УДК 004.415.2
DOI 10.17513/snt.40962

CC BY 4.0

ЭМПИРИЧЕСКИЙ АНАЛИЗ ТЕХНИК ОПТИМИЗАЦИИ ПРОИЗВОДИТЕЛЬНОСТИ FLUTTER: СРАВНЕНИЕ МОБИЛЬНЫХ И ВЕБ-ПЛАТФОРМ

[1]Бисмус А. П. ORCID ID 0009-0009-5165-418X,
[1, 2]Готская И. Б. ORCID ID 0000-0003-3074-8936

[1]Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО», Санкт-Петербург, Российская Федерация, e-mail: alexeybismus@gmail.com;
[2]Федеральное государственное бюджетное образовательное учреждение высшего образования «Российский государственный педагогический университет имени А. И. Герцена», Санкт-Петербург, Российская Федерация

Производительность кроссплатформенных приложений на Flutter определяется совокупностью архитектурных и технических решений, однако их измеримое влияние на метрики рендеринга в реальных условиях исполнения остается слабо верифицированным. Цель работы – установить, оказывают ли пять выбранных техник оптимизации Flutter-приложений измеримое влияние на метрики рендеринга, насколько выявленные эффекты переносимы с мобильной платформы на веб и какие из них могут служить обоснованными ориентирами для выбора оптимизаций при кроссплатформенной разработке на Flutter. Применен двухконтурный дизайн: мобильный контур образуют два Android-устройства среднего сегмента в режиме profile, браузерный – Chrome Canary. Исследование охватывает сценарии: вынесение CPU-интенсивных вычислений в Dart-изолят, адаптивное декодирование изображений, ленивую загрузку списков, сокращение избыточных перестроений дерева виджетов и минимизацию шиммер-анимации. Для каждого сценария фиксировались медиана и 95-й процентиль времени кадра, а также доля кадров с нарушением целевого временного бюджета, обозначаемая как jank. Наиболее выраженный эффект получен при вынесении вычислительно тяжелых задач в изолят: на мобильных устройствах jank устранен полностью, в браузерном контуре его доля снизилась при неизменной медиане. Оптимизация дерева виджетов улучшает фазу сборки при умеренном влиянии на суммарное время кадра. Статичный заполнитель шиммер-анимации обеспечивает предсказуемое поведение на обоих устройствах; частичное упрощение анимации ухудшило хвостовые метрики на более слабом устройстве. Адаптивное декодирование изображений снизило объем кэша изображений на устройстве, где эта метрика доступна; ленивая загрузка сократила число изначально загружаемых элементов. Кадровые метрики в обоих сценариях остались в пределах погрешности. Эффекты оптимизаций согласуются по направлению, однако различаются по масштабу в зависимости от платформы и устройства. Полученные данные дают разработчикам Flutter-приложений обоснованные ориентиры для выбора оптимизаций на мобильных и веб-платформах.

Ключевые слова: производительность рендеринга, кроссплатформенная разработка, время кадра, плавность интерфейса, оптимизация мобильных приложений, эмпирический эксперимент, Flutter

EMPIRICAL ANALYSIS OF FLUTTER PERFORMANCE OPTIMIZATION TECHNIQUES: A COMPARISON OF MOBILE AND WEB PLATFORMS

[1]Bismus A. P. ORCID ID 0009-0009-5165-418X,
[1, 2]Gotskaya I. B. ORCID ID 0000-0003-3074-8936

[1]Federal State Autonomous Educational Institution of Higher Education "National Research University ITMO", Saint Petersburg, Russian Federation, e-mail: alexeybismus@gmail.com;
[2]Federal State Budgetary Educational Institution of Higher Education "Russian State Pedagogical University named after A. I. Herzen", Saint Petersburg, Russian Federation

