Система мониторинга для балансировки нагрузки узлов распределенной вычислительной системы на основе смартфонов

С.А. Балабаев, С.А. Лупин, Ай Мин Тайк

***Аннотация*— В современном мире исследователи часто сталкиваются с необходимостью проводить высокопроизводительные вычисления, не имея мощных рабочих станций или доступа к кластерам. Мощности персонального компьютера недостаточно для решения ресурсоемких прикладных задач, но её можно увеличить, если интегрировать с ним такие устройства, как смартфоны на платформе Android, которые имеют в своем составе многоядерные процессоры и большой объем оперативной памяти и могут участвовать в ресурсоемких вычислениях в качестве узла распределенной системы. Однако, при интеграции в единую вычислительную среду разнородных устройств равномерное распределение нагрузки между ними приводит к низкой эффективности всей системы в целом. Решением проблемы является балансировка нагрузки узлов, учитывающая реальную, а не пиковую производительность узлов распределенной среды. В работе предложен способ балансировки нагрузки узлов распределенной вычислительной системы на основе смартфонов, использующий разработанный монитор для определения реальной производительности смартфонов. Данные об устройстве собираются с помощью клиентского приложения, формируются в пакет и передаются на сервер по сети или сохраняются в виде файла в памяти смартфона. На сервере происходит анализ полученного сообщения и вывод на экран характеристик устройств в виде графиков. Функциональность монитора проверялась в ходе экспериментов по определению параметров смартфона при решении тестовой задачи при разных температурных условиях - при комнатной температуре, при обдуве вентилятором и в условиях пониженной температуры. Полученные характеристики узлов были использованы для балансировки их нагрузки. Проведенные вычислительные эксперименты показали, что полученные с помощью разработанного монитора характеристики позволили устранить дисбаланс вычислительной среды, повысить ее производительность и снизить время вычислений в 1,8 раза. Разработанный монитор может быть использован для балансировки узлов распределенной системы, состоящей из устройств на платформе Android.**

***Ключевые слова*— распределенные вычисления, грид, смартфоны, ОС Android, мобильные приложения**

Статья получена 10 июля 2024.

С.А. Балабаев - аспирант Национального исследовательского университета «МИЭТ»; (email: sergei.balabaev@mail.ru)
С.А. Лупин - профессор, Национальный исследовательский университет «МИЭТ»; (e-mail: lupin@miee.ru)
Ай Мин Тайк - доцент, Национальный исследовательский университет «МИЭТ»; (e-mail: ayeminthike52@gmail.com)

I. Введение

Согласно отчету торговой организации GSM смартфонами обладают более 4,3 миллиардов людей или 54% населения. Широкая популярность таких мобильных устройств делает задачу разработки приложений под *Android* актуальной. Для этого используются такие языки программирования, как Java, Kotlin, C++, C#, JavaScript, Python. Самым популярным из них на сегодняшний момент является Kotlin. На январь 2024 года он занимает 17 строку в рейтинге самых используемых разработчиками языков программирования TIOBE.

В большинстве случаев смартфоны используются в качестве платформы для решения таких прикладных пользовательских задач, как телефонная связь, игры, WEB-навигация, обмен сообщениями и другие. Однако, мощности устройств позволяют проводить на мобильных телефонах и вычисления [1, 2]. Современные смартфоны обладают многоядерными процессорами, позволяющими производить высоконагруженные расчеты параллельно [3]. Операционная система Android также поддерживает возможность параллельных вычислений [4, 5]. Используя нативный код на языках C/C++ возможно добиться высокой производительности, а благодаря технологии OpenMP код для приложения можно распараллелить. В то время, когда владелец не использует свое устройство, смартфоны могут выполнять и ресурсоемкие вычисления, выполняя роль узла распределенной системы [6]. Вместе с устройствами на платформе Android в распределенной системе можно использовать и микрокомпьютеры Raspberry Pi, смартфоны с ОС «Аврора» [7, 8].

Использование смартфонов для проведения вычислений предлагается во многих научных исследованиях. Например, в работах [9, 10] предлагается использовать смартфон на платформе Android для расчета многочастотного фильтра. В настоящее время популярной задачей, для решения которой используются мобильные устройства, является обучение нейронных сетей. Смартфоны в этом случае выступают в качестве узлов для горизонтального федеративного обучения [11].

