САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

На правах рукописи

Сартасов Станислав Юрьевич

УПРАВЛЕНИЕ ЭНЕРГОПОТРЕБЛЕНИЕМ ПРОЦЕССОРА НА ОСНОВЕ СТОХАСТИЧЕСКОЙ ОПТИМИЗАЦИИ

Научная специальность 2.3.5. Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей

Диссертация
на соискание ученой степени кандидата технических наук

Научный руководитель:
доктор физико-математических наук, профессор
Олег Николаевич Граничин

Санкт-Петербург
2023

Оглавление

Введение

На текущий момент задача оптимизации энергопотребления смартфонов, планшетов и других портативных ЭВМ остаётся актуальной как для их пользователей, так и в широком смысле для народного хозяйства. Это связано с тем фактом, что за последние 20 лет изменились паттерны пользовательского доступа к информационным системам и сетевым сервисам: для получения необходимой информации намного чаще используются смартфоны, планшеты и иные виды ЭВМ (далее — *мобильные устройства* или просто *устройства*), поддерживающие режим работы от аккумулятора, а следовательно длительность автономной работы становится одним из основных критериев качества и удобства человеко-машинного интерфейса.

С одной стороны, указанная задача решается электронной и электрохимической промышленностью, разрабатывающей как более энергоёмкие аккумуляторы, так и более энергоэффективные компоненты для таких устройств. К последним относятся центральные процессоры (ЦП), построенные по гетерогенной архитектуре. Например, согласно архитектуре ARM big.LITTLE ЦП состоит из двух кластеров ядер —- менее производительных, но энергоэффективных (кластер LITTLE), и более производительных и энергоёмких (кластер big) [51]. Однако немаловажную роль играет и то, разработано ли системное и прикладное программное обеспечение с учётом особенностей работы этих компонентов от автономного источника питания.

Одной из наиболее проработанных технологий управления энергопотреблением в операционных системах (ОС) является технология динамического масштабирования напряжения и частоты (Dynamic Voltage Frequency Scaling, DVFS). ЦП способен работать на конечном множестве рабочих частот и переключаться между ними в процессе работы. Модуль или регулятор DVFS (DVFS governor) —- это программный модуль ОС, который предписывает ЦП установить ту или иную рабочую частоту в зависимости от наблюдаемого состояния вычислительной системы и ал-

горитма управления частотой. Этот модуль может работать в связке с диспетчером задач ОС или быть независимым от него [103].

Задача оптимизации энергопотребления ЦП состоит в том, чтобы максимизировать производительность и минимизировать затраты энергии, при этом управляющим воздействием является установка рабочей частоты ЦП. Для решения этой задачи можно предложить различные стратегии, но при этом определение лучшей стратегии из множества сопряжено с методологическими трудностями.

На текущий момент существуют два концептуально разных подхода к замерам энергопотребления ЦП. Первый из них, прямой подход, состоит в подключении измерительного прибора к целевому компоненту и анализу результатов измерений при различных рабочих нагрузках. Помимо финансовых затрат и высокой требуемой квалификации исследователей такой подход обладает и тем недостатком, что его не всегда реалистично применить для подключения внешнего мультиметра к цепям питания реального устройства без нарушения его работоспособности. Косвенный подход состоит в замере каких-либо характеристик, опосредованно связанных с энергопотреблением (времени нахождения ЦП на определённой частоте, числа инструкций того или иного вида в рабочей нагрузке), и преобразованию их в данные об энергопотреблении с помощью математической модели. Такой подход существенно дешевле и менее требователен, но при этом точность оценки напрямую связана с качеством модели [78]. Существующие на сегодняшний день модели недостаточно полно описывают поведение современных гетерогенных ЦП. Таким образом, актуальна задача создания новых моделей и методов замера энергопотребления ЦП мобильного устройства для оценки качества работы программных модулей, оптимизирующих энергопотребление.

К настоящему времени разработано несколько подходов к оптимизации энергопотребления ЦП мобильного устройства. Кс. Ли (X. Li), В. Вен (W. Wen) и Кс. Ванг (X. Wang) предложили перед использованием той или иной функциональности включать её специальный профиль в под-

системе DVFS [71]. Использование нейронных сетей для предсказания оптимальной частоты в схожих с текущей ситуациях успешно применяли Й.Л. Чен (Y.L. Chen), М.Ф. Чанг (M.F. Chang), Ч.В. Ю (Ch.W. Yu), Кс.Ж. Чен (X.Zh. Chen), В.Й. Лянь (W.Y. Liang) [33], Дж. Ли (J.Lee), С. Нам (S. Nam) и С. Парк (S. Park) [66]. Подход, заключающийся в преобразовании текущего состояния системы, понимаемом в широком смысле, в прогнозируемую оптимальную частоту посредством эмпирически установленной формулы, встречается в работах К. Пурнамбигаи (K. Poornambigai), М.Л. Раджа (M.L. Raj), П. Мины (P. Meena) [85], Л. Бройд (L. Broyd), К. Никсона (K. Nixon), Кс. Чена (X. Chen), Х. Ли (H. Li) и Й. Чена (Y. Chen) [28]. Среди стандартных модулей, входящих в поставку ОС Android для мобильных устройств, задача оптимизации энергопотребления ЦП решается до определённой степени только в регуляторе schedutil, являющимся частью системы EAS, разработанной Linaro и ARM [60]. Частота выставляется пропорционально оценке энергетической загруженности кластера гетерогенного ЦП исходя из оценочной трудоёмкости задач, назначенных на каждое ядро кластера. Такой подход более энергоэффективен по сравнению с основанными на эмпирических правилах регуляторами OnDemand и Interactive, которые не учитывают энергопотребление ЦП. Однако общим недостатком является недостаточная проработанность теоретических основ предлагаемых методов, особенно при переходе к гетерогенной архитектуре ЦП. Открытым остаётся и вопрос, можно ли дальше снизить энергопотребление при сохранении производительности и отзывчивости человеко-машинных интерфейсов.

Вместе с тем с середины прошлого века развивается подход на основе рандомизированных рекуррентных алгоритмов стохастической оптимизации, начало которому положили процедуры Роббинса-Монро и Кифера-Вольфовица [62,88]. Этот подход хорошо работает во многих задачах, где измерения целевого функционала зашумлены, например, Н.О. Амелиной и А.Л. Фрадковым для формирования консенсуса в стохастических сетях [20], К.С. Амелиным для управления группой квадроко-

птеров [1], Ю.В. Иванским для решения задачи дифференцированного консенсуса [3], А.В. Гасниковым для поиска равновесий в транспортных сетях [5]. Подробно описание подхода сделано в монографии О.Н. Граничина и Б.Т. Поляка «Рандомизированные алгоритмы оценивания и оптимизации при почти произвольных помехах» [11]. В частности, один из наиболее общих алгоритмов, рассмотренных в ней, называется в англоязычной литературе Simultaneous Perturbation Stochastic Approximation (SPSA) и был исследован Дж. Спаллом (J. Spall) [96], Х.-Ф. Ченом (H.-F. Chen), Т. Дункан (T. Duncan), Б. Пасик-Дункан (B. Pasik-Duncan) [31], О.Н. Граничиным [7,9], Б.Т. Поляком и А.Б. Цыбаковым [14]. Идея всех вариаций этого алгоритма состоит в аппроксимации градиента вдоль случайно выбранного направления, и поэтому SPSA относится к разновидности стохастического градиентного спуска. К достоинствам этого подхода следует отнести существенно более слабые ограничения, накладываемые на помехи измерений и целевой функционал, по сравнению с традиционно используемыми методами. В частности, в [7,9] доказана работоспособность алгоритма при почти произвольных помехах (например, при «неизвестных, но ограниченных», unknown-but-bounded).

Задача оптимизации энергопотребления может быть формализована таким образом, чтобы удовлетворять ограничениям этих алгоритмов. Попытка такой формализации и анализ полученного решения были опубликованы в статье О.Н. Граничина и В.Е. Краснощёкова о применимости алгоритмов стохастической оптимизации к оптимизации энергопотребления процессора MP3-плеера [12]. Алгоритм SPSA с одним измерением может быть применён для решения задачи трекинга оптимальной рабочей частоты одноядерного ЦП в условиях нагрузки от 0 до 3 процессов. Однако полученный результат не масштабируется на современные мобильные устройства с гетерогенными ЦП, где количество одновременно запущенных процессов не ограничивается сверху. Работ, посвящённых адаптации алгоритмов стохастической оптимизации для этого случая, до последнего времени не было.

Перечисленные тенденции и проблемы подтверждают актуальность

темы диссертационного исследования.

Целью работы является разработка математического и программного обеспечения для современных мобильных ОС для решения задачи управления энергопотреблением гетерогенного ЦП при работе в широком спектре вычислительных нагрузок. Цель достигается в диссертации через решение следующих задач:

1. Исследовать методы измерения и модели оценки энергопотребления ЦП мобильного устройства, в том числе гетерогенной архитектуры, и связанные с этим особенности взаимодействия программ.

2. Исследовать применимость рандомизированных алгоритмов управления, оптимизации и оценивания параметров для математического и программного обеспечения ОС по управлению энергопотреблением гетерогенного ЦП и разработать алгоритмы управления на их основе.

3. Создать модуль DVFS на базе разработанных алгоритмов управления, встроить его в современную мобильную ОС и оценить качество работы мобильного устройства под его управлением, включая человеко-машинные интерфейсы, в условиях параллельной обработки данных.

Методы исследования. В диссертации применяются методы теорий управления, оптимизации, оценивания, вероятностей и математической статистики; применяются методы системного программирования, стохастической оптимизации, рандомизированные алгоритмы.

Основные результаты:

1. Предложена и обоснована модель оценки энергопотребления ЦП, построенного по гетерогенной архитектуре, учитывающая динамическую вычислительную нагрузку и пребывание в состоянии простоя, даны практические рекомендации по её применению.

2. Разработан подход к решению задачи управления энергопотреблением гетерогенного ЦП на основе рандомизированных алгоритмов стохастической оптимизации, в рамках которого предложены и обоснованы функционалы среднего риска для алгоритмов SPSA с одним и двумя измерениями. Исследована и установлена теоретическая состоятельность оценок, предоставляемых разработанными алгоритмами, в рамках ограничений, накладываемых особенностями работы ЦП.

3. Разработаны модули DVFS, оптимизирующие энергопотребление ЦП с учётом особенностей его работы, основанные на предложенных функционалах и алгоритмах SPSA с одним и двумя измерениями. На базе подготовленных тестовых стендов, управляемых ОС Android, проведено сравнение с существующими аналогами. На модули, основанные на алгоритмах SPSA с двумя измерениями, получно свидетельство о государственной регистрации программ на ЭВМ.

Научная новизна. Все основные научные результаты диссертации являются новыми.

Теоретическая ценность и практическая значимость. Теоретическая ценность работы состоит в исследовании применимости рандомизированных алгоритмов стохастической оптимизации для решения задачи трекинга оптимальной рабочей частоты гетерогенного ЦП на основе его модели энергопотребления в условиях произвольных вычислительных нагрузок и выявлении различных стратегий использования зашумленных измерений.

Разработанное математическое обеспечение, а также стоящие за ним принципы могут быть применены для решения схожих задач по управлению частотой ЦП исходя из других оптимизационных критериев. Введённый понятийный аппарат может быть применён при разработке энергоэффективного программного обеспечения как прикладного, так и си-

стемного уровня, а также для сопровождения программных средств различного назначения.

Созданное во время проведённого диссертационного исследования системное программное обеспечение повышает энергетическую эффективность процессов обработки данных и знаний в вычислительных машинах с гетерогенными ЦП и может быть перенесено на другие ЭВМ, отличные от созданных стендов, а равно послужить основой для других модулей DVFS.

Апробация работы. Результаты диссертации докладывались на Четвёртой конференции по программной инженерии и организации информации (SEIM'20) (г. Санкт-Петербург, Россия, 16 мая, 2020), на Пятой конференции по программной инженерии и организации информации (SEIM'21) (г. Санкт-Петербург, Россия, 17 апреля, 2021), на конференции IEEE 7th International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP'21) (June 23–25, 2021, Krakow, Poland), на конференции 60th IEEE conference on Decision and Control (CDC'21) (December 13–17, 2021, Austin, Texas, USA).

Результаты диссертации были использованы в работах по гранту РНФ 21-19-00516 «Мультиагентное адаптивное управление в сетевых динамических системах с применением к группам робототехнических устройств в условиях неопределенностей» и гранту СПбГУ ID: 94062114.

Публикация результатов. Основные результаты исследований отражены в работах [15,26,27,78,83,92,93]. Соискателем опубликовано 8 научных работ, из которых одна публикация — свидетельство о регистрации программы на ЭВМ, шесть опубликовано в изданиях, индексируемых в базе данных Scopus, и две в журналах, входящих в перечень рецензируемых научных журналов, в которых должны быть опубликованы основные научные результаты диссертаций на соискание ученой степени кандидата наук.

Работы [15, 26, 27, 78, 83, 92, 93] написаны в соавторстве. В работе [78] С.Ю. Сартасову принадлежит методологическое и организационное обес-

печение работы, обзор источников и анализ результатов, соавторам — независимый обзор источников. В [93] С.Ю. Сартасову принадлежит общая постановка задачи, архитектура решения и методологическое обеспечение, соавторам — выбор методов решения и его программирование. В работах [15, 26, 27] О.Н. Граничину принадлежит общее методическое руководство, С.Ю. Сартасову — общая постановка задачи, архитектура функционала среднего риска и решения в целом, методология тестирования, обработка результатов экспериментов, другим соавторам — создание тестового стенда, конкретизация функционала среднего риска, программирование решения и тестовых сценариев, проведение экспериментов. В [92] С.Ю. Сартасову принадлежит общая постановка задачи, выбор методов решения, архитектура решения, обработка результатов экспериментов, соавторам — создание тестовых стендов, программирование решения и проведение экспериментов. В работе [83] О.Н. Граничину принадлежит общее методическое руководство, С.Ю. Сартасову — общая постановка задача, разработка функционала среднего риска, программирование решения, методология тестирования, проведение экспериментов и обработка их результатов, М.А. Пелогейко — модификации функционала среднего риска, программирование решения, подготовка тестового стенда и проведение экспериментов.

На программные модули, разработанные по результатам диссертационного исследования, получены акты о внедрении от ООО «Ланит-Терком» и федерального государственного бюджетного образовательного учреждения высшего образования «Балтийского государственного технического университета «ВОЕНМЕХ» им. Д.Ф. Устинова» (см. Приложение А) и свидетельство о государственной регистрации программ для ЭВМ № 2023666564 от 2 августа 2023 года [16] (см. Приложение В).

Структура и объем диссертации. Диссертация состоит из введения, трех глав, заключения, списка литературы, включающего 110 источников. Текст занимает 97 страниц, содержит 3 рисунка и 12 таблиц.

Краткое содержание работы

Во **введении** обосновывается актуальность диссертационной работы, формулируется цель, ставятся задачи исследования и кратко излагаются основные результаты.

В **первой главе** даётся описание проблемы измерения энергопотребления центрального процессора, сопровождаемое обзором литературы. В разделе 1.1 даётся понятие энергопотребления мобильных устройств, даётся определение задачи оптимизации энергопотребления и рассматриваются существующие подходы к решению этой задачи как на уровне аппаратного обеспечения, так и на уровне операционных систем и отдельных программ и программных систем. Отдельно отмечается, что подтверждением качества решения является сравнение энергопотребления устройства или его компонента до и после его внедрения. В разделе 1.2 рассматривается методология проведения измерительных экспериментов и делается обзор существующих подходов к измерению энергопотребления. Устанавливается разница между измерением и оценкой энергопотребления. Отдельно описываются существующие подходы для энергопотребления ЦП, их достоинства и недостатки, и делается вывод о целесообразности создания новой модели оценки. В разделе 1.3 описана и обоснована новая модель энергопотребления ЦП, учитывающая особенности, накладываемые гетерогенной архитектурой, и использующая авторский подход к учёту времени нахождения ЦП в состояниях простоя.

Во **второй главе** исследуется вопрос применимости рандомизированных алгоритмов стохастической оптимизации для задачи управления энергопотреблением гетерогенного ЦП. В разделе 2.1 вводятся основные понятия теории стохастической оптимизации, описываются наиболее разработанные алгоритмы и их свойства. В подразделе 2.1.1 даётся определение функционала среднего риска и задачи поиска его оптимума. В подразделе 2.1.2 описываются различные варианты алгоритма SPSA и связанные с ними теоретические результаты, в том числе оценки состоятельности. В разделе 2.2 расматривается связь задачи оптимизации энергопотребления процессора с задачей отслеживания изменений (тре-

кинга) оптимальных параметров нестационарного функционала среднего риска, даются теоретические результаты в этой области для алгоритма SPSA. В разделе 2.3 разрабатывается функционал среднего риска для алгоритма SPSA с одним зашумленным наблюдением. В разделе 2.4 разрабатывается функционал среднего риска для алгоритма SPSA с двумя зашумленными наблюдениями. В подразделе 2.4.1 определяется понятие стоимости исполнения программы и на основе этого понятия формируется целевая функция для алгоритма SPSA с двумя измерениями. В подразделе 2.4.2 рассматриваются различные подходы к получению зашумленных наблюдений в рамках цикла DVFS.

Третья глава посвящена разработке регуляторов DVFS на основе принципов, изложенных в Главе 2, и их тестированию. В разделе 3.1 описываются технические аспекты программирования регуляторов SPSA1 и SPSA2 для ОС Android. В разделе 3.2 приводятся методология и результаты экспериментов по оценке качества работы полученных регуляторов. Подраздел 3.2.1 посвящён тестированию регулятора SPSA1, подраздел 3.2.2 — тестированию различных вариантов SPSA2. В разделе 3.3 производится анализ полученных результатов и делается вывод о соответствии свойств полученных регуляторов поставленной цели.

В **заключении** формулируются основные результаты диссертации.

Глава 1. Модель энергопотребления гетерогенного центрального процессора

В этой главе вводятся основные понятия, связанные с архитектурой и энергопотреблением современных мобильных устройств, рассматриваются различные подходы к оптимизации энергопотребления и способы его измерения или оценки, формулируется модель энергопотребления гетерогенного центрального процессора (ЦП).

1.1 Энергопотребление мобильных устройств

Мобильные устройства, такие как смартфоны, планшеты и умные часы, стали неотъемлемой частью современной жизни. Цифровые сервисы, получаемые посредством этих устройств, не только улучшают качество жизни, но и формируют новые виды отношений в социуме. Ожидается, что в 2024 году число активно используемых смартфонов превысит 7 млрд. При этом наиболее распространенной операционной системой (ОС) для мобильных устройств на сегодняшний день является Android [79].

Мобильные устройства — это в первую очередь электрические устройства с автономным источником питания (аккумулятором) и, как следствие, ограниченным временем автономной работы. С физической точки зрения используемые в современных мобильных устройствах литий-полимерные аккумуляторы являются реальными источниками напряжения (РИН). При уровне заряда выше 90% напряжение между контактами аккумулятора выше номинального, от 90% до примерно 20% уровень вы-

даваемого аккумулятором напряжения мало изменяется, а затем начинает резко снижаться. В силу этого и благодаря дополнительным цепям на практике напряжение аккумулятора мобильного устройства принимается постоянным или незначительно меняющимся в рамках эксперимента. Основной характеристикой аккумулятора является ёмкость, измеряемая в ампер-часах (А·ч). Ёмкость, например, в 4000 мА·ч означает, что аккумулятор может обеспечивать для потребителей силу тока в 4 А при его номинальном напржении в течение одного часа, при этом увеличение или уменьшение силы тока пропорционально уменьшает или увеличивает время работы [24].

Разные режимы работы компонентов мобильных устройств, таких как ЦП, экран, модули беспроводной связи Wi-Fi и Bluetooth®, графическая карта и т.д., характеризуются различными уровнями энергопотребления. В частности, ЦП обладает возможностью переключаться между рабочими частотами, и чем ниже выбранная частота, тем ниже энергопотребление. Потребляемая мощность экрана складывается из затрат на формирование пикселов определённого цвета и подсветки экрана. Потребляемая мощность подсветки растёт экспоненциально уровню яркости, однако для схемы расположения пикселов PenTile для матриц AMOLED при отображении зелёного цвета тратится большая мощность по сравнению с красным и синим из-за удвоенного количества зелёных субпикселов на один пиксел [30]. Модули беспроводной связи значительно сильнее нагружают аккумулятор в режимах поиска сетей и подключения к ним по сравнению со стабильной работой с сетью на определённой конфигурации, что приводит к повышенному энергопотреблению в ситуации нестабильного подключения к сети [92].

