УДК 004.272.43 doi: 10.15827/0236-235X.148.504-513 2024. Т. 37. № 4. С. 504–513

Повышение вычислительной мощности персонального компьютера за счет интеграции с распределенной системой из смартфонов

С.А. Балабаев [1], С.А. Лупин [1], П.Н. Телегин [2, 3]✉, Б.М. Шабанов [2, 3]

[1] Национальный исследовательский университет «МИЭТ», г. Москва, 124498, Россия
[2] Межведомственный суперкомпьютерный центр РАН, г. Москва, 119334, Россия
[3] Национальный исследовательский центр «Курчатовский институт», г. Москва, 123182, Россия

Ссылка для цитирования
Балабаев С.А., Лупин С.А., Телегин П.Н., Шабанов Б.М. Повышение вычислительной мощности персонального компьютера за счет интеграции с распределенной системой из смартфонов // Программные продукты и системы. 2024. Т. 37. № 4. С. 504–513. doi: 10.15827/0236-235X.148.504-513
Информация о статье
Группа специальностей ВАК: 2.3.1
Поступила в редакцию: 25.07.2024 После доработки: 27.08.2024 Принята к публикации: 30.08.2024

Аннотация. В статье рассмотрена возможность повышения производительности персонального компьютера за счет интеграции с ним распределенной системы из смартфонов. Приведен обзор основных аппаратных и программных особенностей мобильных устройств, проведена оценка влияния подобных узлов на производительность распределенной вычислительной системы. Было выявлено, что их основными отличиями от персональных компьютеров являются низкое качество охлаждения устройств и архитектура big.LITTLE. Эти особенности мобильных устройств не позволяют задействовать все ядра смартфона на полную мощность и должны быть учтены при их интеграции в единую вычислительную среду. Программное обеспечение для интеграции мобильных устройств с персональными компьютерами состоит из двух приложений – клиента и сервера, разработанных на языке программирования Java. Приложения позволяют загружать вычисляемую задачу на узлы, запускать ее, аккумулировать и отображать полученные с узлов результаты вычислений. Взаимодействие между устройствами происходит по сети. Вычисляемая задача представляет собой apk-приложение, содержащее Java-класс, методы которого могут быть вызваны из приложения, запущенного на клиенте. Используемый алгоритм балансировки нагрузки узлов позволяет интегрировать в единую среду смартфоны разных поколений, значительно отличающиеся по производительности. Показано, что для эффективного распределения нагрузки между узлами системы необходимо использовать значение их реальной производительности. В работе приводятся результаты решения задачи минимизации функции в распределенной среде, организованной при помощи разработанного программного обеспечения. Они подтверждают достижение поставленной цели. Полученные результаты могут быть полезны широкому кругу специалистов.

Ключевые слова: вычислительная мощность, распределенные вычисления, смартфон, операционная система Android, грид-системы, балансировка нагрузки узлов
Благодарности. Работа выполнена в МСЦ РАН и НИЦ «Курчатовский институт» по теме FNEF-2024-0016

Введение. Во второй половине XX века началось бурное развитие микроэлектронной промышленности. Размеры электронных устройств непрерывно уменьшаются, при этом их функциональность столь же неуклонно возрастает. Современные *персональные компьютеры* (ПК) по своей вычислительной мощности многократно превосходят огромные мейнфреймы недалекого прошлого. Благодаря прогрессу в электронике были получены первые фотографические снимки черной дыры, разработаны эффективные лекарства за доступную цену, сделаны многие открытия.

Однако с ростом мощности вычислительных устройств растет и потребность людей в высокопроизводительных вычислениях. Доминирующим способом ускорения работы компьютеров стало использование нескольких вычислительных устройств для решения одной задачи. Такой подход реализован и в рабочих станциях, и в суперкомпьютерах.

Наряду с мощными многоядерными процессорами в вычислениях могут участвовать и более слабые узлы, например, планшеты, смартфоны, телевизионные приставки. Объединив их в сеть, можно сформировать распределенную среду, способную решать ресурсоемкие задачи.

В работе описано специализированное ПО, позволяющее интегрировать в единую среду смартфоны. По характеристикам современные мобильные устройства лишь немногим уступают ПК и позволяют выполнять сложные вычисления [1]. На их основе можно построить, например, распределенную систему HGRID (*Home GRID*) [2].

Существуют похожие решения групп исследователей из разных стран. В работе [3] предлагается объединять мобильные устройства с помощью программной платформы BOINC. В качестве примера применения такой системы рассмотрено ее использование для поиска ла-

тинских квадратов. На особенности разработанной системы влияют ограничения, накладываемые платформой BOINC, – необходимость разработки программного кода на языках C, C++, использование достаточно грубого алгоритма балансировки нагрузки.

Нидерландские ученые разработали систему IBIS, позволяющую объединять различные устройства для распределенных вычислений [4]. Проект написан на языке Java и предоставляет возможность работы с устройствами Android. Авторы предлагают эффективный способ коммуникации устройств между собой с помощью разработанной библиотеки и поддержку MPI. Однако проект был заморожен и доступа к нему нет.

