Вычислительный кластер на основе смартфонов Android и микрокомпьютеров Raspberry Pi

С.А. Балабаев, С.А. Лупин, Р.Н. Шакиров

***Аннотация*— В настоящее время широкое распространения получили такие мобильные устройства, как смартфоны и планшеты. Вычислительные мощности их процессоров сопоставимы с характеристиками персональных компьютеров и это даёт возможность производить на них сложные ресурсоёмкие вычисления. В области IoT большой популярностью пользуются микрокомпьютеры Raspberry Pi. В работе исследуется возможность построения вычислительного кластера из смартфонов с ОС Android и микрокомпьютеров Raspberry Pi. В статье представлены результаты тестирования практической реализации распределенной системы, в узлах которой используются разнородные смартфоны. Определены вычислительные возможности системы и доказана применимость статической балансировки нагрузки узлов для повышения эффективности вычислений. Полученные результаты могут быть полезны широкому кругу специалистов.**

***Ключевые слова*—вычислительный кластер, Android, Raspberry Pi, OpenMP**

I. Введение

Изменяя окружающий мир, приспосабливая его к своим потребностям, люди часто используют решения, подсказанные самой природой. Множество инструментов и механизмов были созданы, опираясь на бионику. Однако в вычислительной технике такой подход практически не используется. Считается, что компьютеры не имеют аналогов в живой природе. Конечно, нейроморфные вычисления и нейронные сети сегодня являются одними из самых популярных направлений исследований, но сам объект, который при этом воспроизводится, всё ещё является загадкой для нас. В этой работе мы постараемся оценить возможность более широкого использования такого свойства живой природы, как «колонии и стаи».

Более 200 миллионов лет назад на Земле уже обитало множество живых существ самого разного размера. Средой их обитания были все природные среды - вода и воздух, суша и подземные пространства. Наибольшего расцвета достигли огромные динозавры, например, такие как Аллозавры, Диплодоки и Тираннозавры. Наряду с этими гигантскими ящерами существовали и гораздо более мелкие обитатели, например гекконы, чей размер составлял всего несколько сантиметров (Рис. 1). До сих пор находят останки подобных существ, застывших в янтаре [1]. Казалось бы, именно могучие динозавры должны были победить в ходе эволюции, но их уже давно нет на планете, а мелких представителей реликтовой фауны можно встретить и в наши дни (табл. 1). Существует множество гипотез, пытающихся объяснить этот феномен, но нам интересен сам факт такого результаты эволюции, а не его причины.

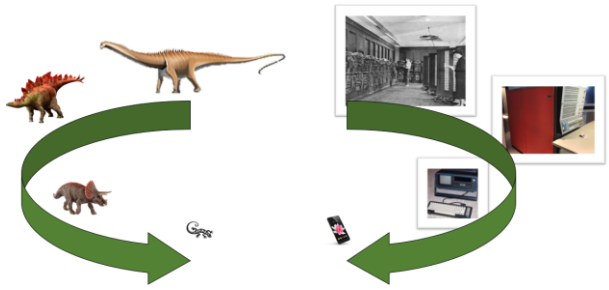

Рис. 1. Лента времени

Таблица 1. Эволюция живых существ

Вид	Вес	Статус
Диплодок	~35-40 т	Вымерли
Трицератопс	~9-10 т	Вымерли
Стегозавр	~3-5 т	Вымерли
Геккон	~50 гр	Выжили

Подобную тенденцию можно проследить и в развитии электронной техники. Еще 80 лет назад вес ЭВМ составлял около 30 тонн. С развитием технологий размеры компьютеров стали уменьшаться и в этой гонке стали «выживать» менее габаритные и более легкие модели (Табл. 2). Например, в 1964 году появляется семейство компьютеров IBM System/360, средний вес которых составлял уже менее 800 кг.

Отметим, что метод интеграции ресурсов отдельных узлов для построения мощных вычислительных сред является сегодня доминирующим. Именно так построены все современные суперкомпьютеры. С архитектурной точки зрения они являются гетерогенными кластерами, в узлах которых

Статья получена 23 мая 2022.

