Научная статья
УДК 004.4
ББК 32.972
А 50
DOI: 10.53598/2410-3225-2023-2-321-74-83

Оценка производительности мобильных приложений

(*Рецензирована*)

Маргарита Федоровна Алиева[1], Вартуи Карекиновна Давтян[2], Сергей Николаевич Коцур[3]

[1–3] *Адыгейский государственный университет*, *Майкоп*, *Россия*
[1] *aliyevamargarita@mail.ru*
[2] *vartuidavtan@gmail.com*
[3] *sergeykotzur@gmail.com*

Аннотация. *Статья посвящена изучению разгрузки вычислений в мобильных приложениях с целью оценки их производительности. Основное внимание обращено на анализ воздействия архитектуры, на использование ресурсов устройства и время выполнения вычислений в мобильных приложениях. Представлены результаты экспериментальных исследований, которые показывают, что использование разгрузки вычислений может значительно ускорить выполнение сложных задач в мобильных приложениях, тем самым повышая их производительность. Приводится краткий анализ того, как использование ресурсов и время выполнения изменяется в зависимости от сложности выполняемой задачи, что может быть полезно для разработчиков мобильных приложений при выборе оптимальной архитектуры приложения.*

Ключевые слова*: разгрузка вычислений, производительность мобильных приложений, оптимизация ресурсов, время выполнения*

Original Research Paper

Evaluation of mobile application performance

Margarita F. Alieva[1], Vartui K. Davtyan[2], Sergey N. Kotsur[3]

[1–3] *Adyghe State University*, *Maikop*, *Russia*
[1] *aliyevamargarita@mail.ru*
[2] *vartuidavtan@gmail.com*
[3] *sergeykotzur@gmail.com*

Abstract. *The aim of the article is to study computational offloading in mobile applications in order to evaluate their performance. The main attention is paid to the analysis of the influence of architecture, the use of device resources and computing time in mobile applications. Experimental results, which show that usage of computation offloading significantly speeds up the execution of complex tasks in mobile applications, thus increasing their performance, are presented. The article provides a brief analysis of how resource usage and execution time vary with the complexity of the task, which can be useful for developers of mobile applications to choose the best application architecture.*

Keywords: *offloading calculations, mobile applications performance, resource optimization, runtime*

1. Введение

В последние годы мобильные приложения нашли применение в различных областях, таких как развлечения, здоровье, бизнес, социальные сети, путешествия и новости. Популярность мобильных приложений обусловлена мобильностью, которую обеспечивают эти устройства, предлагающие услуги независимо от местонахождения пользователя. Однако мобильность связана с рядом проблем, таких как ограниченные ре-

сурсы, ограниченная энергия и плохое соединение [1].

Для преодоления этих ограничений и расширения возможностей мобильных устройств одним из возможных решений является разгрузка вычислений. Под разгрузкой вычислений понимается передача вычислительных задач за пределы конкретного устройства. В последние годы технология облачных вычислений приобрела популярность как новый способ удаленного предоставления вычислительных ресурсов. Концепция облачных вычислений подразумевает передачу приложений в качестве услуги через Интернет и перемещение вычислений и данных к удаленным провайдерам. Облачные вычисления используются для решения проблем производительности при мобильной обработке данных путем использования удаленных провайдеров вместо самого мобильного устройства для выполнения мобильных вычислений.

Такая архитектура, в которой хранение и обработка данных отделены от мобильного устройства, называется мобильными облачными вычислениями [1]. Используя вычислительную мощность и объем памяти мобильных облачных вычислений, можно выполнять сложные операции на устройстве с низкими техническими возможностями. Однако мобильные облачные вычисления вносят высокую задержку, поскольку данные отправляются удаленным провайдерам, находящимся, с точки зрения топологии сети, далеко от мобильного устройства.

Появились мобильные краевые вычисления, предлагающие возможности облачных вычислений в непосредственной близости от мобильных устройств. Пограничные вычисления расширяют возможности мобильных облачных вычислений за счет развертывания хранилищ и вычислительных мощностей в сети радиодоступа.