В большинстве случаев современные исследователи предлагают использовать распределенную систему из Android-смартфонов для решения частной задачи – федеративного обучения [5, 6]. Хотя подобные кластеры и показывают свою эффективность, решать на них другие задачи без изменения программного кода невозможно.

Особенности разработки приложений для мобильных устройств

Разработка приложений для мобильных устройств является одним из ведущих IT-направлений. Число владельцев мобильных устройств, формирующих устойчивый спрос на новое ПО, увеличивается с каждым годом. Высокая производительность и широкая распространенность смартфонов позволяют без дополнительных затрат создать на их основе распределенную вычислительную систему, сопоставимую по мощности с ПК. Для оценки вычислительных возможностей смартфонов опишем особенности этих устройств.

Аппаратные характеристики

Современные мобильные телефоны, например, Honor 8X (JSN-L22), обладают достаточными ресурсами для проведения высокопроизводительных вычислений:

– аппаратная платформа – HiSilicon Kirin710;

– число ядер – 8;

– наличие архитектуры big.LITTLE;

– архитектура процессора – 4x ARM Cortex-A73 @ 2.19 GHz;

– объем оперативной памяти – 3 680 Мб.

Представленные параметры показывают, что смартфон можно рассматривать в качестве полноценной вычислительной платформы. Наличие восьми ядер позволяет производить параллельные вычисления, а высокая тактовая частота дает возможность выполнять их на сопоставимых с ПК скоростях.

Программные характеристики мобильных устройств

Семейство операционных систем, используемых в смартфонах, достаточно развито, среди них Android, iOS, Flume OS, Аврора. Наиболее популярной является система Android. Число мобильных устройств с Android превысило 3 млрд., в то время как устройств, выпущенных их основным конкурентом – компанией Apple, немногим более 1 млрд. штук.

В основе операционной системы Android лежит ядро Linux (рис. 1) [7]. Ядро взаимодействует с уровнем HAL (*Hardware Abstraction Layer* – аппаратный уровень абстракции), обеспечивающим связь между драйверами устройств и библиотеками. В пространство HAL входят несколько библиотечных модулей, каждый из которых реализует интерфейс для определенного аппаратного компонента.

Разработка ПО под архитектуру Android ведется на языках программирования Kotlin и Java. Для запуска программ, написанных на этих языках, требуется виртуальная машина. В Android используется виртуальная машина *Android Runtime* (ART). В версиях Android ниже 5.0 Lollipop использовалась Dalvik.

Отдельный уровень программной архитектуры – Native C/C++ Libraries – включает в себя набор библиотек, написанных на языке C или C++ и используемых различными компонентами ОС. При разработке ресурсоемких приложений со сложными вычислениями можно использовать нативный код для достижения большей производительности.

Для доступа к основным функциям операционной системы Android используется уровень Java API Framework, содержащий в себе набор приложений и библиотек для разработчиков [8].

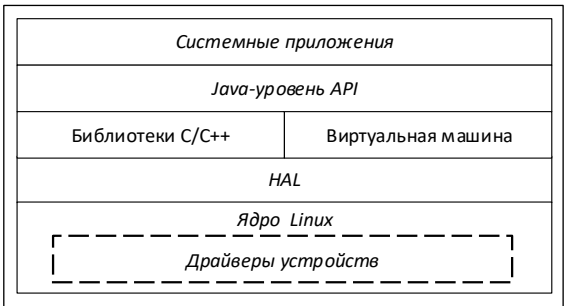

Рис. 1. Архитектура Android

Fig. 1. Android architecture

iOS – мобильная операционная система для устройств компании Apple. Для установки приложения на устройство с iOS требуются сертификаты, подписанные Apple, что делает разработку для этой платформы затруднительной.

Flume OS – операционная система на основе Android, поддерживаемая устройствами компании Meizu. Разработка приложений под нее аналогична разработке под Android.

Аврора – отечественная операционная система, разработанная компанией «Открытая мобильная платформа». В настоящее время она доступна только для ограниченного ряда устройств. Сегодня система менее перспективна для проведения исследований, но с учетом современных реалий ситуация может измениться [9].

Таким образом, можно сделать вывод, что наибольший интерес для использования в качестве узлов распределенной системы представляют мобильные устройства на основе операционной системы Android.

Использование мобильных устройств как вычислительных узлов

Рассмотрим особенности мобильных платформ, которые необходимо учитывать при реализации на них вычислительных задач.

Зависимость производительности процессора от температуры

В отличие от ПК конструкция смартфонов не предусматривает принудительного охлаждения процессора. При проведении интенсивных вычислений, особенно в тех случаях, когда задействованы все ядра, температура процессора мобильного телефона сильно повышается. Например, в случае проведения расчетов на смартфоне ASUS температура повышается до 45 градусов [1]. При достижении критических для устройства значений температуры операционная система снижает нагрузку на процессор, уменьшая тактовую частоту ядер устройства, вследствие чего производительность системы падает. Использование внешнего охлаждения корпуса смартфона с помощью кулеров дает лишь незначительный эффект. Как следствие, при оценке вычислительной мощности мобильного узла невозможно ориентироваться на максимальное значение рабочей частоты процессора.

