Теоретические вопросы информатики, прикладной математики, компьютерных наук и когнитивно-информационных технологий

https://doi.org/10.25559/SITITO.019.202304.874-882
УДК 004.451.25

Метод управления частотой процессора на основе определения интенсивности обращений к памяти

Т. Н. Романова*, Е. А. Варламова

Оригинальная статья

ФГАОУ ВО «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет), г. Москва, Российская Федерация

Адрес: 105005, Российская Федерация, г. Москва, 2-я Бауманская ул., д. 5, стр. 1

* rtn@bmstu.ru

Аннотация

В настоящее время к мобильным устройствам предъявляются требования по высокой производительности и длительному времени работы без зарядки аккумулятора, то есть низкому энергопотреблению. Эти характеристики напрямую зависят от частоты процессора, поскольку на высокой частоте процессор способен выполнить больше инструкций за единицу времени, но при этом должен потратить больше энергии, и наоборот. В работе проведено исследование современных подходов к управлению частотой процессора в ядре операционной системы Linux с целью повышения производительности процессора и уменьшения энергопотребления мобильного устройства. Разработан новый метод управления частотой процессора, который базируется на решении дискретной двухкритериальной оптимизационной задачи. В качестве критериев оптимизации выступают две взаимозависимые характеристики: производительность и энергопотребление. В основе предлагаемого метода лежит идея анализа инструкций выполняемой задачи для выявления случаев, когда повышение частоты процессора неэффективно из-за частых обращений к памяти. Задача решается при наличии определенных ограничений: рассматривается только один процесс, выполняющийся на одном ядре процессора; не рассматриваются гетерогенные архитектуры процессоров; не учитывается аппаратная многопоточность ядер процессора, то есть каждое логическое ядро рассматривается как физическое. На основе предложенного метода создано программное обеспечение, которое было протестировано на 11 тестах из Rodinia Benchmark Suite. Этот набор тестов используется для оценки производительности компьютеров на различных типах задач, таких как вычисления общего назначения, обработка изображений, обработка сигналов и многие другие. Проведенное исследование показало, что использование предложенного подхода позволило уменьшить энергопотребление мобильного устройства в среднем на 7-15%, а производительность увеличить в среднем на 3-5% по сравнению с работой существующих утилит в ядре Linux.

Ключевые слова: управление частотой процессора, энергоэффективность системы, метрика количества обращений в память за инструкцию, конвейер процессора, ядро ОС Linux

Конфликт интересов: авторы заявляют об отсутствии конфликта интересов.

Для цитирования: Романова Т. Н., Варламова Е. А. Метод управления частотой процессора на основе определения интенсивности обращений к памяти // Современные информационные технологии и ИТ-образование. 2023. Т. 19, № 4. С. 874-882. https://doi.org/10.25559/SITITO.019.202304.874-882

Theoretical Questions of Computer Science, Computational Mathematics, Computer Science and Cognitive Information Technologies

A Method for Processor Frequency Management Based on Determining the Intensity of Memory Accesses

T. N. Romanova*, E. A. Varlamova

Original article

Bauman Moscow State Technical University, Moscow, Russian Federation

Address: 5, 2nd Baumanskaya St., building 2, Moscow 105005, Russian Federation

* rtn@bmstu.ru

Abstract

Nowadays, mobile devices are required to have high performance and low power consumption. These characteristics directly depend on the processor frequency, because at high frequency the processor is able to execute more instructions per cycle, but it has to spend more energy, and vice versa. In this paper, a study of modern approaches to processor frequency management in the Linux operating system kernel is conducted to improve processor performance and reduce power consumption of a mobile device. A new method of processor frequency controlling is developed, which is based on solving a discrete two-criteria optimization problem. Two interdependent characteristics act as optimization criteria: performance and power consumption. The proposed method is based on the idea of analyzing the instructions of the executed task to identify cases when increasing the processor frequency is inefficient due to frequent memory accesses. The problem is solved under certain restrictions: only one process running on one processor core is considered; heterogeneous processor architectures are not considered; hardware multithreading of processor cores is not taken into account, i.e. each logical core is considered as a physical core. Based on the proposed method, software was created and tested on 11 tests from Rodinia Benchmark Suite [6]. This test suite is used to evaluate the performance of computers on different types of tasks such as general purpose computing, image processing, signal processing and many others. The conducted study showed that the use of the proposed approach reduced the power consumption of the mobile device by 7-15% on average and increased the performance by 3-5% on average compared to the performance of existing utilities in the Linux kernel.