Системные и прикладные приложения по-разному используют компоненты устройства, но за счёт использования ЦП любая вычислительная нагрузка с неизбежностью разряжает аккумулятор. Таким образом, задача увеличения времени работы мобильного устройства в автономном режиме по сути является задачей оптимизации энергопотребления. Дополнительную актуальность задаче добавляет также и тот факт, что

электрохимические аккумуляторы, используемые в доступных на рынке мобильных устройствах, деградируют с течением времени, то есть их максимальная ёмкость падает с каждым циклом перезарядки [24]. Поэтому дополнительный экономический эффект от решения этой задачи возникает не только благодаря уменьшению потребляемой энергии при перезарядке мобильных устройств с течением времени, но и благодаря возможности увеличить время до покупки нового мобильного устройства по той причине, что у старого время автономной работы уменьшилось до неприемлемого.

С другой стороны, с пользовательской точки зрения важно сохранить комфорт от работы с устройством независимо от выбранной стратегии управления потреблением энергии. Так как уменьшение энергопотребления связано с уменьшением производительности, то, неформально говоря, цель оптимизации состоит в том, чтобы устройство максимально комфортно «тормозило» у пользователя. Однако в связи с тем, что понятие «комфорта» объективно не формализуется, в рамках различных подходов к решению поставленной задачи создаются более формальные критерии оптимизации, которые, по мнению их авторов, связаны с этим понятием.

Классический подход к решению проблемы увеличения времени автономной работы состоит в физическом увеличении ёмкости электрохимических аккумуляторов, и в этом направлении за последние 25 лет наблюдается существенный прогресс: использовавшиеся в начале 2000-х годов в карманных компьютерах никель-кадмиевые аккумуляторы имели ёмкость порядка 1000 мА·ч, в то время как типовая ёмкость литий-полимерных аккумуляторов в современных смартфонах составляет 3500-4000 мА·ч [24]. Другой широко распространённой схемой является выключение или усыпление (гибернация) компонента мобильного устройства, если он долгое время не используется — в таком состоянии его энергопотребление заметно меньше. Результат применения этого подхода чаще всего заметен, когда спустя некоторое время у неиспользуемого мобильного устройства выключается экран, но так же работают драйве-

ры и модулей беспроводной связи, и графических карт [19].

На уровне прикладного программного обеспечения оптимизация энергопотребления достигается путём внедрения в приложения энергетических профилей — совокупностей конфигурационных значений, которые применяются при определённом уровне заряда [90]. Например, при достижении уровня заряда в 20% приложение начинает использовать модуль геопозиционирования (GPS) не постоянно, а раз в 5-60 секунд. Для экономии заряда аккумулятора могут применяться тёмные темы пользовательского интерфейса вместо светлых [38]. Отдельным направлением исследований является обработка вычислительных задач не на устройстве, а в облачной инфраструктуре с передачей результатов обратно на устройство [34,107], что дополняется распространением бессерверной архитектуры приложений [82].

Большое внимание к вопросам энергопотребления уделяется в системном программом обеспечении. Если у компонента мобильного устройства есть технические возможности по оптимизации энергопотребления, то их активно используют на уровне операционной системы.

Современные ЦП способны входить в состояние простоя, во время нахождения в котором приостанавливается выполнение любых программ. Многоядерные процессоры могут переводить одно или несколько своих ядер в состояние простоя, в то время как другие ядра остаются активными. В состоянии простоя снижение энергопотребления достигается за счёт того, что часть аппаратных компонентов процессора отключена. Чем глубже состояние, тем больше компонентов отключено, однако это увеличивает время выхода обратно в активное состояние. В терминах ОС Linux и Android общая задержка входа в состояние простоя и минимальное время, в течение которого аппаратное обеспечение будет в нём находиться, называется целевой резидентностью (target residency).

Например, у двухъядерного процессора ARM CortexA9 [75] есть следующие состояния простоя в порядке увеличения их глубины:

- C0 – активное состояние.

- C1 – большинство таймеров процессора выключены. Задержка выхода составляет 4 мкс.
- C2 — ЦП выключен, блок защиты памяти (Memory Protection Unit, MPU) включен для защиты критических данных, ядро неактивно. Задержка выхода составляет 1100 мкс.
- C3 - аналогично C2, но ядро находится в режиме CSWR. Выходная задержка составляет 1200 мкс.
- C4 - аналогично C3, но ядро находится в режиме OSWR. Выходная задержка составляет 1500 мкс.

Использование технической возможности ЦП переходить в определенное состояние простоя зависит от системы-на-кристалле мобильного устройства и ОС, поэтому на рынке доступны мобильные устройства, способные входить только в состояние C1, в то время как установленный ЦП поддерживает и более глубокие состояния простоя.

Регулятор состояния простоя — это модуль в ядре ОС Android, который может отслеживать текущее состояние системы и отправить сигнал ЦП для входа в некоторое состояние простоя или выхода из него для одного или нескольких его ядер. Существует несколько типовых регуляторов, доступных на устройствах Android, и алгоритмом по умолчанию является Menu. Этот регулятор пытается предсказать текущую продолжительность простоя, затем корректирует полученное значение на основе ряда факторов, а после этого пытается найти самое глубокое состояние простоя, учитывая целевую резидентность и задержку выхода. Прогнозирование текущей продолжительности простоя основано на истории предыдущих простоев.

Установка на мобильные устройства многоядерных процессоров позволило реализовать истинную многозадачность в мобильных ОС. Многие многоядерные процессоры обладают гомогенной архитектурой, то есть вычислительные возможности каждого ядра одинаковы. Использующиеся в настоящее время диспетчеры задач ОС, поддерживающие

такую архитектуру ЦП, не включают в себя дисциплины планирования, направленные на оптимизацию энергопотребления. К таким относится диспетчер Completely Fair Scheduler (CFS, Полностью Справедливый Планировщик). Ситуация существенно изменилась с появлением многоядерных ЦП, обладающих гетерогенной архитектурой. Примером такой архитектуры является big.LITTLE от компании ARM [51]. Ядра разбиты на несколько кластеров, и ядра гомогенны только в пределах своего кластера. Поэтому появляется возможность запускать нетребовательные к вычислительным ресурсам программы на более слабых, но энергоэффективных ядрах (так называемых «малых»), в то время как более мощные в вычислительном и энергетическом плане ядра («большие») используются для приоритетных или ресурсоемких задач.

Energy-aware scheduling (EAS, Энергосберегающий Планировщик) [60] — это планировщик задач, который может корректно работать только в системах с гетерогенными ЦП. Он был разработан для ОС Linux, но поддерживается и в Android. EAS использует нормализованные энергетические модели ядер каждого кластера с учетом доступных частот и энергопотребления. Смоделировать энергетическое поведение для одного ядра кластера достаточно, поскольку внутри кластера ядра гомогенны, и в реальных системах-на-кристалле управление электропитанием и частотой осуществляется для всех ядер кластера одномоментно и унифицированно. Когда необходимо запланировать задачу, EAS выбирает ядро для выполнения этой задачи таким образом, чтобы энергопотребление ЦП увеличивалось как можно меньше. EAS работает, пока загрузка ЦП ниже 80%, в противном случае используется CFS до тех пор, пока нагрузка не уменьшится. В отличие от классических планировщиков, для обеспечения энергосбережения EAS должен иметь возможность подавать кластерам ЦП управляющие сигналы, что делается в других модулях ОС, которые также считаются частью EAS.

Одним из наиболее проработанных подходов является динамическое регулирование частоты и напряжения ЦП (Dynamic Voltage Frequency Scaling, DVFS) [103]. ЦП может работать не на одной частоте, а в неко-

тором диапазоне. Этот диапазон ограничен снизу соображениями удобства пользователя, а сверху — проблемами с теплоотведением, так как с ростом частоты возрастает и тепловыделение. Поскольку существующие механизмы теплоотведения имеют ограничения как по энергопотреблению, так и по форм-фактору, то основной способ охлаждения процессора мобильного устройства — воздушное. На момент написания граница диапазона частот, тепловыделение в котором удаётся эффективно рассеять воздушным охлаждением, составляет около 2 ГГц для имеющиеся на рынке моделей. Хотя в целом рассеиваемая процессором мощность пропорциональна кубу рабочей частоты, в этом диапазоне зависимость между потребляемой мощностью и частотой ещё можно аппроксимировать линейной функцией.

Регулятор DVFS — это алгоритм или модуль ОС, его реализующий[1], который наблюдает за текущим состоянием устройства и посылает сигналы ЦП на увеличение или уменьшение рабочей частоты согласно некоторой стратегии. Необходимо отметить, что сигнал регулятора DVFS носит рекомендательный характер, и поэтому ЦП может отреагировать на этот сигнал с заметной с точки зрения регулятора задержкой.

Существует ряд общедоступных регуляторов DVFS для ОС Android. Некоторые из них специфичны для этой ОС, тогда как другие изначально использовались в ядре Linux [37]:

- Powersave. Этот регулятор устанавливает и удерживает минимальную доступную частоту. Он экономит больше всего энергии, но обеспечивает худшую производительность.

- Performance. Этот регулятор работает с точностью до наоборот: он устанавливает максимальную частоту для лучшей производительности ценой максимального энергопотребления. Performance и PowerSave соответствуют двум экстремальным стратегиям опти-

[1]Здесь и далее в контексте теоретического анализа под регулятором DVFS будет пониматься алгоритм, а в контексте практической реализации — модуль ОС

мизации и зачастую используются в качестве опорных алгоритмов для сравнения.

- OnDemand. Этот регулятор устанавливает частоту ЦП пропорционально максимальной нагрузке, определяемой как отношение активного времени ЦП к общему времени работы, среди ядер ЦП, наблюдаемой между вызовами регулятора, благодаря чему устройство способно реагировать на изменение активности вычислительных процессов. Когда достигается определенный порог нагрузки ЦП (по умолчанию 80%), устанавливается максимальная частота до тех пор, пока нагрузка снова не станет ниже порога.

- Conservative. Этот регулятор представляет собой развитие идей OnDemand: он увеличивает частоту, когда есть активность на ЦП, и снижает до минимума постепенно, а не рывками, характерными для OnDemand.

- Interactive. Этот регулятор разработан специально для ОС Android и учитывает тот факт, что если пользователь начинает взаимодействие с устройством, то он какое-то время будет продолжать это делать дальше. Рабочая частота по-прежнему зависит от уровня нагрузки, но оценка этого уровня происходит не только по истечению фиксированного времени, но и при возникновении определённых аппаратных прерываний, к которым относятся и возникающие в результате взаимодействия пользователя с устройством, например, от прикосновения к экрану.

- schedutil. Этот регулятор предназначен для работы исключительно с диспетчером задач EAS. Его основная идея та же, что и в OnDemand, но определение нагрузки основано на энергетической модели EAS, а не на отношении активного времени к общему времени.

Существуют и качественно иные подходы к этому вопросу.

В отличие от EAS, использующего статическую энергетическую модель, AdaMD реализует процедуру планирования, которая регулярно проверяет состояние различных ресурсов устройства, требующихся на текущий момент исполняемым процессам, и при необходимости переназначает их более подходящим ядрам [23]. По сравнению с другими подходами, привязывающими потоки к ядрам, этот метод позволяет экономить до 28% энергии при удовлетворении 95% требований к производительности.

В последние годы активно развивается направление изучения применимости нейронных сетей для построения регуляторов DVFS общего назначения. Например, для предотвращения проблемы затухания градиента можно использовать эффект «долговременной кратковременной памяти» рекурсивной сети, поскольку управляющие сигналы должны формироваться только для данных, полученных в течение недавнего времени [66]. Такая архитектура сети снижает энергопотребление процессора максимум на 19% по сравнению со стандартными регуляторами DVFS.

Алгоритм DVFS может быть построен с использованием других оптимизационных критериев, нежели производительность или энергоэффективность. Температура кристалла ЦП является еще одним важным фактором, так как повышение температуры влечёт за собой увеличение энергопотребления. За счет использования нейронной сети в цикле DVFS средняя температура кристалла может быть снижена на 18 °C с минимальными затратами на выполнение алгоритма и сопоставимой производительностью [87]. Температура кристалла может сочетаться с энергоэффективностью в качестве критерия оптимизации, и использование нейронной сети глубокого обучения с подкреплением для определения температуры ЦП и оценки температуры окружающей среды может привести к снижению энергопотребления на 23,9% при сохранении требуемого уровня производительности [64].

Алгоритмы DVFS могут строиться и для использования с конкретны-

ми приложениями, а не для общего случая. Например, регулятор DVFS для приложений дополненной реальности, учитывающий требования по частоте кадров и времени отклика, в теории может уменьшить энергопотребление на 80%, но экспериментально это ещё не подтверждено [95].

DVFS применяется и для графического процессора (ГП, Graphical Processing Unit, GPU), который применяется для разгрузки процессора от ряда вычислительных задач, чаще всего связанных с графическими алгоритмами. Он также может работать на разных частотах, поэтому существует концепция единого алгоритма DVFS для обоих устройств. Модель, основанная на правилах, может снизить энергопотребление на 18,11%, а частота кадров смартфона снижается на 3,1% [80]. Другой подход заключается в объединении данных о нагрузке, энергии и температуре для ЦП, ГП и ОЗУ и назначении частоты для каждого компонента с помощью общего списка приоритетов. Этот метод экономит не менее 26,8% энергии по сравнению с регуляторами по умолчанию и современными подходами [39].

Существующие подходы не лишены определённых недостатков. Для начала стоит отметить, что адаптация существующих регуляторов DVFS, включая классические, к работе на гетерогенной архитектуре ЦП сделана путём интерпретации каждого гомогенного кластера как отдельного многоядерного процессора. В силу схемотехнических особенностей как кристаллов, так и систем-на-чипе для всех ядер одного кластера задаётся одна и та же рабочая частота. Таким образом, в теории результатом работы регулятора на каждой итерации цикла DVFS является рабочая частота из списка доступных для кластера частот, на которой он будет работать до следующей итерации, где она, возможно, изменится. На практике среди рассмотренных регуляторов ни один не учитывает тот факт, что кластер может переключать частоту в течение большего времени, чем длительность одной итерации цикла DVFS, что приводит к потере смысла для части управляющих воздействий. Например, в ходе исследования было обнаружено, что хотя типовой номинальной длиной цикла DVFS в ОС Android является 10 мс, реальное время переключе-

ния кластеров ЦП Helio G90T может достигать 30 мс. Таким образом, при успешном запросе на переключение частоты на некоторой итерации цикла DVFS результаты работы до двух последующих циклов DVFS игнорируются. С технической точки зрения было бы полезно учитывать этот факт в работе регуляторов.

Другой проблемой является тот факт, что классические алгоритмы вроде OnDemand, Interactive и EAS+Schedutil оптимизируют энергопотребление на основе эмпирических наблюдений и моделей, в том числе и в модели изменения нагрузки. Главная цель проведённого диссертационного исследования состояла в построении регулятора DVFS, основанного на принципах работы с доказанной состоятельностью оценок к оптимальному значению в условиях неопределённости нагрузки от итерации цикла DVFS к итерации.

1.2 Измерение и оценка энергопотребления, методология экспериментов

Методологически сравнение двух подходов DVFS с точки зрения оптимизации энергопотребления делается следующим образом: на одном и том же устройстве, находящемся перед началом эксперимента в одном и том же состоянии, проводится одна и та же по порядку и во времени последовательность действий, при этом процессы ОС запускаются в одно и то же время и трасса их исполнения одинакова. После этого собираются данные об энергопотреблении устройства или его отдельного компонента во время прохождения последовательности действий. Такая экспериментальная методология требует решения ряда вопросов.

Прежде всего, добиться идеального совпадения состояния устройства перед началом экспериментов и в процессе работы невозможно из-за невозможности контролировать ряд факторов. Например, срабатывание аппаратного прерывания может повлиять на распределение задач между ядрами диспетчером задач ОС. В литературе сложилось видение того,

как минимизировать влияние внешних факторов до и во время эксперимента [78]:

- Использование автоматизированных тестов вместо ручных [36].
- Снижение активности фоновых процессов (как системных, так и несистемных) [29, 52, 59, 81, 89, 102].
- Повышение системного приоритета приложения, являющегося целью эксперимента [52].
- Уменьшение яркости экрана или его полное отключение [29, 53, 55, 58, 81, 89, 100].
- Отключение всех неиспользуемых компонентов системы (Wi-Fi, 4G, GPS, акселерометра, дальномера и т.д.) [17, 29, 59, 81, 91].
- Зарядка аккумулятора до одинакового уровня [22, 29, 100].
- Выключение устройства для охлаждения батареи [29].
- Прогрев путём исполнения тестов без замера энергопотребления [77].
- Выжидание перед началом нового теста для завершения или снижения активности фоновых процессов [42, 89].
- Переустановка исследуемого приложения [53].
- Очистка кэша исследуемых приложений [40, 89].
- Перезагрузка устройства [91].
- Сброс до заводских настроек [22, 91].

Тем не менее невозможно полностью избежать влияния всех внешних факторов на проведение эксперимента, поэтому их проводят несколько раз и результаты замеров статистически обрабатывают.

С экспериментальной точки зрения переключение устройства на испольование другого алгоритма DVFS хорошо изолируется от других факторов в силу организации системы DVFS в ОС Android.

Значительно менее простым вопросом является способ измерения энергопотребления устройства или его отдельного компонента. С концептуальной точки зрения существующие подходы можно разделить на два класса:

- **Подходы прямого измерения** состоят в непосредственном измерении энергопотребления устройством или компонентом посредством специальной аппаратуры.
- **Подходы косвенного измерения** измеряют какие-либо вторичные метрики, связанные с энергетическими характеристиками устройства, и на их основе оценивают итоговое энергопотребление. В основе косвенных методов всегда лежит некоторая модель, связывающая измеряемые метрики и итоговое энергопотребление, поэтому этот класс подходов можно также называть основанными на моделях.

В рамках **подходов прямого измерения** к целевому устройству подключается аппаратура для измерения напряжения, тока или мощности. Такие подходы работают с физическими устройствами и принципиально не могут работать с эмуляторами. В рамках этого класса подхода можно выделить следующие:

Внешнее измерительное устройство: К контактам аккумулятора устройства или, что значительно сложнее, к цепям питания интересующего компонента подключается цифровой мультиметр. Иногда для компенсации падения напряжения в литий-полимерных батареях при разряде и, следовательно, для нормализации значений мощности вместо аккумулятора тестируемое устройство запитывается от внешнего источника постоянного напряжения. В рамках сценария (скрипта) эксперимента запускается как рассматриваемое приложение или модульный тест, так и

запись измерений мощности. Так как мультиметр работает в дискретном времени и усредняет силу тока или мощность за прошедший такт, общее энергопотребление можно оценить с помощью линейной интерполяции:

$$E = \sum_{i=1}^{N_{read}-1} U_i \cdot \frac{I_{i+1} + I_i}{2} \cdot (t_{i+1} - t_i),$$

где E — общее значение энергии, N_{read} — число измерений мультиметра, U_i — i-ое измерение напряжения, I_i — i-ое измерение силы тока, t_i — отметка времени i-го измерения. Некоторые мультиметры могут записывать отметку времени напрямую, для других разность $t_{i+1} - t_i$ может быть определена как величина, обратная частоте мультиметра. Вместо линейной интерполяции может быть использована ступенчатая.

Если необходимо более детальное измерение, например, на уровне энергопотребления отдельных методов программы или блоков кода, то исходный код дополняется инструкциями по протоколированию начала и конца тела метода или блока кода (инструментуется), и записываются две трассы данных — показания мощности и трасса выполнения приложения, как, например, описывает Куто (Couto) и др. [36]. В этом случае перед началом экспериментов системные часы на устройстве с одной стороны и мультиметре или управляющем компьютере с другой должны быть синхронизированы. Затем показания мощности можно сопоставить с трассой выполнения и получить представление об энергоёмкости метода или блока кода. Так как для инструментирования приложения требуется дополнительный программный код, энергозатраты на его исполнение должны быть оценены и вычтены из окончательного результата [29, 35, 41, 53, 63, 65, 69, 99, 105].

Внутреннее измерительное устройство: этот подход использует внутренние измерители мощности, установленные на устройстве производителем, и API ОС Android для доступа к ним. Часто конфигурация измерительного устройства такова, что измеряется энергопотребление устройства в целом, поэтому измерение энергопотребления отдельного компо-

нента требует дополнительной методологической проработки эксперимента. Вместо аккумулятора также может использоваться внешний источник постоянного напряжения. Поскольку устройство и датчик мощности используют одни и те же системные часы, проблем с синхронизацией времени не возникает. Запросы показаний мощности могут быть частью кода инструментирования. Методология получения итогового значения потреблённой энергии не отличается от таковой для подхода с внешним измерителем [32, 45, 57, 59, 65, 102, 105].

Необходимо отметить, что перечисленные стратегии направлены на измерение энергии, потраченной устройством или отдельными компонентами, но изменене уровня заряда аккумулятора позволяет оценить количество энергии, отданной устройству, тем более, что такие датчики общедоступны на мобильных устройствах. На практике измерение уровня заряда аккумулятора оказывается менее предпочтительным, чем измерения потраченной энергии, по следующим причинам:

- Точность оценки датчиков заряда до десятых долей процента оказывается недостаточной для экспериментов длительностью в секунды и минуты.
- Каждый цикл заряда аккумулятора уменьшает его реальную ёмкость, поэтому изменение заряда на 1% может означать значительно разное количество отданной энергии даже в рамках одной серии экспериментов. Первоначальное и последующие определения реальной ёмкости аккумулятора требуют дополнительных трудозатрат.