Зависимость производительности процессора от заряда аккумулятора

При низком уровне заряда аккумулятора, как и при высокой температуре, смартфоны снижают производительность процессора. Для устранения этого эффекта при проведении вычислений в распределенной среде все его мобильные узлы должны быть подключены к сетевому питанию.

Использование архитектуры big.LITTLE

Большинство современных моделей смартфонов используют big.LITTLE. В основе архитектуры лежит совмещение относительно медленных энергосберегающих ядер с более мощными энергоемкими. В большинстве инсталляций число ядер процессора поровну делится между этими двумя типами. Фоновые задачи распределяются по LITTLE-ядрам, а ресурсоемкие выполняются на big-ядрах. Таким образом, при запуске интенсивных вычислений при высокой нагрузке могут быть задействованы и нагружены только big-ядра, на LITTLE-ядрах вычисления производиться не будут. Это снижает максимальную производительность мобильного узла почти в два раза по отношению к пиковому значению.

Пропускная способность сети (модуль Wi-Fi)

При реализации распределенной вычислительной системы необходимо обеспечить взаимодействие ее узлов между собой. Для этого можно использовать беспроводную локальную сеть. В современных смартфонах установлен Wi-Fi-модуль, работающий на частотах 2,4 или 5 МГц. Однако скорость передачи сообщений между устройствами достаточно низкая. Проведенный эксперимент по измерению скорости передачи данных между ПК и мобильным устройством Honor 8X с использованием точки доступа показал, что средняя скорость составляет 13.3 Мбит/с. Этого достаточно для распределенных вычислений, но не позволяет реализовывать параллельные программы с интенсивными межузловыми коммуникациями [10].

Если узлы расположены удаленно друг от друга, то единственным способом связи между устройствами будет Интернет. Для проверки качества связи были измерены входящий и исходящий трафики смартфона и ПК, находящихся в одной сети. Измерения проводились на расстоянии 1 метра от точки доступа при работе контроллера на различных частотах – 2,4 ГГц и 5 ГГц. На частоте 2.4 ГГц скорость оказалась значительно ниже, что связано с зашумленностью эфира (табл. 1).

Из таблицы видно, что скорость связи с сетью Интернет смартфона и ПК находится на одном уровне.

Промежуточные выводы

Приведенные особенности показывают, что смартфоны, как вычислительные узлы, значи-

тельно отличаются от ПК. Однако широкая распространенность и небольшие габариты позволяют использовать их в качестве узлов распределенной системы. Мощность нескольких смартфонов будет уже сопоставима с мощностью одного компьютера. При объединении мобильных устройств в распределенную систему важно учитывать разницу в характеристиках каждого из узлов и с помощью балансировки оптимально распределять вычислительную нагрузку между ними.

Таблица 1

Результаты эксперимента по проверке скорости Интернета

Table 1

Experimental results of the Internet speed test

Частота, ГГц	Трафик, Мбит/сек.			
	исходящий		входящий	
	смарт-фон	ПК	смарт-фон	ПК
2,4	117.05	55.83	35.10	37.86
5	699.52	300.67	198.68	199.92

Программные средства для организации распределенной системы из мобильных устройств

ПО, необходимое для интеграции смартфонов с ПК, должно обеспечивать

- объединение всех узлов в локальную сеть;
- передачу вычислительной задачи с инициирующего узла исполнителям;
- сбор полученных узлами результатов;
- балансировку нагрузки узлов в соответствии с их характеристиками.

Исследуемая распределенная система представлена на рисунке 2. Разработанное ПО состоит из двух приложений: для смартфонов и для ПК. Оба приложения разработаны на языке программирования Java.

Инициирующим узлом является ПК. С него вычислительная задача рассылается на исполняющие узлы и по команде запускается на каждом из них. В качестве среды для передачи данных используется беспроводная локальная сеть. Разработанное ПО может поддерживать вычисления и на узлах глобальной сети при условии соответствующей настройки маршрутизатора. После выполнения вычислений результаты передаются на инициирующий узел.