The performance of cross-platform Flutter applications is determined by a combination of architectural and technical decisions; however, their measurable impact on rendering metrics under real execution conditions remains poorly verified. The aim of this study is to determine whether five selected Flutter optimization techniques produce a measurable effect on rendering metrics, to what extent these effects transfer from mobile to web, and which of them provide evidence-based guidance for selecting optimizations in cross-platform Flutter development. A two-arm design is applied: the mobile arm comprises two mid-range Android devices in profile mode, and the browser arm comprises Chrome Canary. The study covers the following scenarios: offloading CPU-intensive computations to a Dart isolate, adaptive image decoding, lazy list loading, reducing redundant widget-tree rebuilds, and minimizing shimmer animation. For each scenario, the median and 95th-percentile frame time were recorded, as well as the proportion of frames exceeding the target frame-time budget, referred to as jank. The most pronounced effect was observed when offloading computationally heavy tasks to an isolate: on mobile devices, jank was eliminated entirely; in the browser arm, the jank rate decreased while the median remained unchanged. Widget-tree optimization improved the build phase with a moderate effect on total frame time. A static shimmer placeholder achieved predictable behavior on both devices; partial animation simplification degraded tail metrics on the weaker device. Adaptive image decoding reduced the image-cache size on the device where this metric was available; lazy loading reduced the number of initially loaded items, with frame-time metrics unchanged within measurement error. Optimization effects are consistent in direction but differ in magnitude depending on the platform and device. The results provide Flutter developers with preliminary, evidence-based reference points for selecting optimizations on mobile and web platforms.

Keywords: rendering performance, cross-platform development, frame time, UI smoothness, mobile application optimization, empirical experiment, Flutter

Введение

По данным StatCounter, в 2025–2026 гг. через мобильные устройства генерировалось 50–55 % глобального веб-трафика. Кроссплатформенные фреймворки, прежде всего Flutter, получили широкое распространение: по данным Statista, с 2021 г. Flutter занимает лидирующие позиции среди решений данного класса. С усложнением приложений плавность интерфейса и стабильность времени кадра превращаются из технических характеристик в прямой фактор пользовательского восприятия: jank-эффект (срыв плавности, пропуск кадра) снижает удержание и воспринимаемое качество продукта.

Эмпирические исследования производительности мобильных приложений уже сформировали обширный корпус работ. M. Hort с соавт. систематизировали методы оптимизации по четырем измерениям (отзывчивость, запуск, память, энергопотребление), обобщив 156 публикаций [1]. R. Rua и J. Saraiva провели масштабный эмпирический анализ Android-приложений [2], а N. Kumari с коллегами классифицировали типовые причины деградации производительности по данным открытых репозиториев [3]. A. Biørn-Hansen с соавт., оценивая накладные расходы Flutter, React Native и других фреймворков, показали зависимость эффектов от сценария нагрузки [4]; W. Oliveira с коллегами описали неравномерное распределение издержек между CPU, памятью и сетью [5].

Среди Flutter-специфичных работ M. Zulistiyan с соавт. сравнивают библиотеки управления состоянием по нагрузке на CPU [6]; в работе [7] показано снижение нагрузки средствами Skia Shader Language; B. Dornauer и M. Felderer оценивают стратегии энергосбережения в мобильных и веб-приложениях [8]; М. Ф. Алиева с соавт. предлагают систему ключевых метрик производительности [9].

Перечисленные работы охватывают либо общий класс мобильных приложений, либо отдельные техники Flutter в изолированных условиях. Контролируемое же сравнение нескольких техник сразу на мобильной и веб-платформах до сих пор остается за рамками внимания как российских, так и зарубежных исследователей.

Цель исследования – оценить влияние пяти техник оптимизации Flutter-приложений на метрики рендеринга, определить, насколько выявленные эффекты переносятся между мобильной и веб-платформами, и на этой основе сформировать обоснованные ориентиры для выбора оптимизаций при кроссплатформенной разработке. Все измерения проведены в рамках единого контролируемого эксперимента с фиксированным окружением и унифицированными сценариями нагрузки.

Материалы и методы исследования

Основным методом служил контролируемый эксперимент в трактовке эмпирической программной инженерии: в каждом сценарии изменяли единственный фактор при фиксированных прочих условиях [10–12]. Для повторяемости измерений фиксировали программное окружение, использовали унифицированные сценарии нагрузки и автоматический сбор данных [13]. Код стенда, сырые данные и протокол измерений доступны в открытом репозитории: https://github.com/bismus-le-sha/flutter-performance-optimization-experiment.

Экспериментальная среда

Мобильный контур представлен смартфоном Samsung Galaxy A12 (SM-A125F). Основная серия измерений выполнена на нем 11–18 октября 2026 г. Второе устройство, Huawei P Smart (FIG-LX1), было доступно только в предварительной серии (март 2026 г., одна сессия на условие, бета-версия SDK). Его данные приводятся как вспомогательные и в статистический анализ не входят. Приложение собиралось в режиме profile, отдельно для каждого условия; запуск и сбор данных обеспечивали пакеты integration_test и flutter drive.