Подводя итог подходам прямого измерения, следует сказать, что их главным достоинством является именно измерение энергопотребления конкретного физического устройства, а не оценка. Главным недостатком являются финансовые затраты для экспериментаторов и более высокий

требуемый уровень квалификации для создания тестового стенда[2]. Кроме того методология экспериментов в рамках данного подхода должна учитывать ряд особенностей работы как с мультиметрами, так и с самими устройствами:

1. Даже в пределах одной модели различные устройства могут обладать разным энергопотреблением. Хотя в задокументированных случаях это не повлияло ни на внутреннюю согласованность результатов экспериментов на одном устройстве, ни на общие выводы по всех устройствам, потенциально этот риск надо учитывать [110].

2. Подключение измерительных приборов к устройству существенно затруднено в случае, если надо измерить энергопотребление отдельного компонента, — вплоть до использования специализированных плат для тестирования, например, Odroid-A [94], вместо реального устройства. Поскольку аппаратная конфигурация таких плат делается похожей на доступные на рынке мобильные устройства, и в них используются типовые версии мобильных операционных систем, такая замена устройства для тестирования правомерна.

3. Рабочие частоты современных процессоров и периферийных устройств находятся в гигагерцовом диапазоне. Однако измерительные приборы, использованные в литературных источниках работали на значительно более низких частотах. Максимальная частота в 100 КГц задокументирована у Уилка (Wilke) и др. [105, 106], в других работах используются мультиметры, работающие на частотах ниже 10 КГц. Поскольку мультиметры усредняют переменный ток между синхронизирующими импульсами, пики мощности меньшей длительности могут быть сглажены до уровней ошибок измерения и

[2]Хотя при работе над этим исследованием ни одно экспериментальное устройство не было повреждено из-за неправильной сборки тестового стенда, риски такого развития событий и последующих материальных потерь были вполне реальны.

остаться незамеченными. В литературе известны следующие способы учёта этого феномена в методологии эксперимента:

- Миттал (Mittal) и др. [76], Долезал (Dolezal) и Бекват (Becvar) [42], Саксонов (Saksonov) [91], Пандийян (Pandiyan) и Ву (Wu) [81] отключили DVFS или установили рабочую частоту на постоянном уровне, сделав тем самым энергопотребление ЦП более стабильным. Этот метод не подходит в случае, если целью эксперимента является сравнение энергопотребления устройства под управлением различных модулей DVFS.
- Хунг (Hung) и др. [56], Юн (Yoon) и др. [109], Чжан (Zhang) и др. [110] включают данные о частоте процессора в качестве дополнительных входных данных в формулу для оценки общего энергопотребления.
- Куто (Couto) и др. [36], Ли (Li) и Галлахер (Gallagher) [70] устанавливают время выполнения теста таким образом, чтобы его длительность позволила усреднить оценку энергопотребления даже при использовании низкочастотного оборудования.
- Ларссон (Larsson) и Штигелид (Stigelid) [65] регулируют частоту обращений к внутреннему мультиметру за данными об энергопотребления в зависимости от длительности теста. По результатам диссертационного исследования использование этого подхода признано контрпродуктивным и не рекомендуется, так как для модулей Wi-Fi и Bluetooth была экспериментельно показана экспоненциальная зависимость регистрируемого энергопотребления электронного компонента от частоты обращения к внутреннему мультиметру. Тем не менее после калибровки можно ввести поправочный коэффициент, который позволит исключить влияние этого фактора [92].

Подход косвенного измерения (или подход, основанный на моде-

ли) предполагает, что профилирующе программное обеспечение собирает некоторую информацию о выполнении кода и соотносит ее с потреблением энергии посредством математической модели. Работа осуществляется в два этапа: калибровка модели и оценка энергопотребления. На первом этапе коэффициенты модели определяются или настраиваются с помощью вспомогательных экспериментов или справочных данных. Как сказано выше, это важно не только для разных моделей устройств, но и для разных устройств одной модели [53]. После того, как модель настроена под устройство, её можно использовать для оценки энергопотребления на основе полученных в ходе экспериментов косвенных данных.

В зависимости от типа входных данных модели можно классифицировать следующим образом:

Модели времени работы: в рамках этого подхода результатом экспериментов является информация о времени работы различных компонентов устройства [21, 22, 36, 40, 42, 46, 55, 56, 58, 63, 67, 68, 74, 76, 89, 91, 94, 100, 109, 110]. Разным режимам работы могут соответствовать отличающиеся профили энергопотребления. Например, модуль Wi-Fi потребляет разное количество энергии в режиме ожидания, при поиске сети и при передаче данных по сети, а у экрана такими профилямми являются различные уровни энергопотребления при различных уровнях интенсивности подсветки. Оценка энергопотребления рассчитывается как

$$E = \sum_{i=1}^{N_{dev}} \sum_{j=1}^{N_{P_i}} P_{ij} \cdot t_{ij},$$

где E — общее значение энергии, N_{dev} — число компонентов смартфона, участвующих в оценке, N_{P_i} — число профилей энергопотребления i-го устройства, P_{ij} — усреднённое значение потребляемой мощности для j-го режима работы i-го устройства, t_{ij} — общее время работы i-го устройства в j-м режиме работы.

На этапе калибровки проводятся эксперименты по определению профилей энергопотребления для отдельных компонентов с использовани-

ем какого-либо из подходов прямого измерения. С концептуальной точки зрения подходят как внешние, так и внутренние измерительные приборы. Для извлечения параметров мощности, используемых в качестве весовых коэффициентов применяется линейная регрессия. В то же время сами производители устройств могут указывать данные об энергопотреблении устройств в специальном файле профилей мощности (power profile) ОС Android [86].

Для оценки энергопотребления в экспериментах измеряется активное время работы каждого интересующего устройства во время выполнения тестовой нагрузки. Определённые технические неудобства возникают из-за того, что различные компоненты по-разному сообщают о своём времени работы, например, в ОС Android ЦП хранит информацию о времени, проведённом на различных рабочих частотах в папке `proc`, а модуль Wi-Fi генерирует системные события при переходе из одного профиля энергопотребления, связанного с текущим состоянием компонента, в другой. В модель также могут быть включены дополнительные поправочные коэффициенты, например, поправки на скорость разряда батареи [94].

Модели энергопотребления инструкций: эта группа моделей оценивает энергопотребление путём корреляции между отдельными инструкциями или группами инструкций ЦП — условными операторами, инструкциями по управлению циклами, вызовами методов, целочисленными операциями, операциями с плавающей запятой и т. д. — и энергией, которая тратится на их исполнение [18, 52, 70]. Потребление энергии рассчитывается как

$$E = \sum_{i=1}^{N_{instr}} P_i \cdot n_i,$$

где E — общее значение энергии, N_{instr} — число различных инструкций или групп инструкций в модели, P_i — потребляемая мощность одной i-й инструкции или инструкции из i-й группы, n_i — количество i-й инструкции или инструкций из i-й группы в исполненном коде.

Калибровка модели выполняется путем измерения энергопотребления каждого типа инструкций в синтетических тестах с использованием прямого подхода. Общая энергия рассчитывается на основе статистики инструкций в трассе выполнения кода эксперимента. Следует отметить, что реальный запуск тестового кода под ОС Android не требуется, если он не требует вызова Android API. Статистика инструкций может быть агрегирована в любой подходящей среде.

Модели энергопотребления методов или вызовов API: этот подход аналогичен предыдущему, но вместо профиля мощности для одной инструкции рассчитывается энергопотребление системного вызова, вызова API, или метода программной платформы [44, 54, 77, 104]. Модели при таком подходе создаются в предположении, что большая часть времени и энергии тратится вне кода приложения, и поэтому можно получить достаточно хорошую оценку энергопотребления приложения, проанализировав использование его API.

Подводя итог различным вариантам в рамках подхода косвенного измерения, надо отметить, что на практике модели времени работы используются чаще других видов моделей. С одной стороны это связано с тем, что время нахождения компонента устройства в каком-либо состоянии достаточно просто регистрировать, и в ОС Android уже реализовано соответствующее API для различных компонентов. С другой стороны, модели энергопотребления инструкций и методов не учитывают всех нюансов архитектуры ЦП, например, конвейеризации, разделяемых между несколькими ядрами ЦП кэшей, аппаратных технологий вытесняющей многозадачности, например, Intel HyperThreading [72] и т.д., и потому к их точности стоит относиться с осторожностью. Впрочем, аргумент о точности калибровки модели в принципе справедлив ко всему подходу, так как любая модель так или иначе опирается на прямые измерения. Безусловным достоинством подхода является его стоимость и меньшая требовательность к квалификации экспериментаторов.

Для оценки энергопотребления ЦП корпорация Google, поддержи-

вающая ОС Android, рекомендует использовать модель времени работы [73]. Согласно этой модели, ЦП, работая на определённой частоте, при постоянном напряжении потребляет постоянную (усреднённую) силу тока. Поэтому общая энергия оценивается как

$$E = U \sum_{i=1}^{n_f} I(f_i) \cdot t_{f_i}, \tag{1.1}$$

где n_f — число частот, доступных ЦП, f_i — конкретная частота, $I(f)i)$ — сила постоянного тока, потребляемая на этой частоте, t_{f_i} — общее время, которое ЦП работал на частоте f_i. Более того, уравнения без труда расширяется и на гетерогенную архитектуру:

$$E = U \sum_{c=1}^{n_c} \sum_{i=1}^{n_{cf}} I(f_{ci}) \cdot t_{f_{ci}}, \tag{1.2}$$

где n_c — число кластеров гетерогенной архитектуры, а частоты и времена учитываются для каждого кластера отдельно.

Данные для уравнения (1.2) доступны для получения в ОС Android. В частности, значения $t_{f_{ci}}$ для каждого ядра хранятся в специальных файлах `time-in-state` в директории `/sys`. Данные о времени хранятся построчно для каждой частоты в формате "`<frequency><time>`". Количество строк равно n_{cf}. Поскольку кластеры ЦП зарегистрированы в ОС как отдельные процессоры, состав и содержание строк в файлах `time-in-state` меняется от ядра к ядру в зависимости от кластеров, к которым они принадлежат. Время измеряется в интервалах по 10 мс, и отсчёт начинается с момента, когда соответствующий драйвер ОС запускается или перезапускается. Весовые коэффициенты $I(f_{ci})$ хранятся в файле профилей энергопотребления `power_profile.xml` в мА, о котором упоминалось выше. По спецификации Google, этот файл должен

Таблица 1.1: Частоты кластеров ЦП смартфона Xiaomi Redmi Note 8 Pro

A55			A76		
f (Гц)	I (мА)	I/f (мА/ГГц)	f (Гц)	I (мА)	I/f (мА/ГГц)
2000000	90,04	45,02	2050000	324,33	158,21
1933000	85,8	44,39	1986000	307,98	155,08
1866000	80,27	43,02	1923000	291,52	151,60
1800000	72,77	40,43	1860000	269,61	144,95
1733000	66,61	38,44	1796000	247,53	137,82
1666000	62,05	37,24	1733000	233,56	134,77
1618000	58,95	36,43	1670000	209,73	125,59
1500000	52,33	34,89	1530000	177,39	115,94
1375000	44,83	32,60	1419000	152,46	107,44
1275000	39,69	31,13	1308000	130,33	99,64
1175000	35,5	30,21	1169000	105,19	89,98
1075000	31,24	29,06	1085000	91,11	83,97
975000	27,86	28,574	1002000	79,53	79,37
875000	25	**28,571**	919000	70,65	76,88
774000	23,5	**30,36**	835000	61,38	73,51
500000	19,55	**39,10**	774000	56,85	73,45

включаться производителем в сборку ОС Android на устройстве, но, к сожалению, производители этим нередко пренебрегают [93]. Тем не менее, в случае отсутствия файла на выбранном устройстве подойдут и константы с других устройств с такой же моделью ЦП и с такой же или подобной топологией кластеров. Пример содержимого такого файла для ЦП представлен в Табл. 1.1.

В общем случае оценка энергопотребления в мА·ч не является полноценной из-за непостоянного напряжения. Но поскольку считается, что в рассматриваемом случае напряжение на ЦП постоянно, и этим постоянным множителем пренебрегают. Более того, номинальное напряжение мобильных ЦП на современном рынке составляет 1В, таким образом, оценка, полученная по этой модели в мА·ч и переведённая в А·с, на практике оказывается численно равна оценке энергопотребления в Дж.

1.3 Построение модели энергопотребления гетерогенного центрального процессора

Как отмечалось выше, хотя базовая модель энергопотребления, предложенная Google, находит своё применение в исследовательских работах, она не учитывает нахождение ЦП в состоянии простоя. Более того, изучение исходного кода ядер ОС Linux и Android показало, что содержимое файлов `time-in-state` формируется без учёта работы регуляторов состояний простоя. Для решения этой проблемы предлагается следующая модификация базовой модели.

В уравнении (1.2) неявно предполагается, что ЦП, находясь на определенной частоте, потребляет постоянную мощность вне зависимости от того, ведутся ли на нём активные вычисления или нет. Это предположение подтверждается реализацией так называемой задачи бездействия (idle task). Когда планировщик ОС не может запланировать активную задачу для ядра, но при этом ядро находится в состоянии C0 (активно), планировщик назначает специальную задачу бездействия, которая состоит из последовательности инструкций NOP (No Operation, нет операции). Однако уже состояние простоя C1 выключает таймеры ядра, и вычислительный процесс не может быть на нём обработан. Учитывая, что на всех устройствах, использованных в этом исследовании, были доступны только состояния C0 и C1, а также тот факт, что современная реализация гетерогенной архитектуры предполагает наличие нескольких кластеров, все ядра в рамках которых функционируют на одной рабочей частоте, уравнение (1.2) можно изменить следующим образом:

$$E = \sum_{i=1}^{N_{cl}} E_{idle_i}(t) + \sum_{i=1}^{Ncl} U_i \cdot \sum_{j=1}^{N_{f_i}} I_i(f_j) \cdot \sum_{k=1}^{N_{cores_i}} t_{f_{jk}}, \tag{1.3}$$

где N_{cl} — число кластеров ядер, $E_{idle_i}(t)$ — потребляемая энергия вклю-

ченного кластера ЦП в отсутствие активных вычислительных процессов, N_{f_i} — количество доступных частот в i-ом кластере, U_i — номинальное напряжение ядер в i-ом кластере, $I_i(f)$ — номинальная сила тока для ядра в i-ом кластере, работающего на выбранной частоте f, N_{cores_i} — число ядер в i-ом кластере, $t_{f_{jk}}$ — время, проведённое k-ым ядром i-го кластера на j-ой частоте.

Можно заметить, что предложенная модель отличается от базовой модели Google в том, что вместо констант энергопотребления для всего кластера используются константы энергопотребления отдельного ядра. Обоснованием правомерности такой замены служит следующий эксперимент со смартфоном Samsung Galaxy A3 (2016), в котором установлен процессор Exynos 7578 с 4 ядрами Cortex-A53 и максимальной тактовой частотой 1,5 ГГц [15]. Питание от аккумулятора было заменено на питание от источника постоянного напряжения в 3,85 В, и в цепь был добавлен амперметр. Ядра со второго по четвёртое были отключены с помощью системных вызовов ОС Android, а первое обрабатывало только фоновые процессы. Периферийные устройства были отключены или переведены в энергосберегающий режим. Затем на первом ядре был запущен бесконечный вычислительный цикл (разложение ряда натуральных чисел на простые множители полным перебором). Поток, исполняющий программу, был закреплён с помощью системных вызовов ОС Android для исполнения только на первом ядре. Затем, не останавливая первый поток, включалось второе ядро, и такая же программа исполнялась с привязкой к нему. Так ядра включались последовательно вплоть до четвёртого и фиксировалось значение потребляемого тока. Эксперимент проходил в два этапа — с регулятором Performance и регулятором Powersave для максимизации и минимизации производительности и энергопотребления. Результаты замеров представлены на Рис. 1.1.

Стоит заметить, что точке 0 соответствует базовое энергопотребление всего устройства в состоянии покоя и протекания фоновых процессов. Такой график энергопотребления получался при любом порядке включения ядер. Из графика видно, что хотя зависимость энергопотребления

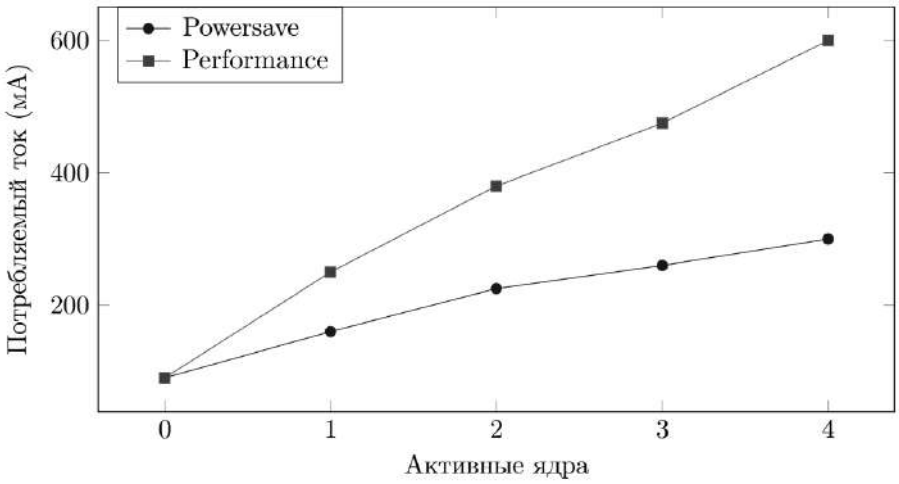


Рис. 1.1: Энергопотребление смартфона Samsung Galaxy A3 (2016) в зависимости от числа активных ядер

от числа активных ядер не является строго линейной, с эмпирической точки зрения она к таковой достаточно близка. Поэтому в этом исследовании переход от констант энергопотребления уровня кластера к константам энергопотребления уровня одного ядра считается оправданным и за неимением более детальнойц информации о работе кластера делается простым делением соответствующей константы кластера на число ядер в нём.

Эта модель сложнее модели, описываемой в (1.2), однако по ряду практических соображений её можно упростить. В первом слагаемом предполагается, что энергопотребление покоя каждого кластера линейно зависит от времени работы. Хотя логично предположить, что для более точного расчета энергопотребления в состоянии простоя необходимо разделить $E_{idle_i}(t)$ на две составляющие (базовое энергопотребление ЦП и энергопотребление в состоянии C1 для каждого ядра), ввиду отсутствия информации о базовом профиле энергопотребления ЦП и о профиле C1 как в `power_profile.xml`, так и в публично доступных спецификациях на ЦП и на систему-на-чипе для использованных в этом исследовании устройств, мы опускаем это слагаемое из дальнейшего анализа.

Далее, вычисление

$$\sum_{k=1}^{Ncores_i} t_{f_{jk}}$$

представляется избыточным, так как ядра одного кластера всегда работают на одной частоте. Однако этот терм необходим в том случае, когда мы пытаемся оценить влияние глубоких состояний простоя на потребление энергии.

Хотя в состоянии простоя останавливаются таймеры ядра ЦП, оно не меняет рабочую частоту ядра саму по себе, то есть ядро, работавщее на частоте 1 ГГц до входа в состояние простоя, будет продолжать работать на частоте 1 ГГц и после выхода из него, если рабочая частота всего кластера не изменилась за это время. Исходное предположение состояло в том, что регуляторы простоя вводят ядро в одно из состояний простоя, когда задач настолько мало, что диспетчер не может нагрузить работой все доступные ядра, и в такой ситуации регулятор DVFS устанавливает минимальную рабочую частоту для кластера. Следовательно, первоначальная идея модели заключалась в том, чтобы вычесть время простоя в C1 из времени, проведенного ядром на самой низкой частоте:

$$E = \sum_{i=1}^{Ncl} U_i \cdot (\sum_{k=1}^{Ncores_i} I_i(f_1)(t_{f0k} - t_k^{idle}) + \sum_{j=2}^{Nf_i} I_i(f_j) t_{fjk}),$$

где t_k^{idle} — время, проведённое k-ым ядром в состоянии простоя. Однако это предположение не подтвердилось, так как в ходе экспериментов были обнаружены ситуации, когда $t_{f0k} < t_k^{idle}$. Это значит, что ОС усыпляла ядро на частоте, отличной от минимальной. Объяснить этот феномен можно тем, что в условиях, когда переключение рабочей частоты кластера может занимать даже больше времени, чем длина целевой резидентности, а нагрузка на отдельном ядре за время последнего цикла DVFS может составлять 100%, у ряда регуляторов DVFS может и не быть оснований понижать частоту для кластера до минимальной. Например, так могут взаимодействовать регулятор DVFS OnDemand и регулятор про-

стоя menu. OnDemand управляет частотой всего кластера, поэтому его оценка загруженности кластера определяется как мажоранта загруженности всех ядер кластера, а значит в ситуации, когда нагружено только одно ядро, регулятор установит высокую частоту всему кластеру. В это же время menu может усыпить ядра с низкой нагрузкой.