Для достижения высокой производительности распределенных систем с большим количеством

однородных узлов, можно применить эффективные решения, описанные в [1]. Если используется среда, объединяющая вычислительные мощности персонального компьютера и нескольких разнородных смартфонов, что характерно для домашнего грида, то применение подобных систем становится неэффективным без балансировки нагрузки мобильных узлов.

Реальную производительность вычислителей определяют различные характеристики устройства – число ядер процессора, их тактовые частоты, объем оперативной памяти и параметры коммуникаций. Конечно, их можно получить из документации и оценить пиковую производительность устройства. В отличие от ПК, во время работы мобильных устройств их характеристики, в частности тактовая частота, будут зависеть от температуры процессора, что влияет на производительность устройства. Для измерения этого показателя непосредственно во время работы узла необходимо использовать специальное приложение, по аналогии с функцией оценки производительности в диспетчере задач ОС Windows. Полученные данные позволят корректировать объем вычислительной нагрузки смартфона в соответствии с его возможностями.

II. Балансировка нагрузки в гетерогенной системе

Мы исследуем возможность проведения высокопроизводительных вычислений на гетерогенной системе **HGRID** (**H**ome **GRID)**. Её особенностью является небольшое число разнородных узлов – смартфонов и персональных компьютеров. Подобную систему можно собрать в домашних условиях, объединив имеющиеся у пользователя ресурсы. Модель работает по архитектуре клиент-сервер, где в качестве сервера выбирается самый мощный узел. Предполагается, что все узлы в системе постоянно подсоединены к вычислительному серверу по локальной сети или через проводное соединение.

Отметим, что **HGRID** состоят из разнородных узлов, поэтому для достижения высокой производительности необходимо балансировать их нагрузку – чем выше вычислительные возможности узла, тем большую часть задачи необходимо ему выделить.

Балансировку нагрузки возможно производить несколькими способами.

• Первый способ – используется значение пиковой производительности устройства. Такой способ применяется в работе [12], где каждому смартфону оператором присваивается ранг – от 0 до 10 в зависимости от характеристик устройства.

• Второй способ – используются реальные характеристики устройства. Для смартфонов её определяет тактовая частота, на которой он работает в текущих условиях.

Для второго способа необходима разработка программного обеспечения, позволяющего снимать характеристики с устройства и отправлять их на сервер для дальнейшего анализа и проведения балансировки.

Существующие приложения для анализа производительности устройств.

Для сбора показателей устройства в реальном времени существуют различные приложения. Приведем наиболее популярные из них:

• CPU-Z
• Device Info HW
• DevCheck Hardware and System Info
• AIDA64
• Cpu Monitor
• G-CPU.

С их помощью можно получить значения тактовой частоты ядер процессора, температуры процессора, уровня нагрузки ядер, температуры и напряжения аккумулятора. Основной недостаток приведенных выше приложений – невозможность сохранять значения параметров для дальнейшей их обработки. Поэтому для организации высокопроизводительных вычислений в распределенной среде со смартфонами в качестве узлов необходимо разработать специальное приложение, обеспечивающее мониторинг их параметров.

Получение характеристик смартфона

Для получения основных параметров мобильного устройства во время вычислений было разработано специальное приложение - монитор. Рассмотрим его основные функции, обеспечивающие возможность получать данные о CPU - время работы каждого из ядер процессора, их тактовой частоты и температуры, а для аккумулятора – уровень заряда, напряжение и температура [13].

Получение характеристик процессора

Для получения характеристик процессора разработан класс CPUinfo, собирающий необходимые данные и передающий их в программу для дальнейшей обработки (рис. 1).

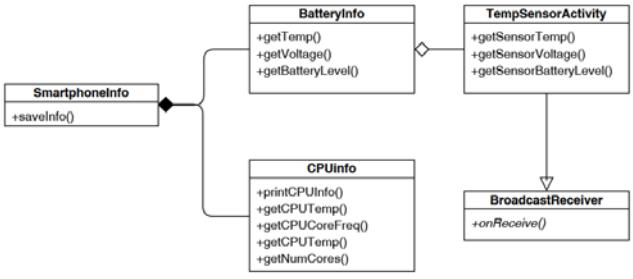

Рис. 1 UML-диаграмма классов

Информация о процессоре хранится в файловой системе устройства по адресу */sys/devices/system/cpu/*. По указанному пути располагаются директории, относящиеся к каждому из ядер смартфона. Для определения числа ядер достаточно посчитать количество таких директорий. Каждая из них имеет вид «*cpuN*», где N – номер ядра. Для определения числа ядер разработан метод *getNumCores()*. Он содержит перегруженный метод *accept*, возвращающий *true* при совпадении имени файла с заданным шаблоном поиска и *false* в обратном случае.