С.А. Балабаев - магистр Национального исследовательского университета «МИЭТ»; (email: sergei.balabaev@mail.ru)

С.А. Лупин - профессор, Национальный исследовательский университет «МИЭТ» ; (e-mail: lupin@miee.ru)

Р.Н. Шакиров - профессор, Университет прикладных наук Бонн; (roustiam.chakirov@h-brs.de)

расположены мощные многоядерные процессоры и графические ускорители.

Таблица 2. Эволюция электронной техники

Название	Год выпуска	Вес	Статус
«Эниак»	1946 г	27 т	Не производится
IBM System/360	1964 г	770 кг	Не производится
IBM 5100	1975 г	25 кг	Не производится
Commodore SX-64	1984 г	10 кг	Не производится
ASUS ExpertBook B9450FA	2020 г	0,9 кг	Производится

Достаточно популярным решением можно считать и GRID-системы, кластеры из персональных компьютеров (CoPC) и рабочих станций (CoWS). Однако такая интеграция совершенно не затрагивает самые маленькие и повсеместно распространенные мобильные устройства - смартфоны. Кластеры на их основе можно назвать – CoMD.

II. Вычислительные возможности смартфонов

Первоначально предназначенные только для коммуникации, мобильные телефоны превратились сегодня в полноценные вычислительные устройства. Одним из первых смартфонов считается выпущенный в 1993 году IBM Simon. Он работал на 16-битном процессоре Vadem VG-230 16 МГц и имел один мегабайт оперативной памяти. Другим представителем первых смартфонов был Nokia 7650, обладающий более мощным процессором и большей оперативной памятью, чем предшественник. В начале 2010 года компания LG представила первый в мире двухъядерный смартфон LG Optimus 2X P990, что положило начало многоядерности в смартфонах [2], [3].

В настоящее время большинство моделей смартфонов оснащены многоядерными процессорами. Например, современный телефон Samsung Galaxy S21 содержит 8 ядер. Подобная архитектура позволяет использовать смартфоны при решении вычислительно сложных задач, если соответствующие приложения разделить на параллельные участки и выполнять одновременно на нескольких ядрах. Еще одной тенденцией развития вычислительных архитектур является ориентация на интеграцию разнородных элементов в единое вычислительное пространство. Устройства, получаемые при этом, называются гетерогенными [4].

Например, в 2020 году компания Intel представила процессоры Lakefield, содержащие пять ядер [5]. Одно из них – мощное, а остальные четыре – экономичные. Подобное решение позволяет повысить производительность и энергоэффективность платформы.

В 2021 году были выпущены более совершенные гетерогенный процессоры линейки Alder Lake.

Таблица 3. Эволюция смартфонов

Смартфон	Год выпуска	Процессор	Число ядер	Объем оперативной памяти
IBM Simon	1993 г	x86 совместимый, 16 MHz, 16 bit	1	1 Мб
Nokia 7650	2002 г	32-bit RISC, 104 MHz ARM 9	1	4 Мб
LG Optimus 2X P990	2010 г	ARM Cortex-A9, 32-bit, 1 GHz	2	512 Мб
Samsung Galaxy S21 Ultra	2021 г	3x Cortex-A78 2.8 GHz, 4x Cortex-A55 2.2 GHz, 1x Cortex-X1 2.9 GHz	8	12 Гб

Например, процессор Intel Core i9-12900HK имеет 14 ядер, из которых 6 предназначены для «тяжелых» задач, и 8 для фоновых [6]. Гетерогенные процессоры применяются и в смартфонах. Упоминаемый выше смартфон Samsung Galaxy S21 Ultra имеет гетерогенный процессор, содержащий три типа ядер. Следует отметить, что разнородность ядер в первую очередь ориентирована на повышение энергоэффективности устройства, а при распределении вычислительной нагрузки между ядрами это создает дополнительную проблему и требует учета мощности каждого ядра для достижения максимальной производительности.

III. Кластерные архитектуры

Вычислительные возможности мобильных устройств, таких как смартфоны и планшеты, значительно возросли за счет увеличения числа ядер и мощности микропроцессоров, повышения объема оперативной памяти. Не случайно, что они вызывают всё больший интерес у исследователей, связанных с проведением сложных ресурсоемких вычислений. Конечно, значительную вычислительную мощность могут обеспечить только кластеры, интегрирующие ресурсы отдельных устройств, подобно тому, как это реализуется в современных суперкомпьютерах. В работах [7], [8] приводится пример подобного кластера. Исследователи отмечают, что существенным недостатком такой архитектуры, если в качестве узлов используются мобильные устройства, является необходимость их постоянной подзарядки [9]. В случае расположения

устройств внутри помещения с постоянным доступом к источникам питания, эта проблема становится неактуальной.

