни.

Каждое новое поколение мобильных устройств двигалось вперед в решении этой проблемы, прежде всего в направлении развития аппаратной части. В свою очередь, рост требований и ожиданий пользователей

мобильных устройств, как и раньше, опережает темпы этого развития. Сегодня нельзя не говорить об энергетически эффективном поведении программного обеспечения, тем более что для этого существуют все основания:

– аппаратное обеспечение управления энергопотреблением;

– поддержка аппаратуры со стороны ОС;

– наличие и возможности доступных инструментальных средств – энергетических профилировщиков кода прикладного ПО.

Но по-прежнему большинство разработчиков не обращает внимания на оптимизацию программного обеспечения по критерию энергопотребления, хотя основные принципы такой оптимизации давно известны.

Результаты экспериментальных исследований, приведенные в статье, призваны убедить всех, что сегодня такое пренебрежение не имеет права на существование, ведь зачастую мизерные затраты труда разработчика прикладного ПО способны кардинально улучшить работу всего мобильного устройства.

Список использованных источников

1. Фараг Ш. Повышение эффективности энергопотребления для приложений / Ш. Фараг, Б. Сроур [Электронный ресурс] – Режим доступа: http://blogs.msdn.com/b/b8_ru/archive/2012/02/15/improving-power-efficiency.aspx – 10.01.2014г.

2. Описание платформы iOS [Электронный ресурс] – Режим доступа: http://en.wikipedia.org/wiki/IOS – 30.11.2013 г.

3. Описание платформы iOS [Электронный ресурс] – Режим доступа: http://en.wikipedia.org/wiki/IOS – 30.11.2013г.

4. Дэйв М. iOS 6 SDK. Разработка приложений для iPhone, iPad и iPod touch = Beginning iOS 6 Development Exploring the iOS SDK / М. Дейв. – М.: «Вильямс», 2013. – 672 с.

5. Бирюков Е. Методы снижения потребления энергии современными портативными устройствами / Е. Бирюков, Д. Василенко [Электронный ресурс] – Режим доступа: http://www.compitech.ru/html.cgi/arhiv/05_06/stat_198.htm – 26.01.2014 г.

6. Power Management – Architecture and Driver Support [Electronic resource] – Access mode: http://www.microsoft.com/whdc/system/pnppwr/powermgmt/default.mspx. – 23.01.2014 г.

7. Спиридонов А. Энергосберегающие технологии / А. Спиридонов [Электронный ресурс] – Режим доступа: http://www.mobile-review.com/articles/2006/smart-energy.shtml – 13.12.2013 г.

8. Watts Up? Plug Load Meters [Electronic resource] – Access mode: https://www.wattsupmeters.com/secure/products.php?pn=0 – 13.02.2014 г.

9. Intel® Power Gadget Meters [Electronic resource] – Access mode: http://software.intel.com/en-us/articles/intel-power-gadget-20 – 04.05.2014 г.

10. XCode [Electronic resource] – Access mode: https://developer.apple.com/xcode/ – 07.04.2014г.

11. Energy-Efficient Software Guidelines [Electronic resource] – Access mode: http://software.intel.com/ru-ru/articles/partner-energy-efficient-software-guidelines – 03.02.2014 г.

12. Снижение энергопотребления приложений [Электронный ресурс] – Режим доступа: http://software.intel.com/ru-ru/articles/conserving-active-power – 01.02.2014г.

Сведения об авторах:

Туркин Игорь Борисович – д.т.н., профессор, зав каф. инженерии программного обеспечения Национального аэрокосмического университета им. Н.Е. Жуковского «ХАИ». Научные интересы: инженерия программного обеспечения.

E-mail: energy@d4.khai.edu.

Вдовитченко Александр Валерьевич – аспирант Национального аэрокосмического университета им. Н.Е. Жуковского «ХАИ». Научные интересы: инженерия программного обеспечения.

E-mail: kentsanya91@gmail.com.