Цель работы – изучить применимость разгрузки вычислений в мобильных приложениях. Основное внимание уделяется оценке производительности мобильных приложений, использующих разгрузку вычислений, и определению того, как такая архитектура влияет на использование ресурсов устройства и время выполнения вычислений. Кроме того, анализируется, как использование ресурсов и время выполнения изменяется в зависимости от сложности выполняемой задачи.

2. Метрики производительности мобильных приложений

Мобильные телефоны или так называемые смартфоны сегодня используются для просмотра веб-страниц, отправки электронной почты, просмотра видео и т.д. Эти функциональные возможности привели к тому, что мобильный телефон стал похож на персональный компьютер. Однако мобильные устройства имеют аппаратные ограничения. Портативность, малые размеры и вес ограничивают размер батареи и, следовательно, мощность аккумулятора, вычислительную мощность и объем памяти. Тепловые факторы ограничивают количество энергии в мобильных устройствах без активного охлаждения примерно тремя ваттами, при превышении этого предела устройство будет нагреваться. Опишем метрики производительности мобильного приложения [2].

1. *Задержка*

Когда речь идет о пользовательском опыте на мобильных устройствах, более быстрое выполнение всегда лучше. Согласно исследованию [3], действия, требующие менее 100 миллисекунд, воспринимаются пользователями как мгновенные, а действия, требующие времени от 1 секунды и более, считаются задержкой в выполнении приложения. Объем данных, передаваемых с мобильного устройства на сервер, производительность и перегрузка сервера могут увеличить задержку в мобильных устройствах.

2. *Использование памяти*

Количество установленных на устройстве приложений, а также объем цифрового контента ограничены объемом памяти устройства. Ограничения также распространяются на оперативную память, в которой хранятся данные в момент их использования.

Мобильные приложения должны быть оптимизированы для использования минимального объема памяти. Если приложение использует много памяти, а аппаратное обеспечение устройства не может его поддерживать, это может привести к его медленному выполнению [4].

Управление памятью в мобильных устройствах сильно отличается от управления памятью в персональных компьютерах. В случае многозадачности в персональных компьютерах, когда несколько программ работают одновременно и потребляют много памяти, пользователь может отключить их. В мобильных устройствах управление памятью происходит автоматически и, если оперативной памяти недостаточно, некоторые из открытых приложений автоматически вытягиваются из оперативной памяти.

3. *Использование процессора*

Центральный процессор (ЦП) или процессор обрабатывает инструкции программных приложений. Мобильные телефоны перешли от одноядерных процессоров к двухъядерным и четырехъядерным. Многоядерные чипы обеспечивают большую мощность для выполнения задач, поскольку нагрузка может быть распределена между процессорами. Хотя большее количество ядер приводит к увеличению скорости обработки данных, существует множество факторов, определяющих скорость работы процессора и общую скорость работы устройства. Объем оперативной памяти и оптимизация программного обеспечения также могут влиять на скорость работы устройства. Программное обеспечение для мобильных устройств должно быть разработано с учетом поддержки многоядерных процессоров, чтобы полностью использовать вычислительную мощность.

4. *Срок службы батареи*

Мобильные устройства работают от аккумулятора, поэтому управление энергопотреблением в этих устройствах очень важно. При включении приложения активируются различные компоненты устройства. Система рассчитывает количество энергии, необходимое для каждого компонента, и распределяет ее между запущенными процессами. Дисплей, передатчики (cellular, Wi-Fi, Bluetooth, GPS) и центральный процессор (для крупномасштабной обработки данных) являются самыми большими потребителями энергии [5].

Потребление энергии во многом зависит от того, как работает мобильное приложение и как оно использует компоненты и ресурсы системы. Использование передатчиков только тогда, когда они действительно необходимы, может существенно повлиять на энергопотребление [4]. При использовании нескольких циклов CPU или GPU потребляется больше энергии из батареи и выделяется больше тепла. Если выполняются графические или сложные вычисления, нагрузка на GPU и CPU велика.

