Компьютерные инструменты в образовании, 2021
№ 2: 26–40
УДК: 004.82
http://cte.eltech.ru
doi:10.32603/2071-2340-2021-2-26-40

ПРОБЛЕМЫ И ПЕРСПЕКТИВЫ ИСПОЛЬЗОВАНИЯ СТОХАСТИЧЕСКОЙ АППРОКСИМАЦИИ ДЛЯ РЕГУЛИРОВАНИЯ ЧАСТОТЫ ПРОЦЕССОРА В ANDROID OS*

Сартасов С. Ю.[1], старший преподаватель, ✉ stanislav.sartasov@spbu.ru
Богданов Е. А.[1], выпускник бакалавриата, evgenij.bogdanov.1999@gmail.com
Божнюк А. С.[1], студент, bozhnyuks@mail.ru
Быков Д. В.[1], студент, bykov.david@gmail.com
Граничин О. Н.[1], доктор физико-математических наук, профессор, Oleg_granichin@mail.ru

[1]Санкт-Петербургский государственный университет,
Университетский пр., д. 28, Старый Петергоф, 198504, Санкт-Петербург, Россия

Аннотация

Алгоритмы стохастической аппроксимации со случайными направлениями демонстрируют хорошие результаты в задачах оптимизации при динамически меняющемся состоянии системы. В работе рассматривается построение регулятора частоты центрального процессора смартфона под управлением Android OS на базе указанных алгоритмов, а также указываются направления для его дальнейшего усовершенствования. Результаты сравнения показывают, что полученный регулятор работает на уровне современных регуляторов, используемых в смартфонах.

Ключевые слова: *стохастическая аппроксимация, регулирование частоты, энергосбережение, Android OS.*

Цитирование: Сартасов С. Ю., Богданов Е. А., Божнюк А. С., Быков Д. В., Граничин О. Н. Проблемы и перспективы использования стохастической аппроксимации для регулирования частоты процессора в Android OS // Компьютерные инструменты в образовании. 2021. № 2. С. 26–40. doi: 10.32603/2071-2340-2021-2-26-49

1. ВВЕДЕНИЕ

Смартфоны и планшеты стали неотъемлемой частью современной жизни. Хотя с их помощью удаётся заметно повысить уровень жизни, основным ограничивающим фактором в их использовании является энергопотребление. С увеличением числа активно работающих приложений и включенных периферийных устройств увеличивается и скорость разряда аккумулятора. Актуальность этой проблемы понятна любому, кто сталкивался с разряженным смартфоном, когда надо было сделать важный звонок или найти на нём срочно необходимые сведения.

* Исследование поддержано Санкт-Петербургским государственным университетом, проект № 73555239.

Заряд аккумулятора конечен и ограничен сверху. Для повышения эффективности его использования существуют различные подходы, например, достижения в области электрохимии за последние десятилетия заметно увеличили ёмкость аккумуляторов. Аппаратное обеспечение проектируется с возможностью регулирования энергопотребления в своей работе, особенно в ситуациях, когда оно мало используется или не используется вообще. В прикладном программном обеспечении применяются практики «зелёной» разработки, например энергетические профили или энергетические рефакторинги [1]. Конечный пользователь устройства может выключить неиспользуемые функции или удалить энергозатратные приложения. В результате общая экономия электроэнергии может быть существенной.

В этой работе рассматривается оптимизация энергопотребления центрального процессора (ЦП) на уровне операционной системы (ОС). Современные мобильные ЦП поддерживают изменение частоты и напряжения в процессе работы (Dynamic Voltage Frequency Scaling, DVFS) — метод, позволяющий увеличивать или уменьшать рабочее напряжение и тактовую частоту для экономии электроэнергии или, напротив, для увеличения производительности системы. Специальные модули ОС, регуляторы DVFS (DVFS governors), напрямую меняют частоту ЦП в зависимости от текущей вычислительной нагрузки на уровне всего ЦП или отдельных ядер в зависимости от аппаратной архитектуры. На сегодняшний день разработано существенное количество регуляторов.

В определённом смысле регуляторы DVFS, оптимизирующие энергопотребление, замедляют исполнение программ настолько, чтобы это не вызывало дискомфорт у пользователя. Если формализовать качество исполнения вычислительной нагрузки для заданных настроек DVFS, то задачу регулировки частоты и напряжения можно рассматривать как оптимизационную задачу. Для её решения можно применить алгоритмы стохастической аппроксимации со случайными направлениями (Simultaneous Perturbation Stochastic Approximation, SPSA). SPSA можно рассматривать как разновидность случайного поиска, и в среднем алгоритм будет почти следовать в направлении градиентного спуска. [2]. Хотя ранее этот подход применялся в различных областях информатики [3, 4], возможности его использования для разработки регуляторов DVFS в современных смартфонах не изучалась. Таким образом, главным результатом этой работы является разработка нового алгоритма для динамической регулировки частоты ЦП на основе подхода SPSA, сравнение его с широко распространёнными в смартфонах на базе Android OS алгоритмами регулирования DVFS OnDemand и Interactive, а также анализ проблем и перспектив этого алгоритма.

Работа построена следующим образом. В разделе 2 сделан обзор алгоритмов DVFS в современных смартфонах, описан общий алгоритм SPSA, а также представлены результаты научных работ в смежных областях. В разделе 3 описывается архитектура предлагаемого регулятора DVFS. Экспериментальная методология представлена в разделе 4, сами же эксперименты задокументированы в разделе 5. В разделе 6 рассказывается о дальнейших улучшениях для этого алгоритма, и в разделе 7 делается заключение.

2. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

2.1. Регулирование частоты и напряжения центрального процессора в Android OS

Среди реализованных алгоритмов DVFS некоторые стандартные алгоритмы были унаследованы Android OS от операционной системы Linux, другие же были специально раз-

работаны для неё [5]. Как правило, в смартфонах на базе этой ОС встречается какое-либо подмножество из далее описанных алгоритмов:

PowerSave. Этот регулятор уменьшает частоту ЦП до минимального доступного значения и поддерживает её на этом уровне для минимизации энергопотребления.

Performance. Этот регулятор работает диаметрально противоположным образом — для максимизации быстродействия и энергопотребления он устанавливает для ЦП максимальную доступную частоту и держит её на этом уровне.

OnDemand. Этот регулятор повышает частоту ЦП при появлении вычислительных задач, чтобы система могла поддерживать взаимодействие с пользователем. В отсутствие активных вычислений частота снижается. При достижении определённого уровня загруженности ЦП (~80% активного времени от общего времени работы) регулятор повышает частоту до максимальной и держит её на этом уровне до уменьшения вычислительной нагрузки.

Conservative. В отсутствие нагрузки этот регулятор использует минимальную частоту, а при её появлении частота постепенно увеличивается.

Interactive. Этот регулятор был разработан для вычислительных нагрузок, чувствительных к задержкам их обработки, например для обработки событий интерактивных пользовательских интерфейсов. Рабочая частота зависит от уровня нагрузки, и проверка на её изменение происходит не по таймеру, а при наступлении определённых событий, к примеру, при взаимодействии пользователя с устройством.

На базе указанных алгоритмов DVFS сторонними разработчиками было создано значительное количество их улучшений, однако в литературе описаны и качественно иные подходы для регулирования частоты.

Используя данные, полученные от ЦП (температура и потребляемая мощность), управлять DVFS можно на основе регрессионной модели, обученной выставлять требуемые параметры для каждой целевой вычислительной нагрузки. По наблюдениям авторов работы, энергосбережение по сравнению с типовыми алгоритмами составляет в среднем 10 % [6].

Алгоритм на основе нейронных сетей встречного распространения анализирует текущее поведение системы (быстродействие, статистику исполняемых инструкций, статус кэшей) и выбирает подходящую частоту [7]. Для многоядерных процессоров задача определения оптимальной частоты решается для каждого ядра независимо. При измерении энергопотребления в одноядерном режиме энергосбережение от 5 до 20 % сопровождалось потерей быстродействия в 10 %, а энергосбережение от 9 до 42 % — падением быстродействия на 30 %. В экспериментах с многоядерной работой этот подход позволил сэкономить от 2,3 до 8,8 % потреблённой энергии с падением быстродействия на 10 % и от 3,9 до 22 % с падением быстродействия на 30%. «Длинная краткосрочная память» рукурсивных нейронных сетей также может использоваться при создании регулятора DVFS [8]. Вследствие проблемы затухания градиента для этого подхода важно делать выводы только на коротких интервалах наблюдения. Такая архитектура сети уменьшает энергопотребление процессора до 19% по сравнению с существующими алгоритмами. Обучение с подкреплением для предсказания нагрузки на ЦП позволяет увеличить энергосбережение в среднем до 22% [9].

Другой класс алгоритмов основан на подстройке регулятора к специфической пользовательской нагрузке до начала фактического использования смартфона. Например, на основании особенностей пользовательского взаимодействия (к примеру, частоты прикосновений к экрану или прокрутки информации на нём) можно выбирать оптимальную частоту для работы вплоть до последнего обновления экрана в последовательности взаимодействий [10]. Если во время использования устройства детектируется заранее известное взаимодействие, то выставляется предварительно вычисленная частота. Энергосбережение при этом составляет от 10 до 36 % без потерь в комфорте использования.

Вычислять оптимальную частоту можно и с помощью эмпирической формулы или алгоритма, исходя из текущего состояния системы. Например, главным критерием смены частоты может быть текущая температура процессора [6]. Вторичным оптимизационным критерием может выступать либо быстродействие, либо энергопотребление. Получившийся алгоритм уменьшает энергопотребление в среднем на 12,7 %, если оптимизировать по энергопотреблению, и на 6,7 %, если оптимизировать по быстродействию. Во втором случае время исполнения возрастает в среднем на 6,3 %. Энергопотребление может быть также оптимизировано посредством DVFS совместно с контролем пропускной способности ЦП [11]. Этот подход определяет оптимальное число активных ядер ЦП, а также их частоту и контролирует выделенную пропускную способность для достижения наилучшего состояния процессора с точки зрения производительности и энергосбережения. Для более и менее равномерно распределённых во времени рабочих нагрузок авторы сообщают об энергосбережении в 14 и 23 % соответственно.

2.2. Стохастическая аппроксимация со случайными направлениями

Различные задачи управления могут быть решены нахождением экстремума эмпирически построенного функционала, моделирующего управляемую систему. В нашем случае нагрузка ЦП за определённый период времени может быть использована для определения оптимальной частоты из множества доступных частот в рамках установленных критериев качества.