Веб-контур – браузер Chrome for Testing 155.0.8059.39 (зафиксированная сборка, не обновлявшаяся в ходе серии) на ноутбуке под Windows 11 со встроенным графическим адаптером Intel UHD (ANGLE, Direct3D 11) и дисплеем 144 Гц. Приложение компилировалось dart2js в режиме profile; журнал каждой сессии подтверждает загрузку рендерера CanvasKit. HTML-рендерер в используемой версии Flutter отсутствует, поэтому сравнение рендереров не проводилось. Каждая сессия запускалась в новом профиле браузера без расширений, в видимом окне 1600×1024 пикселей.

Все измерения основной серии выполнены на Flutter 3.41.9 (канал stable) и Dart 3.11.5; версия SDK и режим сборки записывались в каждую строку данных. Конфигурация среды приведена в табл. 1.

Сценарии оптимизации

Стенд включал пять изолированных сценариев, охватывающих основные классы нагрузки Flutter-приложений (табл. 2).

В каждом сценарии включали единственный метод оптимизации, оставляя прочие компоненты стенда неизменными; изоляцию вмешательств обеспечивали конфигурационные флаги и dependency injection [14]. Показатели сравнивали до и после оптимизации, а SCN-SHIMMER дополнительно тестировали в трех состояниях: baseline, optimized и static.

Таблица 1

Среда эксперимента

Параметр	Мобильный контур	Веб-контур
Устройство	Samsung Galaxy A12 (SM-A125F), Android 12, 3 ГБ ОЗУ; вспомогательно – Huawei P Smart (FIG-LX1), Android 9	Ноутбук, Windows 11, 16 ГБ ОЗУ, Intel UHD (ANGLE / Direct3D 11)
Дисплей	1600×720, 60 Гц	1920×1080, 144 Гц
Среда исполнения	Flutter engine (Android)	Chrome for Testing 155.0.8059.39, CanvasKit
Сборка и запуск	profile; integration_test, flutter drive	dart2js, profile; flutter drive
SDK	Flutter 3.41.9 (stable), Dart 3.11.5	То же
Повторы	10 запусков на условие	10 сессий на условие
Период	11–18.10.2026	11–18.10.2026

Примечание: составлена авторами на основе полученных данных в ходе исследования. FIG-LX1 – предварительная серия, март 2026 г., одна сессия на условие.

Таблица 2

Сценарии оптимизации

Сценарий	Нагрузка	Метод оптимизации
SCN-FIB	Вычислительный	Вынесение CPU-задач в Dart-изолят
SCN-IMG	Ресурсоемкий	Декодирование с учетом размера отображения
SCN-LAZY	Ленивые операции	Ленивая загрузка элементов списка
SCN-REBUILD	UI-перестроения	Условные обновления через setState
SCN-SHIMMER	Анимационный	Минимизация шиммер-эффекта; static placeholder

Примечание: составлена авторами на основе полученных данных в ходе исследования.

В веб-сборке функция compute() не создает отдельного изолята: вычисление выполняется в основном потоке после передачи управления циклу событий (исходный код Flutter SDK, foundation/_isolates_web.dart). Поэтому в веб-контуре SCN-FIB сравнивает синхронный и асинхронный вызов одной функции, а не вынесение вычислений в отдельный поток.

В мобильных сценариях SCN-IMG и SCN-LAZY изображения загружались с локального зеркала (по USB) вместо внешнего сервера: при загрузке из интернета часть запусков завершалась сетевыми ошибками. Зеркало отдает побайтные копии исходных файлов, поэтому вычислительная нагрузка не меняется, а сетевая вариативность исключается. При подготовке серии обнаружена ошибка служебного кода сбора метрик: он повторно загружал и декодировал в полном разрешении каждое отображаемое изображение. Ошибка исправлена до измерения SCN-IMG и SCN-LAZY; в SCN-REBUILD оба состояния отображают одни и те же изображения, и дополнительная нагрузка в них одинакова.

Конвейер измерений и метрики

На мобильной платформе для каждого кадра через интерфейс FrameTiming фиксировали время сборки (build), время растеризации (raster), полное время кадра t_frame и момент начала кадра (vsync). В веб-контуре для каждого кадра фиксировали метку requestAnimationFrame и вычисляли межкадровый интервал t_int.