Для получения времени работы каждого из ядер необходимо прочитать значение из файла */sys/devices/system/cpu/cpuN/cpufreq/stats/time_in_state,* где N – номер ядра процессора. Чтение значений производится в цикле от 0 до N, где N – переменная, хранящая возвращенное значение методом *getNumCores().* Значения читаются с помощью методов классов, предназначенных для буферизированного чтения из файла, и сохраняются в массив из строк *output,* который передается пользователю для дальнейшей обработки.

Аналогично производится чтение текущей частоты каждого из ядер. Требуемое значение расположено по пути:

/sys/devices/system/cpu/cpuN/cpufreq/scaling_cur_freq.

Температура процессора может быть получена чтением файла:

*/sys/devices/*virtual*/thermal/thermal_zone0/temp.*

Для перевода полученного значения в градусы Цельсия его необходимо разделить на 1000.

Получение характеристик батареи

Для получения параметров аккумулятора используется предоставляемый разработчикам класс языка Java BatteryManager. Класс содержит переменные, характеризующие параметры батареи. В приложении использованы значения: EXTRA_TEMPERATURE – температура аккумулятора, EXTRA_VOLTAGE – напряжение на выходе, EXTRA_LEVEL – уровень заряда. Для получения значений только при изменениях показаний датчика разработан класс TempSensorActivity, унаследованный от класса BroadcastReceiver. При получении сообщения от датчика (срабатывание Intent.ACTION_BATTERY_CHANGED) вызывается переопределенный метод onReceive и значения с датчика передаются в программу [15].

Пересылка на сервер

После сбора всех параметров их значения необходимо передать на сервер для проведения анализа. Соединение устройств возможно с использованием кабеля, Bluetooth или WiFi [15]. В качестве решения была выбрана беспроводная сеть в связи с её доступностью и высокой пропускной способностью. Передача значений параметров происходит по сети в реальном времени после их считывания, или предварительно сохранив их в файл.

Передача данных в реальном времени

В отельном потоке открывается еще одно сетевое соединение, использующее UDP-сокет, по которому передаются значения параметров с интервалом в 1 секунду. Результаты передаются в приложение сервера, где графически представляются в отдельном окне в виде графиков.

В случае критических значений параметров, например, выход температуры процессора за допустимые пределы, инициируется вывод информационного сообщения на экран.

Передача данных в виде пакета

Для сохранения результатов измерений параметров устройства в директории, связанной с приложением, создается текстовый файл, в который с помощью разработанного метода writeToFile записываются все полученные значения. Для записи создается отдельный поток threadWriter.

Полученный файл можно передать по сети на сервер или загрузить его из памяти телефона вручную.

Для обработки и графической визуализации полученных характеристик разработан скрипт на языке python. Он принимает на вход файл со снятыми данными и отображает результат в графическом виде.

III. Система мониторинга состояния узлов распределенной системы

Для отображения в реальном времени параметров узла на экране ПК было разработано приложение на языке Java. Монитор отображает следующие параметры: температура процессора, уровень заряда, температура и напряжение батареи (рис. 2).