Более формально, пусть $F_t(x, w)$ — функция от дискретного времени t, некоторого параметра x и рандомизированного вектора w. Определим «текущий» функционал среднего риска как

$$f_t(x) = E_w F_t(x, w)$$

и точку минимума $f_t(x)$ как

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

Тогда задача состоит в построении последовательности таких оценок $\{\hat{\theta}_n\}$, что $||\hat{\theta}_n - \theta_t|| \to \min$ по наблюдениям за случайными векторами $F_t(x_n, w_n)$, $n = 1, 2, \ldots$

Будем понимать под *пробным одновременным возмущением* последовательность наблюдаемых одинаково симметрично распределенных случайных векторов Δ_n с матрицами ковариаций

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

где $\delta_{nj} \in \{0, 1\}$ — символ Кронекера, $0 < \sigma_\Delta < \infty$. Бернуллиевские случайные вектора часто используются в качестве пробного одновременного возмущения, так как координаты векторов Δ_n не зависят друг от друга и принимают значения ± 1 с равной вероятностью.

Было показано, что при зашумленных наблюдениях возможно использовать следующий алгоритм:

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

для построения последовательности оценок точек минимума для функционала $F(x)$ без существенных потерь в скорости сходимости [12]. На каждой итерации алгоритм использует только одно зашумленное вычисление эмпирической функции:

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

$\{\alpha_n\}$ и $\{\beta_n\}$ — последовательности неотрицательных чисел, удовлетворяющих некоторым условиям, w_n^+ — стохастический вектор возмущений для наблюдений y_n^+, v_n^+ — произвольный внешний шум во время этих наблюдений. Эта рекуррентная процедура называется *стохастической аппроксимацией со случайными направлениями* (simultaneous perturbation stochastic approximation, SPSA), так как она включает в себя рандомизированное пробное возмущение, являющееся одновременным по всем координатам. Оно также используется для задания направления следующего изменения оценки и для выбора новой точки. Среди условий устойчивости оценок следует отдельно выделить условие слабой корреляции между пробным возмущением $\{\Delta_n\}$ и последовательностями неопределённостей $\{w_n\}$ и $\{v_n\}$ как наиболее важное. Хотя среднеквадратичная скорость сходимости этого алгоритма ниже, чем у других алгоритмов этого класса, с практической точки зрения он полезен, поскольку при оптимизации систем реального времени и в задачах адаптивного управления два и более наблюдений с шумами, независимыми от Δ_n, как правило, недоступны.

В общем виде алгоритм SPSA в задачах управления может быть сформулирован следующим образом:

1. Определить эмпирический функционал $F(x)$.
2. Сделать первоначальную оптимальную оценку $\hat{\theta}_0$.
3. Внести в текущую оптимальную оценку возмущение.
4. Получить новое зашумленное наблюдение F и обновить текущую оптимальную оценку $\hat{\theta}_n$.
5. Перейти к шагу 3.

3. ОПИСАНИЕ АЛГОРИТМА

Предлагаемая нами модель основывается на нескольких допущениях. Определим нагрузку на ЦП как процент времени, который ЦП или его ядро проводит, занимаясь вычислительной работой. Первое допущение состоит в том, что ЦП часто не загружен полностью, и нагрузка на ЦП редко достигает 100 %. В широком смысле рабочая частота ЦП определяет объём работы, который процессор может выполнить за фиксированное время, поэтому при уменьшении частоты, время расчёта одной и той же вычислительной задачи увеличится.

Во-вторых, большинство современных архитектур процессоров, особенно варианты архитектуры big.LITTLE, работают с фиксированным множеством рабочих частот $Freq$, поэтому наши оценки рабочей частоты округляются до ближайшего доступного значения.

В-третьих, мы предполагаем, что работа регулятора DVFS опирается на принципы работы подсистемы CPUFreq: после того, как регулятор выставляет частоту, система какое-то время работает на ней, после чего регулятор анализирует текущее состояние системы и выставляет новую частоту на следующий временной интервал.

Общая идея алгоритма состоит в установке на основе текущих данных по нагрузке ЦП рабочей частоты на следующий временной интервал таким образом, чтобы уровень нагрузки был как можно ближе к заранее выбранному значению. Это значение не должно быть близким к 100%, чтобы в случае недооценки система могла выделить дополнительное время на работу, но также должно быть удалено от 0 %, так как это связано с повышением рабочей частоты, а следовательно и негативно сказывается на энергопотреблении.

Эмпирический функционал модели определим как

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

где workload(f) — нагрузка на ЦП, определяемая по системным метрикам, λ — целевая частота, table(f) — номер частоты в списке доступных ЦП частот, отсортированном по возрастанию.

Для обновления рабочей частоты нужно вычислить

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

где $\mathscr{P}$ — проекция во $Freq$, $\hat{f}_{n-1}$ — текущая оценка частоты, β — параметр размера шага SPSA. Затем для новой оценки частоты необходимо вычислить

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

где $\mathscr{L}$ — проекция в отрезок $[min(Freq), max(Freq)]$, α, β — параметры размера шага SPSA.

Таким образом, алгоритм регулятора DVFS может быть описан следующим образом:

1. Выбрать первоначальное значение оценки $\hat{f}_0$.
2. Вычислить Δ_n.
3. Внести возмущение в текущую оптимальную оценку $f_n = \mathscr{P}(\hat{f}_{n-1} + \beta\Delta_n)$.
4. Выставить текущую частоту процессора в f_n.
5. Получить новое зашумленное наблюдение $y_n = F(\hat{f}_{n-1} + \beta\Delta_n) + \upsilon_n$
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.