Частоту выдачи кадров вычисляли как $F = N / T$, где N – число кадров, T – длительность окна наблюдения. В отличие от величины, обратной медиане времени кадра, F не превышает частоту обновления экрана и учитывает периоды, когда кадры не формировались. На мобильной платформе частоту перерисовки задает цикл тестового стенда (шаг 16 мс), поэтому F не превышает примерно 30 кадров/с и сравнивается только между состояниями одного сценария. Для SCN-FIB дополнительно приводится максимальный интервал между соседними кадрами – длительность заморозки интерфейса.

Пропуском кадра (англ. jank) считали кадр, не уложившийся в бюджет 16,67 мс (60 Гц): на мобильной платформе – t_frame > 16,67 мс, в веб-контуре – t_int > 16,67 мс. Для веб-контура дополнительно рассчитывали долю интервалов длиннее 1,5 периода кадра при 144 Гц (t_int > 10,42 мс). Под хвостовыми показателями понимаются верхние процентили распределения (p95).

Статистический анализ

Единицей анализа служил запуск – отдельный процесс приложения или отдельная сессия браузера, а не кадр: кадры одного запуска зависимы друг от друга. Каждое условие измерялось в 10 запусках; запуск включал 2 прогревочные и 10 измерительных итераций, прогревочные в анализ не входили. Порядок условий перемешивался в 10 рандомизированных блоках (каждое условие один раз в блоке). Перед каждым мобильным запуском процесс приложения принудительно завершался; запуск начинался, только если температура аккумулятора не превышала 31,0 °C.

Для каждого запуска вычисляли медиану и p95 времени кадра (межкадрового интервала), долю пропусков кадров и F. Медиану предпочли среднему, поскольку распределения временны́х метрик не подчиняются нормальному закону [2, 15]. Эффект оптимизации оценивали как разность медиан показателя между состояниями (по 10 запусков в каждом) с 95 % доверительным интервалом: перцентильный бутстрэп, 10 000 повторений, ресэмплинг запусков. Непараметрическим размером эффекта служила дельта Клиффа (δ); значимость проверяли U-критерием Манна – Уитни с поправкой Холма на множественные сравнения. Коэффициент Коэна d не применялся: он опирается на среднее и стандартное отклонение и плохо описывает асимметричные распределения времени кадра.

Выбросы не удалялись: медленные кадры и составляют предмет измерения. Для проверки чувствительности к данным применен фильтр по межквартильному размаху (1,5 IQR), использованный в предварительной серии. Он удаляет 4–15 % кадров и в большинстве условий снижает долю пропусков кадров почти до нуля, то есть скрывает исследуемый эффект; сравнение до и после фильтрации приведено в табл. 6. Распределения дополнительно проверяли визуально – по графикам ECDF и диаграммам размаха [16, 17].

Результаты исследования и их обсуждение

Основная серия включает 220 запусков, по 10 на каждое условие. В таблицах приведены медианы по запускам и разности между состояниями с 95 % доверительными интервалами.

Мобильные результаты

SCN-FIB. Синхронное вычисление замораживает интерфейс примерно на 12 с за итерацию; вынесение в изолят устраняет заморозку и снижает долю пропусков кадров с 20 до 1 % (табл. 3, 4; рисунок, а, б). Время отдельного кадра здесь плохо описывает деградацию: основная потеря плавности – заморозка между кадрами.

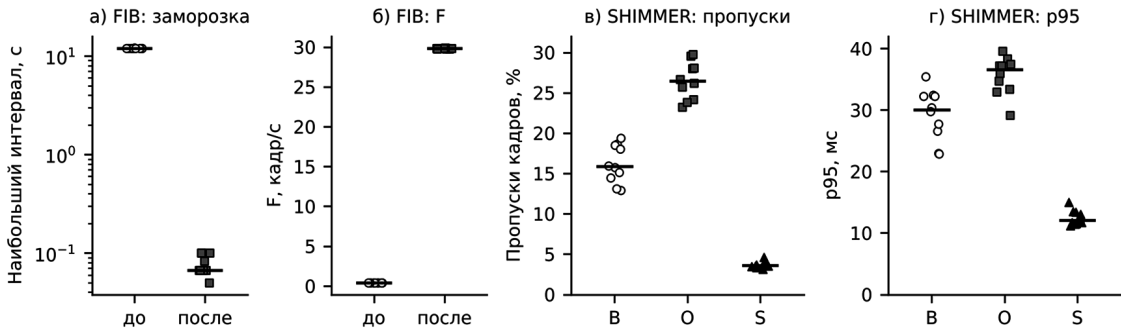
SCN-REBUILD. Оптимизация вдвое сократила фазу сборки, не затронув растеризацию (табл. 4); изменение полного времени кадра незначимо, доля пропусков кадров снизилась на 2,2 п. п.