Потребности в энергии значительно возросли с появлением мобильных приложений, требующих высокой вычислительной мощности. Однако увеличение емкости аккумуляторов не может следовать этой тенденции из-за медленного развития аккумуляторных технологий, ограниченного тем, что мобильные устройства должны удовлетворять потребности пользователей в небольших размерах устройства.

5. *Сеть*

Из-за мобильности мобильного устройства нельзя ожидать, что мобильное приложение будет иметь постоянное и стабильное соединение с Интернетом. Мобильные устройства часто переключаются между различными типами соединений, такими как 3G, 4G и Wi-Fi, с переменной скоростью [6]. Подключение через переполненные и ненадежные мобильные сети может привести к проблемам с мобильным приложением: медленная загрузка изображения, замораживание экрана, блокировка приложения или прекращение начатого действия. Гетерогенная сеть является дополнительной проблемой для мобильных пользователей из-за различных протоколов, топологии и различной архитектуры.

3. Разработка и тестирование мобильных приложений

1. *Определение задач*

Для сравнения производительности между локальным выполнением и выполнением в облаке было выбрано несколько задач различной сложности. Для определения задач разной сложности использована нотация BigO в качестве меры.

Первая задача – получить доступ к элементу массива.

Это простейшая задача с постоянной сложностью O(1). Идея заключается в том, чтобы создать массив фиксированного размера и получить доступ к первому элементу массива.

Вторая задача заключается в итерации по массиву. В этой задаче создается массив определенного размера. Размер определяется входным параметром n. Затем выполняется итерация по массиву и заполняется значениями от 0 до n-1. Сложность этой задачи линейная O(n), она зависит от размера массива.

Третья задача – итерация по матрице. В этой задаче создается квадратная матрица определенного размера. Количество строк и столбцов определяется входным параметром n. Затем итерируется матрица и заполняется ее значениями. Эта задача имеет квадратичную сложность O(n^2).

Четвертое задание – вычисление числа Фибоначчи. Числа Фибоначчи или так называемый массив Фибоначчи представляет собой математический ряд чисел, который применяется во многих областях [7]. В этом массиве сумма двух предыдущих чисел дает значение следующего числа. Первые несколько членов этого ряда: 1, 1, 2, 3, 5, 8, 13, 21, 34 и 55. Число n, которое нужно вычислить, будет определяться входным параметром n. Сложность этой задачи экспоненциальная O(2^n).

Во втором, третьем и четвертом заданиях при проведении испытаний дополнительно задаются различные диапазоны для входных параметров.

2. *Разработка мобильных приложений*

Разработано мобильное приложение, реализующее ранее описанные задачи. В качестве платформы для разработки выбрана мобильная операционная система Android, поскольку она доминирует на рынке мобильных устройств. По данным статистики [8], эта операционная система установлена на 85% мобильных устройств. Приложение написано на языке программирования Java, а в качестве среды разработки использована Android Studio. Мобильное приложение состоит из front-end части, которая представляет собой пользовательский интерфейс, и back-end, где выполняются вычисления. Пользовательский интерфейс очень упрощен, как показано на рисунке 1. Для каждой задачи есть четыре радиокнопки, поле ввода для входного параметра и две кнопки – одна для локальных, другая для автономных вычислений.

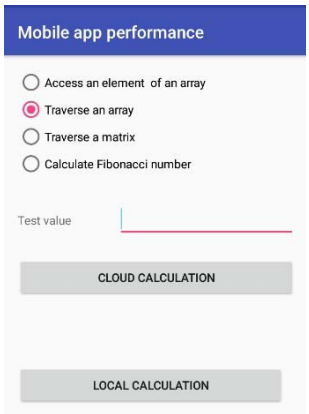

Рис. 1. Интерфейс приложения для синтетического тестирования

Fig. 1. Synthetic testing application interface

Для локальных вычислений код находится внутри приложения и выполняется локально на устройстве. Для выгруженных расчетов разработан Web API, в котором все задачи реализованы так же, как и в локальной версии. Здесь код и выполнение кода выгружены на удаленный сервер. Мобильное приложение выполняет вызовы к Web API. В запросе к конечной точке API оно отправляет тестовое значение.