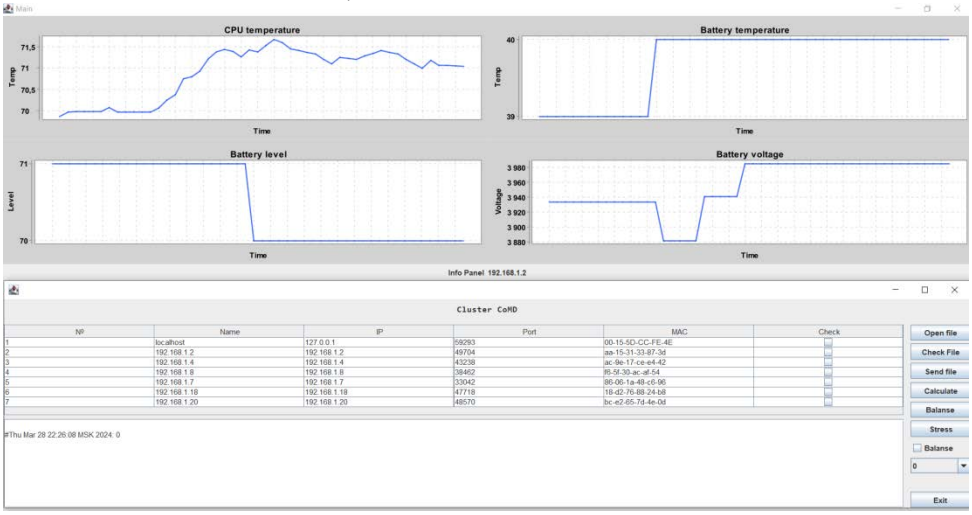

Рис.2 Интерфейс системы мониторинга

Работа системы мониторинга происходит по следующему алгоритму. После запуска программы открывается TCP соединение для подключения пользователей к серверу **HGRID**. После подключения узла его MAC адрес заносится в таблицу с информацией об устройствах, работающих в системе. При подключении нового клиента в отдельной форме выводится состояние устройства в реальном времени.
По сети через UDP-соединение программа получает пакет, содержащий сообщение в формате:

АДРЕС_ИСТОЧНИКА;ТИП_СООБЩЕНИЯ;ДАННЫЕ

В качестве адреса источника указывается MAC адрес устройства, отправившего сообщение. В случае, если он совпадает с одним из адресов устройств в таблице узлов, то сообщение передается дальше на этап анализа. В противном случае пришедший пакет отбрасывается.
На этапе анализа из сообщения выделяется второй параметр – тип сообщения, содержащий одно из четырех значений (табл. 1).

Таблица 1. Типы сообщений

Тип сообщения	Значение
CPU_T	температура процессора
BAT_T	температура батареи
BAT_V	напряжение батареи
BAT_L	уровень заряда батареи

В зависимости от типа пришедшего сообщения, его содержимое с данными передается в кольцевой буфер. Он создан для синхронизации анализатора пришедших пакетов данных и потока, выводящего значения на экран в виде графика. Для каждого из типов сообщения создан отдельный кольцевой буфер (рис. 3).

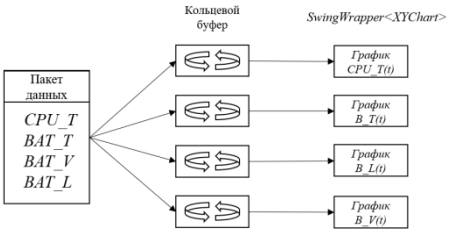

Рис. 3 Схема передачи и обработки пакета данных

Для вывода графиков на экран использовалась библиотека *xchart*. Для каждого из графиков был создан отдельный экземпляр класса *XYChart*, объединенных для удобства работы с ними в *XChartPanel*. Для параллельного вывода значений для каждого из графиков создан отдельный экземпляр класса *MySwingWorker*, унаследованный от *SwingWorker*. Этот класс позволяет создавать отдельные потоки для работы с графиками [16].

IV. Проведение экспериментов по снятию характеристик узла

Условия эксперимента

Для тестирования монитора и анализа характеристик смартфонов был проведен ряд экспериментов. Известно, что при высокой нагрузке температура процессора мобильного устройства повышается, а тактовая частота его ядер автоматически снижается, что приводит и к снижению производительности устройства. Для того, чтобы проверить функциональность монитора и его способность оценивать производительность смартфона эксперименты проводились в разных условиях:

1. при комнатной температуре (21° C),
2. смартфон охлаждался с помощью вентилятора – внешний обдув,
3. смартфон помещался в морозильную камеру при температуре -10°C.

В экспериментах мы использовали смартфон с характеристиками, указанными в табл. 2. Во всех случаях на смартфоне была запущена нагрузочная задача. Первые 15 минут смартфон находился при указанных выше условиях без нагрузки. Далее на 30 минут циклически запускалась вычислительная задача, нагружающая все ядра устройства. Во время экспериментов со смартфона снимались перечисленные выше параметры, сохранялись на устройстве и по окончании передавались на сервер для дальнейшей обработки и анализа.

Таблица 2. Характеристики исследуемого смартфона

Марка	Модель	SoC	Процессор	RAM (Гб)	Число логических ядер	Тактовая частота (ГГц)
Honor	JSN-L22	Hisilicon Kirin 710	ARM 4×Cortex-A73 4×Cortex-A53	4	8	4*2,2 4*1,7

Результаты экспериментов

В ходе экспериментов было получено среднее время выполнения задачи на устройстве при различных температурных режимах. Результаты вычислений приведены на рис. 4–6. На графиках по оси X отложено количество последовательных запусков задачи, по оси Y – время ее выполнения. Без охлаждения среднее время выполнения задачи составляет 11.5 секунд, при разбросе значений 4 секунды. С использованием вентилятора среднее время снижается до 10.5 секунд, уменьшается и разброс значений.