Приложение для смартфона (*Client*) обеспечивает ввод IP-адреса и порта для подключения к хосту (рис. 3). В программе предусмотрена возможность сбора логов основных характеристик смартфона: температура CPU, тактовые

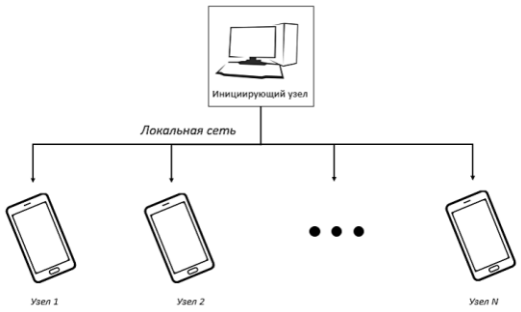

Рис. 2. Распределенная система с мобильными узлами

Fig. 2. Distributed system with mobile nodes

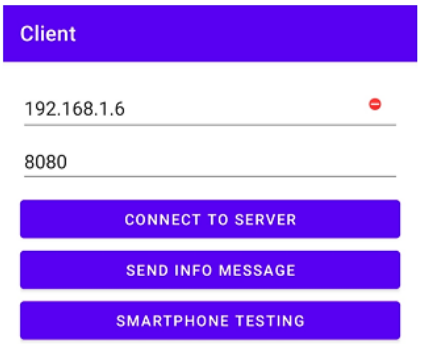

Рис. 3. Интерфейс ПО для смартфона (Client)

Fig. 3. Smartphone software interface (Client)

частоты каждого из ядер во время работы программы, время работы ядер на каждой из частот, температура, напряжение и уровень заряда батареи.

Интерфейс приложения для ПК (*Server*) представлен на рисунке 4. На экране в таблице отображаются все подключенные к нему узлы. Основные функции для работы запускаются нажатием кнопок, расположенных на панели справа.

Представим основные функции ПО инициирующего узла (функции приложения Server):

Open File – открытие файла с вычислительной задачей;

Send File – пересылка приложения на выбранные узлы;

Check File – проверка наличия приложения на выбранных узлах;

Calculate – запуск вычислений;

Balance – запуск балансировщика нагрузки узлов;

Stress – нагрузочное тестирование; функция предназначена для определения пиковых характеристик узла.

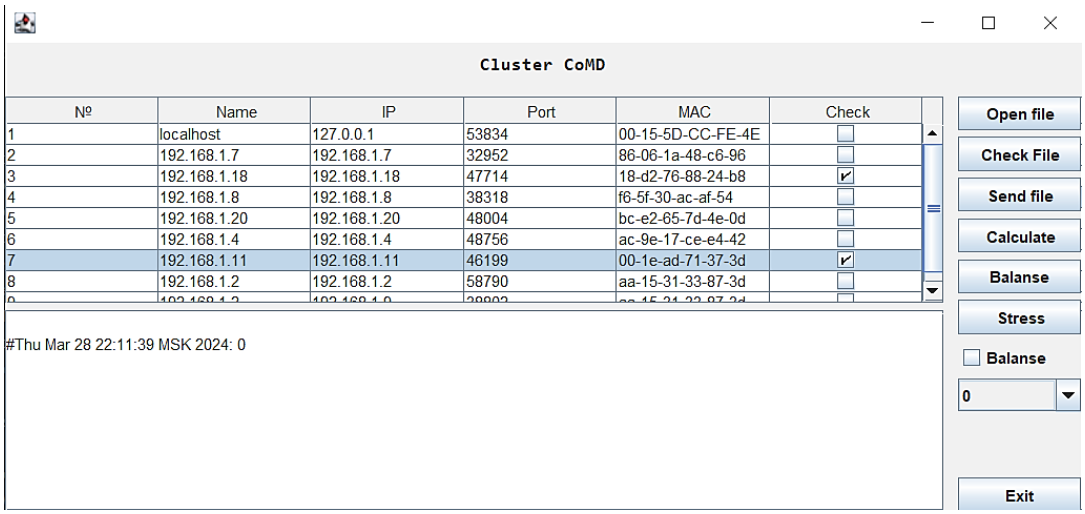

Рис. 4. Интерфейс ПО для ПК (Server)

Fig. 4. PC software interface (Server)

Разработанное ПО позволяет запускать распределенные приложения, не требующие интенсивного обмена сообщениями между узлами сети. Программный код и параметры рассылаются вычислительным узлам (рис. 4) перед началом работы. На каждом узле запускается приложение с полученными параметрами, а после завершения вычислений узел записывает полученный результат в файл. К этому классу задач относится, например, численное интегрирование.

Программные средства для организации распределенных вычислений

ПО для серверной части и клиентских узлов было разработано на языке Java, обеспечивающем необходимую производительность. Основной особенностью разработанного ПО является возможность включения в программу участков кода на языке C, которые и будут выполнять ресурсоемкие вычисления. Рассмотрим специфику реализации некоторых функций.

Механизм взаимодействия сервера и клиентов

После запуска приложения открывается порт (рис. 4), выбранный на прослушивание TCP-соединения, по которому к серверу подключаются клиенты-смартфоны. Взаимодействие между клиентами и сервером реализуется с помощью механизма сообщений, каждое из которых начинается и заканчивается ключевыми словами. Процесс взаимодействия происходит следующим образом: с инициирующего устройства отправляется сообщение, которое содержит ключевое слово, определяющее тип дальнейшей операции.

Синтаксис ключевых слов:

BALANCE – расчет коэффициента мощности узла;

SNDF – отправка приложения на узел;

CHS – проверка наличия приложения на узле;

STRESS – проведение нагрузочного тестирования;

CALC – проведение вычислений.