3. *Тестовая среда и план тестирования*

Среда, в которой проводятся тесты, а также инструменты и необходимое оборудование, составляют тестовую среду.

Тестовая среда в данном исследовании состоит из:

- сервера, на котором расположен Web-API;
- мобильных устройств для тестирования;
- инструментов для мониторинга использования жестких ресурсов в мобильных устройствах;
- инструментов для анализа и визуализации данных.

Веб-сервер имеет 2 ГБ оперативной памяти и четырехъядерный процессор 3,3 ГГц. Список мобильных телефонов и их технические характеристики, используемые для тестирования, представлены в таблице 1.

Таблица 1

Характеристики устройств для тестирования

Table 1. Characteristics of testing devices

Модель	Операционная система	Процессор	Оперативная память
HuaweiP8	Android 8	4 ядра, 1,2 ГГц, Cortex-A53	4 Гбайта
SamsungGalaxyJ5	Android 7.1	4 ядра, 1,2 ГГц, Cortex-A53	4 Гбайта
LGL90	Android 9.0	4 ядра, 1,2 ГГц, Cortex-A7	4 Гбайта

Для каждого теста измеряется время, необходимое для выполнения задачи, использование оперативной памяти и процессора. Тесты проводятся в сетях Wi-Fi и 4G. Время, необходимое для выполнения задачи, рассчитывается в коде. Время выполнения для облачных вычислений включает общее время, необходимое для завершения запроса. Кроме того, измеряется время, необходимое серверу для обработки, для наблюдения задержки для каждого облачного запроса. Для измерения использования ресурсов устройства, оперативной памяти и процессора используется инструмент Android Profiler. Android Profiler является частью Android Studio 3.0, обеспечивает графическое отображение в реальном времени использования процессора, памяти и сети для запущенного мобильного приложения.

Процесс точного измерения энергопотребления мобильного телефона является сложной задачей. Один из подходов заключается в проведении физических измерений мощности на реальном оборудовании с помощью специализированного измерительного оборудования. Существуют программные инструменты для выполнения такого рода измерений, но трудно получить точную информацию. Поэтому здесь основное внимание уделяется времени выполнения, использованию процессора и памяти функций мобильного приложения, где можно получить более точные значения.

4. *Выполнение тестов*

Определено четыре тестовых случая для каждой задачи: доступ к элементу массива, итерация по массиву, итерация по матрице и вычисление числа Фибоначчи. Для тестовых случаев, требующих входные параметры, определено три диапазона: малый, средний и большой. Для малого диапазона берется арбитрное и достаточно маленькое значение, способствующее изменению загрузки памяти и процессора. Для большого диапазона берется приблизительное значение верхней границы, после которого у устройства не хватает ресурсов для выполнения действия. Средний диапазон определяется

между верхней и нижней границами. Максимальные значения отличаются для устройств с различной аппаратной конфигурацией. Определены следующие значения:

• итерация по массиву: малый: 1000; средний: 1000000; большой: 40000000 (Huawei), 30000000 (Samsung) и 20000000 (LG);

• итерация по матрице: малые: 100; средние: 1000; большие: 6000 (Huawei), 5000 (Samsung) и 4000 (LG);

• число Фибоначчи: малое:10; среднее: 30; большое: 40 (Huawei), 40 (Samsung) и 35 (LG).

5. *Результаты*

Тесты проводились в сети со следующей средней скоростью загрузки/выгрузки: 15–16 Мбит/с для Wi-Fi соединения и 1–8 Мбит/с для 4G соединения. Для каждого случая тестирования и соответствующего диапазона взято среднее время выполнения пяти измеренных значений. Значения отображаются в миллисекундах. На рисунке 2 показано время выполнения доступа к элементу для локального соединения, Wi-Fi и 4G. В этом тесте создается массив заданной длины и возвращается его первый элемент. Данные показывают, что локальная обработка намного быстрее, чем облачные вычисления. Локальная обработка занимает от 1 мс до 3 мс, в то время как облачная обработка в сети Wi-Fi занимает от 166 мс до 305 мс, из которых время выполнения сервера составляет 0 мс.