Keywords: processor frequency management, system energy efficiency, metric of the number of memory accesses per instruction, processor pipeline, Linux kernel

Conflict of interests: The authors declare no conflict of interest.

For citation: Romanova T.N., Varlamova E.A. A Method for Processor Frequency Management Based on Determining the Intensity of Memory Accesses. Modern Information Technologies and IT-Education. 2023;19(4):874-882. https://doi.org/10.25559/SITITO.019.202304.874-882

Введение

Высокая производительность и низкое энергопотребление системы являются важными и взаимозависимыми характеристиками для мобильных устройств, поэтому задача поиска баланса между ними является актуальной. Для решения указанной проблемы разработчики аппаратного обеспечения вводят всё большее количество ядер, уровней частоты ядер и их типов. Примером могут служить так называемые гетерогенные системы (популярные в области мобильных устройств), имеющие 2 типа ядер: Big и Little, которые отличаются по микроархитектуре и диапазоном доступных частот. Ядра типа Big используют для задач, требующих высокой производительности, а ядра типа Little, напротив, для фоновых задач. Таким образом, разнообразие доступных ядер и частот позволяет разработчикам операционных систем более гибко подстраивать конфигурацию под требования исполняемых задач.

Целью работы является создание метода управления частотой процессора на основе определения нагрузочных характеристик для поиска компромисса между минимизацией энергопотребления и максимизацией производительности. Для этого были решены следующие задачи:

1. проведён анализ и классификация подходов к управлению частотой процессора, являющихся частью ядра ОС Linux, и предложенных в статьях по соответствующей теме (раздел 1);
2. изложена концепция предлагаемого метода управления частотой (раздел 2);
3. разработан метод управления частотой процессора и изложены его особенности (раздел 3);
4. выполнено сравнение характеристик энергопотребления и производительности мобильного устройства при использовании программного модуля, созданного на основе предложенного метода управления частотой процессора, и использовании стандартных утилит, содержащимися в ядре ОС Linux (раздел 4).

1 Обзор существующих решений

В современных операционных системах (ОС) алгоритмы управления частотой процессора реализованы в соответствии со следующим общим принципом: чем больше задачи используют ядро процессора, тем выше частота процессора (алгоритмы из ядра Linux «schedutil», «conservative», «ondemand»)[1]. При этом использование процессора вычисляется на основании времени, проведенного на ядре процессора отдельными задачами. Однако время, проведённое на ядре процессора, не является показательной характеристикой использования процессора. Если задача интенсивно использует память, то она будет занимать значительное время на процессоре из-за значительной разницы в быстродействии памяти и процессора, но при этом фактически не будет использовать ресурсы процессора, так как процессор будет ожидать данные. Следовательно, повышение частоты процессора в таком случае неэффективно.

Также в современных ОС реализованы алгоритмы управления частотой, фиксирующие максимальную (алгоритм из ядра Linux «performance») или минимальную (алгоритм из ядра Linux «powersave») доступную частоты процессора. Такие алгоритмы, как показано в [1], неэффективны на нагрузках, наиболее часто исполняемых в пользовательских системах.

В данной работе предлагается метод управления частотой процессора, в основе которого лежит идея анализа инструкций выполняемых задач для выявления случаев, когда повышение частоты процессора неэффективно из-за частых обращений к памяти.

В работах ([2]-[23]) была использована эта идея. Однако предлагаемые в этих работах подходы имеют недостатки по сравнению с предлагаемым в данной работе. В работах ([2]-[8]) методы предполагают выставление частот без оценки энергоэффективности. В работах ([9]-[11]) рассматривается неполный набор частот процессора из всех доступных на процессоре и ограниченный набор приложений, для которых подход применим. Так, например, предлагаемый в работе ([11]) подход применим только к приложениям, связанным с просмотром веб-страниц. В работах ([11]-[23]) авторы оптимизируют только энергоэффективность, но не акцентируют внимание на производительности.