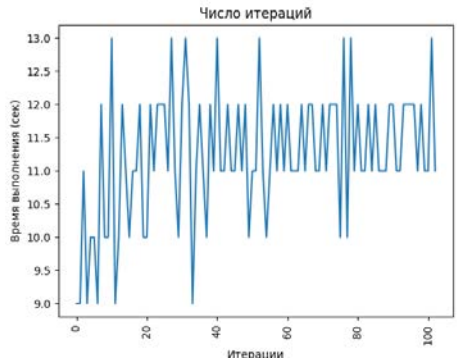

Рис. 4 Без охлаждения

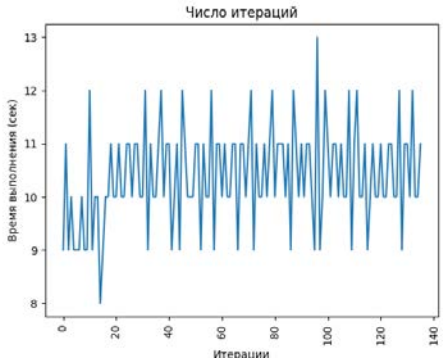

Рис. 5 Вентилятор

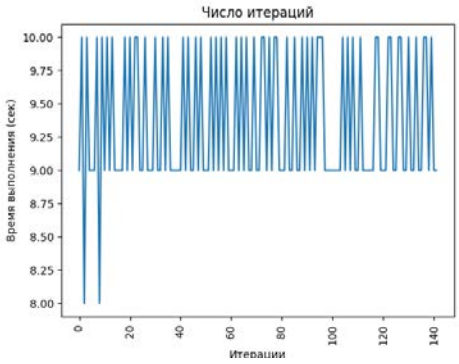

Рис. 6 Морозильная камера

Исследуемый смартфон содержит два 4-х ядерных процессора - Cortex-A73 и Cortex-A53. В таблице 4 представлены частоты, на которых могут работать их ядра.

Таблица 4. Тактовые частоты процессоров смартфона

Процессор	Тактовые частоты (ГГц)			
Cortex-A73	0,807	1,037	1,268	1,460
	1,671	1,824	1,997	2,189
Cortex-A53	0,48	0,96	1,152	1,325
	1,440	1,536	1,709	

В процессе вычислений монитор собирал значения времени работы каждого из ядер на каждой из возможных частот.

Полученные результаты были представлены в виде графиков зависимости времени работы на определенной частоте от времени проведения эксперимента (рис.7 – 9).