SCN-SHIMMER. Состояние optimized снизило медиану времени кадра, но увеличило p95 и долю пропусков кадров; статичный заполнитель улучшил все показатели (табл. 3; рисунок, в, г).

SCN-LAZY и *SCN-IMG.* Обе техники снизили память процесса на 110–130 МБ (табл. 4). Кадровые метрики SCN-LAZY не изменились; SCN-IMG снизил p95 и долю пропусков кадров при росте медианы на 0,25 мс.

Веб-результаты

Медиана межкадрового интервала во всех условиях составила 6,9–7,0 мс (144 Гц). Значимые различия получены в двух сценариях: в SCN-FIB вызов через compute() ухудшил отрисовку, в SCN-LAZY частота кадров выросла на треть, а наибольшая пауза отрисовки сократилась с 590 до 48 мс (табл. 5).



SCN-FIB (а – наибольший интервал между кадрами, логарифмическая шкала; б – частота выдачи кадров F) и SCN-SHIMMER (в – доля пропусков кадров; г – p95 времени кадра; B/O/S – baseline/optimized/static) на SM-A125F. Точки – запуски (n = 10), линия – медиана
Примечание: составлен авторами по результатам данного исследования

Таблица 3

Кадровые метрики мобильного контура (SM-A125F)

Сценарий	Состояние	N	t̃, мс	p95, мс	Пропуски, %	Δ p95, мс [95 % ДИ]	Δ пропусков, п. п. [95 % ДИ]
SCN-FIB	before	500	13,6	20,9	20,0		
	after	37 359	7,5	9,7	1,0	−11,2 [−11,5; −11,0]	−19,0 [−19,9; −16,9]
SCN-REBUILD	before	8413	10,9	18,7	8,7		
	after	8411	10,7	17,5	6,6	−1,2 [−2,0; −0,6]*	−2,2 [−3,6; −1,8]
SCN-SHIMMER	baseline	5947	14,0	30,1	15,9		
	optimized	5936	11,6	36,6	26,4	+6,5 [2,4; 10,6]	+10,6 [7,2; 13,3]
	static	5688	9,5	12,1	3,6	−18,0 [−20,6; −13,3]	−12,3 [−14,9; −10,6]
SCN-LAZY	before	5723	8,8	12,4	4,3		
	after	5866	8,7	11,5	4,0	−1,0 [−2,6; 0,2]*	−0,3 [−0,7; 0,2]*
SCN-IMG	before	17 016	10,9	22,2	11,7		
	after	17 035	11,2	20,3	9,8	−1,9 [−2,4; −1,3]	−1,8 [−2,1; −1,1]

Примечание: составлена авторами на основе полученных данных в ходе исследования. Медианы по 10 запускам; N – число кадров; Δ – относительно первого состояния (для SCN-SHIMMER – baseline); ДИ – бутстрэп по запускам; * – незначимо после поправки Холма.

Таблица 4

Фазы кадра, заморозка и память (SM-A125F)

Сценарий	Показатель	До	После	Разность [95 % ДИ]
SCN-FIB	Наибольший интервал между кадрами, мс	12 000	67	-11 934 [-11 958; -11 901]
SCN-FIB	F, кадр/с	0,41	29,8	+29,4 [29,4; 29,4]
SCN-REBUILD	Медиана сборки / растеризации, мс	1,99/7,42	1,05/7,62	-0,94 [-1,02; -0,70] / +0,20 [-0,11; 0,36]
SCN-LAZY	Память процесса (PSS), МБ	289	175	-114 [-117; -103]
SCN-IMG	Кэш изображений / PSS, МБ	85,9/316	10,5/190	-75,3 / -126 [-157; -114]

Примечание: составлена авторами на основе полученных данных в ходе исследования. Медианы по 10 запускам; PSS – dumpsys meminfo в конце запуска; кэш изображений одинаков во всех запусках состояния.