Хотя технически возможно модифицировать ядро ОС Android и отслеживать, на какой частоте ядро было приостановлено, это достаточно дорогостоящий путь. Каждая отдельная сборка ОС Android может потребовать по-разному модифицировать одну и ту же функциональность. Для реализации более простого подхода будем считать, что управление состоянием простоя не зависит от работы регуляторов DVFS, но зависит от работы диспетчера задач ОС, и ядро может быть переведено в спящий режим на любой частоте в любой момент времени. В рамках модифицированной модели энергопотребления предлагается вычитать из каждого отдельного времени, проведённого ядром на конкретной частоте, такую долю общего времени простоя, которую это время работы на частоте составляет от общего времени работы:

$$E = \sum_{i=1}^{Ncl} U_i \cdot \sum_{j=1}^{Nf_i} I_i(f_j) \cdot \sum_{k=1}^{Ncores_i} (t_{fjk} - t_k^{idle} \frac{t_{fjk}}{\sum_{m=1}^{Nf_i} t_{fmk}}).$$

Этот подход не является точным, но даёт хорошую оценку фактического энергопотребления. Вместе с тем, данные о времени нахождения в состоянии простоя несложно получить из ОС Android аналогично информации о времени работы ядер на определённой частоте. Так как эта модель была разработана во второй половине исследования, для представления полученных результатов используется как исходная базовая, так и модифицированная модель.

Глава 2. Рандомизированные алгоритмы стохастической оптимизации в задаче оптимизации энергопотребления мобильного устройства

В этой главе даётся обзор рандомизированных алгоритмов стохастической оптимизации и показывается их применимость для разработки энергосберегающих регуляторов DVFS. Вводится понятие стоимости исполнения программы, и для различных вариантов алгоритма одновременно возмущаемой стохастической аппроксимации вводятся целевые функционалы среднего риска.

2.1 Рандомизированные алгоритмы стохастической оптимизации

2.1.1 Функционал среднего риска

Во многих задачах управления поведение целевой системы может быть описано в виде эмпирического функционала качества, усреднение которого определяет *функционал среднего риска*. На основе экстремумов этого функционала осуществляется выбор оптимального управляющего воздействия.

Более формально, пусть распределение вероятностей $P_w(\cdot)$ порождает последовательность p-мерных случайных векторов $\{w_n\}$ из $\mathbb{R}^p$. Тогда нахождение минимума функции $f(\cdot)$ минимизирует функционал средне-

го риска, если функция имеет вид

$$f(x) = E_w F(x, w) = \int_{\mathbb{R}^p} F(x, w) P_w(dw),$$

где $F(x, w) : \mathbb{R}^q \times \mathbb{R}^p \to \mathbb{R}$ — дифференцируемая по первому аргументу функция потерь, $E_w\{\cdot\}$ — математическое ожидание по $P_w(\cdot)$.

Когда функция распределения $P_w(\cdot)$ неизвестна, задачу можно попробовать решить в тех случаях, когда существует возможность в заданных точках $\{(x_n, w_n)\}$ произвести наблюдение (измерение), возможно, с дополнительными помехами, или значений функции $F(x_n, w_n)$, или значений вектора–градиента $\nabla_x F(x_n, w_n)$. Предполагается, что экспериментатор может задавать или наблюдать значения последовательности $\{x_n\}$, а соответствующие значения последовательности случайных величин $\{w_n\}$ из $\mathbb{R}^p$ порождаются одинаковым распределением $P_w(\cdot)$. При этом значения $\{w_n\}$ могут быть неизвестны экспериментатору, или он не может оказывать на них влияния.

Измерения значений функции $F(x_n, w_n)$ в общем случае могут производиться с некоторой аддитивной ошибкой $v_n \in \mathbb{R}$ (шумом наблюдения). Если помехи v — центрированные и независимые, то вектор w может быть расширен дополнительной компонентой v:

$$\bar{w} = \begin{pmatrix} w \\ v \end{pmatrix}.$$

Тогда вместо $F(x, w)$ можно рассматривать новую функцию $\bar{F}(x, \bar{w}) = F(x, w) + v$, схема наблюдения за которой не будет включать дополнительных возмущений, а взамен неизвестного распределения $P_w(\cdot)$ будет фигурировать новое неизвестное распределение $P_{w,v}(\cdot)$. Но так упрощать схему наблюдений, если ошибки измерения не обладают «хорошими» статистическими свойствами, нельзя [4].

На практике актуальны нестационарные постановки задач, в которых точка минимума функционала $f(x)$ и сам функционал изменяются

со временем. В частности, задачу управления энергопотрблением гетерогенного ЦП можно сформулировать в терминах трекинга (отслеживания) минимума нестационарного функционала среднего риска на основе поступающих данных наблюдений (измерений).

Определим точку минимума $f_t(x)$ как:

$$\theta_t = \arg\min_x f_t(x).$$

Тогда для её нахождения, необходимо построить последовательность таких оценок $\{\hat{\theta}_n\}$, что $||\hat{\theta}_n - \theta_t|| \to \min$, основываясь на наблюдениях с помехами за случайными переменными $F_t(x_n, w_n)$, n = 1, 2,.. [50]. Иначе говоря, последовательность оценок $\{\hat{\theta}_n\}$ неизвестного вектора θ, минимизирует функцию

$$f(x) = E_w\{F(x,w)\} = \int_{\mathbb{R}^p} F(x,w)P_w(dw)$$

типа функционала среднего риска.

Положим, что на функцию накладываются следующие ограничения:

Ограничение 1: Функция $f(\cdot)$ — сильновыпуклая, т.е.

$$\langle x - \theta, \nabla f(x)\rangle \geq \mu\|x-\theta\|_2^2, \forall x \in \mathbb{R}^q.$$

Ограничение 2: Градиенты функции $F(\cdot, w)$ удовлетворяют условию липшицевости:

$$\|\nabla_x F(x,w) - \nabla_x F(y,w)\|_2 \leq M\|x-\theta\|_2, \forall x, \theta \in \mathbb{R}^q.$$

Хотя согласно доказательству А.Т. Вахитова и О.Н. Граничина [101] можно сформулировать и более слабые ограничения, из которых следуют вышеуказанные, для целей этого исследования их будет достаточно.

Далее рассмотрим рандомизированные алгоритмы, которыми эту задачу можно решить, а затем спроектируем целевые функции, удовлетворяющие ограничениям этих алгоритмов.

2.1.2 Алгоритм одновременно возмущаемой стохастической аппроксимации

Методы стохастической оптимизации появились в пятидесятых годах XX в. с алгоритмов Роббинса-Монро и Кифера-Вольфовица [62, 88].

Дж. Кифер (J. Kiefer) и Дж. Вольфовиц (J. Wolfowitz) предложили следующее решение задачи о нахождении стационарного локального минимума или максимума θ некоторой функции $f(\cdot)$, когда для каждого значения $x \in \mathbb{R}$ доступно наблюдение

$$Y(x) = f(x) + v,$$

где $Y(x)$ — зашумленное значение $f(x)$ [62]. При некоторых дополнительных ограничениях рекуррентная последовательность

$$\hat{\theta}_n = \hat{\theta}_{n-1} - \alpha_n \frac{Y(\theta_{n-1} + \beta_n) - Y(\theta_{n-1} - \beta_n)}{2\beta_n}$$

сходится к точке θ, где $\{\alpha_n\}$ и $\{\beta_n\}$ — некоторые убывающие числовые последовательности, которые должны обладать определёнными свойствами. Можно заметить, что указанный алгоритм фактически является разновидностью псевдоградиентного спуска — изменение оценки «в среднем» идет в направлении градиента.

Существенным ограничением процедуры Кифера-Вольфовица является имплицитное допущение об условной центрированности помех наблюдения, то есть для функции

$$g(x, \beta) = \frac{Y(x + \beta) - Y(x - \beta)}{2\beta},$$

выборочные значения которой точно наблюдаются, математическое ожидание при малом β близко к значению производной функции:

$$E\{g(x,\beta)\} \approx f'(x).$$

Этот инвариант не всегда верен в практических задачах, и из-за этого процедура Кифера-Вольфовица не всегда сходится к искомой точке. Однако из указанного выше определения также видно, что поведение последовательности оценок зависит от выбора функции $g(x,\beta)$, а следовательно лучшего поведения оценок можно добиться выбором иной функции, лучше аппроксимирующей «в среднем» производную $f'(\cdot)$.

Пусть функция $f(\cdot) : \mathbb{R}^q \to \mathbb{R}$ дважды непрерывно дифференцируема, и задана последовательность независимых случайных векторов $\{\Delta_n\}$, компоненты которых независимы и имеют распределение Бернулли, т.е. принимающих значение +1 или -1 с равной вероятностью, и на каждом шаге n между значениями этих случайных величин и ошибками наблюдения нет корреляции. Дж. Спалл (J. Spall) предложил определить новую функцию как

$$\widetilde{g}(x,\beta,\Delta) = g(x,\beta\Delta)$$

и назвал $\{\Delta_n\}$ *пробным одновременным возмущением* (trial simultaneous perturbation). Он также показал, что при указанных выше условиях

$$E\{\widetilde{g}(x,\beta,\Delta)\} = \nabla(x) + \mathcal{O}(\beta).$$

Это значит, что при выполнении условия

$$\lim_{n\to\infty} \beta_n \to 0$$

в пределе эта функция «в среднем» совпадает с градиентом $\nabla f(\cdot)$, то есть для больших n вероятностное распределение соответствующим об-

разом отмасштабированных ошибок оценивания является приблизительно нормальным. Более того, такие же свойства демонстрирует и предложенная в [6, 14] функция, использующая только одно наблюдение на каждой итерации,

$$\bar{g}(x, \beta, \Delta) = \frac{\Delta}{\beta} Y(x + \beta\Delta).$$

В [6] доказано, что функция $\bar{g}(x, \beta, \Delta)$ позволяет строить последовательность оценок, состоятельных при почти произвольных помехах в наблюдении.

Актуальным является вопрос о скорости сходимости оценок алгоритмов стохастической аппроксимации [2, 13, 25]. Было показано, что для процедуры Кифера-Вольфовица и ряда её обобщений она зависит от гладкости функции $f(\cdot)$. Для дважды дифференцируемой функции среднеквадратичная ошибка первоначальной процедуры Кифера-Вольфовица убывает как $\mathcal{O}(n^{-\frac{1}{2}})$, для трижды дифференцируемой — $\mathcal{O}(n^{-\frac{2}{3}})$. Спалл показал, что его модификация обладает такой же скоростью сходимостью, что и исходная процедура Кифера-Вольфовица. Пусть характеристика $\gamma = \mathscr{X} + 1$, если все частные производные некоторой функции до $\mathscr{X}$ порядка включительно удовлетворяют условию липшицевости. В [14] предлагается использовать

$$\widetilde{g_\gamma}(x, \beta, \Delta) = \mathcal{K}(\Delta) \frac{Y(x + \beta\Delta) - Y(x - \beta\Delta)}{2\beta}$$

и

$$\bar{g}_\gamma(x, \beta, \Delta) = \frac{1}{\beta} Y(x + \beta\Delta),$$

где $\mathcal{K}(\cdot)$ — дифференцирующее ядро, определяемое с помощью ортогональных многочленов Лежандра степени меньшей γ, тогда для двух рандомизированных алгоритмов, полученных с использованием этих функций, среднеквадратичная скорость оценки равна $\mathcal{O}(n^{-\frac{\gamma-1}{\gamma}})$, и для широ-

кого класса задач она оптимальна в некотором асимптотически минимаксном смысле.

Для многомерного случая сходимость алгоритмов стохастической аппроксимации доказана в [8, 84, 97]. Хотя асимптотически алгоритм с одним измерением ведёт себя хуже, чем версия с двумя [14], это компенсируется меньшей вычислительной трудоёмкостью алгоритма [10].

Определим *пробное одновременное возмущение* как последовательность одинаково симметрично распределённых независимых случайных векторов Δ_n с матрицами ковариации:

$$cov\{\Delta_n \Delta_j^T\} = \delta_{nj}\sigma_\Delta^2 I,$$

где $\delta_{nj} \in \{0, 1\}$ — символ Кронекера, $0 < \sigma_\Delta < \infty$. В качестве случайных векторов удобно использовать бернуллиевские случайные векторы, так как их компоненты Δ_n независимы друг от друга и принимают равновероятные значения ± 1.

Существуют три возможных способа построения последовательности оценок точек минимума функции $F(x)$ без существенной потери скорости сходимости:

$$\hat{\theta}_n = \hat{\theta}_{n-1} - \frac{\alpha_n}{\beta_n}\Delta_n y_n,$$

$$\hat{\theta}_n = \hat{\theta}_{n-1} - \frac{\alpha_n}{2\beta_n}\Delta_n (y_n^+ - y_n^-),$$

$$\hat{\theta}_n = \hat{\theta}_{n-1} - \frac{\alpha_n}{\beta_n}\Delta_n (y_n^+ - y_n),$$

опирающихся на три типа защумленных наблюдений с одним или двумя измерениями на итерации y_n [47]:

$$y_n = F(\hat{\theta}_{n-1}, w_n^+) + v_n,$$

$$y_n^- = F(\hat{\theta}_{n-1} - \beta_n \Delta_n, w_n^+) + v_n^+,$$

$$y_n^+ = F(\hat{\theta}_{n-1} + \beta_n \Delta_n, w_n^+) + v_n^-,$$

где $\{\alpha_n\}$ и $\{\beta_n\}$ — последовательности неотрицательных чисел, удовлетворяющих некоторым условиям, w_n^+ — вектор стохастических возмущений для наблюдения y_n^+, v_n^+ — вектор произвольного внешнего шума во время наблюдений. Третий вариант удобен в тех случаях, когда в силу специфики задачи нельзя сделать два измерения таким образом, чтобы они были некоррелированы с Δ_n [31]. Эти рекуррентные процедуры называются *одновременно возмущаемой стохастической аппроксимацией* (Simultaneous Perturbation Stochastic Approximation, SPSA), поскольку в них конститутивно используется случайное пробное возмущение по всем направлениям. SPSA относится к типу алгоритмов стохастического градиентного спуска.

Первый алгоритм использует только одно зашумленное наблюдение, а второй и третий — два зашумленных наблюдения. В целях различия между этими вариациями алгоритма SPSA будем называть версию с одним зашумленным наблюдением как SPSA1, а две последние вариации — SPSA2.

Среди условий состоятельности оценок отдельно выделеим условие слабой корреляции между пробным возмущением $\{\Delta_n\}$ и последовательностями неопределенностей $\{w_n\}$ и $\{v_n\}$ как наиболее важное.

Обе вариации алгоритма SPSA работают следующим образом:

1. Определяется целевая эмпирическая (измеряемая) функция $F(x, w)$.
2. Делается первоначальная оценка точки оптимума $\hat{\theta}_0$.
3. Текущая оценка оптимума водвергается случайному возмущению.
4. Получается требуемое количество зашумленных наблюдений функции F, и обновляется текущая оценка оптимума $\hat{\theta}_n$.
5. Производится переход на шаг 3.

Следует отметить, что SPSA применимы, как для поиска стационарного оптимума, так и в задачах, где оптимум может смещаться от итерации к итерации, т.е. при решении задачи трекинга [48, 49, 98].

2.2 Теоретические результаты о свойствах оценок рандомизированных алгоритмов стохастической оптимизации в задаче отслеживания изменений оптимальных параметров

Во многих практических задачах функционал среднего риска является нестационарным, то есть его точка оптимума может смещаться с течением времени.

В дискретном времени нестационарный функционал среднего риска можно определить следующим образом [49]. Пусть $\{F_\xi(\theta, w)\}_{\xi\in\Xi}$ — множество дифференцируемых функций $F_\xi(\theta, w) : \mathbb{R}^q \times \mathbb{R}^p \to \mathbb{R}$. Множество Ξ можно определить как многомерное, и все дальнейшие выводы будут справедливы и для него с незначительными изменениями, несущественными для этого исследования, но далее будет достаточно рассмотрения простого случая $\Xi = \mathbb{N}$. Пусть последовательность $\{x_t\}$ — план наблюдения или последовательность точек, в которых наблюдатель производит измерения, в моменты времени $t = 1, 2, ...$, а $\{y_t\}$ — последовательность результатов измерения

$$y_t = F_{\xi_t}(x_t, w_t) + v_t,$$

где $\{\xi_t\}$ и $\{w_t\}$ — последовательности, на которые наблюдатель не может оказать влияния. В каждый момент дискретного времени t наблюдается значение некоторой функции $F_t(\cdot)$.

Тогда задача отслеживания изменений (трекинга) оптимальных параметров нестационарного функционала среднего риска состоит в том, чтобы по последовательности зашумленных измерерий $\{y_t\}$, построенных по точкам $\{x_t\}$ построить оценку $\hat{\theta}_t$ неизвестного вектора θ_t, минимизирующего изменяющийся во времени функционал

$$f_t(\theta) = E_{\mathscr{F}_{t-1}} F_t(\theta, w) \to \min_{\theta},$$

где $\mathscr{F}_{t-1}$ — σ-алгебра всех вероятностных событий, произошедших вплоть до момента времени $t = 1, 2, ...$, $E_{\mathscr{F}_{t-1}}$ — условное математическое ожидание в σ-алгебре $\mathscr{F}_{t-1}$.

Определение алгоритмов SPSA не делает различений при его использовании в стационарном и нестационарном случаях, но для нестационарного варианта были получены важные теоретические результаты для варианта с двумя измерениями. Введём следующие допущения для функционала среднего риска:

Допущение 1: Разница между последовательными значениями шума $\bar{v}_n = v_{2n} - v_{2n-1}, n = 1, 2, ...$, ограничена: $|\bar{v}_n| \leq c_v < \infty$.

Допущение 2: Смещение точки оптимума ограничено: $\|\theta_t - \theta_{t-1}\| \leq \delta_\theta < \infty$.

Допущение 3: Скорость смещения точки оптимума ограничена таким образом, что для любой точки $x : \|E_{\mathscr{F}_{2n-2}} \nabla\varphi_n(x)\| \leq a_1\|x - \theta_{2n-2}\| + a_0, E_{\mathscr{F}_{2n-2}}\varphi_n(x)^2 \leq a_2\|x-\theta_{2n-2}\|^2+a_3, \varphi_n(x) = F_{\xi_{2n}}(x, w_{\xi_{2n}})-F_{\xi_{2n-1}}(x, w_{\xi_{2n-1}})$.

Допущение 4: Функции $f_t(\cdot)$ имеют уникальные точки минимума θ_t, и для них справедливо

$$\forall x \in \mathbb{R}^d : \langle x - \theta_t, E_{\mathscr{F}_{t-1}} \nabla f_t(x)\rangle \geq \mu\|x - \theta_t\|^2,$$

где $\mu > 0$ — константа, $\langle\cdot,\cdot\rangle$ — скалярное произведение векторов.

Допущение 5: Градиент ∇f_t ограничен в среднеквадратичном смысле в точках минимума θ_t:

$$E\|\nabla F_t(\theta_t, w_t)\|^2 \leq 0, E\langle \nabla F_t, \nabla F_{t-1}(\theta_{t-1}, w_{t-1})\rangle \leq 0.$$

Допущение 6: $\forall \xi \in \Xi$ градиент $\nabla f_\xi(x)$ удовлетворяет условию липшицевости:

$$\forall x' x'' \in \mathbb{R}^d : \|\nabla f_\xi(x', w_\xi) - \nabla f_\xi(x'', w_\xi)\| \leq M\|x' - x''\|,$$

где M — константа, $M \geq \mu$.

Два последующих допущения касаются вектора Δ_n и его дифференцирующего ядра $\mathcal{K}_n(\Delta_n)$ в алгоритме SPSA:

Допущение 7: Для любых $n = 1, 2, ...,$

1. Δ_n не зависит от σ-алгебры $\mathscr{F}_{2n-2}$.

2. Шум $\hat{v}_n$ и вектор Δ_n не зависят друг от друга.

Допущение 8: Для любых $n = 1, 2, ...,$ вектора Δ_n и $\mathcal{K}_n(\Delta_n)$ ограничены: $\|\Delta_n\| \leq c_\Delta < \infty, \|\mathcal{K}_n(\Delta_n)\| \leq \kappa < \infty$, и векторные функции $\mathcal{K}_n(\cdot)$ наряду с симметричными функциями распределения одновременных возмущений $P_n(\cdot)$ удовлетворяют условиям:

$$\int \mathcal{K}_n(x) P_n(dx) = 0, \int \langle \mathcal{K}_n(x), x\rangle P_n(dx) = I,$$

где I — единичная матрица.