Следует отметить, что построенная модель параметризована λ, α, β и γ, поэтому регулятор способен демонстрировать различное поведение для различных наборов параметров.

4. МЕТОДОЛОГИЯ ЭКСПЕРИМЕНТОВ

4.1. Выбор устройства

Для экспериментов был выбран смартфон Xiaomi Redmi Note 8 Pro. На нём установлена Android OS 10, а в его ЦП, Helio G90T, есть два кластера ядер — 2 ядра A76 и 6 ядер A55 [14]. Важно, что каждый кластер может независимо контролироваться регулятором DVFS.

На выбор этой модели повлияло наличие существующего набора инструментов для запуска модифицированных сборок Android OS. В качестве основы для сборки использовалось ядро `begonia` (https://github.com/AgentFabulous/begonia), в список регуляторов DVFS которого была добавлена реализация регулятора SPSA. Отметим, что для установки модифицированной сборки потребовалось получить корневой доступ к смартфону. Переключение на регулятор SPSA производилось штатными системными вызовами подсистемы `cpufreq` [15].

В качестве базовых регуляторов для сравнения мы выбрали регуляторы OnDemand и Interactive, так как они являются наиболее часто использующимися на практике из стандартных адаптивных регуляторов.

Исходный код нашего регулятора DVFS свободно доступен (https://github.com/jackbogdanov/DVFS-for-begonia).

4.2. Тестовые сценарии

Регулятор DVFS общего назначения должен справляться с различными видами нагрузки. Хотя по сути каждый регулятор является определённой точкой в балансе между энергопотреблением и производительностью, в основе его архитектуры лежит некоторая модель нагрузки на ЦП. Так как эта модель может как адекватно, так и неадекватно описывать реальные рабочие нагрузки, необходимо протестировать регулятор DVFS на различных примерах нагрузок. Был реализован следующий набор тестовых сценариев:

1. Запись видео посредством *Open Camera* (https://play.google.com/store/apps/details?id=net.sourceforge.opencamera&hl=en&gl=US).
2. Проигрывание видеофайла из внутреннего хранилища с помощью *VLC player for Android* (https://play.google.com/store/apps/details?id=org.videolan.vlc&hl=en&gl=US).
3. Быстрое написание текста в приложении *Notes* (https://play.google.com/store/apps/details?id=com.ogden.memo&hl=en&gl=US).
4. Игра во *Flappy Bird* (https://en.wikipedia.org/wiki/Flappy_Bird).
5. Игра в *Trial Xtreme 3* (https://www.youtube.com/watch?v=iGu1R090pYk).
6. Игра в *Hill Climb Racing* (https://play.google.com/store/apps/details?id=com.fingersoft.hillclimb).
7. Проигрывание музыки из *Spotify*.
8. Отображение и периодическая прокрутка PDF-файла в *Foxit PDF Editor* (https://play.google.com/store/apps/details?id=com.foxit.mobile.pdf.lite).
9. Просмотр видео на *YouTube* в браузере по умолчанию в качестве 480p.
10. Просмотр видео на *YouTube* в браузере по умолчанию в качестве 1080p.

Все тестовые сценарии были запущены 4 раза по 15 минут для усреднения влияния фоновых системных процессов. Они были реализованы в виде скриптов на Python, запускаемых с помощью утилиты `Monkeyrunner` (https://developer.android.com/studio/test/monkeyrunner). Для запуска тестового сценария смартфон должен быть подключен к компьютеру, и `Monkeyrunner` управляет выполнением сценария посредством команд `adb`.

4.3. Подготовка устройства

Для обеспечения методологической корректности экспериментов и консистентности результатов были предприняты следующие шаги согласно [16]:

- Были удалены все ненужные приложения. Приложения, которые по тем или иным причинам было невозможно удалить, были отключены в настройках Android OS.
- Неиспользуемые в конкретном тесте периферийные устройства были выключены (например, Wi-Fi, 3G, GPS).
- Перед началом нового теста был добавлен интервал ожидания в две минуты для снижения температуры и завершения фоновых процессов, которые могли начаться во время предыдущего теста.
- Каждый тестовый сценарий предварялся «разогревочным» прогоном.

Следует отметить, что в силу заинтересованности только в энергопотреблении ЦП не было необходимости начинать каждый тест с одного и того же уровня заряда батареи. Более того, поскольку смартфон контролировался компьютером через USB, аккумулятор заряжался при проведении тестов.

4.4. Оценка энергопотребления

В Android OS есть специальные файлы `time-in-state` в папке `/sys`, содержащие информацию о промежутках времени, которые каждое ядро процессора провело на той или иной рабочей частоте. Информация в этих файлах хранится в формате "`<частота><время>`". Количество таких пар равно количеству частот, поддерживаемых ядром процессора. Так как разные кластеры ядер ЦП работают на разных наборах частот, частоты в файлах `time-in-state` также отличаются у различных ядер в зависимости от принадлежности к тому или иному кластеру. Время измеряется как количество интервалов по 10 мс и считается от момента установки соответствующего регулятора или сброса данных по процессору. Перед каждым запуском теста производился сброс информации, а её сбор для анализа производился сразу после завершения тестового сценария.