Рассмотрим программную реализацию основных операций.

Расчет коэффициента мощности узла

Критерием эффективности работы распределенной системы является минимизация времени выполнения задачи. Гетерогенность среды не позволяет добиться этого простейшим способом: разбивая задачу на равные части, требуется учитывать характеристики каждого узла, то есть выполнять балансировку нагрузки узлов. Для настройки балансировки на сервере хранится таблица, содержащая в себе MAC-адрес устройства и значение, равное времени выполнения тестовой задачи на узле. Эта информация позволяет вычислить относительную мощность всех интегрированных в единую среду устройств.

Коэффициент относительной мощности используется для разбиения задачи на части, пропорциональные возможностям устройства. Клиентам с высокой производительностью необходимо выделить больший объем вычислений, а для смартфона с низкими вычислительными возможностями выделить маленький участок.

В качестве характеристики мощности узла можно использовать и значение пиковой про-

изводительности, основанное на его архитектуре и частоте работы ядер процессора. При этом нет необходимости тестировать смартфон перед выполнением задачи, но в то же время результат балансировки будет значительно уступать нагрузочному тестированию [11].

Полученные результаты подтверждают это утверждение (рис. 5). При балансировке с использованием тестовой задачи время работы узлов сопоставимо друг с другом, тогда как при балансировке, основанной на пиковой производительности, оно сильно отличается.

Отправка приложения на узел

Для разработки более гибкого приложения, позволяющего запускать разнообразные задачи на стороне клиента, исполняемый код задачи передается узлу в виде apk-приложения. Пересылаемая программа должна содержать java-класс, которому необходимо дать название DynamicCalculator. Класс должен содержать в себе метод Task, который будет вызван на клиенте. Выгрузка программного кода будет происходить с помощью класса DexClassLoader, разработанного для Android-приложений, и позволяющего загружать классы из файлов .jar и .apk.

Для приложения, отправляемого клиентам, на сервере вычисляется контрольная сумма файла с помощью алгоритма md5. После передачи файла контрольная сумма рассчитывается и на клиентском узле, а полученные значения сравниваются между собой. При совпадении в лог программы выводится сообщение об успешном завершении операции пересылки, в противном случае – о неудаче и предложении повторить пересылку. На рисунке 6 представлена соответствующая этой операции диаграмма обмена сообщениями.

В случае отправки вычислительной задачи приложению клиента обмен сообщениями будет происходить следующим образом. Первым сообщением отправляется ключевое слово SNDF (сокращение от *Sendfile*), запускающее на клиенте метод класса для приема файла. В ответ клиент отправляет сообщение о готовности к приему данных, после чего последовательно получает имя файла и его содержимое. По завершении процесса передачи на клиенте рассчитывается контрольная сумма полученного файла и отправляется на сервер, где сравнивается с исходной. В случае совпадения процесс передачи завершается.

Проверка наличия задачи на узле

Пересылка задачи с сервера на клиентский узел занимает продолжительное время. Чтобы

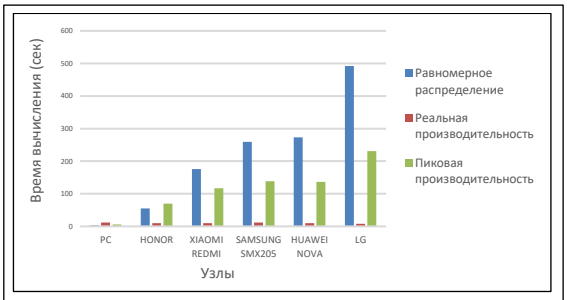

Рис. 5. Эффективность методов балансировки

Fig. 5. Effectiveness of balancing methods

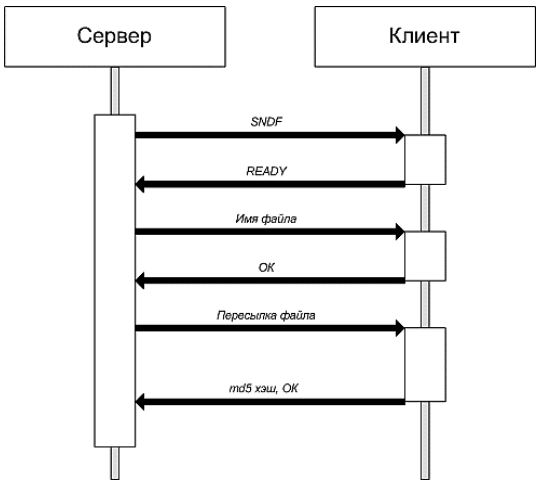

Рис. 6. Диаграмма обмена сообщениями при отправке вычислительной задачи

Fig. 6. Diagram of message exchange when sending a computational task

избежать повторной передачи данных, можно проверить наличие на узлах выбранной на исполнение задачи. Для этого используется метод сравнения контрольной суммы файла с задачей на сервере и на клиентском узле. Если значения совпадают, то соответствующий файл уже находится на узле и дополнительная пересылка не требуется.