2 Концепция предлагаемого метода

Любое ядро процессора представляет собой конвейер, условно разделенный на две части: *frontend* и *backend*. *Frontend* часть отвечает за загрузку инструкций из кэша и их декодирование (также может включать модули предсказателя инструкций для повышения производительности). *Backend* часть отвечает за выполнение инструкций в различных устройствах (арифметико-логическое устройство, блок управления памятью, блок вычислений с плавающей точкой). Максимальная эффективность заполнения конвейера ядра процессора достигается, если на каждом цикле инструкции проходят обе части конвейера. Чем больше инструкций пройдено за цикл *(Instructions Per Cycles, IPC)*, тем выше эффективность заполнения конвейера ядра процессора. При этом могут возникнуть два типа ситуаций, при которых конвейер будет простаивать из-за ожидания ресурсов:

1. на *frontend* конвейера процессора не поступает инструкция, поскольку отсутствует в кэше;
2. на *backend* части конвейера ожидают выполнения несколько инструкций, требующих блок управления памятью, а блок управления памятью в это время ожидает загрузку требуемой памяти в кэш.

Такие ситуации неизбежны, так как требуемые ресурсы (инструкции и данные) не обязаны всегда находиться в кэше. Однако их негативное влияние на энергопотребление и эффективность заполнения конвейера ядра процессора можно контролировать с

[1] Brodowski D. Linux CPUF req Governors [Электронный ресурс] // Linux Kernel Organization, 2013. URL: https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt (дата обращения: 16.07.2023).

помощью уменьшения частоты процессора. Снижение частоты процессора приведёт к тому, что будет потрачено меньше циклов, проведенных в ожидании ресурсов, соответственно, будет наблюдаться повышение эффективности заполнения конвейера. Кроме того, снижение частоты процессора позволит снизить энергопотребление мобильного устройства.

Когда ядро процессора выполняет инструкцию загрузки данных из памяти, то сначала осуществляется поиск нужной информации в кэш-памяти. Если информация найдена в кэше, то она загружается в регистры процессора. Этот случай называют «кэш-попадание». Если нет – ядро ожидает, пока медленная оперативная память доставит всё необходимое. Этот случай называют «кэш-промах». Современные системы имеют несколько уровней кэш-памяти, которые имеют одну общую особенность: чем больше размер кэш-памяти, тем больше время доступа к ней.

В основе предлагаемого метода лежит следующая идея. Учитывая, что время доступа (количество циклов) к кэшу всех уровней небольшое по сравнению со временем доступа к оперативной памяти, то будем определять негативное влияние ожидания ресурсов в конвейере процессора на производительность и энергопотребление по метрике **MPI** *(Misses Per Instruction)* – **количества промахов в кэше последнего уровня за инструкцию.** Если процесс имеет большое количество промахов в кэше последнего уровня за инструкцию, то предлагается выставлять оптимальную частоту процессора для данного значения метрики MPI. Оптимальная частота процессора определяется как результат решения двухкритериальной дискретной оптимизационной задачи, при наличии следующих ограничений:

1. рассматривается только один процесс, выполняющийся на одном ядре процессора;
2. не рассматриваются гетерогенные архитектуры процессоров;
3. не учитывается аппаратная многопоточность ядер процессора, то есть каждое логическое ядро рассматривается как физическое.

3 Описание предлагаемого метода

Предлагаемый метод разделяется на 3 этапа:

1. *сбор данных*: составление таблицы выполнения нагрузок с разными значениями метрики MPI на всех частотах процессора. Таблица выполнения включает в себя следующие данные: энергопотребление мобильного устройства, время выполнения нагрузок, частоту процессора и значения метрик MPI;
2. *составление списка оптимальных частот*. На данном этапе на основе собранных данных (таблицы выполнения нагрузок) выбираются оптимальные частоты для каждой нагрузки, характеризующейся минимальным значением метрики MPI;
3. *управление частотой процессора* при выполнении приложения. На основании полученного списка оптимальных частот процессора при выполнении каждого приложения выставляется автоматически нужная частота мобильного устройства.

3.1 Сбор данных

Было сгенерировано N приложений с возрастающей последовательностью метрики MPI. Для создания нагрузок, варьирующих метрику MPI, использован следующий подход:

1. минимальное значение MPI может быть достигнуто в том случае, если ни одна инструкция не будет обращаться к памяти. Примером такой нагрузки может служить циклический инкремент регистровой переменной;
2. максимальное значение может быть достигнуто, если максимальное количество инструкций будут вызывать промах в кэше последнего уровня. Примером такой нагрузки может служить циклическое обращение к случайным ячейкам памяти, гарантированно отсутствующим в кэше;
3. смешивая итерации приложений из предыдущих двух пунктов, то есть, изменяя пропорцию итераций, обращающихся в память, можно получить нагрузки, характеризующиеся промежуточными значениями метрики MPI.