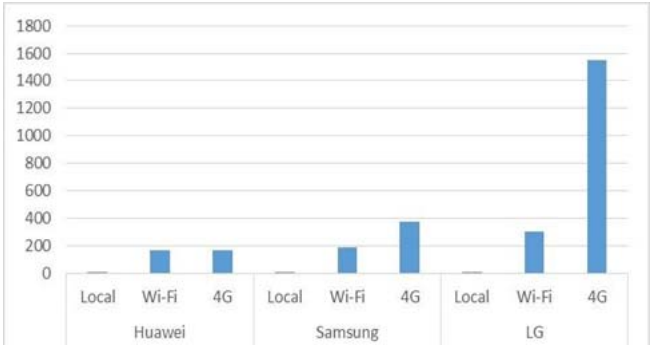

Рис. 2. Время выполнения доступа к элементу

Fig. 2. Item access time

На рисунке 3 показано время выполнения итерации по массиву. Данные отображаются для каждого типа выполнения: локального, Wi-Fi и 4G, и для каждого определенного диапазона. Данные показывают, что локальное время быстрее облачного в первом и втором диапазоне. В первом диапазоне локальное время выполнения занимает от 1 мс до 7,8 мс, облачное время выполнения в сети Wi-Fi занимает от 132 мс до 202 мс, из которых время выполнения сервера составляет 0 мс.

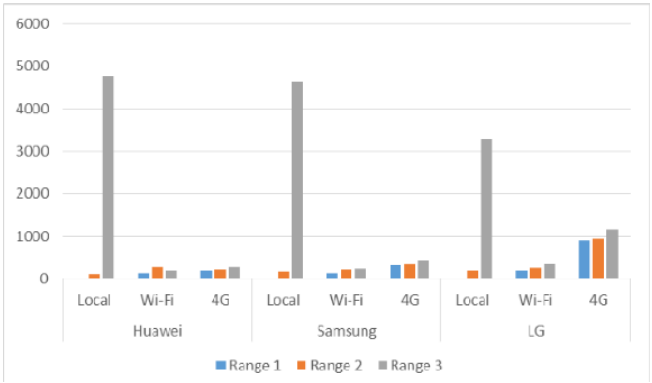

Рис. 3. Время выполнения обработки массива

Fig. 3. Array processing time

Во втором диапазоне локальное время занимает 119–191 мс, а время выполнения в облаке – 212–275 мс, из которых 3 мс – серверное время. Облачные вычисления намного быстрее локальных в третьем диапазоне, где локальное время занимает около 4235 мс, а облачное – около 264 мс, из которых 56 мс – серверное время.

На рисунке 4 показано время выполнения итерации через матрицу. Локальное время в первом диапазоне занимает от 3 мс до 5,8 мс, облачное время занимает от 116 мс до 196 мс и 0 мс серверного времени. Во втором диапазоне локальное время составляет 127–191 мс, облачное время – 211–239 мс и 3 мс серверного времени. В третьем диапазоне локальное время занимает около 4133 мс, а облачное – около 300 мс, из которых 100 мс приходится на серверное время.

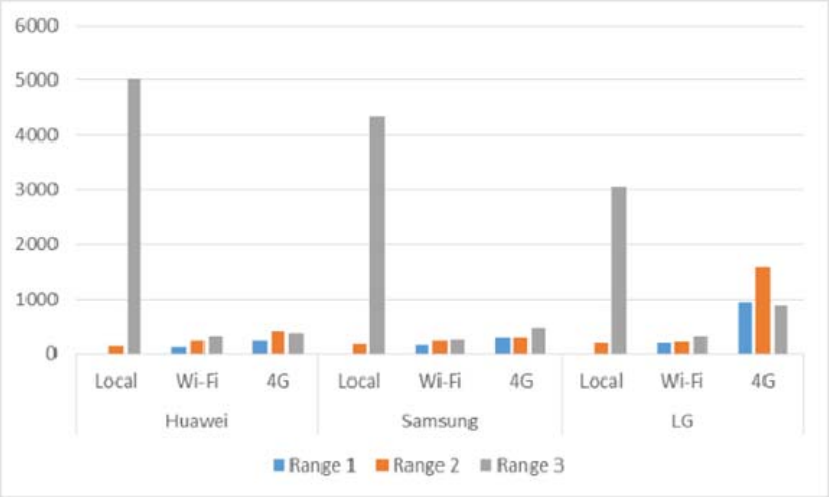

Рис. 4. Время выполнения итерационной матрицы мыслей

Fig. 4. Iterative thought matrix execution time

На рисунке 5 показано время выполнения вычисления числа Фибоначчи. Локальные вычисления быстрее только в первом диапазоне. Во втором и третьем диапазонах облачные вычисления быстрее.