Проведение нагрузочного тестирования

Для оценки параметров устройства во время вычислений используется нагрузочное тестирование. Оно реализовано с помощью выполнения в бесконечном цикле нагрузочной задачи до получения сигнала прерывания со стороны сервера. Во время ее исполнения можно снимать с узла с определенным промежутком времени следующие характеристики: время работы каждого из ядер процессора, значение тактовой частоты ядер, температура процессора, напряжение на батарее, уровень заряда батареи, температура батареи.

Проведение вычислений в распределенной системе

После загрузки вычислительной задачи на каждый из узлов запускается процесс вычисления. Для этого на каждый из выбранных узлов (рис. 4) отправляется ключевое слово CALC и после получения подтверждения от принимающей стороны происходит отправка параметров для запуска задачи. По окончании вычисления его результат возвращается на сервер. Если алгоритм предполагает многократный запуск задачи на узле, то возможна повторная отправка запроса на вычисления с новыми параметрами вычислительной задачи.

Эксперименты

Для проверки функциональности разработанного ПО была организована распределенная система, состоящая из шести узлов (http://www.swsys.ru/uploaded/image/2024-4/6.jpg), характеристики которых представлены в таблице 2.

Для узлов распределенной системы была рассчитана пиковая производительность по формуле

$$Perf = N_{cores} F_{cores} OpC,$$

где N_{cores} – число ядер процессора; F_{cores} – рабочая частота ядра (ГГц); OpC – число операций, выполняемых за такт.

Коэффициент относительной мощности узла определяется по формуле

$$Coef_i = \frac{\sum_1^6 Perf_i}{Perf_i}.$$

В таблице 3 приведены значения коэффициентов относительной мощности узлов, рассчитанные на основе их пиковой и реальной производительности.

Пиковая производительность распределенной системы составляет 210 Гигафлопс. Для

Таблица 2

Характеристики узлов распределенной системы

Table 2

Characteristics of distributed system nodes

Узел	Тип устройства	Марка	Модель	SoC	Процессор	RAM (Гб)	Число логических ядер	Тактовая частота (ГГц)
1	ПК	-	-	-	Intel core i5 11400H CPU @ 2.70 GHz	32	4	4*2.20
2	Смартфон	Huawei (Honor)	8X (JSN-L22)	Hisilicon Kirin 710	ARM 4×Cortex-A73 ARM 4×Cortex-A53	4	8	4*2,2 4*1,7
3	Смартфон	Xiaomi	Readme	Mt6877	ARM 2× Cortex-A78 ARM 6× Cortex-A55	8	8	2*2,6 6*2.0
4	Планшетный компьютер	Samsung	SM-X205	T618	ARM 6×Cortex-A55 ARM 2×Cortex-A75	4	8	8*2.02
5	Смартфон	Huawei	Huawei-Nova	Qualcomm Snapdragon 625	ARM 8×Cortex-A53	3	8	8*2.02
6	Смартфон	LG	Spirit LG-H422	My70ds	ARM 4×Cortex-A7	1	4	4*1.3

Таблица 3

Коэффициенты относительной мощности узлов

Table 3

Coefficients of node relative power

Узел	Тип устройства	Марка	Модель	$Perf_i$ (Гигафлопс)	$Coef_i$		
					Пиковая производительность	Реальная производительность	
						Однопоточное выполнение	Многопоточное выполнение
1	ПК	-	-	64,8	0,31	0,633	0,455
2	Смартфон	Huawei (Honor)	8X (JSN-L22)	31,2	0,15	0,181	0,161
3	Смартфон	Xiaomi	Readme	44,8	0,21	0,115	0,242
4	Планшетный компьютер	Samsung	SM-X205	32,0	0,15	0,011	0,050
5	Смартфон			32,0	0,15	0,049	0,088
6	Смартфон			5,2	0,02	0,011	0,002

оценки реальной производительности распределенной системы были проведены вычислительные эксперименты, в ходе которых запускалось приложение, выполняющее поиск экстремума функции с помощью метода дихотомии. Подобные методы используются и при обучении нейронных сетей.

Эксперименты проводились как в однопоточном, так и в многопоточном режимах, причем задействовались все доступные ядра узлов. Для балансировки нагрузки узлов были использованы коэффициенты из таблицы 3 для реальной производительности. Полученные результаты представлены на рисунке 7 и в таблице 4. На оси абсцисс рисунка отражается состав распределенной среды, то есть ПК, ПК+HONOR, ПК+HONOR+XIAOMI, REDMI и т.д.

Таблица 4

Эффективность HGRID

Table 4

HGRID effectiveness

Режим	**Время решения задачи (сек.)**		**Ускорение**
	ПК	**HGRID**	
Однопоточный	27.3	16.7	1.6
Многопоточный	6.6	3.5	1.8

Время решения задачи в распределенной среде снизилось в 1,6 раза в однопоточном режиме и в 1,8 в многопоточном по сравнению с вычислением на ПК.