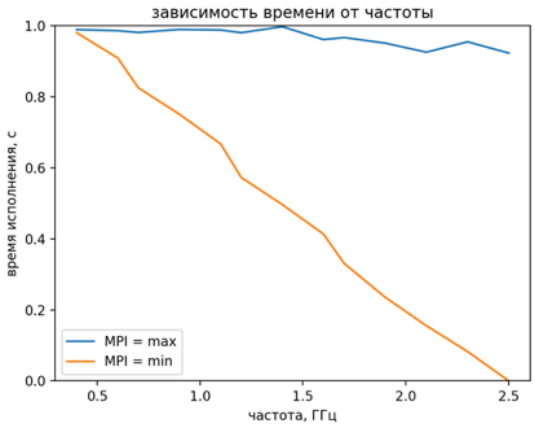
Для каждого из таких приложений были собраны данные об энергопотреблении и времени выполнения на всех доступных частотах процессора. Приложения были сгенерированы при условии, что на минимальной частоте процессора они выполнялись одинаковое время.

3.2 Составление списка оптимальных частот

Определим t_{max} как время выполнения любого приложения на минимальной частоте процессора (приложения составлены таким образом, что это время одинаково для всех).

Время исполнения приложения, характеризующегося минимальным MPI, при максимальной частоте процессора обозначим t_{min}.

На рисунке 1 представлен график зависимости времени выполнения приложений от частоты процессора.



Р и с. 1. Зависимость времени выполнения приложений от частоты процессора

F i g. 1. The dependence of application execution time on the processor frequency

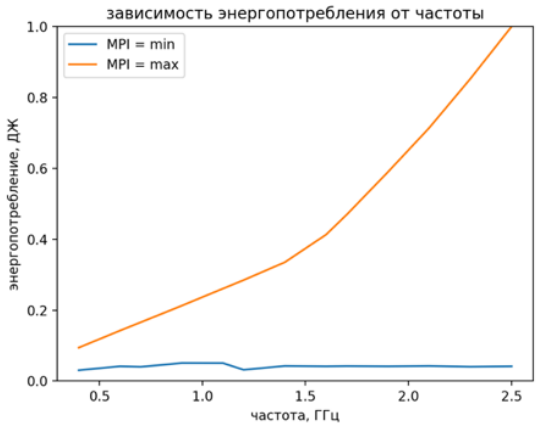
Источник: здесь и далее в статье все таблицы и рисунки составлены авторами.

Source: Hereinafter in this article all tables and figures were made by the authors.

На этом графике видно, что для всех исполняемых приложений с минимальным значением метрики MPI время выполнения будет уменьшаться с повышением частоты процессора. Для приложений с максимальным значением MPI время выполнения приложений практически не зависит от частоты.

Энергопотребление приложения, которое характеризуется максимальным значением метрики MPI при максимальной частоте процессора, обозначим E_{max}. Энергопотребление приложения, характеризующегося минимальным MPI на минимальной частоте процессора, обозначим E_{min}.

Энергопотребление по определению является произведением мощности на время выполнения. Мощность, как показано в ([24]), является суммой мощностей процессора и шины памяти *пропорционально ее использованию*. Там же было доказано, что мощность процессора пропорциональна частоте процессора. Таким образом, для приложения, характеризующегося минимальным MPI, энергопотребление постоянно при изменении частот (так как мощность пропорциональна изменению частоты, а время выполнения обратно пропорционально). На рисунке 2 показана зависимость энергопотребления от частоты.



Р и с. 2. Зависимость энергопотребления от частоты
F i g. 2. Power consumption as a function of frequency

На этом графике видно, что для приложений, характеризующихся максимальным значением MPI, энергопотребление увеличивается с увеличением частоты, поскольку мощность пропорциональна частоте, при этом время выполнения не меняется с увеличением частоты.

Математическая постановка двухкритериальной оптимизационной задачи

Пусть n – количество доступных частот процессора f_i, m – количество сгенерированных приложений a_j, где $i \in [1;n]$, $j \in [1; m]$. Обозначим $E_{i,j}'$ экспериментально полученные данные по энергопотреблению на частоте f_i для приложения a_j.