При всем многообразии существующих подходов к созданию приложений для параллельных вычислительных систем, в зависимости от архитектуры памяти их можно разделить на два класса. В случае, когда память для всей системы является общей, используется механизм распределения частей задачи между потоками. Потоки – самостоятельные цепочки последовательно выполняемых операторов программы, соответствующих некоторой подзадаче, при этом все глобальные данные будут общими для каждого из потоков [10].

В случае, когда память системы является распределенной в качестве механизма распараллеливания, используются процессы. Процесс – программа во время исполнения или объект, которому выделяются ресурсы вычислительной системы. Каждый процесс обладает индивидуальной областью памяти, которая может содержать как локальные, так и глобальные переменные, которые будут индивидуальными для каждого из процессов [10].

Чтобы добиться максимальной производительности, нужно эффективно использовать все вычислительные ресурсы устройства. Необходимо распараллелить исходный код программы так, чтобы он мог использовать для вычислений все имеющиеся ядра узла. Задачу необходимо разбить на процессы, каждый из которых будет выполняться на отдельном узле. Если узел содержит многоядерный процессор, способный выполнять параллельный код, то каждый процесс необходимо разбивать и на отдельные потоки. Это возможно сделать как вручную, с помощью стандартных функций создания потоков, так и с помощью директив OpenMP [11].

Для создания распределенной системы из мобильных устройств мы можем использовать несколько вариантов. Одним из самых популярных вариантов является построение грид-системы. Грид-вычисления — это форма распределённых вычислений, в которой «виртуальный суперкомпьютер» представлен в виде кластера из соединённых с помощью сети, слабосвязанных гетерогенных компьютеров, работающих вместе для выполнения огромного количества заданий [12].

Это достаточно дешевый способ, который часто используют для кластеров из микрокомпьютеров Raspberry Pi, где он показывает неплохие результаты [13]. Микрокомпьютеры объединяются в локальную сеть и производят распределенные вычисления. На таких устройствах решают задачи компьютерного зрения и распознавания образов.

В работе [14] представлен вариант объединения Raspberry Pi Compute Module в кластер на коммутационной печатной плате.

Проведенные исследования показывают, что такое решение является компактным и при этом имеет высокую производительность. Схема разработанного кластера показана на рисунке 2.

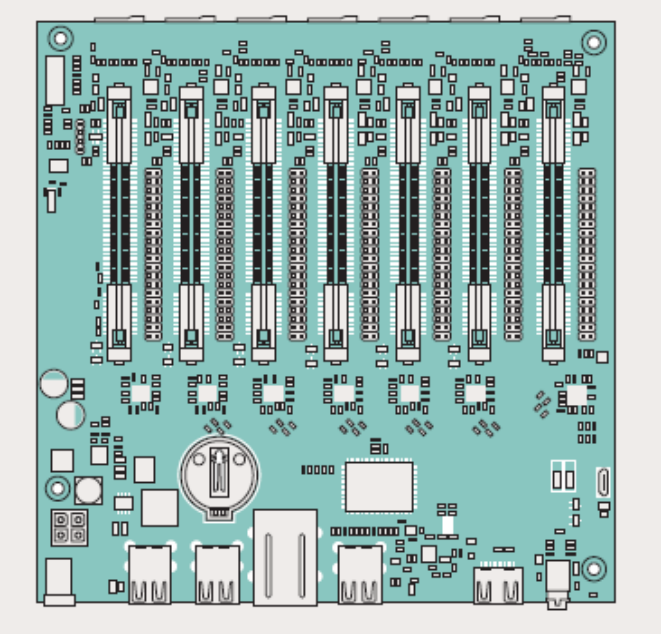

Рис. 2. Кластер из Raspberry Pi

Однако у всех подобных устройств есть существенный недостаток – сложно найти назначение подобному кластеру, поскольку его мощность сопоставима с мощностью персонального компьютера. Для промышленных масштабов такие кластеры бесполезны, а для домашнего использования – дорогостоящи и габаритны.