Частным случаем, удовлетворяющим допущениям 7 и 8, является бернуллиевская последовательность $\{\Delta_n\}$, описанная ранее, и $\mathcal{K}_n(x) \equiv x$.

Будем считать, что последовательность оценок $\hat{\theta}_{2n}$ имеет асимптотически эффективную верхнюю границу $\bar{L} > 0$ остатков оценки, если $\forall \epsilon > 0 \exists N$ такое, что $\forall n > N$

$$\sqrt{E\|\hat{\theta}_{2n} - \theta_{2n}\|^2} \leq \bar{L} + \epsilon.$$

Без потери общности будем интерпретировать определение алгоритма SPSA с двумя наблюдениями таким образом, что на каждой итерации наблюдения совершаются в точках

$$x_{2n} = \hat{\theta}_{2n-2} + \beta_n^+ \Delta_n, x_{2n-1} = \hat{\theta}_{2n-2} + \beta_n^- \Delta_n,$$

где $\{\beta_n^+\}$ и $\{\beta_n^-\}$ — последовательности неотрицательных целых чисел, и $\beta_n = \beta_n^+ + \beta_n^- > 0$.

О.Н. Граничин и Н.О. Амелина доказали в [49], что если выполняются допущения 1–8, $\beta_{max} + \bar{\beta} < \infty$, и константа α достаточно мала:

$$k\alpha < 1, \alpha < \frac{\mu}{3\kappa^2(a_2\bar{\beta}^2 + c_\Delta^2 M^2)},$$

то последовательность оценок, выдаваемая алгоритмом SPSA с двумя измерениями имеет асимптотически эффективную верхнюю границу

$$\bar{L} = h + \sqrt{h^2 + l/k},$$

где $\beta_{max} = \max_n \beta_n, \bar{\beta} = \max_n \frac{1}{\beta_n}, \bar{\beta}^+ = \max_n \frac{\beta_n^+}{\beta_n}, \bar{\beta}^- = \max_n \frac{\beta_n^-}{\beta_n}, k = 2\mu - 6\alpha\kappa^2(a_2\bar{\beta}^2 + c_\Delta^2 M^2), h = \frac{2\delta_\theta}{\alpha k} - \delta_\theta + \frac{\bar{\beta}^- a_1 + 6\alpha c_\Delta^2 M^2 (c_\Delta + \bar{\beta}^- \sigma_\theta)}{k}, c_1 = M((\bar{\beta}^+)^2 + 3(\bar{\beta}^-)^2), c_2 = M^2(c_\Delta^2 + (\bar{\beta}^+)^2\sigma_\theta^2 + 2\beta^+\beta_{max}\sigma_\theta(c_\Delta + \sigma_\theta)), \bar{l} = 2(\beta^- a_0 + 3\kappa^2\alpha a_3\bar{\beta}^2 + \kappa c_\Delta^2(\beta_{max}c_1 + \kappa\alpha(3c_2 + c_v^2)))$.

Допущения 2–6 накладывают ограничения на нестационарный функционал среднего риска, однако допущение 1 — это допущение на модель шума. Поскольку в задаче оптимизации энергопотребления процессора нагрузка на ядро или кластер под влиянием вычислительных процессов, интерпретируемых как шум, изменяется в ограниченных пределах (от 0 до 100%), если проектируемый функционал среднего риска удовлетворит допущениям 2–6, то благодаря указанной выше теореме оценки, сделанные алгоритмом SPSA с двумя зашумленными измерениями, будут сходиться к текущей точке оптимума.

2.3 Целевая функция для алгоритма одновременно возмущаемой стохастической аппроксимации с одним наблюдением в задаче оптимизации энергопотребления процессора

Функционал среднего риска для алгоритма SPSA с одним измерением был разработан первым, и при его разработке был сделан ряд допущений. Первое из них состоит в том, что ресурсы ЦП чаще недоиспользуются, чем используются на 100%. Так как при прочих равных условиях частота ЦП определяет объем работы, которую он может выполнить за выбранный интервал времени, уменьшая рабочую частоту, мы увеличиваем количество времени, которое требуется для выполнения той же вычислительной задачи.

Пусть нагрузка L на ядро ЦП или кластера ЦП определена как процент активного времени от общего времени работы ЦП (активного и неактивного) за некоторый период времени. Аналогично подходу в регуляторе OnDemand нагрузка на кластер ЦП определяется как мажоранта нагрузок входящих в кластер ядер. Общая идея, которую необходимо реализовать с помощью алгоритма SPSA, состоит в том, чтобы установить частоту на следующую итерацию цикла DVFS исходя из текущего состояния процессора с точки зрения уровня нагрузки L таким образом, чтобы наблюдаемый во время следующей итерации цикла DVFS уровень нагрузки был максимально близок к заранее выбранному значению L_T. Это значение не должно быть близко к 100%, чтобы можно было выделить дополнительные вычислительные ресурсы, если фактическая нагрузка оказалась выше предсказанной, и не должно быть близко к 0%, так как тогда уже небольшая нагрузка будет обсчитываться на энергетически неэффективной высокой частоте.

Второе допущение обусловлено спецификой работы ЦП. Так как и гомогенные, и гетерогенные ЦП работают только на фиксированном для каждого процессора или кластера ядер наборе частот $Freq$, непрерывная оценка оптимальной частоты округляется до ближайшего доступного значения.

Целевая функция определяется как

$$F(f) = 2^{((\text{workload}(f)-L_T)/2)} + \gamma 1.5^{\text{table}(f)},$$

где $\text{workload}(f)$ — нагрузка ЦП или кластера ЦП, полученная по метрикам ОС, L_T — целевой уровень нагрузки, $\text{table}(f)$ — номер частоты в массиве частот, доступных ЦП или кластеру ЦП, отсортированном по возрастанию. Первое слагаемое функции — это штраф за слишком низкую нагрузку по сравнению с L_T. Поскольку одна и та же вычислительная нагрузка будет занимать меньше времени с увеличением частоты, функция $\text{workload}(f)$ невозрастающая. Второе слагаемое – это штраф за использование слишком высокой частоты из доступного набора, и оно является монотонно возрастающим [27].

Чтобы установить новую рабочую частоту, вычисляется

$$f_n = \mathscr{P}(\hat{f}_{n-1} + \beta\Delta_n),$$

где $\mathscr{P}$ — проектор во множество $Freq$, $\hat{f}_{n-1}$ — текущая оценка частоты, β — параметр шага SPSA. Для новой оценки частоты вычисляется

$$\hat{f}_n = \mathscr{L}(\hat{f}_{n-1} - \frac{\alpha}{\beta}\Delta_n y_n),$$

где $\mathscr{L}$ — проектор в отрезок $[min(Freq), max(Freq)]$, α, β — параметры шага алгоритма SPSA [26]. В этом алгоритме от итерации к итерации используются постоянные значения обоих параметров.

В рамках предлагаемого подхода изменение вычислительной нагрузки, вызванное началом и окончанием приложений, интерпретируется как

изменение точки оптимума в нестационарном функционале среднего риска. С другой стороны время цикла DVFS в 10 мс превышает продолжительность многих вычислительных процессов, запускаемых в современных операционных системах и на мобильных устройствах. Системные прерывания и обработчики пользовательских событий в прикладных программах длятся, как правило, не более нескольких миллисекунд, однако изнутри операционной системы предсказать их возникновение в тот или иной момент времени часто невозможно, например, нельзя предсказать, когда пользователь дотронется до экрана. Изменение вычислительной нагрузки, вызванное подобными кратковременными факторами, в рамках данного подхода интерпретируется как аддитивная помеха измерений v_n.

В итоге цикл DVFS для регулятора SPSA1 выглядит следующим образом:

1. Установить первончальное значение оценки $\hat{f}_0$, выбрать α и β.
2. Сгенерировать Δ_n.
3. Добавить возмущение к текущей оценке $f_n = \mathscr{P}(\hat{f}_{n-1} + \beta\Delta_n)$.
4. Установить текущую частоту в f_n.
5. Получить новое зашумленное наблюдение $y_n = F(\hat{f}_{n-1} + \beta\Delta_n) + v_n$
6. Обновить оценку частоты $\hat{f}_n = \mathscr{L}(\hat{f}_{n-1} - \frac{\alpha}{\beta}\Delta_n y_n)$.
7. Перейти к шагу 2.

Чтобы показать, что оценки алгоритма состоятельны, покажем, что целевая функция удовлетворяет требованиям, изложенным в разделе 2.1.2. Во-первых, в силу особенностей предметной области изменения оптимальной частоты ограничены:

$$\|f_n - f_{n-1}\| \leq \delta < \infty.$$

Далее, $F(f)$ удовлетворяет обоим допущениям, которые необходимы для доказательства состоятельности оценок алгоритма:

Допущение 1: Функция $F(f)$ строго выпуклая и имеет точку минимума f^*:

$$\langle f - f^*, \nabla F(f) \rangle \geq \gamma \mathrm{ln}1.5 - 1, \forall \mathrm{f} \in \mathbb{R}.$$

Допущение 2: Градиент $\nabla F(f)$ удовлетворяет условию липшицевости:

$$\|\nabla F(f_1) - \nabla F(f_2)\| \leq \gamma 1.5^{\mathrm{max(f_1;f_2)}} \mathrm{ln}^2 1.5 \|\mathrm{f_1} - \mathrm{f_2}\|,$$
$$\forall f_1, f_2 \in \mathbb{R}.$$

2.4 Целевая функция для алгоритма одновременно возмущаемой стохастической аппроксимации с двумя наблюдениями в задаче оптимизации энергопотребления процессора

2.4.1 Стоимость исполнения программы и функционал среднего риска

С теоретической точки зрения при решении одной и той же задачи различные определения функционалов среднего риска могут значительно различаться, но приводить к тому же решению [50]. При определении функционала среднего риска для SPSA2 были использованы наработки, полученные при изучении SPSA1, при этом они получили дальнейшее развитие.

Как и в случае с SPSA1 определим нагрузку L на ядро ЦП или кла-

стера ЦП как процент активного времени от общего времени работы ЦП (активного и неактивного) за некоторый период времени. В ситуации, когда частота устанавливается одновременно для всех ядер кластера, будем определять не нагрузку отдельного ядра, а нагрузку кластера как мажоранту нагрузок всех ядер, входящих в него. Определим *вычислительный объём* или просто *объём* как произведение частоты на нагрузку. По существу объём — это количество тактов ЦП, выделенных на активный вычислительный процесс от его постановки на исполнение до снятия с исполнения. Хотя объём коррелирует с количеством выполненных инструкций при выполнении процесса, соответствие не является взаимно однозначным: конвейерная обработка позволяет процессору исполнять разные фазы последовательно идущих инструкций одновременно, и длительность в тактах различных инструкций может быть различной. Для процесса DVFS важно, что один и тот же объём создает разную нагрузку на разных частотах: чем выше частота, тем ниже нагрузка, и наоборот.

Определение частоты в цикле DVFS для следующей итерации цикла нетривиально в том смысле, что мы можем основывать нашу стратегию помимо прочего на истории наблюдений за нагрузкой процессора и на характеристиках процессов ОС, как это делается в EAS, однако нагрузка в будущем не может быть достоверно определена — например, невозможно предсказать, что в следующую секунду на устройство поступит телефонный звонок. Ввиду этой неопределённости введём *целевую* или *пороговую нагрузку* L_T — конкретное постоянное значение нагрузки в течение некоторого времени наблюдения. В литературе встречаются эмпирически определённые значения L_T от 60% до 80%, и они использовались и в наших экспериментах — такие целевые уровни гарантируют наличие дополнительных вычислительных ресурсов в сценарии, где фактическая нагрузка оказалась намного выше, чем ожидалось по истории наблюдений. L_T используется в качестве границы между двумя разными стратегиями DVFS:

- Если текущая нагрузка превышает L_T, то ядро выполняет доста-

точно длительную вычислительную задачу. В этом случае соображения производительности перевешивают экономию энергии. Следовательно, выделение ресурсов для решения этой задачи является приоритетным, и оптимальная частота, которую должен установить регулятор DVFS, — это та, которая обеспечивает наиболее близкую к L_T нагрузку при текущем наблюдаемом объеме. В долгосрочной перспективе, если нагрузка не падает ниже L_T, будет устанавливаться всё более и более высокая рабочая частота вплоть до максимальной, и эта стратегия будет использоваться до тех пор, пока нагрузка не упадет.

- Если нагрузка не превышает L_T, это означает, что при сохранении такой тенденции существует возможность оптимизировать энергопотребление путём нахождения такой частоты, которая обеспечивает оптимальную энергоэффективность, а прогнозируемая нагрузка по-прежнему не превышает L_T.

Определим *эффективность выполнения* или *стоимость выполнения* (cost of execution, CoE) рабочей частоты f следующим образом:

$$CoE = \frac{E(t)}{n_{ticks}} = \frac{\int_0^t U(t)I(t)\,dt}{ft} = \frac{I_f U}{f},$$

где CoE — стоимость выполнения, t — длительность времени наблюдения, $E(t)$ — энергия, затраченная за период времени t, n_{ticks} — количество тактов ЦП, наблюдаемых в течение этого времени. По сути CoE — это затраты энергии на один такт ядра процессора на определенной рабочей частоте.

CoE можно рассматривать и как приоритет регулятора DVFS при выборе частоты — чем ниже значение, тем выше приоритет. С этой точки зрения вместо CoE можно использовать некоторую метрику *обобщенной стоимости выполнения* (generalized cost of execution, $GCoE$) для определения приоритета при выборе частот. В этом случае $GCoE$

определяется как отображение конечного множества частот кластера ЦП в некоторое множество, для которого определено линейное отношение порядка. Строгость этого отношения необязательна: если несколько рабочих частот обладают одинаковым приоритетом, то выбор одной из них остаётся на усмотрение регулятора DVFS [83].

В этом исследовании $GCoE$ определеляется как отношение усреднённой силы тока при работе ядра на определённой частоте к этой частоте:

$$GCoE = \frac{I_f}{f},$$

Рабочее напряжение процессора допустимо рассматривать как постоянное, поэтому при удалении постоянного множителя U из определения CoE при переходе к $GCoE$ отношение порядка не изменится[3].

Ключевым моментом, повлиявшим на изучение и использование CoE и $GCoE$ в этой работе является тот факт, что для имеющихся профилей энергопотребления кластеров ЦП с ростом f растет и I_f, но $GCoE(f)$ в общем случае является немонотонно возрастающей функцией. Пример такого поведения показан в Табл. 1.1, где ядро A55 потребляет наименьшее количество энергии на частоте 500 МГц, но наиболее энергоэффективно на частоте 875 МГц.

Таким образом, когда текущая нагрузка становится ниже L_T, оптимальной будет частота с наименьшим $GCoE$ среди частот, способных обработать такой же объём прогнозируемой нагрузки, не превышая L_T.

Объединяя оба режима работы регулятора DVFS (режим производительности и режим экономии) воедино, определим результат вычисления функционала среднего риска и оптимальность текущей рабочей частоты как расстояние между индексами текущей и оптимальной частот в отсортированном списке частот:

[3]Более того, для использованных устройств U=1V, поэтому CoE и $GCoE$ были численно равны.

$$F(f) = \begin{cases} \|\mathscr{P}(f) - \mathscr{P}(f\frac{L}{L_T})\|, L_T < L \leq 1 \\ \|\mathscr{P}(f) - \mathscr{S}(\min_{f_1..f_n} GCoE(f_i) : L\frac{f}{f_i} <= L_T)\|, 0 \leq L \leq L_T, \end{cases}$$

где $\mathscr{P}(\cdot)$ — проектор в индекс ближайшей частоты в массиве частот, отсортированном по возрастанию, $\mathscr{S}(\cdot)$ — проектор $GCoE$ частоты в её индекс в массиве частот, отсортированном по возрастанию.

Уровень нагрузки L может изменяться от измерения к измерению, однако для такого определения целевой функции необходимо рассмотреть особый случай, когда одно зашумленное измерение соответствует режиму экономии, а другое — режиму производительности. Такая ситуация возникает, когда кончается период интенсивной работы, или, наоборот, начинается новый вычислительно ёмкий процесс. Для преодоления этого препятствия будем считать, что в такой ситуации нельзя сделать внутренне непротиворечивый вывод о грядущей нагрузке, поэтому эту пару измерений будем отбрасывать без дальнейшего изменения частоты в этой итерации цикла DVFS, а со следующей итерации начнём новую итерацию алгоритма SPSA согласно выбранной схеме.

Отметим, отсутствие у устройства частот с одинаковым $GCoE$ означает, что для любого наблюдаемого L оптимальная частота всегда будет уникальной. С практической точки зрения, если такая ситуация встречается, то коллизию можно разрешить назначением меньшей частоте меньшего $GCoE$.

Хотя «внутри себя» целевая функция знает, на какую частоту оптимальнее всего переключиться в данный момент, эта информация целенаправленно не выносится «вовне», чтобы алгоритм сделал шаг в сторону оптимальной частоты, но не сразу же её установил. Такая стратегия приводит к лучшей адаптивности алгоритма как в ситуации, когда прогноз алгоритма совпал с реальностью, так и в ситуации, когда прогноз не оправдался.

2.4.2 Получение зашумленных наблюдений

Исходя из определения SPSA2, для его реализации требуется два зашумленных наблюдения, после чего устанавливается текущая оптимальная оценка, то есть для целей этого исследования — три итерации цикла DVFS:

1. Пусть i_i — индекс текущей рабочей частоты в массиве частот, отсортированной по возрастанию. Сгенерируем случайное число $\Delta = \pm 1$, после чего установим в качестве текущей частоту с индексом $i_i + \Delta\beta$ для получения первого зашумленного наблюдения y^+ функционала среднего риска.

2. Далее в качестве текущей устанавливается частота с индексом $i_i - \Delta\beta$ для получения второго зашумленного наблюдения y^-.

3. Новая оценка оптимальной частоты $i_{i+1} = i_i - \frac{\alpha(y^+ - y^-)}{2\Delta\beta}$ устанавливается в качестве текущей.

Заметим, что из-за отсутствия в ядре ОС Android операций с плавающей запятой проще работать с индексами частот в отсортированном массиве, чем с самими частотами. Во всех случаях, когда индекс выходит за границы массива, в качестве результата операции вычитания или сложения берётся нижняя или верхняя граница соответственно.

Вместе с тем в самом определении SPSA2 заключается возможность для оптимизации — установка новой оценки оптимальной частоты может быть интерпретирована в качестве первого зашумленного наблюдения на следующем цикле, и следовательно цикл работы алгоритма потребует только двух итераций цикла DVFS:

1. Текущее значение функции при выбранной частоте с индексом i_i используется в качестве первого зашумленного наблюдения y^0. Генерируется случайное число $\Delta = \pm 1$, и для получения второго

зашумленного наблюдения функции y^+ устанавливается частота с индексом $i_i + \Delta\beta$.

2. Устанавливается новая частота с индексом $i_{i+1} = i_i - \frac{\alpha(y^+ - y^0)}{\Delta\beta}$.

Однако ожидание двух или трёх циклов DVFS для отработки одной итерации алгоритма может оказаться пагубным для точности оценки в динамично изменяющихся условиях и как следствие — для качества работы регулятора. Поэтому предлагается вариант алгоритма, который занимает только одну итерацию цикла DVFS, хотя и отходит от строгого определения SPSA2:

1. Пусть i_i — индекс текущей рабочей частоты в массиве частот, отсортированной по возрастанию. Сгенерируем случайное число $\Delta = \pm 1$. Вычисляется оценка нагрузка для частот с индексами $i_i \pm \Delta\beta$ в предположении, что объём останется неизменным, и для смоделированных оценок вычисляются значения функции y^+ и y^-. Далее устанавливается рабочая частота с индексом $i_{i+1} = i_i - \frac{\alpha(y^+ - y^-)}{2\Delta\beta}$.

Далее каждую схему мы будем называть, исходя из числа циклов DVFS, необходимых для одного цикла алгоритма, SPSA2_3, SPSA2_2 и SPSA2_1 соответственно.

Глава 3. Регуляторы частоты процессора на основе алгоритмов одновременно возмущаемой стохастической аппроксимации с одним и двумя наблюдениями

В этой главе рассказывается о технических особенностях разработки регуляторов DVFS, основанных на алгоритме SPSA и функционалах среднего риска из Главы 2, описывается методология и результаты сравнения разработанных регуляторов с существующими и делаются выводы по результатам их тестирования.

3.1 Особенности программирования регуляторов на основе алгоритмов одновременно возмущаемой стохастической аппроксимации с одним и двумя наблюдениями

Регуляторы SPSA1 и все три схемы для регулятора SPSA2 были реализованы на базе инфраструктуры стандартного регулятора OnDemand на языке программирования C. При разработке приходилось учитывать, что в ядре ОС Android, как и в ядре ОС Linux, на котором оно основано, не поддерживаются операции с плавающей запятой, поэтому при необходимости функции от аргумента с плавающей запятой переводились на использование целочисленных аргументов. В силу этого в ряде случаев

применялось табличное задание функций, в том числе и непрерывных, если было известно, что в рамках ограничений алгоритма они могут быть вычислены от конечного числа различных аргументов. Дополнительные проверки на выход за пределы массива частот делались во всех случаях перед установкой текущей оценки оптимальной частоты.