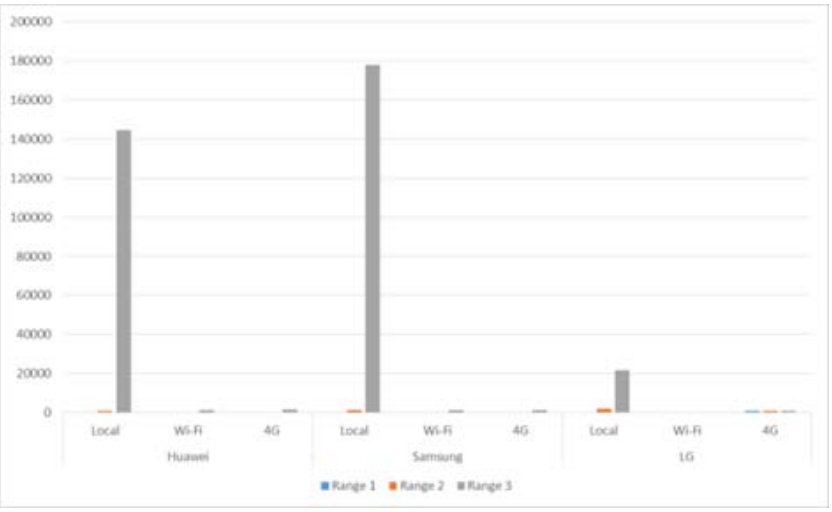

Рис. 5. Время выполнения для чисел Фибоначчи

Fig. 5. Execution time for Fibonacci sequence

Локальное время в первом диапазоне составляет 1–4,8 мс, а облачное время – 138–206 мс при 0 мс серверного времени. Во втором диапазоне локальное время занимает от 1166 мс до 197 мс, а облачное время –143–276 мс с серверным временем 4 мс. В третьем диапазоне локальное время занимает около 114–с, а облачное – около 1 с, из

которых 575 мс – серверное время. Если проанализировать время выполнения всех тестовых примеров, то наибольшая разница наблюдается в третьем диапазоне тестовых примеров, где локальное время занимает от 3 секунд до 178 секунд (около 3 минут), что ухудшает пользовательский опыт. В среднем обработка данных в сети Wi-Fi всегда быстрее, чем в сети 4G, примерно на 55%. Локальное время выполнения в третьем диапазоне показывает, что время выполнения LG мало по сравнению с другими телефонами. Это связано с тем, что на Huawei и Samsung тест выполняется при значении 40, а на LG – при значении 35. LG имеет более низкие аппаратные характеристики, и при более высоком значении приложение аварийно завершает работу.

На рисунке 6 показано использование памяти на всех трех устройствах.

Использование памяти скачет для локальных вычислений для самых больших диапазонов в тестовых примерах для массива и матричной итерации. Самые высокие пиковые значения памяти составляют 188 Мб для массива 40.00.00 элементов в Huawei и 139 Мб для 30000000 элементов в Samsung. Во всех остальных случаях нет существенной разницы между локальными и облачными вычислениями. Нет существенной разницы между объемом памяти при расчетах Wi-Fi и 4G на всех устройствах.

На рисунке 7 показана загрузка процессора, выраженная в процентах.