Мы полагаем, что напряжение на ЦП в рамках наших экспериментов изменяется настолько незначительно, что его можно считать постоянным, поэтому для оценки энергопотребления достаточно умножить времена, проведённые на конкретных частотах, на соответствующие весовые коэффициенты в файле смартфона `power_profile.xml`. В этом файле, предоставляемом производителем смартфона, содержатся значения номинального мгновенного потребляемого тока для каждого устройства смартфона. Так как данные по ЦП хранятся в *мА* в разбивке по кластерам, а время считается в 10-миллисекундных интервалах, итоговое энергопотребление переводится в *мАч*.

5. ЭКСПЕРИМЕНТЫ

Суммарное энергопотребление в мАч представлено в таблице 1.

В сценарии записи видео SPSA экономит больше всего энергии. В целом этот сценарий умеренно энергозатратный, так как каждую секунду сохраняется фиксированное количество кадров, и OnDemand стабильно устанавливает частоты ближе к верхнему краю диапазоноа как для A55, так и для A76. Гистограмма частот SPSA более выражена и в нижнем, и в верхнем краю диапазона.

В сценарии набора текста SPSA оказывается между OnDemand и Interactive по энергоэффективности. Сценарий не очень вычислительно ёмкий, так как скорость ввода существенно ограничена взаимодействием с пользователем. SPSA лучше работает с ядрами A76, чем OnDemand, но переоценивает выделяемые ресурсы для A76. Interactive хорошо управляет A55, но для A76 выставляет частоты слишком высоко.

Хотя на первый взгляд *Trial Xtreme 3* и *Flappy bird* похожи с точки зрения игрового процесса, их вычислительная нагрузка на ЦП и как следствие энергопотребление различны. Для нагрузки *Trial Xtreme 3* регулятор SPSA определил, что ядра A76 можно держать на минимальной частоте, а A55 — в середине диапазона. Спектр частот, используемых OnDemand, уже, чем у SPSA, но он склонен держать A55 в середине диапазона частот. Interactive ведёт себя подобно SPSA, но его спектр частот значительно уже в обоих случаях, и суммарно их энергопотребление сопоставимо. В сценарии *Flappy bird* SPSA и Interactive держат ядра A55 близко к минимуму, а OnDemand — немного выше. Однако существенный выигрыш в энергопотреблении OnDemand происходит от того, что ядра A76 работают на минимальной частоте, в то время как SPSA иногда поднимает энергопотребление до максимума, а Interactive переоценивает требуемые вычислительные ресурсы.

Аналогичная ситуация и в сценарии с воспроизведением видео. Сценарий требует достаточно высоких вычислительных ресурсов от ЦП вследствие необходимости регулярно декодировать видеокадры, и в силу частоты декодирования и сравнительной плавности переключения частот SPSA переоценивает вычислительные затраты как для A55, так и для A76, в то время как OnDemand держит все ядра близко к минимуму. Хотя Interactive близок по энергоэффективности к OnDemand на низкопроизводительных ядрах, он существенно переоценивает вычислительную нагрузку для A76.

Тест с игрой в Hill Climb Racing оказался худшим для регулятора OnDemand — он выставляет оба кластера ядер на максимальную частоту. SPSA показывает несколько лучшие результаты, чем Interactive, благодаря более низким частотам у A55, хотя активно используемый диапазон частот для A76 у него шире.

В тесте Spotify регуляторы показали существенно отличное друг от друга поведение. Interactive завышает частоту A76, SPSA использует максимально широкий диапазон частот A55, а OnDemand держит и те, и другие близко к минимальным значениям.

Хотя просмотр файла в Foxit PDF Editor не кажется на первый взгляд энергозатратным тестом, рендеринг новой страницы — ресурсоёмкая задача. SPSA показывает наилучшие результаты благодаря консервативному использованию кластера A76.

Несмотря на значительную разницу в размере передаваемых данных, энергопотребление ЦП в двух тестах с YouTube практически идентично. Interactive чрезмерно завышает частоту ядер A76 и проигрывает двум другим регуляторам. OnDemand в свою очередь более консервативен в регулировке обоих кластеров ядер.

Хотя целью работы является энергосбережение, важно, чтобы достижение этой цели не снизило ощутимо производительность для конечного пользователя. Для этого мы сравнили SPSA со стандартными регуляторами с помощью утилиты оценки производительности AnTuTu (https://www.antutu.com/en/index.htm). Это часто используемая утилита в среде разработчиков для Android OS позволяет оценить быстродействие смартфона как интегральную метрику.

Утилита проводит несколько тестов, сгруппированных по четырём категориям: центральный процессор, видеокарта, память и удобство пользовательского интерфейса. В таблице 2 представлены результаты регуляторов SPSA, OnDemand и Interactive во всех четырёх категориях, и в колонке «Итог» отмечена их сумма как итоговая интегральная метрика качества. Результаты для регуляторов Powersave и Performance также показаны как минимально и максимально возможные значения.

Разница в производительности между регуляторами SPSA, OnDemand и Interactive незаметна для пользователя (3,06 % и 0,36 % разницы с OnDemand и Interactive соответственно) и сравнима с максимально возможной производительностью (разница в 6,38 %

Таблица 1. Энергопотребление регуляторов SPSA, OnDemand и Interactive

Тест	SPSA (в мАч)	OnDemand (в мАч)	Interactive (в мАч)
Камера	57,119	70,918	44,823
Ввод текста	77,187	75,536	84,306
Trial Xtreme 3	34,100	40,066	35,801
Flappy bird	27,690	24,643	43,710
VLC	31,723	25,736	47,660
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 2. Результаты оценки утилитой AnTuTu

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

с регулятором Performance). При анализе оставшихся трёх категорий видно, что регулятор DVFS влияет на производительность и остальных систем смартфона, вследствие чего наблюдается общесистемное падение производительности на более консервативных регуляторах.