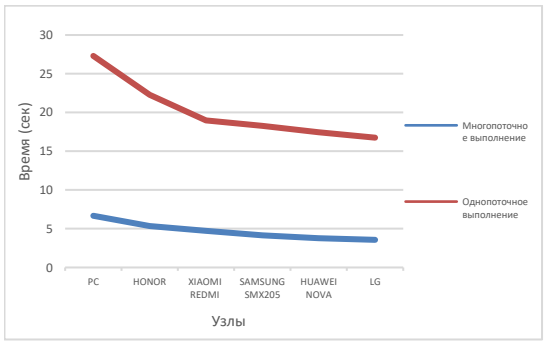

Рис. 7. Время решения задачи

Fig. 7. Problem solving time

Заключение

Проведенные исследования подтверждают возможность повышения вычислительной мощности ПК за счет интеграции с ним распределенной системы из смартфонов. Приложение, разработанное для проведения вычислений в HGRID, поддерживает узлы с операционными системами Android и Windows. Хотя использование старых смартфонов с низкими вычислительными возможностями и не оказывает существенного влияния на производительность HGRID, механизм балансировки позволяет использовать и их вклад в решение задачи. Конечно, HGRID-системы ориентированы прежде всего на реализацию распределенных приложений, но могут быть использованы, например, для отладки параллельных приложений.

Дальнейшие исследования будут направлены на оценку применимости HGRID для решения задачи обучения нейронной сети.

Список литературы

1. Балабаев С.А, Балабаев А.А. Применение CoMD-систем для обучения нейронных сетей // Микроэлектроника и информатика: матер. науч.-технич. конф. 2024. С. 24–28.

2. Балабаев С.А., Лупин С.А., Шакиров Р.Н. Вычислительный кластер на основе смартфонов Android и микрокомпьютеров Raspberry Pi // INJOIT. 2022. Т. 10. № 7. С. 86–93.

3. Kurochkin I., Dolgov A., Manzyuk M., Vatutin E. Using mobile devices in a voluntary distributed computing project to solve combinatorial problems. In: CCIS. Proc. RuSCDays, 2021, vol. 1510, pp. 525–537. doi: 10.1007/978-3-030-92864-3_40.

4. Gurusamy V., Nandhini K. IBIS: The new era for distributed computing, IJESTR, 2018, 7(1), pp. 61–65.

5. Li K.H., de Gusmão P.P.B., Beutel D.J., Lane N.D. Secure aggregation for federated learning in flower. Proc. DistributedML, 2021, pp. 8–14. doi: 10.1145/3488659.3493776.

6. Hasumi M. Azumi T. Federated learning platform on embedded many-core processor with flower. Proc. RAGE Workshop, 2024, pp. 1–6. doi: 10.1109/RAGE62451.2024.00015.

7. Таненбаум Э.С., Херберт Б. Современные операционные системы. СПб: Питер, 2018. 1120 с.

8. Ераннсон А. Эффективное использование потоков в операционной системе Android. М.: ДМК Пресс, 2015. 304 с.

9. Балабаев С.А., Лупин С.А. Оценка вычислительных возможностей мобильных устройств на платформе ОС Аврора // Микроэлектроника и информатика: матер. науч.-технич. конф. 2023. С. 51–56.

10. Таненбаум Э., Фимстер Н., Уэзеролл Д. Компьютерные сети; [пер. с англ.]. СПб: Питер, 2023. 992 с.

11. Мин Тху Кхаинг, Лупин С.А., Аунг Тху. Оценка эффективности методов балансировки нагрузки в распределенных вычислительных системах // INJOIT. 2021. Т. 9. № 11. С. 30–36.

Software & Systems doi: 10.15827/0236-235X.148.504-513 2024, 37(4), pp. 504–513

Increasing the PC computing power: Integration with a distributed smartphone system

Sergey A. Balabaev [1], Sergey A. Lupin [1], Pavel N. Telegin [2, 3]✉, Boris M. Shabanov [2, 3]

[1] National Research University of Electronic Technology, MIET, Moscow, 124498, Russian Federation
[2] Joint Supercomputer Center of RAS, Moscow, 119334, Russian Federation
[3] National Research Centre "Kurchatov Institute", Moscow, 123182, Russian Federation

For citation
Balabaev, S.A., Lupin, S.A., Telegin, P.N., Shabanov, B.M. (2024) 'Increasing the PC computing power: Integration with a distributed smartphone system', *Software & Systems*, 37(4), pp. 504–513 (in Russ.). doi: 10.15827/0236-235X.148.504-513
Article info
Received: 25.07.2024 After revision: 27.08.2024 Accepted: 30.08.2024