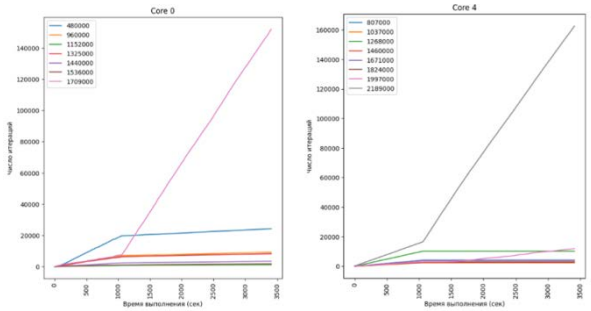

Рис. 7 Морозильная камера

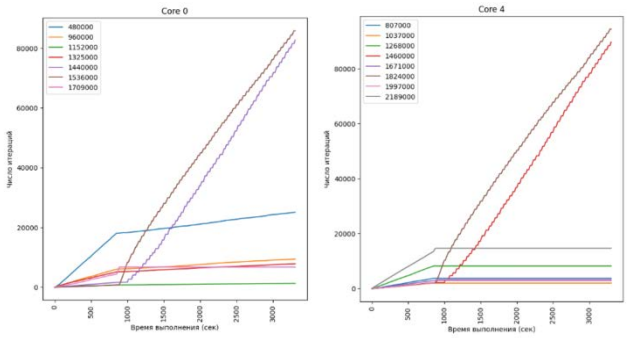

Рис. 8 Вентилятор

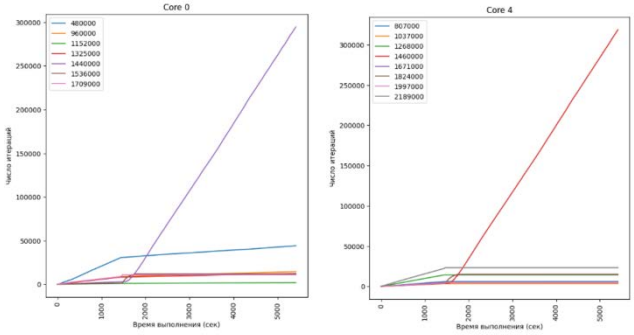

Рис. 9 Без охлаждения

Поскольку все ядра процессора работают на одинаковой частоте, на рисунках показаны графики только для одного из ядер процессора Cortex-A53 (Core 0) и Cortex-A73 (Core 4).

Рассмотрим, какой вывод позволяют сделать полученные результаты. После запуска нагрузочной задачи во время экспериментов 1 и 3 процессоры через некоторое время начинает работать на одной преобладающей частоте. Во втором эксперименте процессоры переключаются с одной частоты на другую. Преобладающие значения частот показаны в таблице 5.

Таблица 5. Преобладающие частоты

Номер эксперимента	Тип охлаждения	Преобладающие частоты (ГГц)
1	Без охлаждения	Core 0-3 – 1.44 Core 4-7 – 1.46
2	Вентилятор	Core 0-3 – 1.44 / 1.536 Core 4-7 – 1.46 / 1.824
3	Морозильная камера	Core 0-3 – 1.709 Core 4-8 – 2.189

Результаты экспериментов показывают, что при нагрузке смартфон начинает выполнять задачу на максимально возможной частоте, однако после повышения температуры процессора понижает её, что приводит к снижению скорости вычислений. Это хорошо согласуется и с результатами измерения времени выполнения задачи, приведенными на рис. 4–6.

Исходя из поставленного эксперимента, можно сделать вывод, что пиковые значения частоты ядер процессора устройств не являются оптимальными характеристиками для балансировки нагрузки. Их использование возможно только при постоянном внешнем охлаждении, что трудно реализуется на практике.

Поэтому, в качестве характеристики было выбрано значение тактовой частоты, получаемой с помощью программы мониторинга.

V. Балансировка нагрузки узлов HGRID

Оценим эффективность работы гетерогенной системы **HGRID** при разных способах балансировки нагрузки узлов системы. Для сравнения были реализованы два способа балансировки узлов – первый, учитывающий пиковые значения частоты устройств (табл. 4) и второй, основывающийся на получаемой монитором рабочем значении тактовой частоты.

В качестве узлов использованы устройства, представленные в таблице 6. Время решения задачи при первом способе балансировки составило 5,1 сек., а при втором – 2,86 сек.

Таблица 6. Узлы HGRID

Узел	Тип устройства	Марка	Модель	SoC	Процессор	RAM (Гб)	Число логических ядер	Тактовая частота (ГГц)
1	Персональный компьютер (ПК)	-	-	-	Intel core i5 11400H CPU @ 2.70 GHz	32	4	4*2.20
2	Смартфон	Huawei	Huawei-Nova	Qualcomm Snapdragon 625	ARM 8×Cortex-A53	3	8	8*2.02
3	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
4	Смартфон	Xiaomi	Readme	Mt6877	ARM 2× Cortex-A78 ARM 6× Cortex-A55	8	8	2*2,6 6*2.0

Для оценки уровня сбалансированности нагрузки рассчитана дисперсия для выборки значений времени работы узлов при разных видах балансировки. Получено, что при первом способе балансировки дисперсия составляет D = 3,275, а при втором D = 0,358. Это доказывает, что монитор позволяет более точно балансировать нагрузку узлов, что хорошо видно на рисунке 10.

Рис. 10 Время выполнения задачи на узлах

VI. Заключение

Приложение, разработанное для мониторинга параметров смартфона при решении вычислительных задач, в ходе экспериментов подтвердило свою функциональность. Балансировка нагрузки узлов **HGRID**, базирующаяся на получаемом монитором значении тактовой частоты ядер процессора, привела к существенному снижению дисбаланса и повышению производительности системы.

Разработанный монитор позволяет наблюдать и за параметрами аккумулятора, однако все вычисления проводились в режиме питания от сети, и эта возможность не использовалась.