Этот недостаток можно устранить, если объединять в кластер смартфоны. Их мощности позволяют решать многие вычислительные задачи. При этом создание распределенной вычислительной системы на их основе не требуется никаких денежных вложений – в наше время смартфоны есть в каждой семье по несколько штук у каждого из её членов. При выходе новой модели телефона старая обычно ложиться на полку. Её можно использовать для кластерных вычислений.

IV. Распределенная вычислительная система CoMD

Для подтверждения тезиса о возможности практического использования вычислительных систем типа CoMD был построен кластер из устройств, приведенных в таблице 4.

Учитывая то, что CoMD системы относятся к архитектурам с распределенной памятью, для оценки их производительности необходимо выбрать приложение, обеспечивающее эффективное распараллеливание задачи при минимальном взаимодействии узлов. Этому требованию хорошо удовлетворяет задача численного интегрирования методом прямоугольников. Алгоритм расчета можно построить таким образом, чтобы каждый

узел кластера вычислял значение интеграла на выделенном ему отрезке. Для параллельного запуска приложения необходимо передать узлам только границы интервала интегрирования и собрать полученные результаты после завершения вычислений.

Для проведения исследований был выбран интеграл:

$$y = \int_0^{10} x^2 \sin(x) dx \quad (1)$$

При вычислениях использовалась формула:

$\int_a^b f(x)dx \approx h\sum_1^{N-1} f_i$, где $h = \frac{b-a}{N}$, N — число разбиений. Сложность последовательного алгоритма можно оценить как $O(N)$. Тестовое приложение создано на языке программирования C. Для компиляции и запуска программы использовалось приложение Termux, эмулирующее командную строку Linux и имеющее возможность компилировать и запускать программы на языке C. Таком образом нет необходимости разрабатывать отдельную программу специально для смартфонов. Более того, программный код можно использовать и для тестирования платформы на основе Raspberry Pi. Встроенный компилятор поддерживает технологию OpenMP, что позволяет эффективно распараллелить вычисление, распределив нагрузку и между ядрами процессоров узлов кластера.

Таблица 4. Узлы экспериментального гетерогенного кластера

Узел	**Тип устройства**	**Марка**	**Модель**	**SoC**	**Процессор**	**RAM (Гб)**	**Число логических ядер**	**Тактовая частота (ГГц)**
1	Смартфон	Honor	JSN-L22	Hisilicon Kirin 710	ARM 4×Cortex-A73 4×Cortex-A53	4	8	4*2,2 4*1,7
2	Смартфон	Huawei	ANA-NX9	-	ARM 8×Cortex-A55	4	8	4*1,92 4*2,36
3	Смартфон	Huawei	Huawei-Nova	Qualcomm Snapdragon 625	ARM 8×Cortex-A53	3	8	8*2.02
4	Персональный компьютер (ПК)	-	-	-	Intel core i5 5200U CPU @ 2.20 GHz	6	4	4*2.20
5	Raspberry Pi 4	-	Model B	-	Cortex-A72 (ARM v8)	6	4	4*1.5

Анализ загруженности процессоров при запуске программы осуществлялся с помощью приложения Cpu Monitor. Оно измеряет нагрузку на каждое ядро процессора и его температуру.

При проведении экспериментов по оценке производительности узлов кластера число шагов интегрирования было задано равным $N = 10^9$. Отметим, что приложение запускалось только на одном ядре в последовательном режиме, и на максимально возможном числе физических ядер в параллельном.

Результаты представлены в таблице 5.

Таблица 5. Оценка производительности узлов

Узел	Время выполнения (*сек*)	
	В последовательном режиме (*Tcore(i)*)	В параллельном режиме ($Tprc(i)$)
1	34,8	8,22
2	15,3	4,31
3	90,5	11,74
4	34,6	11,02
5	64.5	17.3

Мы видим, что и в последовательном, и в параллельном режиме быстрее работает современный смартфон. Он опережает даже не очень новый, но еще успешно работающий ПК. Полученный результат говорит о значительных вычислительных мощностях смартфонов и подтверждает возможность создания эффективных многопоточных приложений для смартфонов с использованием технологии OpenMP.