Таблица 5

Результаты веб-контура (Chrome for Testing 155, 144 Гц)

Сценарий	Состояние	N	F, кадр/с	p95 t_int, мс	t_int > 16,67 мс, %	t_int > 10,42 мс, %	Наибольший интервал, мс
SCN-FIB	before	3204	7,2	34,8	6,4	9,3	4225
	after	2354	5,4	48,5	8,9	9,4	4197
SCN-REBUILD	before	36 927	143,2	7,2	0,07	0,49	35
	after	37 039	143,2	7,2	0,05	0,40	42
SCN-SHIMMER	baseline	28 534	142,0	7,2	0,42	0,91	35
	optimized	28 443	141,5	7,2	0,49	0,92	42
	static	28 509	141,8	7,2	0,49	0,91	42
SCN-LAZY	before	21 283	97,2	12,4	2,5	10,5	590
	after	22 457	129,5	9,9	1,4	5,8	48
SCN-IMG	before	65 702	143,0	7,2	0,07	0,46	77
	after	65 616	143,0	7,2	0,08	0,46	73

Примечание: составлена авторами на основе полученных данных в ходе исследования. Медианы по 10 сессиям; t_int – межкадровый интервал; 10,42 мс – 1,5 периода кадра при 144 Гц. Значимо после поправки Холма: SCN-FIB – F -1,8 [-2,0; -1,7] кадр/с, доля > 16,67 мс +2,4 [2,3; 2,5] п. п.; SCN-LAZY – F +32 [14; 48] кадр/с, доля > 16,67 мс -1,2 [-1,5; -0,7] п. п.

Таблица 6

SM-A125F: без фильтрации и после фильтрации 1,5 IQR

Сценарий	Состояние	Удалено кадров, %	p95, мс: без / с фильтром	Пропуски, %: без / с фильтром
SCN-FIB	before	6,2	21,0/20,1	19,6/14,3
	after	4,6	9,7/8,7	1,1/0,0
SCN-REBUILD	before	6,4	18,7/15,7	8,8/2,5
	after	6,5	17,6/14,8	6,4/0,0
SCN-SHIMMER	baseline	15,0	29,8/16,2	16,2/3,7
	optimized	10,3	35,4/22,3	26,5/18,1
	static	4,1	12,2/11,0	3,7/0,0
SCN-LAZY	before	4,8	12,6/10,3	4,3/0,0
	after	4,5	11,4/10,2	4,1/0,0
SCN-IMG	before	11,2	22,1/14,6	11,5/0,3
	after	9,9	20,4/15,0	9,8/0,0

Примечание: составлена авторами на основе полученных данных в ходе исследования. По объединенным кадрам 10 запусков условия.

Фильтр 1,5 IQR, применявшийся в предварительной серии, удаляет 4–15 % кадров и в 7 из 11 условий снижает долю пропусков кадров почти до нуля (табл. 6), поэтому выводы построены на нефильтрованных данных. Предварительная серия на FIG-LX1 (одна сессия; табл. S1 в репозитории) согласуется с основной по направлению эффекта SCN-FIB; в SCN-SHIMMER состояние optimized на этом устройстве, напротив, снизило долю пропусков кадров. Это расхождение требует проверки на повторных запусках.

Вычисления. Синхронное вычисление в основном изоляте останавливает кадровый конвейер, поэтому на мобильной платформе тяжелые вычисления следует выносить в изолят. В веб-сборке compute() не создает отдельного потока, и эффект отсутствует; для браузера требуются Web Workers или разбиение работы на части, которые в работе не исследовались.

Перестроения. Сборка занимает около 2 мс из 11 мс кадра, а растеризация, не затрагиваемая оптимизацией, – около 7,5 мс. Поэтому сокращение перестроений заметно влияет на кадр лишь там, где сборка составляет его значимую долю. Перенос нагрузки на растеризацию, отмеченный в предварительной серии, не подтвердился.

Анимации. Частичное упрощение анимации удешевило типичный кадр, но сделало редкие кадры дороже. Причина не устанавливалась; гипотеза – перераспределение работы между фазами кадра при объединении слоев ShaderMask. Отсюда практический вывод: оптимизацию следует оценивать по p95 и доле пропусков кадров, а не только по медиане; статичный заполнитель оказался наиболее предсказуемым вариантом.

Ресурсные оптимизации. Ленивая загрузка и декодирование с учетом размера отображения уменьшают объем одновременно декодированных изображений, поэтому их выбор оправдан прежде всего экономией памяти. Рост медианы в SCN-IMG предположительно связан с масштабированием при декодировании; это не проверялось.

Переносимость. Переносимость эффектов между платформами ограничена: в браузере проявился только эффект ленивой загрузки. Отсутствие различий в SCN-SHIMMER и SCN-IMG может объясняться тем, что при 144 Гц и аппаратном ускорении все состояния укладываются в бюджет кадра; это гипотеза. Выводы относятся только к исследованным устройствам и условиям.