Abstract. The paper considers the possibility of improving PC performance by integrating a distributed smartphone system with it. There is an overview of the main hardware and software features of mobile devices. The authors evaluated the effect of such nodes on the performance of a distributed computing system. It was found that the main differences from PCs are low quality of device cooling and big.LITTLE architecture. These mobile device features do not allow using all smartphone cores at full capacity and should be taken into account when integrating them into a unified computing environment. Software for integrating mobile devices with personal computers consists of two applications: a client and a server developed in Java programming language. The applications allow uploading a computable task to nodes, running it, accumulating and displaying the computational results obtained from the nodes. The devices interact over the network. The computable task is an apk application containing a Java class with methods that can be called from the application running on the client. The utilized node load-balancing algorithm allows integrating smartphones of different generations, which differ significantly in performance, into a single environment. The authors show that in order to distribute the load efficiently between the system nodes, it is necessary to use their real performance value. The paper presents the results of solving the problem of function minimization in a distributed environment organized using the developed software. They confirm the achievement of the set goal. The obtained results are useful for a wide range of specialists.
Keywords: distributed computing, smartphone, Android OS, grid systems, node load balancing
Acknowledgements. The paper was carried out under the government assignment, project no. FNEF-2024-0016

References

1. Balabaev S., Balabaev A. (2024) 'Application of CoMD systems for training neural networks', *Proc. Sci. and Tech. Conf. Microelectronics and Informatics*, pp. 24–28 (in Russ.).
2. Balabaev, S.A., Lupin, S.A., Shakirov, R.N. (2022) 'Computing cluster based on Android smartphones and Raspberry Pi microcomputers', *INJOIT*, 10(7), pp. 86–93 (in Russ.).
3. Kurochkin, I., Dolgov, A., Manzyuk, M., Vatutin, E. (2021) 'Using mobile devices in a voluntary distributed computing project to solve combinatorial problems', in *CCIS. Proc. RuSCDays*, 1510, pp. 525–537. doi: 10.1007/978-3-030-92864-3_40.
4. Gurusamy V., Nandhini K. (2018) 'IBIS: The new era for distributed computing', *IJESTR*, 7(1), pp. 61–65.
5. Li, K.H., de Gusmão, P.P.B., Beutel, D.J., Lane, N.D. (2021) 'Secure aggregation for federated learning in flower', *Proc. DistributedML*, pp. 8–14. doi: 10.1145/3488659.3493776.
6. Hasumi, M., Azumi, T. (2024) 'Federated learning platform on embedded many-core processor with flower', *Proc. RAGE Workshop*, pp. 1–6. doi: 10.1109/RAGE62451.2024.00015.
7. Tanenbaum, E.S., Herbert, B. (2014) *Modern Operating Systems*. NJ, Hoboken: Pearson Education Publ., 1185 p. (Russ.ed.: (2018) St. Petersburg, 1020 p.).
8. Göransson, A. (2014) *Efficient Android Threading*. O'Reilly Publ., 277 p. (Russ. ed.: (2015) Moscow, 304 p.).
9. Balabaev, S.A., Lupin, S.A. (2023) 'Assessment of computing capabilities of mobile devices on the Aurora OS platform', *Proc. Sci. and Tech. Conf. Microelectronics and Informatics*, pp. 51–56 (in Russ.).
10. Tanenbaum, E., Feamster, N., Wetherall, D. (2021) *Computer Networks*. NJ, Hoboken: Pearson Education Publ., 944 p. (Russ. ed.: (2023) St. Petersburg, 992 p.).

11. Min Thu Khaing, Lupin, S.A., Aung Thu (2021) 'Evaluating the effectiveness of load balancing methods in distributed computing systems', *INJOIT*, 9(11), pp. 30–36 (in Russ.).

Авторы
Балабаев Сергей Андреевич,
аспирант, sergei.balabaev@mail.ru
Лупин Сергей Андреевич [1], к.т.н.,
профессор, lupin@miee.ru
Телегин Павел Николаевич [2, 3], к.т.н.,
ведущий научный сотрудник, ptelegin@jscc.ru
Шабанов Борис Михайлович [2, 3],
д.т.н., чл.-корр. РАН, директор,
зам. директора, shabanov@jscc.ru

[1] Национальный исследовательский
университет «МИЭТ»,
г. Москва, 124498, Россия
[2] Межведомственный суперкомпьютерный
центр РАН, г. Москва, 119334, Россия
[3] Национальный исследовательский центр
«Курчатовский институт», г. Москва, 123182, Россия

Authors
Sergey A. Balabaev [1], Postgraduate Student,
sergei.balabaev@mail.ru
Sergey A. Lupin [1], Cand. of Sci. (Engineering),
Professor, lupin@miee.ru
Pavel N. Telegin [2, 3], Cand. of Sci. (Engineering),
Leading Researcher, ptelegin@jscc.ru
Boris M. Shabanov [2, 3], Dr.Sci. (Engineering),
Corresponding Member of the RAS,
Director, Deputy Director, shabanov@jscc.ru

[1] National Research University
of Electronic Technology, MIET,
Moscow, 124498, Russian Federation
[2] Joint Supercomputer Center of RAS,
Moscow, 119334, Russian Federation
[3] National Research Centre "Kurchatov Institute",
Moscow, 123182, Russian Federation