Для более детального подтверждения этого утверждения были проведены исследования зависимости времени расчета от числа работающих потоков для узлов 1, 2 и 3. На каждом смартфоне приложение запускалось с различным числом потоков – от 1 (последовательное исполнение) до 8 (число физических ядер). Графики зависимости времени выполнения задачи от числа потоков приведены на рисунке 3.

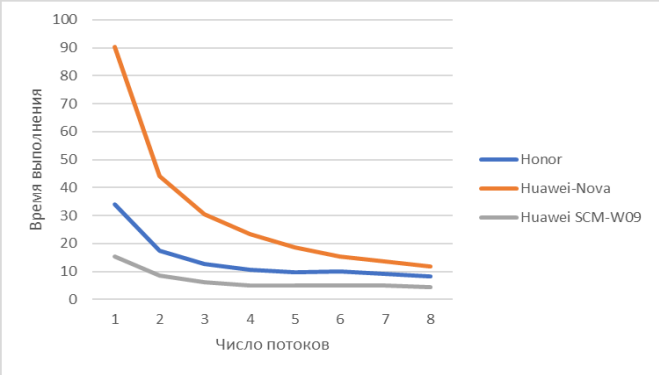

Рис. 3. Оценка масштабируемости приложения

Для большей наглядности представим графики зависимости ускорения от числа потоков.

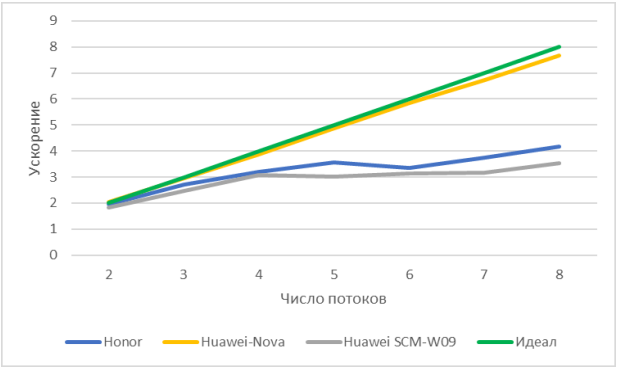

Рис. 4. График ускорения

В случае идеальной масштабируемости приложения зависимость ускорения вычислений от количества потоков будет линейной. На рисунке 4 это зеленый график. Полученные данные показывают, что только один смартфон демонстрирует близкую к идеальной эффективность работы. Это связано с тем, что в современных аппаратах используется система *big.LITTLE,* которая ограничивает возможности использования всех ядер на полную мощность.

После оценки производительности всех узлов был организован вычислительный кластер (Grid-система), структура которого представлена на рисунке 5.

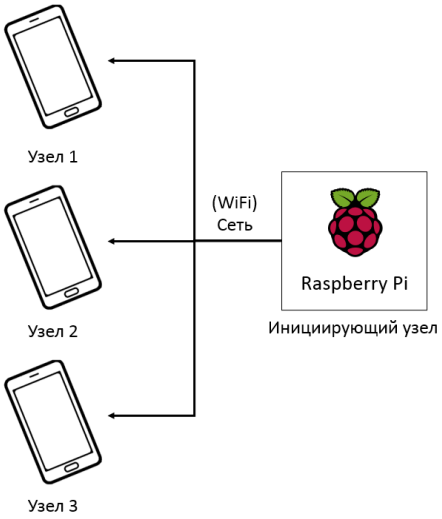

Рис. 5. Структура grid системы

К кластеру было подключено 3 смартфона, характеристики которых представлены в таблице 4. Первый узел кластера — это смартфон 1, второй – смартфон 2 и третий – смартфон 3. В качестве инициирующего устройства использовался микрокомпьютер Raspberry Pi, что позволило использовать все смартфоны непосредственно для вычислений. Реализованная система позволяет делать инициирующим любой узел, например тот, который обладает наименьшей производительностью.

Ведущий или инициирующий узел отправляет на остальные узлы задачу и все параметры, необходимые для распределения вычислительной нагрузки. По окончании вычислений все узлы отправляют полученные локальные результаты инициирующему узлу, который и формирует общее решение. В нашем случае это значение интеграла.