VII. Литература

[1] Kurochkin I. et al. Using Mobile Devices in a Voluntary Distributed Computing Project to Solve Combinatorial Problems //Supercomputing: 7th Russian Supercomputing Days, RuSCDays 2021, Moscow, Russia, September 27–28, 2021, Revised Selected Papers 7. – Springer International Publishing, 2021. – С. 525-537.

[2] Долгов А.А. Разворачивание Грид-системы из Мобильных устройств на платформе BOINC // Облачные и распределенные вычислительные системы в электронном управлении ОРВСЭУ-2022 в рамках национального супрекомпьютерного форума (НСКФ-2022), 2022 с. 24-29

[3] Phuc B. H. et al. Enhancing the performance of android applications on multi-core processors by selecting parallel configurations for source codes //2017 4th NAFOSTED Conference on Information and Computer Science. – IEEE, 2017. – С. 225-229.

[4] Kumar T. U., Senthilkumar R. CWC*—Secured distributed computing using Android devices //2016 International Conference on Recent Trends in Information Technology (ICRTIT). – IEEE, 2016. – С. 1-7.

[5] Mateeva G. et al. Some capabilities of android os for distributed computing //2021 Big Data, Knowledge and Control Systems Engineering (BdKCSE). – IEEE, 2021. – С. 1-6.

[6] Балабаев С.А. Оценка вычислительных возможностей мобильных платформ //28-я Всероссийская межвузовская научно-техническая конференция студентов и аспирантов «Микроэлектроника и информатика - 2021», 2021.

[7] Балабаев С.А., Лупин С.А. Оценка вычислительных возможностей мобильных устройств на платформе ОС Аврора //Микроэлектроника и информатика - 2023. Материалы научно-технической конференции, pp. С. 51-56. , 20 04 2023.

[8] Балабаев С.А., Лупин С.А., Шакиров Р.Н. Вычислительный кластер на основе смартфонов Android и микрокомпьютеров Raspberry Pi //International Journal of Open Information Technologies. – 2022. – Т. 10. – №. 7. – С. 86-93.

[9] Acosta A., Almeida F. Parallel implementations of the particle filter algorithm for android mobile devices //2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing. – IEEE, 2015. – С. 244-247.

[10] Acosta A., Almeida F. The particle filter algorithm: parallel implementations and performance analysis over Android mobile devices //Concurrency and Computation: Practice and Experience. – 2016. – Т. 28. – №. 3. – С. 788-801.

[11] Tang J. et al. PE-FedAvg: A Privacy-Enhanced Federated Learning for Distributed Android Malware Detection //2023 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom). – IEEE, 2023. – С. 474-481.

[12] Salem H. Distributed computing system on a smartphones-based network //Software Technology: Methods and Tools: 51st International Conference, TOOLS 2019, Innopolis, Russia, October 15–17, 2019, Proceedings 51. – Springer International Publishing, 2019. – С. 313-325.

[13] Hardy B., Phillips B. Android programming: the big nerd ranch guide. – Addison-Wesley Professional, 2013.

[14] Васильев А.Н. Java. Объектно-ориентированное программирование. Учебное пособие. Стандарт третьего поколения. – " Издательский дом"" Питер""", 2021

[15] Aljohani M., Alam T. Design an M-learning framework for smart learning in ad hoc network of Android devices //2015 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC). – IEEE, 2015. – С. 1-5.

[16] Fain Y. Java programming 24-hour trainer. – John Wiley & Sons, 2011.

Monitoring system for load balancing of distributed computing system nodes based on smartphones

Sergey Balabaev, Sergey Lupin, Aye Min Thike

***Abstract*— In the contemporary era, researchers frequently encounter the necessity of executing high-performance computing without the availability of robust workstations or access to clusters. The processing power of a personal computer is insufficient for the resolution of resource-intensive applied tasks. However, this can be augmented by the integration of such devices as Android smartphones, which are equipped with multi-core processors and a substantial amount of random-access memory (RAM), enabling them to perform resource-intensive computations as a node of a distributed system. However, when heterogeneous devices are integrated into a single computing environment, even distribution of load between them results in low efficiency of the whole system. The solution to this problem is node load balancing, which takes into account the real, rather than peak, performance of the nodes of the distributed environment. This paper proposes a method for load balancing the nodes of a smartphone-based distributed computing system using the developed monitor to determine the real performance of smartphones. The device data is collated by the client application, assembled into a packet and transmitted to the server via the network or stored as a file within the smartphone's memory. The server then performs an analysis of the received message and presents the device characteristics in graphical form. The functionality of the monitor was evaluated through experimentation to ascertain the parameters of the smartphone when solving the test problem under varying temperature conditions, namely at room temperature, when blown by a fan, and in conditions of reduced temperature. The characteristics of the nodes were employed to achieve a state of equilibrium in the computing environment. The computational experiments demonstrated that the characteristics obtained with the assistance of the developed monitor enabled the elimination of the imbalance in the computing environment, an increase in its performance and a reduction in the computation time by a factor of 1.8. The developed monitor can be utilised to achieve equilibrium in the nodes of a distributed system comprising Android devices.**