Стоит особо отметить, что в рамках диссертационного исследования во всех разработанных регуляторах константы энергопотребления или метрики $GCoE$ для всех частот всех кластеров конкретнных ЦП тестовых стендов были прописаны как константы в программном коде. Более правильным с инженерной точки зрения подходом являлась бы модификация ядра ОС, которая бы предоставляла эти константы по аналогии с информацией о поддерживаемых частотах, однако в силу заметных различий кода ядер ОС Android различных производителей, целесообразно, чтобы подобный программный интерфейс был изначально предоставлен разработчиком ядра.

Для схемы SPSA2_1 была добавлена эмпирическая оптимизация для режима производительности. Если нагрузка за последний период наблюдений составляет 100%, значение β изменяется таким образом, чтобы рабочая частота увеличивалась более существенно. В целом, это не противоречит идее SPSA2, так как в общем случае на каждой итерации могут использоваться значения последовательности β_n, а не конкретная константа β. Если же высокий уровень нагрузки не продлился долго, и решение изменить β оказывается избыточным с точки зрения энергопотребления, то рабочая частота также быстро убывает, и β возвращается к исходному значению.

Время, необходимое процессору для изменения частоты, может превышать длительность итерации цикла DVFS. Например, длительность цикла DVFS регулятора OnDemand по умолчанию составляет 10 мс. Но так как OnDemand разрабатывался для настольных компьютеров и серверов с более продвинутой схемотехникой ЦП, такого интервала было достаточно для переключения рабочей частоты. При этом во время пред-

варительных экспериментов с Xiaomi Redmi Note 8 Pro было обнаружено, что после первоначального запроса на изменение частоты регулятором OnDemand или иным регулятором DVFS на его основе может потребоваться до трёх последовательных вызовов DVFS, чтобы в программных интерфейсах ОС отобразилась обновлённая частота ЦП. При этом регулятор DVFS может отправлять новый запрос на изменение частоты при каждом вызове, который будет проигнорирован до исполнения первоначального. Наблюдаемый феномен свидетельствует о том, что даже наиболее распространенные регуляторы DVFS могут быть основаны на предположениях и инвариантах, которые могут оказаться неверными для управляемого ими оборудования.

Так как разработанные регуляторы SPSA2 также используют модель тактирования OnDemand, и для алгоритма критично получать зашумленные наблюдения на целевых частотах, в программе пропускаются те вызовы DVFS, для которых частота кластера ЦП все ещё не совпадает с запрошенной. Алгоритм продолжается тогда и только тогда, когда установлена запрошенная частота. Дополнительно предполагается, что в пределах последнего цикла DVFS, в конце которого запрошенная частота была отражена в ОС, запрошенная частота была активна на всём времени его длительности, и благодаря этому не делается изменений в процессе оценки зашумленной нагрузки кластера ЦП. Для регулятора SPSA1 и схемы SPSA2_1 этот фактор не столь актуален в силу совпадения длительности алгоритма и цикла DVFS и потому не учитывался. Однако учёт этого явления может быть использован для незначительного снижения времени работы алгоритма.

Диспетчер задач EAS, который учитывает энергоэффективность в своей работе, активно взаимодействует с регулятором DVFS schedutil. Согласно документации, EAS не гарантирует снижения энергопотребления, если schedutil не выбран в качестве активного регулятора [61]. С алгоритмической точки зрения schedutil реализует ту же стратегию, что и регулятор OnDemand, то есть выбирает частоту кластера пропорционально мажоранте нагрузки на ядрах кластера. Минимальная часто-

та соответствует нагрузке в 0%, а максимальная частота соответствует нагрузке в 100% нагрузке. Однако определение нагрузки ЦП у EAS существенно отличается от того, что используется в OnDemand. Нагрузка кластера ЦП оценивается с точки зрения модели энергопотребления EAS и вычислительной ёмкости всех задач, назначенных конкретному ядру ЦП. Вместо апостериорной оценки она оценивается априори.

Для предложенных схем использовалось определение нагрузки как в OnDemand даже при включенном EAS, и, как показали эксперименты, подбора параметров α, β и L_T достаточно для эффективной работы SPSA2 с EAS.

На программные модули, реализующие три схемы SPSA2, было получено свидетельство о государственной регистрации программы для ЭВМ № 2023666564 «Энергосберегающие регуляторы DVFS для Android OS на основе алгоритма SPSA с двумя измерениями и регулятора OnDemand» от 02.08.2023 [16] (см. Приложение В).

3.2 Результаты тестирования разработанных регуляторов на основе алгоритмов одновременно возмущаемой стохастической аппроксимации

3.2.1 Результаты тестирования регулятора на основе алгоритма с одним наблюдением

Для тестирования SPSA1 был выбран смартфон Xiaomi Redmi Note 8 Pro. В него встроен гетерогенный ЦП Helio G90T с двумя кластерами ядер — 2 ядра Cortex-A76 (big) и 6 ядер Cortex-A55 (LITTLE). ОС по умолчанию — Android 10. Эта модель была выбрана благодаря существующему набору утилит для запуска на устройстве альтернатив-

ных сборок ядра ОС Android. В качестве основы для экспериментальной сборки было взято ядро begonia[4], в список регуляторов DVFS которого был добавлен регулятор SPSA1. Для установки экспериментальной сборки потребовались права root-доступа. Переключение на регулятор SPSA1 производилось стандартными системными вызовами подсистемы `cpufreq` [108].

В качестве регуляторов для сравнения были выбраны OnDemand и Interactive, в качестве планировщика задач — CFS, в качестве регулятора состояний простоя — menu.

Так как цель работы состоит в создании регулятора DVFS общего назначения, приспособленного к обработке различных нагрузок, набор тестовых сценариев отражает различные паттерны человеко-машинного взаимодействия:

1. Сессия игры в *Hill Climb Racing*[5].
2. Проигрывание песни в *Spotify*.
3. Отображение и периодическая прокрутка файла в формате PDF, открытого в *Foxit PDF Editor*[6].
4. Просмотр видео на сервисе *YouTube* в качестве 480p в браузере по умолчанию.
5. Просмотр видео на сервисе *YouTube* в качестве 1080p в браузере по умолчанию.

Для запуска тестовых сценариев и управления ими устройство было подключено через USB к управляющему компьютеру.

Методология тестирования была составлена согласно рекомендациям, собранным в [78]:

[4] `https://github.com/AgentFabulous/begonia`
[5] `https://play.google.com/store/apps/details?id=com.fingersoft.hillclimb`
[6] `https://play.google.com/store/apps/details?id=com.foxit.mobile.pdf.lite`

- Все не требующиеся для обеспечения работы или проведения тестирования приложения были удалены. Для приложений, которые нельзя удалить, были выключены фоновые активности или максимально снижена их частота.

- Были выключены все не использующиеся в текущем тесте компоненты смартфона, например, Wi-Fi, 3G, GPS.

- Перед началом каждого последующего теста бралась пауза в две минуты, чтобы на устройстве завершились фоновые процессы, возможно, вызванные запуском теста, а само устройство остыло.

- Каждый сценарий запускался пять раз по пятнадцать минут для уменьшения влияния фоновых процессов.

- Перед каждой серией запусков каждого тестового сценария проводился однократный запуск сценария, и энергопотребление при его прохождении не учитывалось. Таким образом, осуществлялся прогрев тестового сценария с целью исключить из дальнейших запусков однократные процессы, которые могут повлиять на энергопотребление (например, загрузка программных библиотек).

- Поскольку устройство было подключено к USB, оно работало в режиме внешнего источника питания, поэтому перезарадка аккумулятора к одному и тому же исходному уровню не производилась.

В качестве модели оценки энергопотребления была использована базовая модель Google. Для сравнения производительности использовалась утилита `AnTuTu`[7] — распространённая в среде разработчиков под ОС Android утилита оценки производительности устройств, дающая интегральные оценки производительности устройства в условиях различных вычислительных нагрузок для ЦП, графической платы (Graphical Processing Unit, GPU), памяти и отзывчивости человеко-машинных интерфейсов (User Experience, UX).

Таблица 3.1: Сравнение энергопотребления регуляторов SPSA1, OnDemand и Interactive

Сценарий	SPSA1 (в мА·ч)	OnDemand (в мА·ч)	Interactive (в мА·ч)
Hill Climb Racing	**34,196**	91,862	37,565
Spotify	33,913	**24,678**	41,343
Foxit PDF Editor	**54,729**	65,322	77,950
YouTube 480p	32,957	**24,579**	47,980
YouTube 1080p	32,378	**25,319**	47,254

Таблица 3.2: Оценки производительности утилиты AnTuTu регуляторов Powersame, SPSA1, OnDemand, Interactive и Performance

Регулятор	ЦП	GPU	Память	UX	Общая
SPSA	80555	74357	40897	48442	244251
OnDemand	**83704**	76335	**42020**	**49893**	**251972**
Interactive	79459	**76880**	39403	49412	245154
Performance	86756	79952	42364	51789	260861
Powersave	28883	50193	30792	18458	128326

В Табл. 3.1 содержатся усреднённые данные по энергопотреблению по результатам экспериментов, переведённые в мА·ч, **жирным** выделены лучшие результаты.

В сценарии *Hill Climb Racing* OnDemand показал себя хуже всего — он всегда выводит на максимальную частоту ядра A55 и A76. SPSA1 работает немного лучше, чем Interactive, потому что держит A55 на более низких частотах, хотя диапазон использованных частот для A76 шире.

В сценарии *Spotify* Interactive считает, что потребность в производительности у ядер A76 больше необходимой и завышает устанавливаемую частоту, SPSA1 использует все доступные частоты для A55, тогда как OnDemand держит рабочую частоту ближе к нижней границе.

Хотя на первый взгляд сценарий *Foxit PDF Editor* не самый энергозатратный для процессора, рендеринг новой страницы PDF-файла требует больших вычислительных ресурсов. Здесь SPSA1 выигрывает, поскольку

[7] `https://www.antutu.com/en/index.htm`

ядра A76 в работе активно не используются.

Несмотря на то, что в тесте *YouTube 1080p* передаётся значительно больший объем данных, чем в *Youtube 480p*, разница в энергопотреблении между этими сценариями незначительна. Interactive выставляет ядрам A76 максимальную частоту и поэтому проигрывает другим регуляторам по энергопотреблению. OnDemand более консервативен в оценках как для ядер A55, так и для ядер A76, чем SPSA1, поэтому потребляет меньше энергии.

На Рис. 3.1 и 3.2 различие в поведении регуляторов в тесте Spotify изображено более детализированно в виде гистограмм распределения времени, проведённых кластерами на определённой частоте[8]. Из них видно, что принципиально разные стратегии по выставлению рабочей частоты рассматрвиаемых регуляторов приводят к принципиально разным профилям распределения времени и, как следствие, энергопотребления.

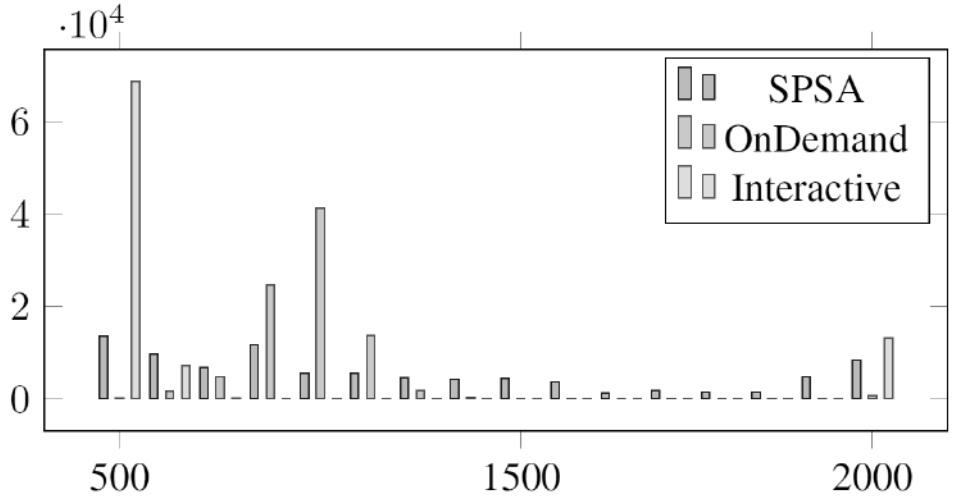


Рис. 3.1: Распределение времён (в мс), проведённых кластером A55 под управлением различных регуляторов DVFS на различных частотах (в МГц) в тесте Spotify

В Табл. 3.2 представлены оценки производительности SPSA1, OnDemand и Interactive утилитой `AnTuTu` по четырём перечисленным категориям, а в столбце «Общая» показана их сумма как интегральная метрика

[8]Гистограммы для всех экспериментов доступны по адресу `https://drive.google.com/file/d/17ki2HxrFVwYjccV4-pPwIZuD8g8iqoLG/view`

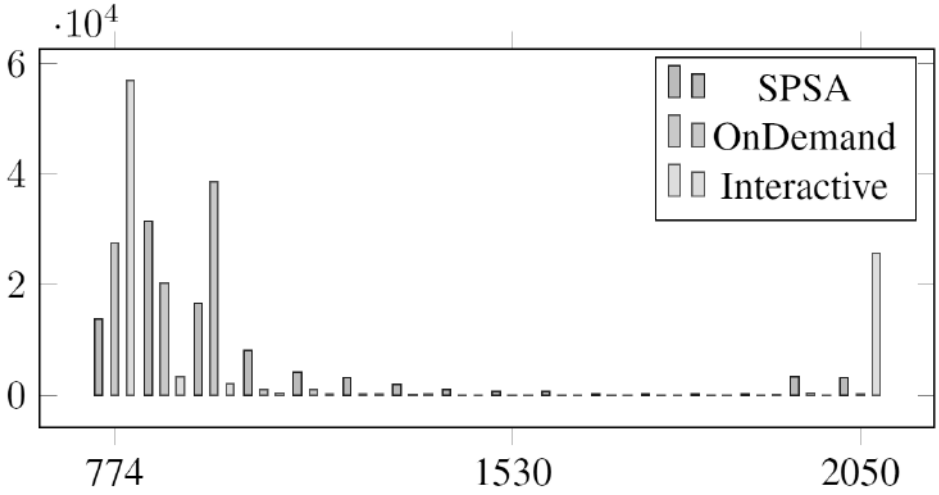


Рис. 3.2: Распределение времён (в мс), проведённых кластером A76 под управлением различных регуляторов DVFS на различных частотах (в МГц) в тесте Spotify

качества, **жирным** выделены лучшие результаты. Результаты регуляторов `Powersave` и `Performance` показаны в качестве минимального и максимального достижимых значений.

Разница по производительности между SPSA1, OnDemand и Interactive практически незаметна для конечного пользователя (3,06% и 0,36% по сравнению с OnDemand и Interactive соответственно), и достигнутая SPSA1 производительность сопоставима с наилучшей достижимой (разница 6,38% с регулятором `Performance`). При сравнении метрик производительности графического процессора, памяти и отзывчивости человеко-машинных интерфейсов, можно заметить, что стратегия регулятора DVFS косвенно влияет и на производительность других компонентов мобильного устройства и, следовательно, на производительность всей системы в целом.

3.2.2 Результаты тестирования регулятора на основе алгоритма с двумя наблюдениями

Для тестирования SPSA2 был использован смартфон Samsung Galaxy s7 SM-G930F. Установленный на нём процессор Exynos 8 Octa (8890) построен по гетерогенной архитектуре с двумя кластерами: 4 ядра Cortex-A53 (LITTLE) и 4 ядра Exynos M1 (big).

Хотя изначально на нём стояла ОС Android 8, для тестирования были установлены сборки ОС Android версий 10 и 11. В частности, на устройство была установлена сборка herolte[9] ядра Android 11, в которой используется планировщик задач EAS. Для тестирования с планировщиком CFS использовалась сборка ядра от корпорации Samsung — android_kernel_samsung_universal8890 для Android 10[10]. Благодаря такому выбору сборок ядра появилась возможность добавить к сравнению с ранее использованными регуляторами Interactive и OnDemand и регулятор schedutil.

По большей части, подготовка тестового стенда и методология экспериментов осталась без изменений. Как и раньше, регуляторы $SPSA2_3$, $SPSA2_2$ и $SPSA2_1$ были добавлены как отдельные модули в сборку, и переключение между регуляторами также производилось с помощью стандартных системных вызовов подсистемы `cpufreq`.

Аналогичными соображениями был продиктован и выбор тестовых сценариев для этой серии экспериментов:

- Проигрывание файла в формате MP4 с помощью проигрывателя *VLC video player*, который был заранее выбран в качестве проигрывателя по умолчанию.
- Сессия игры в *Trial Xtreme 3*.
- Сессия игры в *Flapy Bird*.

[9] `https://github.com/pascua28/herolt`

[10] `https://github.com/8890q/android_kernel_samsung_universal8890/tree/lineage-17.1`

- Создание заметки в приложении *Notes*, написание текста и удаление заметки.
- Запуск приложения для *камеры* по умолчанию и запись видео с его помощью. Получившийся видеофайл удаляется в конце сценария.
- Просмотр видеострима в сервисе *Twitch* в браузере и закрытие браузера спустя какое-то время.

Как и раньше, для запуска тестовых сценариев и управления ими устройство было подключено через USB к управляющему компьютеру и находилось в режиме работы от внешнего источника питания.

Подготовка устройства к прохождению сценариев тестирования проходила по тем же самым принципам, что и в Разделе 3.2. Единственное отличие состояло в том, что тесты запускались десять раз по пять минут на каждый прогон — опыт SPSA1 показал, что уже на пятиминутном интервале влияние фоновых процессов незначительно.

В отличие от предыдущей серии экспериментов для тестирования SPSA2 использовалась как классическая модель энергопотребления от Google, так и предложеннная в этом исследовании. Кроме того, для тестирования производительности устройства использовалась утилита Geekbench 5.5.1, поскольку её методология, в отличие от `AnTuTu`, является открытой. Эта методология подошла для целей исследования [43]. В силу продолжительности тесты производительности запускались 3 раза для каждого регулятора, и были найдены средние значения метрик.

Сперва эксперименты были проведены для ОС Android 11 с планировщиком задач EAS. Подбор целочисленных параметров алгоритма SPSA показал, что лучшими являются следующие наборы:

1. $\alpha = 2, \beta = 1, L_T = 70\%$;
2. Кластер LITTLE: $\alpha = 2, \beta = 1, L_T = 80\%$; Кластер big: $\alpha = 3, \beta = 1, L_T = 98\%$.

Таблица 3.3: Потребление энергии (мА·ч) при использовании Android 11 и диспетчера EAS (базовая энергетическая модель), тесты videoVLC, trialXTreme3 и flappyBird

Алгоритм	videoVLC	trialXTreme3	flappyBird
$SPSA2_3$	21,67	27,62	24,86
	21,43	**24,53**	**24,62**
$SPSA2_2$	23,40	37,44	40,64
	22,65	32,50	34,36
$SPSA2_1$	23,75	43,43	47,04
	22,17	30,43	31,23
Schedutil	24,56	44,30	43,56
Interactive	141,21	119,08	104,73
OnDemand	34,86	86,06	92,86

Таблица 3.4: Потребление энергии (мА·ч) при использовании Android 11 и диспетчера EAS (базовая энергетическая модель), тесты notes, camera и twitch

Алгоритм	notes	camera	twitch
$SPSA2_3$	32,61	22,43	30,59
	30,91	**22,02**	**24,86**
$SPSA2_2$	46,46	25,27	38,62
	39,96	23,67	28,93
$SPSA2_1$	52,19	26,44	47,06
	35,66	22,86	27,32
Schedutil	47,09	27,36	51,83
Interactive	122,31	55,36	91,33
OnDemand	97,43	56,16	80,68

Таблица 3.5: Потребление энергии (мА·ч) при использовании Android 11 и диспетчера EAS (модифицированная энергетическая модель), тесты videoVLC, trialXTreme3 и flappyBird

Алгоритм	videoVLC	trialXTreme3	flappyBird
$SPSA2_3$	**5,90**	14,71	10,23
	6,11	**13,49**	**10,15**
$SPSA2_2$	6,34	19,16	16,27
	6,24	16,71	14,16
$SPSA2_1$	6,31	19,68	18,21
	5,93	15,61	12,75
Schedutil	5,91	20,24	15,70
Interactive	33,74	46,78	35,46
OnDemand	8,59	34,84	32,48

Таблица 3.6: Потребление энергии (мА·ч) при использовании Android 11 и диспетчера EAS (модифицированная энергетическая модель), тесты notes, camera и twitch

Алгоритм	notes	camera	twitch
$SPSA2_3$	8,99	14,41	22,84
	8,67	**13,71**	**18,98**
$SPSA2_2$	**7,69**	17,18	28,18
	12,30	15,24	21,88
$SPSA2_1$	13,02	17,76	33,25
	9,36	15,43	20,70
Schedutil	10,24	15,62	34,92
Interactive	21,83	30,56	58,48
OnDemand	21,35	25,07	47,53

Таблица 3.7: Оценки производительности GeekBench 5.5.1

Алгоритм	Одноядерный режим	Многоядерный режим
$SPSA2_3$	287	970
	282	913
$SPSA2_2$	302	1025
	290	962
$SPSA2_1$	**331**	1073
	324	966
Schedutil	319	970
Interactive	321	**1112**
OnDemand	296	1068

Табл. 3.3–3.6 содержат медианные значения энергопотребления для базовой и модифицированной модели энергопотребления. Табл. 3.7 содержит данные о производительности. **Жирным** выделены лучшие результаты. Первая строка для регуляторов SPSA2 соответствует первому набору параметров, а вторая строка — второму набору.