6. ПРОБЛЕМЫ И ПЕРСПЕКТИВЫ РЕГУЛЯТОРА SPSA

Уже в текущей реализации регулятор SPSA работает на уровне стандартных алгоритмов Android OS. Созданный для него эмпирический функционал не использует априорную информацию о виде вычислительной нагрузки и подстраивается к текущему её уровню на лету, благодаря чему регулятор выгодно отличается от других подходов, предложенных в литературе. Хотя с помощью дальнейшего подбора параметров SPSA можно сделать переход между частотами более резким, как в Interactive, представляется более перспективным развитие нашего регулятора в следующих направлениях.

6.1. Регуляторы простоя и power_profile.xml

Файл textttpower_profile.xml упоминался в контексте оценки энергопотребления, но так как он содержит номинальное значение энергопотребления устройств смартфона и, в частности, кластеров ЦП на различных частотах, эту информацию можно использовать для лучшей калибровки функционала.

С другой стороны, помимо регуляторов частоты в Android OS существуют и регуляторы простоя (idle state governors). Если на процессоре нет вычислительной нагрузки, то

эти регуляторы могут переводить его в одно из энергетически выгодных состояний простоя. Представляя из себя два слабо связанных друг с другом модуля ОС, оба регулятора могут как улучшать общее энергопотребление системы без существенных потерь производительности, так и мешать друг другу, например, если процессор будет переведён в состояние простоя, когда SPSA ещё не снизил частоту процессора до минимальной. Таким образом, исследование взаимодействия двух регуляторов представляется нам важным.

6.2. Нелинейное энергопотребление процессора

Со смартфоном Samsung Galaxy A3 (2016), в котором установлен процессор Exynos 7578 с 4 ядрами Cortex-A53 и максимальной тактовой частотой 1,5 ГГц [17], был проведён следующий эксперимент. 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 Android OS, а первое обрабатывало только фоновые процессы. 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 Android OS 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 обобщены на рис. 1.

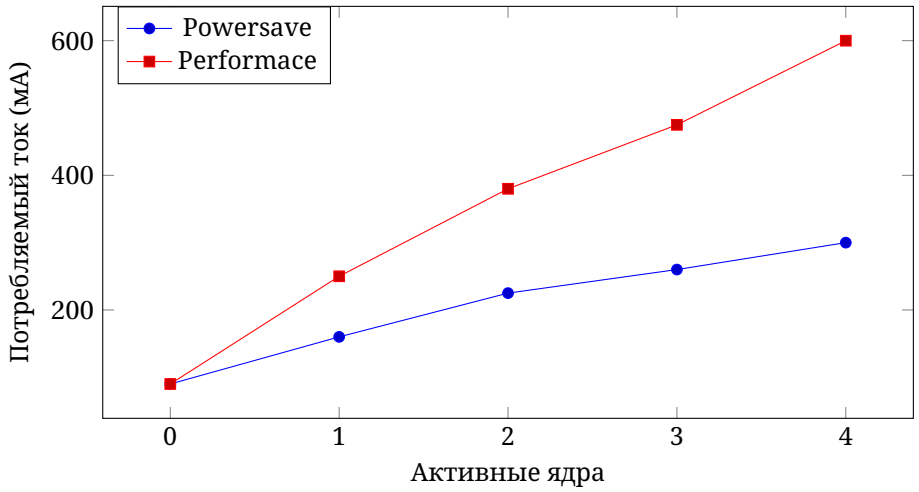

Рис. 1. Энергопотребление процессора в зависимости от числа активных ядер

Из графика видно, что энергопотребление нелинейно зависит от числа полностью нагруженных ядер как для минимальной, так и для максимальной рабочей частоты. Таким образом, регулятор, учитывающий эту нелинейность и способный при необходимости отключать и включать ядра в целях энергоэффективности, если текущая вычислительная нагрузка это позволяет, может показать лучшие результаты по сбережению энергии без потери производительности.

7. ЗАКЛЮЧЕНИЕ

В работе было рассмотрено использование алгоритма SPSA с одним наблюдением для построения регулятора частоты ЦП в смартфоне. Результаты показывают, что полученный регулятор, подходит для современных процессорных архитектур с несколькими кластерами ядер, справляется с различными видами вычислительной нагрузки и показывает результаты, сравнимые со стандартными алгоритмами, а иногда и превосходящие их. Были намечены пути развития разработанного регулятора как отдельного модуля, так и совместно с другими модулями подсистемы энергоэффективности Android OS.