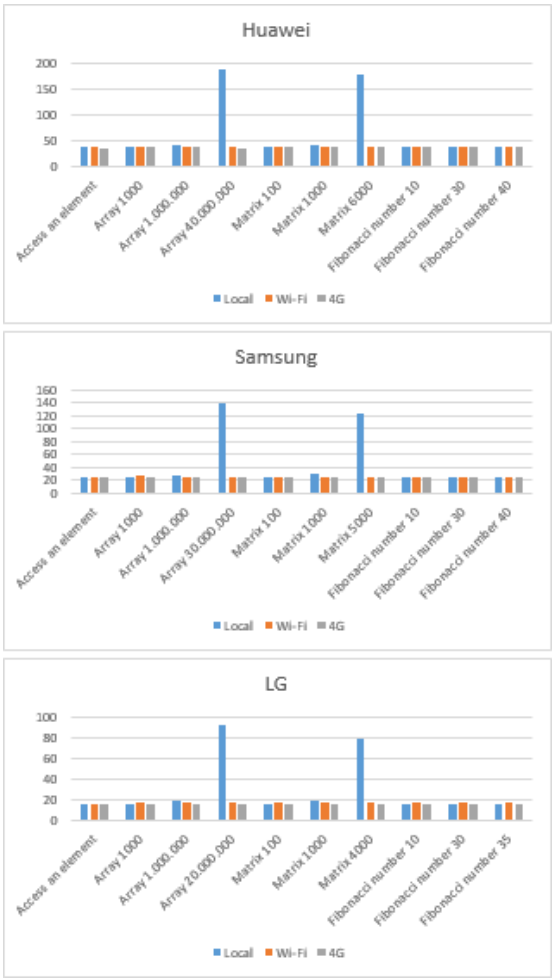

Рис. 6. Использование памяти, в %

Fig. 6. Memory usage, in %

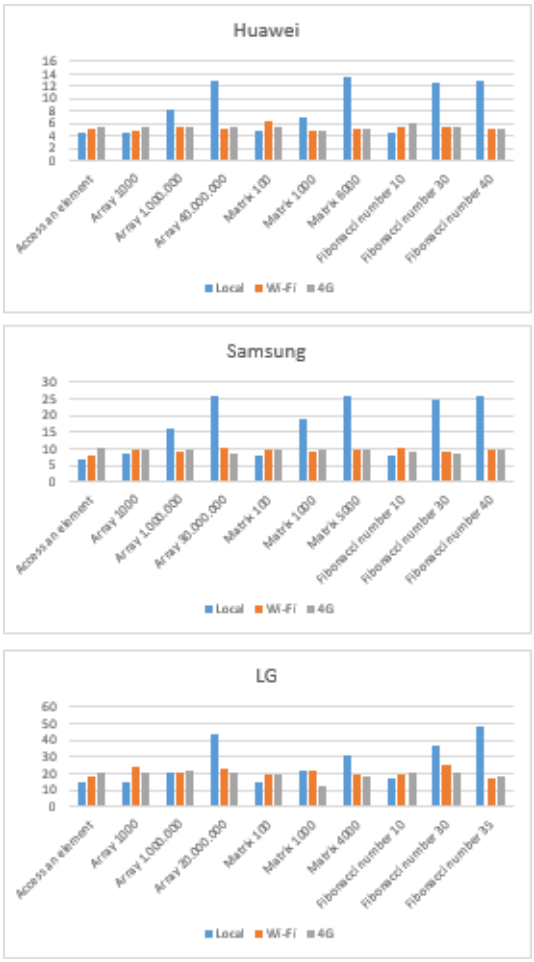

Рис. 7. Загрузка процессора в нагрузке, в %

Fig. 7. CPU activity, in %

Представленный процент использования процессора относится ко всей мощности процессора. Поскольку применяется однопоточный код, для обработки используется одно ядро. В тестовых случаях с последними двумя диапазонами в Huawei и Samsung загрузка процессора скачет для локальной обработки. В остальных случаях

нет существенной разницы в использовании между локальной и облачной обработкой. В Huawei использование варьируется от 4,4% до 13,6% для локальных и от 4,8% до 6,4% для облачных вычислений. В Samsung использование процессора варьируется от 7% до 26% для локальных и от 8,2% до 10,4% для облачных вычислений. В LG локальная обработка скачет в третьем диапазоне второго и третьего тестовых примеров и в последних двух диапазонах четвертого тестового примера. В остальных случаях нет существенной разницы в использовании локальной и облачной обработки. Локальная загрузка процессора составляет 14,6–48,6%, а в облаке – 18–25,2%. Существенной разницы в использовании между Wi-Fi и 4G нет.