Ограничения исследования. Область применения выводов ограничена несколькими факторами. Мобильный контур – это два устройства с 3 ГБ ОЗУ и частотой обновления 60 Гц; устройства нижнего (менее 3 ГБ) и верхнего (90+ Гц) сегментов он не охватывает, а эффекты на них могут заметно отличаться [3]. Измерения шли только в режиме profile: он близок к release-сборке по отсутствию отладочной инфраструктуры, но не тождествен ей. Каждый сценарий – это одна сессия с прогревом; разброс между независимыми запусками не оценивался [15]. Наконец, в SCN-IMG метрика кэша недоступна на FIG-LX1, что ограничивает сравнение устройств в этом сценарии.

Использованные в веб-контуре конфигурации html и canvaskit на данной версии Flutter SDK не гарантированно используют разные рендер-движки: флаг --web-renderer в этой версии не поддерживается, и сборочный скрипт не передает его. Поэтому обе конфигурации рассматриваются как повторные измерения одного условия, а не как независимое сравнение рендереров.

Точная версия Flutter SDK инструментально не зафиксирована: в коде харнесса присутствует строка версии, не привязанная к фактическому значению SDK на момент выполнения. Достоверно известны только нижняя граница из pubspec.lock и канал сборки.

Сбор данных занял несколько месяцев; данные по двум мобильным устройствам получены в разное время, что могло внести дополнительную вариативность, не связанную с самими техниками оптимизации.

Из веб-контура исключен дополнительный сценарий разбора JSON-данных SCN-JSON, не входящий в основной набор из пяти сценариев (не удалось корректно воспроизвести условия), а также SCN-IMG и SCN-REBUILD (специфичная для них метрика не сохранилась в конвейере агрегации данных).

Формулировка «вынесение в изолят» для веб-контура отражает используемый программный интерфейс (compute()), но не гарантирует независимое параллельное исполнение: поведение зависит от целевой компиляции и версии Flutter и отдельно не проверялось.

Внешнюю валидность ограничивает однородность стенда: изолированные сценарии хорошо контролируют переменные, но не воспроизводят реальное приложение, где несколько техник работают одновременно и влияют друг на друга. Поэтому приоритетные направления дальнейшей работы – расширить парк устройств, оценить доверительные интервалы для изменения медианы, p95 и доли jank-кадров по нескольким независимым запускам, применить непараметрические критерии (например, U-критерий Манна – Уитни) и изучить взаимодействие техник между собой [11, 15, 3].

Заключение

Проведена эмпирическая оценка пяти техник оптимизации Flutter-приложений в контролируемых условиях, в двухконтурном дизайне: мобильном (Samsung Galaxy A12, Huawei P Smart, режим profile) и браузерном (Chrome Canary). Основой анализа стали медиана, 95-й процентиль и доля кадров с нарушением временно́го бюджета.

Работа дает систематическую оценку пяти техник в единой среде измерений, охватывающей мобильную и браузерную платформы. Показано, что межплатформенный перенос правомерен на уровне направления эффекта, но не его величины: браузерное окружение существенно меняет масштаб изменений. Совокупный анализ центральных и хвостовых статистик позволил увидеть случаи, когда улучшение медианы сопровождается деградацией хвостовых характеристик. По устойчивости и переносимости эффектов выделены техники, показавшие наиболее согласованный результат между платформами в проведенных испытаниях; это может служить предварительным ориентиром при выборе оптимизаций при кроссплатформенной разработке на Flutter.

Полученные результаты носят предварительный характер и относятся только к исследованным устройствам, версии SDK и условиям измерений. Выводы ограничены двумя устройствами среднего сегмента, одной сессией на сценарий и профильным режимом сборки. Перенос результатов на другие аппаратные конфигурации, термические условия и реальные многокомпонентные приложения требуют повторной проверки.

Список литературы

1. Hort M., Kechagia M., Sarro F., Harman M. A survey of performance optimization for mobile applications // IEEE Transactions on Software Engineering. 2022. Vol. 48. Is. 8. P. 2879–2904. DOI: 10.1109/TSE.2021.3071193.

2. Rua R., Saraiva J. A large-scale empirical study on mobile performance: energy, run-time and memory // Empirical Software Engineering. 2024. Vol. 29. Art. 31. DOI: 10.1007/s10664-023-10391-y.