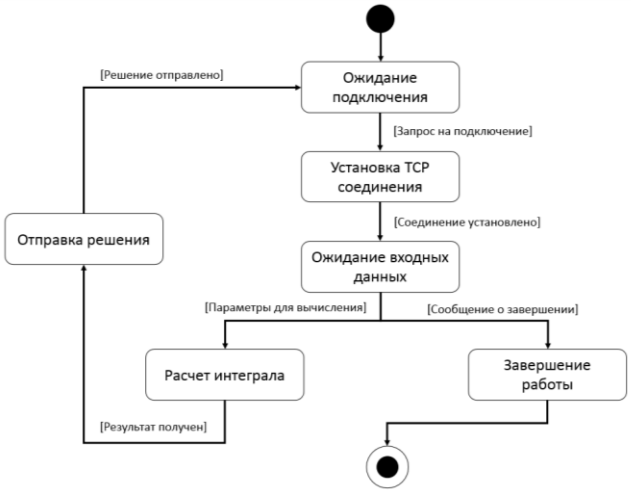

Рис. 6. Схема работы кластера

Обмен между узлами происходит по сети с помощью клиент-серверного приложения, использующего сокеты. Схема работы кластера указана на рисунке 6.

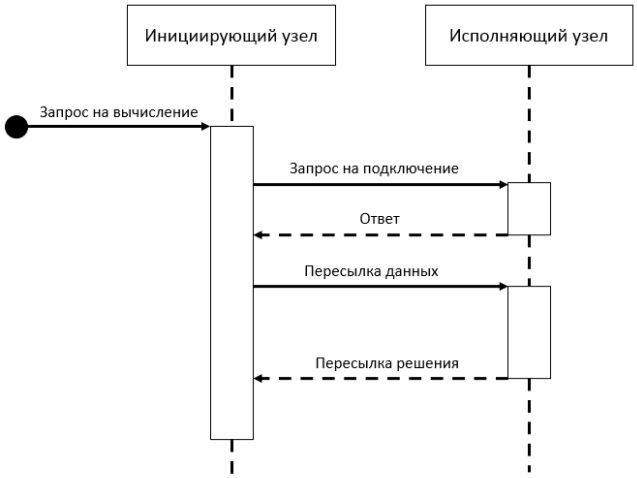

Рис. 7. Схема работы клиент-серверного приложения

Инициирующий узел, после установки соединения, отправляет в сеть пакет, содержащий текст программы и требуемые для ее запуска на узле параметры. В рассматриваемом примере это границы интегрирования и число интервалов. Далее на узле происходит распаковка пакета, компиляция программы и запуск с

заданными параметрами. В случае сбоя работы, инициирующему узлу возвращается сообщение об ошибке. При успешном завершении задачи программа отправляет рассчитанное значение интеграла (Рис. 7).

Для того чтобы получить максимальную производительность гетерогенного кластера, необходимо решить задачу балансировки нагрузки его узлов. Кластеры CoMD в подавляющем большинстве случаев также можно считать гетерогенными, поскольку смартфоны в их узлах обладают различными характеристиками. В проводимых исследованиях в качестве способа статической балансировки нагрузки был выбран метод, основанный на учёте реальной, а не пиковой производительности узлов [15].

Реализация статической балансировки основана на тестировании производительности каждого из узлов и измерения времени работы однопоточного приложения (*Tcore(i)*) и приложения, использующего максимальное число потоков ($Tprc(i)$) (Табл. 6).

Разнородность узлов отражают следующие параметры:

$$K_1(i) = \frac{\min(T_{core})}{T_{core}(i)} \quad (2)$$

$$K_2(i) = \frac{\min(T_{prc})}{T_{prc}(i)} \quad (3)$$

Доля нагрузки или интервала интегрирования в задаче (1), передаваемая соответствующему узлу, будет определяться как:

$Load_1(i) = \frac{K_1(i)}{\sum_1^3 K_1(i)}$ - для однопоточного приложения;

$Load_2(i) = \frac{K_2(i)}{\sum_1^3 K_2(i)}$ -для многопоточного приложения.

Результаты соответствующих расчетов приведены в таблице 6.