***Keywords*— distributed computing, grid, smartphones, Android OS, mobile applications**

References

[1] Kurochkin I. et al. Using Mobile Devices in a Voluntary Distributed Computing Project to Solve Combinatorial Problems //Supercomputing: 7th Russian Supercomputing Days, RuSCDays 2021, Moscow, Russia, September 27–28, 2021, Revised Selected Papers 7. – Springer International Publishing, 2021. – S. 525-537.
[2] Dolgov A.A. Razvorachivanie Grid-sistemy iz Mobil'nyh ustrojstv na platforme BOINC // Oblachnye i raspredelennye vychislitel'nye sistemy v jelektronnom upravlenii ORVSJeU-2022 v ramkah nacional'nogo suprekomp'juternogo foruma (NSKF-2022), 2022 s. 24-29
[3] Phuc B. H. et al. Enhancing the performance of android applications on multi-core processors by selecting parallel configurations for source codes //2017 4th NAFOSTED Conference on Information and Computer Science. – IEEE, 2017. – S. 225-229.
[4] Kumar T. U., Senthilkumar R. CWC*—Secured distributed computing using Android devices //2016 International Conference on Recent Trends in Information Technology (ICRTIT). – IEEE, 2016. – S. 1-7.
[5] Mateeva G. et al. Some capabilities of android os for distributed computing //2021 Big Data, Knowledge and Control Systems Engineering (BdKCSE). – IEEE, 2021. – S. 1-6.
[6] Balabaev S.A. Ocenka vychislitel'nyh vozmozhnostej mobil'nyh platform //28-ja Vserossijskaja mezhvuzovskaja nauchno-tehnicheskaja konferencija studentov i aspirantov «Mikrojelektronika i informatika - 2021», 2021.
[7] Balabaev S.A., Lupin S.A. Ocenka vychislitel'nyh vozmozhnostej mobil'nyh ustrojstv na platforme OS Avrora //Mikrojelektronika i informatika - 2023. Materialy nauchno-tehnicheskoj konferencii, pp. S. 51-56. , 20 04 2023.
[8] Balabaev S.A., Lupin S.A., Shakirov R.N. Vychislitel'nyj klaster na osnove smartfonov Android i mikrokomp'juterov Raspberry Pi //International Journal of Open Information Technologies. – 2022. – T. 10. – #. 7. – S. 86-93.
[9] Acosta A., Almeida F. Parallel implementations of the particle filter algorithm for android mobile devices //2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing. – IEEE, 2015. – S. 244-247.
[10] Acosta A., Almeida F. The particle filter algorithm: parallel implementations and performance analysis over Android mobile devices //Concurrency and Computation: Practice and Experience. – 2016. – T. 28. – #. 3. – S. 788-801.
[11] Tang J. et al. PE-FedAvg: A Privacy-Enhanced Federated Learning for Distributed Android Malware Detection //2023 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom). – IEEE, 2023. – S. 474-481.
[12] Salem H. Distributed computing system on a smartphones-based network //Software Technology: Methods and Tools: 51st International Conference, TOOLS 2019, Innopolis, Russia, October 15–17, 2019, Proceedings 51. – Springer International Publishing, 2019. – S. 313-325.
[13] Hardy B., Phillips B. Android programming: the big nerd ranch guide. – Addison-Wesley Professional, 2013.
[14] Vasil'ev A.N. Java. Ob"ektno-orientirovannoe programmirovanie. Uchebnoe posobie. Standart tret'ego pokolenija. – " Izdatel'skij dom"" Piter"""", 2021
[15] Aljohani M., Alam T. Design an M-learning framework for smart learning in ad hoc network of Android devices //2015 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC). – IEEE, 2015. – S. 1-5.
[16] Fain Y. Java programming 24-hour trainer. – John Wiley & Sons, 2011.