4. Заключение

Результаты показывают, что задачи с постоянной сложностью лучше выполнять на устройстве. Эти задачи очень просты, и локальное время их выполнения намного меньше, чем время выполнения вычислений на устройстве. Локальное время выполнения также быстрее в первых двух диапазонах задач с линейной и квадратичной сложностью и в первом диапазоне задач с экспоненциальной сложностью. Общим для всех этих задач является то, что они используют определенное количество оперативной памяти и процессора на целевом устройстве: использование памяти составляет менее 30%, а использование процессора – менее 50%.

Результаты третьего диапазона задач с линейной, квадратичной и экспоненциальной сложностью показывают, что сервер имеет лучшую производительность по сравнению с локальным выполнением даже при дополнительной задержке. Эти задачи используют почти полную емкость выделенной памяти и процессора (полное использование одного ядра) и являются кандидатами на разгрузку. Мобильное устройство может справиться с их выполнением, но ценой высокой загрузки ресурсов и длительного времени выполнения, что приводит к ухудшению пользовательского опыта.

Этот подход к анализу может быть использован при разработке решений для мобильных приложений. Определив сложность каждой функции, мобильное решение можно разделить на части, которые должны выполняться локально, и части, которые должны быть выгружены.

Примечания

1. Android Studio. URL: https://developer.android.com/studio?hl=ru (дата обращения: 01.12.2022).
2. Kotlin. URL: https://kotlinlang.org/ (дата обращения: 01.12.2022).
3. Develop Android apps with Kotlin. URL: https://developer.android.com/kotlin?hl=ru (дата обращения: 29.01.2023).
4. Oracle Java. URL: https://www.oracle.com/ru/java/ (дата обращения: 17.02.2023).
5. Нативная или кроссплатформенная разработка? Наглядное сравнение. URL: https://appcraft.pro/blog/pochemu-my-vybiraem-nativnuyu-razrabotku/ (дата обращения: 20.02.2023).
6. Material Design. URL: https://material.io/design (дата обращения: 01.03.2023).
7. Human Interface Guidelines. URL: https://developer.apple.com/design/human-interface-guidelines/ (дата обращения: 01.12.2020).
8. App Annie State of Mobile in 2020 Report: 10 Highlights. URL: https://medium.com/@nadir/app-annie-state-of-mobile-in-2020-report-10-highlights-7746d7aad044 (дата обращения: 01.03.2023).

References

1. Android Studio. URL: https://developer.android.com/studio?hl=ru (access date: 01.12.2022).

2. Kotlin. URL: https://kotlinlang.org/ (access date: 01.12.2022).
3. Develop Android apps with Kotlin. URL: https://developer.android.com/kotlin?hl=ru (access date: 29.01.2023).
4. Oracle Java. URL: https://www.oracle.com/ru/java/ (access date: 17.02.2023).
5. Native or cross-platform development? Visual comparison. URL: https://appcraft.pro/blog/pochemu-my-vybiraem-nativnuyu-razrabotku/ (access date: 20.02.2023).
6. Material Design. URL: https://material.io/design (access date: 01.03.2023).
7. Human Interface Guidelines. URL: https://developer.apple.com/design/human-interface-guidelines/ (access date: 01.12.2020).
8. App Annie State of Mobile in 2020 Report: 10 Highlights. URL: https://medium.com/@nadir/app-annie-state-of-mobile-in-2020-report-10-highlights-7746d7aad044 (access date: 01.03.2023).

Авторы заявляют об отсутствии конфликта интересов.

The authors declare no conflicts of interests.

Статья поступила в редакцию 05.05.2023; *одобрена после рецензирования 29.05.2023*; *принята к публикации 30.05.2023.*

The article was submitted 05.05.2023; *approved after reviewing 29.05.2023*; *accepted for publication 30.05.2023.*