Следует отметить, что полученные значения для регуляторов SPSA и schedutil имеют заметно более низкий разброс во всех экспериментах по сравнению с OnDemand и особенно с Interactive. Этим обусловлено, что энергопотребление сравнивается по медианному, а не по среднему значению. Например, Interactive показал в тесте videoVLC минимальное и максимальное значения при базовой модели энергопотребления 42,42 и 155,91 мА·ч соответственно. Впрочем, наблюдаемый феномен не влияет на достоверность анализа, так как даже минимальное энергопотребление обоих регуляторов было выше, чем максимальное значение у schedutil или SPSA.

С точки зрения производительности как существующие, так и новые алгоритмы работают сопоставимо. Явных выбросов нет, и для одноядерного сценария разница между лучшей и худшей производительностью находится в пределах 15%, а для многоядерного — в пределах 18%. Отметим, что первый набор параметров SPSA2 обеспечивает лучшую производительность, чем второй набор. Производительность $SPSA2_3$ хуже

из-за того, что алгоритм требуеут трёх итераций цикла DVFS, сопровождающихся ожиданием переключения частоты.

Перед рассмотрением оценок энергопотребления следует отметить, что модифицированная модель энергопотребления заметно изменяет наблюдаемые значения по сравнению с базовой моделью, а в некоторых случаях и их соотношение между отдельными экспериментами. Модифицированная модель, по нашему мнению, лучше отражает реальность, если прямое измерение энергопотребления недоступно, и дальнейший анализ сделан на её основе несмотря на то, что базовая модель более комплиментарна по отношению к регуляторам SPSA2.

Среди стандартных регуляторов Interactive и OnDemand стабильно потребляют больше энергии, чем schedutil при использовании EAS, и при наличии EAS в прошивке устройства для экономии энергопотребления и одновременно сохранения отзывчивости человеко-машинных интерфейсов рекомендуется использовать пару EAS+schedutil.

Регуляторы SPSA2 лучше справляются со сценариями *Trial Xtreme 3* и *Twitch*, чем schedutil. Это связано с использованием параметра $\beta = 1$ в экспериментах, так как более высокие значения β приводят к более быстрому градиентному спуску, а в ситуации быстро меняющейся нагрузки на процессор при наличии задержек переключения частоты относительно плавное изменение рабочей частоты показывает лучший результат. SPSA2_1 с первым набором параметров потребляет больше энергии в большинстве тестов (до 27%), но только на 10% лучше по производительности по сравнению с schedutil, поэтому для него рекомендуется использовать второй набор параметров, так как производительность остаётся на том же уровне, а энергопотребление может быть на 31% лучше. SPSA2_2 является наиболее сбалансированным регулятором, так как, хотя в одних сценариях его энергопотребление до 20% лучше, а в других до 10% хуже, устройство под его управлением показывает прирост производительности на 5,6% в многоядерном случае, что является более реалистичным сценарием с точки зрения современных паттернов человеко-

Таблица 3.8: Потребление энергии (мА·ч) при использовании Android 10 и диспетчера CFS (базовая энергетическая модель), тесты videoVLC, trialXTreme3 и flappyBird

Алгоритм	videoVLC	trialXTreme3	flappyBird
$SPSA2_3$	23,52	49,49	48,67
	23,55	**43,87**	**44,68**
$SPSA2_2$	26,14	66,62	71,74
	24,34	63,82	62,85
$SPSA2_1$	26,58	101,24	116,08
	23,65	57,40	85,69
Interactive	**22,25**	44,31	53,47
OnDemand	29,18	76,41	94,63

машинного взаимодействия. $SPSA2_3$ является наиболее консервативным по сравнению с другими регуляторами семейства SPSA с точки зрения энергопотребления и производительности, но общая многоядерная производительность сравнима с schedutil.

Особым случаем является сценарий videoVLC, где регуляторы SPSA тратят больше энергии, чем schedutil, хотя разница не слишком велика (не более 7,2% для $SPSA2_2$). Соотношение продолжительностей интенсивных вычислений (декодирование кадров) и простоя таково, что регуляторы SPSA не успевают понизить частоту. Единственным исключением является $SPSA2_3$, но разница находится в пределах статистической погрешности.

Таким образом, при работе с планировщиком EAS все реализации регулятора SPSA2 могут использоваться вместо schedutil.

Табл. 3.8–3.11 содержат медианные значения энергопотребления для базовой и модифицированной модели энергопотребления для сценариев, выполненных устройством под управлением Android 10 и планировщика CFS. **Жирным** выделены лучшие результаты. Ситуация в этом случае существенно отличается. Во-первых, schedutil нельзя использовать в качестве эталона, так как он будет работать в условиях, для которых он не предназначен. Регулятор Interactive демонстрирует заметное улучшение

Таблица 3.9: Потребление энергии (мА·ч) при использовании Android 10 и диспетчера CFS (базовая энергетическая модель), тесты notes, camera и twitch

Алгоритм	notes	camera	twitch
$SPSA2_3$	57,73	26,25	38,71
	51,96	25,08	**30,22**
$SPSA2_2$	81,87	36,10	45,66
	70,70	29,28	33,41
$SPSA2_1$	116,36	31,28	50,54
	68,21	25,23	32,04
Interactive	59,52	**24,32**	41,73
OnDemand	101,57	41,39	66,24

Таблица 3.10: Потребление энергии (мА·ч) при использовании Android 10 и диспетчера CFS (модифицированная энергетическая модель), тесты videoVLC, trialXTreme3 и flappyBird

Алгоритм	videoVLC	trialXTreme3	flappyBird
$SPSA2_3$	5,70	13,51	12,07
	5,57	**11,13**	**11,07**
$SPSA2_2$	6,35	17,53	17,27
	5,66	14,57	15,38
$SPSA2_1$	6,20	19,83	25,58
	5,44	11,67	20,35
Interactive	**1,31**	13,28	12,30
OnDemand	7,08	21,65	24,03

Таблица 3.11: Потребление энергии (мА·ч) при использовании Android 10 и диспетчера CFS (модифицированная энергетическая модель), тесты notes, camera и twitch

Алгоритм	notes	camera	twitch
$SPSA2_3$	13,22	4,88	16,53
	11,33	4,48	**13,54**
$SPSA2_2$	17,44	4,88	18,98
	15,22	4,18	14,61
$SPSA2_1$	23,59	3,27	20,50
	14,63	**2,14**	13,91
Interactive	11,71	2,93	16,92
OnDemand	22,03	4,98	23,61

энергопотребления. Все регуляторы SPSA2 демонстрируют энергетическое поведение, похожее на регулятор SPSA1, где в большинстве тестовых сценариев с точки зрения энергопотребленииея они находятся между OnDemand и Interactive, а в некоторых отдельных случаях демонстрируют лучшие или худшие результаты.

Причиной такого энергетического поведения является изменение стратегии планирования. В отличие от EAS, CFS не отдает приоритета ядрам LITTLE над ядрами big, и в подобных ситуациях ядра big загружаются больше, что в свою очередь приводит к большему энергопотреблению. Данная тенденция становится особенно заметно в гетерогенных конфигурациях, где число ядер big равно числу ядер LITTLE или сопоставимо с ним.

Хотя в целом $SPSA2_3$ показывает приемлемое энергопотребление, обнаружилось, что производительность устройства под его управлением ниже, так как при уровнях нагрузки 99—100% он не сразу рекомендует повышать частоту. Поскольку другие алгоритмы демонстрируют более высокое общее энергопотребление, чем Interactive, что соответствует работе на более высоких частотах, а последний обеспечивает приемлемую отзывчивость человеко-машинных интерфейсов, тесты производительности для регуляторов SPSA2 не проводились.

3.3 Анализ результатов тестирования регуляторов

Первый вывод, который можно сделать по результатам тестирования разработанных регуляторов, состоит в том, что на базе всех вариаций алгоритмов одновременно возмущаемой стохастической аппроксимации возможно построить регулятор DVFS, пригодный для повседневного использования в современных мобильных устройствах. Более того, хотя регуляторы разрабатывались под гетерогенную архитектуру ЦП, они могут быть использованы и в устройствах с гомогенной архитектурой.

Алгоритмы, с которыми проводилось сравнение, являются детерминированными, и пытаются найти оптимальное решение в условиях неопределённости. При этом анализ результатов показывает, что регуляторы на основе рандомизированных алгоритмов могут принимать в отдельных случаях неоптимальные решения на основе имеющихся данных, которые в действительности оказываются верными. Таким образом, получение оптимального результата в среднем в долгосрочной перспективе оказывается конкурентным преимуществом разработанных регуляторов.

Эксперименты показывают, что одного адекватно подобранного набора параметров для каждого кластера хватает для построения энергосберегающего регулятора общего назначения, который может работать в широком диапазоне нагрузок. Вместе с тем различное поведение как стандартных, так и разработанных регуляторов показывает, что разные вычислительные нагрузки по-разному нагружают ЦП. Ранее упоминалось, что прикладное программное обеспечение может обладать функцией энергетических профилей, то есть изменять своё энергетическое поведение в зависимости от уровня заряда аккумулятора [90]. По аналогии с этим подходом набор параметров регулятора DVFS, оптимальный по энергопотреблению и производительности для конкретного приложения, можно назвать энергетическим профилем уровня системного программного обеспечения. С учётом результатов, полученных в данном исследовании, задача развития ОС по автоматическому или направляемому пользователем применению таких энергетических профилей для различных нагрузок перестаёт быть концептуальной и становится чисто технической.

Заключение

Основные научные результаты диссертационного исследования, полученные в рамках выполнения поставленных задач:

1. Предложена и обоснована модель оценки энергопотребления ЦП, построенного по гетерогенной архитектуре, учитывающая динамическую вычислительную нагрузку и пребывание в состоянии простоя, даны практические рекомендации по её применению.

2. Разработан подход к решению задачи управления энергопотреблением гетерогенного ЦП на основе рандомизированных алгоритмов стохастической оптимизации, в рамках которого предложены и обоснованы функционалы среднего риска для алгоритмов SPSA с одним и двумя измерениями. Исследована и установлена теоретическая состоятельность оценок, предоставляемых разработанными алгоритмами, в рамках ограничений, накладываемых особенностями работы ЦП.

3. Разработаны модули DVFS, оптимизирующие энергопотребление ЦП с учётом особенностей его работы, основанные на предложенных функционалах и алгоритмах SPSA с одним и двумя измерениями. На базе подготовленных тестовых стендов, управляемых ОС Android, проведено сравнение с существующими аналогами. На модули, основанные на алгоритмах SPSA с двумя измерениями, получено свидетельство о государственной регистрации программ на ЭВМ.

Литература

[1] *Амелин К. С. и др.* Адаптивное управление автономной группой беспилотных летательных аппаратов // Стохастическая оптимизация в информатике. -– 2009. — Т. 5. -– С. 157–166.

[2] *Вазан М.* Стохастическая аппроксимация. – М.: Мир, 1972. – 295 с.

[3] *Амелина Н. О., Иванский Ю. В.* Задача достижения дифференцированного консенсуса при стоимостных ограничениях // Вестник Санкт-Петербургского университета. Математика. Механика. Астрономия. — 2015. — Т. 2. — № 4. — С. 495–506.

[4] *Вахитов А. Т., Граничин О. Н.* Рандомизированные алгоритмы оценивания при нерегулярных помехах // Стохастическая оптимизация в информатике. – 2006. – Т. 2. – С. 3–37.

[5] *Гасников А. В. и др.* Численные методы поиска равновесного распределения потоков в модели Бэкмана и в модели стабильной динамики // Математическое моделирование. – 2016. — Т. 28. — № 10. — С. 40–64.

[6] *Граничин О. Н.* Об одной стохастической рекуррентной процедуре при зависимых помехах в наблюдении, использующей на входе пробные возмущения // Вестник Ленинградского университета. Серия 1: Математика, механика, астрономия. – 1989. – № 1. – С. 19–21.

[7] *Граничин О. Н.* Процедура стохастической аппроксимации с возмущением на входе // Автоматика и телемеханика. — 1992. — № 2. — С. 97–104.

[8] *Граничин О. Н.* Оценивание точки минимума неизвестной функции, наблюдаемой на фоне зависимых помех // Проблемы передачи информации. – 1992. – Т. 28. – № 2. – С. 16–20.

[9] *Граничин О. Н.* Рандомизированные алгоритмы стохастической аппроксимации при произвольных помехах // Автоматика и телемеханика. — 2002. — № 2. — С. 44–55.

[10] *Граничин О. Н.* Оптимальная скорость сходимости рандомизированных алгоритмов стохастической аппроксимации при произвольных помехах // Автоматика и телемеханика. – 2003. – № 2. – С. 88–99.

[11] *Граничин О.Н., Поляк Б.Т.* Рандомизированные алгоритмы оценивания и оптимизации при почти произвольных помехах. – М.: Наука, 2003. – 291 с.

[12] *Граничин О. Н., Краснощеков В. Е.* Алгоритмы оптимизации энергопотребления в мобильных устройствах // Нейрокомпьютеры: разработка, применение. — 2007. — № 6. — С. 81–86.

[13] *Михалевич В.С., Гупал А.М., Норкин В.И.* Методы невыпуклой оптимизации. – М.: Мир, 1987. – 279 с.

[14] *Поляк Б. Т., Цыбаков А. Б.* Оптимальные порядки точности поисковых алгоритмов стохастической оптимизации // Проблемы передачи информации. – 1990. – Т. 26. – № 2. — С. 45–53.

[15] *Сартасов С.Ю., Богданов Е.А., Божнюк А.С., Быков Д.В., Граничин О.Н.* Проблемы и перспективы использования стохастической аппроксимации для регулирования частот процессора в Android OS // Компьютерные инструменты в образовании. – 2021. – № 2. – С. 26–40.

[16] *Сартасов С.Ю., Пелогейко М.А., Граничин О.Н.* Энергосберегающие регуляторы DVFS для Android OS на основе алгоритма SPSA с двумя измерениями и регулятора OnDemand. – Свидетельство о регистрации программы для ЭВМ №2023666564 от 02.08.23.

[17] *Ahmad R. W. et al.* A case and framework for code analysis–based smartphone application energy estimation // International Journal of Communication Systems. – 2017. – Vol.30, №. 10.

[18] *Ahmad R. W. et al.* Enhancement and assessment of a code-analysis-based energy estimation framework // IEEE Systems Journal. — 2018. — Vol. 13, no. 1. – P. 1052–1059.

[19] *Aldahlawi A., Kim Y.B., Kim K.K.* GPU architecture optimization for mobile computing // 2019 International SoC Design Conference (ISOCC). -– IEEE, 2019. -– P. 247–248.

[20] *Amelina N. et al.* Approximate consensus in stochastic networks with application to load balancing //IEEE Transactions on Information Theory. – 2015. – Vol.61. – №. 4. – P. 1739–1752.

[21] *Banerjee A. et al.* Detecting energy bugs and hotspots in mobile apps // In: Proc. of the 22nd ACM SIGSOFT international symposium on foundations of software engineering. – 2014. – P. 588–598.

[22] *Bareth U.* Simulating power consumption of location tracking algorithms to improve energy-efficiency of smartphones // In: Proc. of 2012 IEEE 36th Annual Computer Software and Applications Conference. – IEEE, 2012. – P. 613–622.

[23] *Basireddy K. R. et al.* AdaMD: Adaptive mapping and DVFS for energy-efficient heterogeneous multicores // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. – 2019. – Vol. 39, no. 10. – P. 2206–2217.

[24] *Beard K.W.* Linden's Handbook of Batteries, 5th Edition. – McGraw-Hill Education, 2019. – 1456 p.

[25] *Blum J. R.* Multidimensional stochastic approximation methods // The annals of mathematical statistics. – 1954. – p. 737–744.

[26] *Bogdanov E., Bozhnyuk A., Bykov D., Sartasov S., Sergeenko A., Granichin O.* Dynamic Voltage-Frequency Optimization using Simultaneous Perturbation Stochastic Approximation // In: Proc. of 60th IEEE Conference on Decision and Control (CDC) 2021. – 2021. – P. 3774–3779.

[27] *Bogdanov E., Bozhnyuk A., Sartasov S., Granichin O.* On Application of Simultaneous Perturbation Stochastic Approximation for Dynamic Voltage-Frequency Scaling in Android OS // In: Proc. of 7th International Conference on Event-Based Control, Communication, and Signal Processing (EBCCSP) 2021. – 2021. – P. 1–7.

[28] Broyde L. et al. MobiCore: An adaptive hybrid approach for power-efficient CPU management on Android devices // In: Proc. of 30th

IEEE International System-on-Chip Conference (SOCC 2017). — 2017. – P. 221–226.

[29] *Carette A. et al.* Investigating the energy impact of android smells // In: Proc. of 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER). – IEEE, 2017. – P. 115–126.

[30] *Chae S.-H., Yoo Ch.-H, Sun J.-Y., Kang M.-Ch., Ko S.-J.* Subpixel rendering for the pentile display based on the human visual system // IEEE Transactions on Consumer Electronics. 2017. – Vol. 63, № 4. – P. 401–409.

[31] *Chen H.-F., Duncan T. E., Pasik-Duncan B.* A Kiefer-Wolfowitz algorithm with randomized differences // IEEE Transactions on Automatic Control. – 1999. — Vol. 44, no. 3. – P. 442–453.

[32] *Chen X., Zong Z.* Android app energy efficiency: The impact of language, runtime, compiler, and implementation // Proceedings of 2016 IEEE international conferences on big data and cloud computing (BDCloud), social computing and networking (socialcom), sustainable computing and communications (sustaincom)(BDCloud-socialcom-sustaincom). – IEEE, 2016. – P. 485–492.

[33] *Chen Y.L., Chang M.F., Yu Ch.W., Chen, X.Zh., Liang W.Y.* Learning-directed dynamic voltage and frequency scaling scheme with adjustable performance for single-core and multi-core embedded and mobile systems // Sensors. – 2018. – Vol. 12, no. 9.

[34] *Chung J. M. et al.* Adaptive Cloud Offloading of Augmented Reality Applications on Smart Devices for Minimum Energy Consumption // Ksii Transactions on Internet & Information Systems. -– 2015. — Vol. 9, no. 8.

[35] *Chung Y. F., Lin C. Y., King C. T.* Aneprof: Energy profiling for android java virtual machine and applications // Proceedings of 2011 IEEE 17th International Conference on Parallel and Distributed Systems. – IEEE, 2011. – P. 372–379.

[36] *Couto M. et al.* GreenDroid: A tool for analysing power consumption in the android ecosystem // In: Proc. of 2015 IEEE 13th International Scientific Conference on Informatics. – IEEE, 2015. – P. 73–78.

[37] CPU frequency and voltage scaling code in the Linux (TM) kernel. Linux CPUFreq. CPUFreq Governors. Available at: https://android.googlesource.com/kernel/common/+/a7827a2a60218b25f222b54f77ed38f57aebe08b/Documentation/cpu-freq/governors.txt. (accessed 10.08.2023).

[38] *Cruz L., Abreu R.* Catalog of energy patterns for mobile applications // Empirical Software Engineering. – 2019. – Vol. 24. – P. 2209–2235.

[39] *Dey S. et al.* CPU-GPU-Memory DVFS for Power-Efficient MPSoC in Mobile Cyber Physical Systems // Future Internet. – 2022. – Vol. 14, no. 3.

[40] *Di Nucci D. et al.* Petra: a software-based tool for estimating the energy profile of android applications // Proceedings of the 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C). – IEEE, 2017. – P. 3–6.

[41] *Dong M., Lan T., Zhong L.* Rethink energy accounting with cooperative game theory // Proceedings of the 20th annual international conference on Mobile computing and networking. – 2014. – P. 531–542.

[42] *Dolezal J., Becvar Z.* Methodology and tool for energy consumption modeling of mobile devices // Proceedings of 2014 IEEE Wireless Communications and Networking Conference Workshops (WCNCW). – IEEE, 2014. – P. 34–39.