Список литературы

1. *Sahin C., Cayci F., Manotas I., Clause J., Kiamilev F., Pollock L., Winbladh K.* Initial explorations on design pattern energy usage // Proc. of 1st International Workshop on Green and Sustainable Software, GREENS, June 2012.
2. *Spall C.* Multivariate stochastic approximation using a simultaneous perturbation gradient approximation // IEEE Trans. on Automatic Control. 1992. Vol. 37, N 3. P. 332–341.
3. *Granichin O. N., Amelina N.* Simultaneous perturbation stochastic approximation for tracking under unknown but bounded disturbances. // IEEE Trans. on Automatic Control. 2015. Vol. 60, N 6. P. 1653–1658.
4. *Granichin O. N., Gurevich L., Vakhitov A.* Discrete-time minimum tracking based on stochastic approximation algorithm with randomized differences. // Proc. of Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference, CDC/CCC, 2009.
5. *Brodowski D., Golde N.* Cpu frequency and voltage scaling code in the Linux™ kernel. Linux cpufreq. cpufreq governors. Доступно на: https://android.googlesource.com/kernel/msm/+/android-7.1.0r0.2/Documentation/cpu-freq/governors.txt (дата обращения: 01.08.2021).
6. *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, N 1.
7. *Chen Y. L., Chang M. F., Yu C. W., Chen X. Z., 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, N 9.
8. *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.
9. *Das A., Walker M. J., Hansson A., Al-Hashimi B. M., Nerrett G. V.* Hardware-software interaction for run-time power optimization: A case study of embedded Linux on multicore smartphones. // Proc. of the International Symposium on Low Power Electronics and Design, 2015.
10. *Li X., Wen W., Wang X.* Usage history-directed power management for smartphone. // Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2015.
11. *Broyde L., Nixon K., Chen X., Li H., Chen Y.* MobiCore: An adaptive hybrid approach for power-efficient CPU management on Android devices. // Proc. of 30th IEEE International System-on-Chip Conference, SOCC, 2017.
12. *Granichin O. N.* Linear regression and filtering under nonstandard assumptions (arbitrary noise). // IEEE Transactions on Automatic Control. 2004. Vol. 49, N 10. P. 1830–1837.
13. *Bogdanov E. A., Bozhnyuk A. S., Sartasov S. Yu., Granichin O. N.* On Application of Simultaneous Perturbation Stochastic Approximation for Dynamic Voltage-Frequency Scaling in Android OS. // Proc. of 7th International Conference on Event-Based Control, Communication and Signal Processing, EBCCSP, 2021.
14. Xiaomi Redmi Note 8 pro specifications. Доступно на: https://www.mi.com/global/redmi-note-8-pro/specs/ (дата обращения: 01.08.2021)
15. Android OS cpufreq specification. Доступно на: https://android.googlesource.com/kernel/common/+/

a7827a2a60218b25f222b54f77ed38f57aebe08b/Documentation/cpu-freq/index.txt (дата обращения: 01.08.2021)
16. *Myasnikov V., Sartasov S., Slesarev I., Gessen P.* Energy consumption measurement frameworks for Android OS: A systematic literature review. // Proc. of the Fifth Conference on Software Engineering and Information Management (ser. CEUR Workshop Proceedings), SEIM, 2020.
17. Samsung Galaxy A3 specifications. Доступно на: https://www.gsmarena.com/samsung_galaxy_a3_(2016)-7791.php (дата обращения: 01.08.2021)

Поступила в редакцию 16.05.2021, окончательный вариант — 17.06.2021.

Сартасов Станислав Юрьевич, старший преподаватель кафедры Системного программирования математико-механического факультета СПбГУ, ✉ stanislav.sartasov@spbu.ru

Богданов Евгений Алексеевич, выпускник образовательной программы бакалавриата «Программная инженерия» СПбГУ, evgenij.bogdanov.1999@gmail.com

Божнюк Александр Сергеевич, студент образовательной программы бакалавриата «Программная инженерия» СПбГУ 3 года обучения, bozhnyuks@mail.ru

Быков Давид Вадимович, студент образовательной программы бакалавриата «Программная инженерия» СПбГУ 3 года обучения, bykov.david@gmail.com

Граничин Олег Николаевич, доктор физико-математических наук, профессор кафедры Системного программирования математико-механического факультета СПбГУ., Oleg_granichin@mail.ru

Computer tools in education, 2021
№ 2: 26–40
http://cte.eltech.ru
doi:10.32603/2071-2340-2021-2-26-40

On Using Stochastic Approximation for Frequency Scaling in Android OS

Sartasov S. Yu.[1], Senior Lecturer, ✉ stanislav.sartasov@spbu.ru
Bogdanov E. A.[1], Graduate, evgenij.bogdanov.1999@gmail.com
Bozhniuk A. S.[1], Student, ✉ kosovtm@gmail.com
[1], Student, bykov.david@gmail.com
Granichin O. N.[1], Doctor of Physical and Mathematical Sciences, Professor, Oleg_granichin@mail.ru

[1]Saint Petersburg State University, 28, Universitetskiy pr, 198504, Saint Petersburg, Russia

Abstract

Simultaneous perturbation stochastic approximation demonstrate good results for control theory problems where system states dynamically changes. In this paper a dynamic voltage frequency scaling governor creation based on this approach for smartphone CPU under Android OS is described and ways to improve it are considered. The new governor shows results comparable to default Android OS governors

Keywords: *stochastic approximation, dynamic voltage frequency scaling, energy efficiency, Android OS.*

Citation: S. Yu. Sartasov, E. A. Bogdanov, A. S. Bozhniuk, D. V. Bykov, O. N. Granichin, "On Using Stochastic Approximation for Frequency Scaling in Android OS," *Computer tools in education*, no. 2, pp. 26–40, 2021; doi: 10.32603/2071-2340-2021-2-26-40