Таблица 6. Статическая балансировка нагрузки узлов

Узел	*Tcore(i) (сек)*	$K_1(i)$	$Load_1(i)$	*Tprc(i) (сек)*	$K_2(i)$	$Load_2(i)$
1	37,27	0,41	0,26	8,69	0,50	0,27
2	95,17	0,16	0,10	11,78	0,37	0,20
3	15,32	1,00	0,64	4,33	1,00	0,54

Распределив вычислительную нагрузку в соответствии с полученными значениями, мы получили результаты, представленные в таблице 7.

Таблица 7. Результаты эксперимента

Узел	Время вычисления *(сек)*
1	2,02
2	2,07
3	2,51
CoMD	2,51

Время решения задачи на кластере определяется по максимальному времени работы узла. В эксперименте оно составило 2,51 секунды. Хотя нам и не удалось добиться полностью сбалансированной работы всех узлов, кластер продемонстрировал результаты близкие к теоретическому максимуму. Время, потраченное на пересылку, компиляцию и запуск программы следует отнести к накладным затратам на организацию вычислительного процесса. В среднем, оно составляло 5,6 секунды.

Поведенные опыты доказывают, что, как и в мире живой природы, в компьютерной технике даже узлы с низкой производительностью, объединившись, могут обеспечить необходимую вычислительную мощность.

V. Заключение

Проведенные исследования подтверждают возможность построения вычислительных кластеров на базе смартфонов. Полученные результаты говорят и о высокой эффективности CoMD систем при реализации параллельных и распределенных приложений на устройствах с операционной системой Android. Накладные расходы на организацию вычислительного процесса невелики даже для относительно коротких задач.

Дальнейшие исследования будут направлены на оценку возможности использования пересылок между узлами CoMD для приложений с интенсивными коммуникациями.

Библиография

[1] M. Bauer, W. Bohme и W. Weitschat, «An Early Eocene gecko from Baltic amber and its implications for the evolution of gecko adhesion,» *The Zoological Society of London,* 2005

[2] D. Nenni и D. Dingee, The Origin and Evolution of ARM Processors In Our Devices, Danville: SemiWiki LLC, 2015.

[3] M. P. Singh и M. K. Jain, «Evolution of Processor Architecture in Mobile Phones,» *International Journal of Computer Applications,* т. 90, № 4, pp. 34-39, 2014.

[4] «Intel Experience Day 2021,» 26 11 2021. [В Интернете]. Available: https://www.youtube.com/watch?v=VETzK1HsVWE&t=28s. [Дата обращения: 11 12 2021].

[5] «Intel® Core i5-L16G7 Processor,» [В Интернете]. Available: https://ark.intel.com/content/www/us/en/ark/products/202777/intel-core-i5l16g7-processor-4m-cache-up-to-3-0ghz.html. [Дата обращения: 18 04 2022].

[6] «Процессор Intel® Core i9-12900HK,» [В Интернете]. Available: https://ark.intel.com/content/www/ru/ru/ark/products/132215/intel-core-i912900hk-processor-24m-cache-up-to-5-00-ghz.html. [Дата обращения: 18 04 2022].

[7] Yuki Sawada, Yusuke Arai, Kanemitsu Ootsu, Takashi Yokota, Takeshi Ohkawa, «An Android Cluster System Capable of Dynamic Node Reconfiguration,» *International Conference on Ubiquitous and Future Networks,* 2015.

[8] M. M. Juno, A. R. Bhangwar и A. A. Laghari, «Grids of Android Mobile Devices,» *ICICTT,* 2013.

[9] С. А. Балабаев, «28-я Всероссийская межвузовская научно-техническая конференция студентов и аспирантов «Микроэлектроника и информатика - 2021»,» в *Оценка*

вычислительных возможностей мобильных платформ, Зеленоград, 2021.

[10] Таненбаум Э., Хербет Б. 2. Процессы и потоки // Современные операционные системы. Санкт-Петербург: Питер, 2018. pp. 111-213.

[11] Лупин С.А., Посыпкин М.А. Среда программирования OpenMP // In: Технологии параллельного программирования. Москва: ИД "ФОРУМ" - ИНФРА-М, 2011. pp. 119-133.

[12] Г. И. Радченко, Распределенные вычислительные системы, Челябинск: Фотохудожник, 2021.

[13] Govindaraj, Parallel Programming in Raspberry Pi Cluster, Ithaca, 2016.

[14] S. Cass, «Home Clustering Made Easier,» *IEEE Spectrum,* № 11, pp. 16-19, 2021.