[43] Geekbench 5 CPU Workloads. Available at: https://www.geekbench.com/doc/geekbench5-cpu-workloads.pdf. (accessed 10.05.2023).

[44] *Feghhi M.* Multi-layer tracing of android applications for energy-consumption analysis. – Master Thesis. – 2017.

[45] *Fischer L. M., de Brisolara L. B., De Mattos J. C. B.* Sema: An approach based on internal measurement to evaluate energy efficiency of android applications // 2015 Brazilian Symposium on Computing Systems Engineering (SBESC). – IEEE, 2015. – P. 48–53.

[46] *Gao X. et al.* E-android: A new energy profiling tool for smartphones // 2017 IEEE 37th international conference on distributed computing systems (ICDCS). – IEEE, 2017. – P. 492–502.

[47] *Granichin O.* Linear regression and filtering under nonstandard assumptions (Arbitrary noise) // IEEE Transactions on Automatic Control. – 2004. – Vol. 49, no. 10. – P. 1830–1837.

[48] *Granichin O., Gurevich L., Vakhitov A.* Discrete-time minimum tracking based on stochastic approximation algorithm with randomized differences // Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference. – IEEE, 2009. – P. 5763–5767.

[49] *Granichin O., Amelina N.* Simultaneous perturbation stochastic approximation for tracking under unknown but bounded disturbances // IEEE Transactions on Automatic Control. – 2015. – Vol. 60. – No. 6. – P. 1653–1658.

[50] *Granichin O., Volkovich Z., Toledano-Kitai D.* Randomized Algorithms in Automatic Control and Data Mining. – 2015. – 249 p.

[51] *Greenhalgh, P.* big.LITTLE technology: The future of mobile [White paper] // ARM Limited. – 2013.

[52] *Hao S. et al.* Estimating Android applications' CPU energy usage via bytecode profiling // 2012 First international workshop on green and sustainable software (GREENS). – IEEE, 2012. – P. 1–7.

[53] *Hindle A. et al.* Greenminer: A hardware based mining software repositories software energy consumption framework // In: Proc. of the 11th Working Conference on Mining Software Repositories. – 2014. – P. 12–21.

[54] *Hu Y. et al.* Lightweight energy consumption analysis and prediction for Android applications // Science of Computer Programming. – 2018. – Vol. 162. – P. 132–147.

[55] *Huang J. et al.* A close examination of performance and power characteristics of 4G LTE networks // Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services. – 2012. – P. 225–238.

[56] *Hung S. H. et al.* Performance and power estimation for mobile-cloud applications on virtualized platforms // Proceedings of the 2013

Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing. – IEEE, 2013. – P. 260–267.

[57] *Jung W., Kim K., Cha H.* Userscope: A fine-grained framework for collecting energy-related smartphone user contexts // 2013 International Conference on Parallel and Distributed Systems. – IEEE, 2013. – P. 158–165.

[58] *Kamiyama T., Inamura H., Ohta K.* A model-based energy profiler using online logging for Android applications // Proceedings of the 2014 Seventh International Conference on Mobile Computing and Ubiquitous Networking (ICMU). – IEEE, 2014. – P. 7–13.

[59] *Kapetanakis K., Panagiotakis S.* Efficient energy consumption's measurement on android devices // Proceedings of 2012 16th Panhellenic Conference on Informatics. – IEEE, 2012. – P. 351–356.

[60] Energy Aware Scheduling. Available at: `https://www.kernel.org/doc/html/next/scheduler/sched-energy.html`. (accessed 06.08.2023).

[61] Schedutil. Available at: `https://docs.kernel.org/scheduler/schedutil.html`. (accessed 12.08.2023)

[62] *Kiefer J., Wolfowitz J.* Statistical estimation on the maximum of a regression function // The Annals of Mathematical Statistics. – 1952. – Vol. 23. – P. 462–466.

[63] *Kim H. J., Kyong J., Lim S. S.* A systematic power and performance analysis framework for heterogeneous multiprocessor system // IEMEK Journal of Embedded Systems and Applications. – 2014. – Vol. 9, no. 6. – P. 315–321.

[64] *Kim S., Bin K., Ha S., Lee K., Chong S.* ZTT: learning-based DVFS with zero thermal throttling for mobile devices // In: Proc. of the 19th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '21). – 2021. – P. 41-–53.

[65] *Larsson M., Stigelid M.* Energy Efficient Data Synchronization in Mobile Applications. A Comparison Between Different Data Synchronization Techniques. – Ph.D. Thesis – 2015.

[66] *Lee J., Nam S., Park S.* Energy-Efficient Control of Mobile Processors Based on Long Short-Term Memory // IEEE Access. – 2019. – Vol. 7. – P. 80552-80560.

[67] *Lee S., Yoon C., Cha H.* User interaction-based profiling system for android application tuning // Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing. – 2014. – P. 289–299.

[68] *Lee S. et al.* EnTrack: a system facility for analyzing energy consumption of Android system services // Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing. – 2015. – P. 191–202.

[69] *Li D. et al.* Energy-directed test suite optimization // 2013 2nd International Workshop on Green and Sustainable Software (GREENS). – IEEE, 2013. – P. 62–69.

[70] *Li X., Gallagher J. P.* Fine-grained energy modeling for the source code of a mobile application // Proceedings of the 13th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services. – 2016. – P. 180–189.

[71] *Li X., Wen W., Wang X.* Usage history-directed power management for smartphones // Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). – Springer. – 2015.

[72] *Marr D. T. et al.* Hyper-threading technology architecture and microarchitecture // Intel Technology Journal. – 2002. – Vol. 6, no. 1.

[73] Measuring Power Values. Available at: `https://source.android.com/docs/core/power/values`. (accessed 06.08.2023).

[74] *Metri G.C.* Energy Efficiency Analysis and Optimization for Mobile Platforms. – Wayne State University, 2014.

[75] *Metri G.C. et al.* A simplistic way for power profiling of mobile devices // In: Proc. of 2012 International Conference on Energy Aware Computing. — 2012. — P. 1–6.

[76] *Mittal R., Kansal A., Chandra R.* Empowering developers to estimate app energy consumption // In: Proc. of the 18th Annual International Conference on Mobile Computing and Networking. – 2012. – P. 317–328.

[77] *Oliveira W. et al.* Recommending energy-efficient java collections // 2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR). – IEEE, 2019. – P. 160–170.

[78] *Myasnikov V., Sartasov S., Slesarev I., Gessen P.* Energy consumption measurement frameworks for Android OS: A systematic literature review // In: Proc. of the Fifth Conference on Software Engineering and Information Management (SEIM 2020). – 2020.

[79] Number of smartphone mobile network subscriptions worldwide from 2016 to 2022, with forecasts from 2023 to 2028. Available at: `https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/`. (accessed 06.08.2023)

[80] *Ohk S. R., Kim Y. S., Kim Y. J.* Phase-based low power management combining CPU and GPU for Android smartphones // Electronics. — 2022. — Vol. 11, no. 16. — P. 2480.

[81] *Pandiyan D., Wu C. J.* Quantifying the energy cost of data movement for emerging smart phone workloads on mobile platforms // Proceedings of 2014 IEEE International Symposium on Workload Characterization (IISWC). – IEEE, 2014. – P. 171–180.

[82] *Patros P. et al.* Toward sustainable serverless computing // IEEE Internet Computing. — IEEE, 2021. — Vol. 25, no. 6. — P. 42–50.

[83] *Pelogeiko M., Sartasov S., Granichin O.* On stochastic optimization for smartphone CPU energy consumption decrease // Informatics and Automation (SPIIRAS Proceedings). – 2023. – Vol. 22. – no. 5. – P. 1004–1033.

[84] *Polyak B. T., Tsybakov A. B.* On stochastic approximation with arbitrary noise (the KW case) / Topics in Nonparametric Estimation. Khasminskii R.Z. eds. // Advances in Soviet Mathematics. Amer. Math. Soc. Providence. – 1992. – Vol. 12. – P. 107–113.

[85] *Poornambigai K., Raj M. L., Meena P.* Reducing the energy consumption using dvfs performance optimizing scheme //EPRA

International Journal of Research and Development (IJRD). — 2017. — Vol. 2. -– no. 1. — P. 79–88.

[86] Power Profiles for Android. Available at: `https://source.android.com/docs/core/power`. (accessed 06.08.2023).

[87] *Rapp M., Krohmer N., Khdr H., Henkel J.* NPU-accelerated imitation learning for thermal- and QoS-aware optimization of heterogeneous multi-cores // In: Proc. of the 2022 Conference & Exhibition on Design, Automation & Test in Europe (DATE '22). – 2021. – P. 584–587.

[88] *Robbins H., Monro S.* A stochastic approximation method // The Annals of Mathematical Statistics. – 1951. – Vol. 22. – P. 400–407.

[89] *Sahar H., Bangash A. A., Beg M. O.* Towards energy aware object-oriented development of android applications // Sustainable Computing: Informatics and Systems. – 2019. – Vol. 21. – P. 28–46.

[90] *Sahin C. et al.* Initial explorations on design pattern energy usage // 2012 First International Workshop on Green and Sustainable Software (GREENS). -– IEEE, 2012. – P. 55–61.

[91] *Saksonov A.* Method to Derive Energy Profiles for Android Platform. – Master Thesis. – 2014.

[92] *Sartasov S., Miroshnikov V., Kuznetsov I.* Frequency-independent smartphone peripherals energy consumption estimation // Cybernetics and Physics. – 2023. – Vol. 12. – no. 1. – P. 42–50.

[93] *Sartasov S. et al.* Navitas Framework: A Novel Tool for Android Applications Energy Profiling // In: Proc. of the Sixth Conference on Software Engineering and Information Management (SEIM 2021). – 2021.

[94] *Shin D. et al.* Online estimation of the remaining energy capacity in mobile systems considering system-wide power consumption and battery characteristics // Proceedings of 2013 18th Asia and South Pacific Design Automation Conference (ASP-DAC). – IEEE, 2013. – P. 59–64.

[95] *Song S., Kim J., Chung J.-M.* Energy consumption minimization control for augmented reality applications based on multi-core smart devices // In: Proc. of 2019 IEEE International Conference on Consumer Electronics (ICCE). – 2019. – P. 1–4.

[96] *Spall J.C.* Multivariate stochastic approximation using a simultaneous perturbation gradient approximation // IEEE Transactions on Automatic Control. – 1992. – Vol. 37, no. 3. – P. 332–341.

[97] *Spall J.C.* A one measurement form of simultaneous perturbation stochastic approximation // Automatica. – 1997. – Vol. 33. – P. 109–112.

[98] *Spall J. C.* An overview of the simultaneous perturbation method for efficient optimization / Johns Hopkins apl technical digest. – 1998. – Vol. 19. – No. 4. – P. 482–492.

[99] *Tsao S. L. et al.* Powermemo: A power profiling tool for mobile devices in an emulated wireless environment // Proceedings of 2012 International Symposium on System on Chip (SoC). – IEEE, 2012. – P. 1–5.

[100] *Tuysuz M. F., Ucan M., Trestian R.* A real-time power monitoring and energy-efficient network/interface selection tool for android smartphones // Journal of Network and Computer Applications. – 2019. –Vol. 127. – P. 107–121.

[101] *Vakhitov A. T., Granichin O. N.* SPSA-based adaptive control: accuracy of estimates // IFAC Proceedings Volumes. – 2007. – Vol. 40, no. 13. – P. 429–434.

[102] *Walcott-Justice K.* MAUE: A Framework For Detecting Energy Bugs From User Interactions On Mobile Applications. – Master Thesis. – 2016.

[103] *Weiser M., Welch B., Demers A., Shenker S.* Scheduling for reduced CPU energy // Mobile Computing. – Springer. – 1996. – P. 449–471.

[104] *Westfield B., Gopalan A.* Orka: A new technique to profile the energy usage of Android applications // In: Proc. of 2016 5th International Conference on Smart Cities and Green ICT Systems (SMARTGREENS). – IEEE, 2016. – P. 1–12.

[105] *Wilke C., Götz S., Richly S.* Jouleunit: a generic framework for software energy profiling and testing // Proceedings of the 2013 Workshop on Green in/by Software Engineering. – 2013. – P. 9–14.

[106] *Wilke C.* Energy-Aware Development and Labeling for Mobile Applications. – Ph.D. Thesis. – 2014.

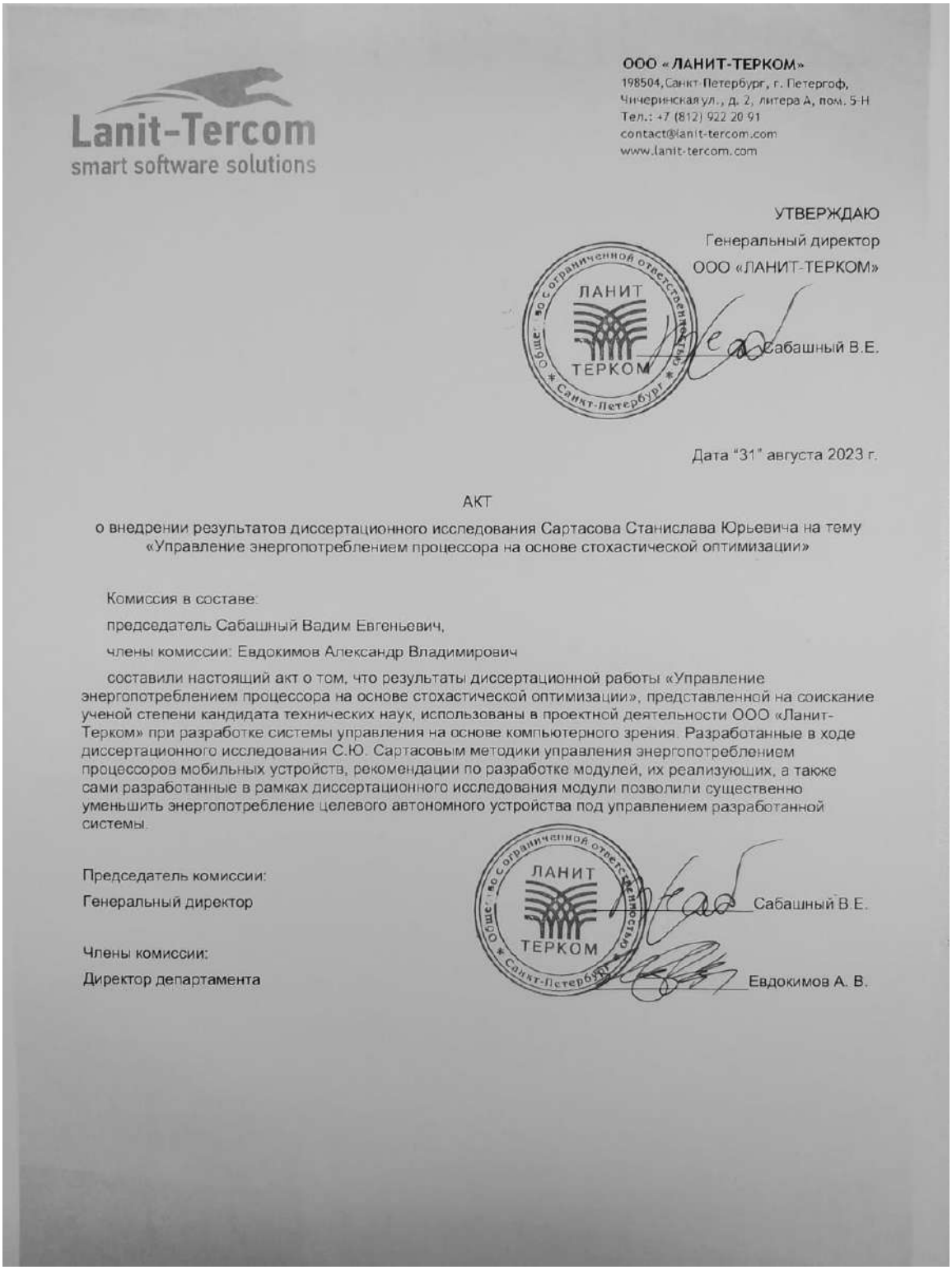
[107] *Wu H., Sun Y., Wolter K.* Energy-efficient decision making for mobile cloud offloading // IEEE Transactions on Cloud Computing. — IEEE, 2018. — Vol. 8, no. 2. – P. 570–584.

[108] *Wysocky R.J.* CPU Performance Scaling. Available at: `https://www.kernel.org/doc/html/latest/admin-guide/pm/cpufreq.html`. (accessed 06.08.2023).

[109] *Yoon C. et al.* AppScope: Application energy metering framework for Android smartphone using kernel activity monitoring // Proceedings of the 2012 USENIX Annual Technical Conference (USENIX ATC 12). – 2012. – P. 387–400.

[110] *Zhang L. et al.* Accurate online power estimation and automatic battery behavior based power model generation for smartphones // Proceedings of the Eighth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis. – 2010. – P. 105–114.

Приложение А. Акты о внедрении



Lanit-Tercom
smart software solutions

ООО «ЛАНИТ-ТЕРКОМ»
198504, Санкт-Петербург, г. Петергоф,
Чичеринская ул., д. 2, литера А, пом. 5-Н
Тел.: +7 (812) 922 20 91
contact@lanit-tercom.com
www.lanit-tercom.com

УТВЕРЖДАЮ
Генеральный директор
ООО «ЛАНИТ-ТЕРКОМ»
Сабашный В.Е.

Дата "31" августа 2023 г.

АКТ

о внедрении результатов диссертационного исследования Сартасова Станислава Юрьевича на тему «Управление энергопотреблением процессора на основе стохастической оптимизации»

Комиссия в составе:

председатель Сабашный Вадим Евгеньевич,

члены комиссии: Евдокимов Александр Владимирович

составили настоящий акт о том, что результаты диссертационной работы «Управление энергопотреблением процессора на основе стохастической оптимизации», представленной на соискание ученой степени кандидата технических наук, использованы в проектной деятельности ООО «Ланит-Терком» при разработке системы управления на основе компьютерного зрения. Разработанные в ходе диссертационного исследования С.Ю. Сартасовым методики управления энергопотреблением процессоров мобильных устройств, рекомендации по разработке модулей, их реализующих, а также сами разработанные в рамках диссертационного исследования модули позволили существенно уменьшить энергопотребление целевого автономного устройства под управлением разработанной системы.

Председатель комиссии:
Генеральный директор Сабашный В.Е.

Члены комиссии:
Директор департамента Евдокимов А. В.

МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Балтийский государственный технический университет «ВОЕНМЕХ» им. Д.Ф. Устинова»
(БГТУ «ВОЕНМЕХ» им. Д.Ф. Устинова)

Санкт-Петербург, 190005, 1-я Красноармейская ул., д. 1
Тел.: (812) 316-2394, Факс: (812) 490-0591
E-mail: komdep@bstu.spb.su. www.voenmeh.ru
ИНН 7809003047

12.09.2023 № 3/506

На № _____ от _____

УТВЕРЖДАЮ
Проректор по НР и ИР
С.А. Матвеев
«__» _____ 2023 г.

АКТ

о внедрении результатов диссертационного исследования Сартасова Станислава Юрьевича на тему «Управление энергопотреблением процессора на основе стохастической оптимизации»

Комиссия в составе:

председатель Тучкин Игорь Ревович,

члены комиссии: Лукичев Вадим Юрьевич

составили настоящий акт о том, что результаты диссертационной работы «Управление энергопотреблением процессора на основе стохастической оптимизации», представленной на соискание ученой степени кандидата технических наук, использованы в опытно-конструкторской работе БГТУ «ВОЕНМЕХ» им. Д.Ф. Устинова при разработке СЧ ОКР «Разработка макета системы документирования изображений подстилающей поверхности для стенда» АО «ВПК «НПО машиностроения» в виде методик управления энергопотреблением процессора и рекомендаций по разработке модулей, их реализующих.

Председатель комиссии:

Начальник Научно-Исследовательской лаборатории

«Сенсорики и Искусственного интеллекта»

Тучкин И.Р.

Члены комиссии:

Научный руководитель Научно-Исследовательской лаборатории

«Сенсорики и Искусственного интеллекта», к.т.н.

Лукичев В.Ю.

Приложение В. Свидетельство о государственной регистрации программы для ЭВМ

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2023666564

Энергосберегающие регуляторы DVFS для Android OS на основе алгоритма SPSA с двумя измерениями и регулятора OnDemand

Правообладатели: ***Сартасов Станислав Юрьевич (RU), Пелогейко Макар Андреевич (RU), Граничин Олег Николаевич (RU)***

Авторы: ***Сартасов Станислав Юрьевич (RU), Пелогейко Макар Андреевич (RU), Граничин Олег Николаевич (RU)***

Заявка № **2023664962**
Дата поступления **12 июля 2023 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***02 августа 2023 г.***

Руководитель Федеральной службы по интеллектуальной собственности

Ю.С. Зубов