3. Kumari N., Das T., Mikkonen T. An Empirical Study on Understanding Performance Issues and Causes in Android Apps // 2024 IEEE International Conference on Source Code Analysis and Manipulation (SCAM). Flagstaff, AZ, USA, 2024. P. 242–247. DOI: 10.1109/SCAM63643.2024.00032.

4. Biørn-Hansen A., Rieger C., Grønli T.-M., Majchrzak T. A., Ghinea G. An empirical investigation of performance overhead in cross-platform mobile development frameworks // Empirical Software Engineering. 2020. Vol. 25. Is. 4. P. 2997–3040. DOI: 10.1007/s10664-020-09827-6.

5. Oliveira W., Moraes B., Castor F., Fernandes J. P. Analyzing the Resource Usage Overhead of Mobile App Development Frameworks // Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering (EASE 2023). ACM, 2023. DOI: 10.1145/3593434.3593487.

6. Zulistiyan M., Adrian M., Wibowo Y. F. A. Performance Analysis of BLoC and GetX State Management Library on Flutter // Journal of Information System Research (JOSH). 2024. Vol. 5. Is. 2. P. 583–591. DOI: 10.47065/josh.v5i2.4698.

7. Нуждин Д. Г. Оптимизация производительности мобильных приложений: стратегии и лучшие практики // Вестник науки. 2023. Т. 3. № 11 (68). С. 850–862. URL: https://www.вестник-науки.рф/article/10987 (дата обращения: 20.05.2026). ISSN 2712-8849.

8. Dornauer B., Felderer M. Energy-saving strategies for mobile web apps and their measurement: Results from

a decade of research // 2023 IEEE/ACM 10th International Conference on Mobile Software Engineering and Systems (MOBILESoft). IEEE, 2023. P. 75–86. DOI: 10.1109/MOBILSoft59058.2023.00017.

9. Алиева М. Ф., Давтян В. К., Коцур С. Н. Оценка производительности мобильных приложений // Вестник Адыгейского государственного университета. Серия 4: Естественно-математические и технические науки. 2023. № 2 (321). URL: https://cyberleninka.ru/article/n/otsenka-proizvoditelnosti-mobilnyh-prilozheniy (дата обращения: 21.05.2026).

10. Easterbrook S., Singer J., Storey M.-A., Damian D. Selecting Empirical Methods for Software Engineering Research // Guide to Advanced Empirical Software Engineering. 2008. P. 285–311. DOI: 10.1007/978-1-84800-044-5_11.

11. Wohlin C., Runeson P., Höst M., Ohlsson M. C., Regnell B., Wesslén A. Experimentation in Software Engineering. Berlin, Heidelberg: Springer, 2012. DOI: 10.1007/978-3-642-29044-2. ISBN 978-3-642-29043-5.

12. Montgomery D. C. Design and Analysis of Experiments. 9th ed. Hoboken: Wiley, 2017. 630 p. ISBN 978-1-119-29936-3.

13. Fehr J., Heiland J., Himpe C., Saak J. Best practices for replicability, reproducibility and reusability of computer-based experiments exemplified by model reduction software // AIMS Mathematics. 2016. Vol. 1. Is. 3. P. 261–281. DOI: 10.3934/Math.2016.3.261.

14. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Boston: Prentice Hall, 2017. ISBN 978-0-13-449416-6.

15. Georges A., Buytaert D., Eeckhout L. Statistically rigorous Java performance evaluation // ACM SIGPLAN Notices. 2007. Vol. 42. Is. 10. P. 57–76. DOI: 10.1145/1297105.1297033.

16. Shatz I. Assumption-checking rather than (just) testing: The importance of visualization and effect size in statistical diagnostics // Behavior Research Methods. 2024. Vol. 56. P. 826–845. DOI: 10.3758/s13428-023-02072-x.

17. Kitchenham B. A., Pfleeger S. L., Pickard L. M., Jones P. W., Hoaglin D. C., El Emam K., Rosenberg J. Preliminary guidelines for empirical research in software engineering // IEEE Transactions on Software Engineering. 2002. Vol. 28. Is. 8. P. 721–734. DOI: 10.1109/TSE.2002.1027796.

Конфликт интересов: Авторы заявляют об отсутствии конфликта интересов.

Conflict of interest: The authors declare that there is no conflict of interest.

Финансирование: Авторы заявляют об отсутствии внешнего финансирования.

Financing: The research was performed without external funding.