[15] Мин Тху Кхаинг, Аунг Тху и С. А. Лупин, «Оценка эффективности методов балансировки нагрузки в распределенных вычислительных системах,» *International Journal of Open Information Technologies,* № 11, 2021.

Computing cluster based on Android smartphones and Raspberry Pi microcomputers

Sergey Balabaev, Sergey Lupin, Roustiam Chakirov

Abstract— Currently, mobile devices such as smartphones and tablets are widely used. The computing power of their processors is comparable to the characteristics of personal computers, and this makes it possible to perform complex resource-intensive calculations on them. In the field of IoT, Raspberry Pi microcomputers are very popular. The paper explores the possibility of building a computing cluster from smartphones with Android OS and Raspberry Pi microcomputers. The article presents the results of testing the practical implementation of a distributed system, the nodes of which use heterogeneous smartphones. The computational capabilities of the system are determined and the applicability of static load balancing of nodes to improve the efficiency of computations is proved. The results obtained can be useful to a wide range of specialists.

Keywords— Computing cluster; Android, Raspberry Pi, OpenMP.

REFERENCES

[1] M. Bauer, W. Bohme i W. Weitschat, «An Early Eocene gecko from Baltic amber and its implications for the evolution of gecko adhesion,» The Zoological Society of London, 2005

[2] D. Nenni i D. Dingee, The Origin and Evolution of ARM Processors In Our Devices, Danville: SemiWiki LLC, 2015.

[3] M. P. Singh i M. K. Jain, «Evolution of Processor Architecture in Mobile Phones,» International Journal of Computer Applications, t. 90, # 4, pp. 34-39, 2014.

[4] «Intel Experience Day 2021,» 26 11 2021. [V Internete]. Available: https://www.youtube.com/watch?v=VETzK1HsVWE&t=28s. [Data obrashhenija: 11 12 2021].

[5] «Intel® Core i5-L16G7 Processor,» [V Internete]. Available: https://ark.intel.com/content/www/us/en/ark/products/202777/intel-core-i5l16g7-processor-4m-cache-up-to-3-0ghz.html. [Data obrashhenija: 18 04 2022].

[6] «Processor Intel® Core i9-12900HK,» [V Internete]. Available: https://ark.intel.com/content/www/ru/ru/ark/products/132215/intel-core-i912900hk-processor-24m-cache-up-to-5-00-ghz.html. [Data obrashhenija: 18 04 2022].

[7] Yuki Sawada, Yusuke Arai, Kanemitsu Ootsu, Takashi Yokota, Takeshi Ohkawa, «An Android Cluster System Capable of Dynamic Node Reconfiguration,» International Conference on Ubiquitous and Future Networks, 2015.

[8] M. M. Juno, A. R. Bhangwar i A. A. Laghari, «Grids of Android Mobile Devices,» ICICTT, 2013.

[9] S. A. Balabaev, «28-ja Vserossijskaja mezhvuzovskaja nauchno-tehnicheskaja konferencija studentov i aspirantov «Mikrojelektronika i informatika - 2021»,» v Ocenka vychislitel'nyh vozmozhnostej mobil'nyh platform, Zelenograd, 2021.

[10] Tanenbaum Je., Herbet B. 2. Processy i potoki // Sovremennye operacionnye sistemy. Sankt-Peterburg: Piter, 2018. pp. 111-213.

[11] Lupin S.A., Posypkin M.A. Sreda programmirovanija OpenMP // In: Tehnologii parallel'nogo programmirovanija. Moskva: ID "FORUM" - INFRA-M, 2011. pp. 119-133.

[12] G. I. Radchenko, Raspredelennye vychislitel'nye sistemy, Cheljabinsk: Fotohudozhnik, 2021.

[13] Govindaraj, Parallel Programming in Raspberry Pi Cluster, Ithaca, 2016.

[14] S. Cass, «Home Clustering Made Easier,» IEEE Spectrum, # 11, pp. 16-19, 2021.

[15] Min Thu Khaing, Aung Thu i S. A. Lupin, «Ocenka jeffektivnosti metodov balansirovki nagruzki v raspredelennyh vychislitel'nyh sistemah,» International Journal of Open Information Technologies, # 11, 2021.