References

1. C. Sahin, F.Cayci, I. Manotas, J. Clause, F. Kiamilev, L. Pollock, and K. Winbladh, "Initial explorations on design pattern energy usage," in *Proc. of 1st International Workshop on Green and Sustainable Software, GREENS, June 2012*. 2012, pp. 55–61; doi: 10.1109/GREENS.2012.6224257
2. J. C. Spall, "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation," *IEEE Trans. on Automatic Control*, vol. 37, no. 3, pp. 332–341, 1992; doi: 10.1109/9.119632
3. O. Granichin and N. Amelina, "Simultaneous perturbation stochastic approximation for tracking under unknown but bounded disturbances," *IEEE Trans. on Automatic Control*, vol. 60, no. 6, pp. 1653–1658, 2015; doi: 10.1109/TAC.2014.2359711
4. O. Granichin, L. Gurevich, and A. Vakhitov, "Discrete-time minimum tracking based on stochastic approximation algorithm with randomized differences," in *Proc. of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, 2009, pp. 5763–5767; doi: 10.1109/CDC.2009.5400839
5. D. Brodowski and N. Golde, "Cpu frequency and voltage scaling code in the Linux™ kernel. Linux cpufreq. cpufreq governors," in *The Linux Kernel Archives*. [Online]. Available: https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt
6. K. Poornambigai, M. L. Raj, and P. Meena, "Reducing the energy consumption using DVFS performance optimizing scheme," *EPRA International Journal of Research and Development (IJRD)*, vol. 2, no. 1, pp. 79–88, 2017.
7. Y. L. Chen, M. F. Chang, C. W. Yu, X. Z. Chen, and W. Y. Liang, "Learning-directed dynamic voltage and frequency scaling scheme with adjustable performance for single-core and multi-core embedded and mobile systems," *Sensors*, vol. 18, no. 9, art. 3068, pp. 1–28, 2018; doi: 10.3390/s18093068
8. J. Lee, S. Nam, and S. Park, "Energy-Efficient Control of Mobile Processors Based on Long Short-Term Memory," *IEEE Access*, vol. 7, pp. 80552–80560, 2019; doi: 10.1109/ACCESS.2019.2923334
9. A. Das, M. J. Walker, A. Hansson, B. M. Al-Hashimi, and G. V. Merrett, "Hardware-software interaction for run-time power optimization: A case study of embedded Linux on multicore smartphones," in *Proc. of 2015 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, 2015, pp. 165–170; doi: 10.1109/ISLPED.2015.7273508
10. X. Li, W. Wen, and X. Wang, "Usage history-directed power management for smartphone," in *Algorithms and Architectures for Parallel Processing. ICA3PP 2015. Lecture Notes in Computer Science*, vol. 9528, Springer, Cham., pp. 288-302; doi: 10.1007/978-3-319-27119-4_20
11. L. Broyde, K. Nixon, X. Chen, H. Li, and Y. Chen, "MobiCore: An adaptive hybrid approach for power-efficient CPU management on Android devices," in *Proc. of 2017 30th IEEE International System-on-Chip Conference (SOCC)*, 2017, pp. 221–226; doi: 10.1109/SOCC.2017.8226044
12. O. Granichin, "Linear regression and filtering under nonstandard assumptions (arbitrary noise)," *IEEE Transactions on Automatic Control*, vol. 49, no. 10, pp. 1830–1837, 2004; doi: 10.1109/TAC.2004.835585
13. E. Bogdanov, A. Bozhnyuk, S. Sartasov, and O. Granichin, "On Application of Simultaneous Perturbation Stochastic Approximation for Dynamic Voltage-Frequency Scaling in Android OS," in *Proc. of 2021 7th International Conference on Event-Based Control, Communication, and Signal Processing (EBCCSP)*, 2021, pp. 1–7; doi: 10.1109/EBCCSP53293.2021.9502396
14. Xiaomi Corp. "Xiaomi Redmi Note 8 pro specifications," in *MI Official site*. [Online]. Available: https://www.mi.com/global/redmi-note-8-pro/specs/
15. ? "Android OS cpufreq specification," in *Google Git*. [Online]. Available: https://android.googlesource.com/kernel/common/+/a7827a2a60218b25f222b54f77ed38f57aebe08b/Documentation/cpu-freq/index.txt
16. V. Myasnikov, S. Sartasov, I. Slesarev, and P. Gessen, "Energy consumption measurement frameworks

for Android OS: A systematic literature review," in *Proc. of the Fifth Conference on Software Engineering and Information Management (ser. CEUR Workshop Proceedings), SEIM, St. Petersburg, May 16, 2020*, 2020, pp. 18–29.

17. "Samsung Galaxy A3 specifications," in *Gsmarena*. [Online]. Available: https://www.gsmarena.com/samsung_galaxy_a3_(2016)-7791.php

Received 16-05-2021, the final version — 17-06-2021.

Stanislav Sartasov, Senior Lecturer of the Department of System Program-Department of Mathematics and Mechanics, St. Petersburg State University, ✉ stanislav.sartasov@spbu.ru

Evgenii Bogdanov, graduate of the educational program of the bachelor's degree "Software Engineering" SPbSU, evgenij.bogdanov.1999@gmail.com

Alexander Bozhniuk, student of the educational bachelor's program "Software Engineering" SPbSU 3 years of study, bozhnyuks@mail.ru

David V. Bykov, student of the educational bachelor's program "Software Engineering" SPbSU 3 years of study, bykov.david@gmail.com

Oleg N. Granichin, Doctor of Physical and Mathematical Sciences, Professor of the Department System Programming, Faculty of Mathematics and Mechanics, St. Petersburg State University, Oleg_granichin@mail.ru