Для перехода от абсолютных значений параметров к безразмерным значениям осуществим нормирование энергопотребления $E_{i,j}$ приложения a_j на частоте f_i по следующей формуле: $E_{i,j} = \frac{(E_{i,j}' - E_{min})}{(E_{max} - E_{min})}$,

Аналогично нормируем время выполнения:

$$t_{i,j} = \frac{(t_{i,j}' - t_{min})}{(t_{max} - t_{min})},$$

где $t_{i,j}'$ экспериментально полученные данные времени исполнения приложения a_j на частоте f_i.

Для каждого приложения a_j рассмотрим двумерное пространство с началом координат в точке (0,0). По оси абсцисс откладываем время выполнения приложения a_j на всех частотах, а по оси ординат – энергопотребления этих приложений $E_{i,j}$. Получаем множество точек в двумерном пространстве, которые находятся на разных расстояниях от начала координат. Для каждого приложения a_j ведём функцию $R_j(f_i)$, которая будет отражать расстояние от начала координат (0,0) до точки в двумерном пространстве $(t_{i,j}, E_{i,j})$. Тогда можно записать: $R_j^2(f_i) = E_{i,j}^2 + t_{i,j}^2$

Функцию $R_j(f_i)$ выбираем в качестве целевой функции, зависящей от двух взаимозависимых критериев оптимизации $E_{i,j}$ и $t_{i,j}$, которую следует минимизировать, то есть: $R_j(f_i) \to min$.

Алгоритм решения оптимизационной задачи

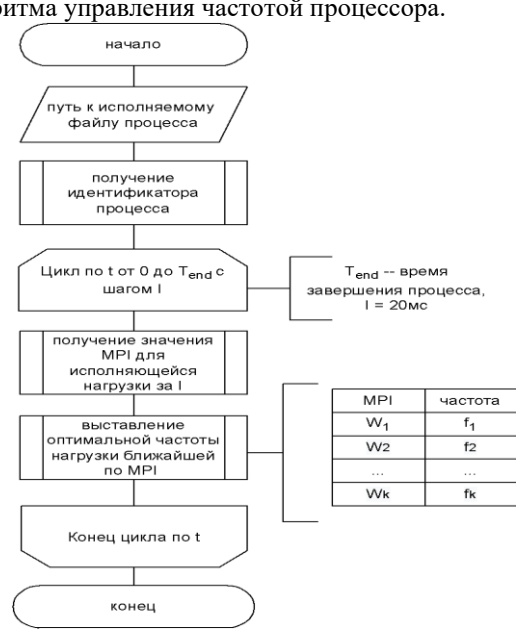
Вычисляем расстояния всех полученных экспериментально точек $(t_{i,j}, E_{i,j})$ до начала координат. Получаем матрицу $R(n \times m)$ расстояний всех точек пространства, имеющую m строк по количеству приложений a_j и n столбцов по количеству частот процессора f_i для каждого приложения a_j. Находим в каждой строке матрицы $R(n \times m)$ минимальные значения расстояний. Тогда для каждого приложения a_j, $j \in [1;m]$, которое характеризуется определённым значением метрики MPI, полученному минимальному значению расстояния соответствующей точки пространства до начала координат будет соответствовать некоторая частота процессора, которую и будем считать оптимальной частотой. Получим последовательность оптимальных частот:

$$F = \{f_1, f_2, \ldots, f_m\} \quad (1)$$

3.3 Управление частотой процессора

При выполнении целевого приложения предлагается каждые 20 миллисекунд определять значение метрики MPI и выставлять оптимальную частоту в соответствии с последовательностью оптимальных частот (1). Так как данные не могут быть собраны для всех значений метрики, то предлагается использовать модель масштабирования. Модель масштабирования выполняет поиск ближайшей по метрике MPI оптимальной частоты. Метрика MPI определяется с помощью сбора значений соответствующих счётчиков

процессора за интервал времени 20 мс: счётчика количества инструкций и счётчика промахов в кэше последнего уровня. На рисунке 3 представлена схема алгоритма управления частотой процессора.



Р и с. 3. Схема алгоритма управления частотой
F i g. 3. Diagram of the frequency control algorithm

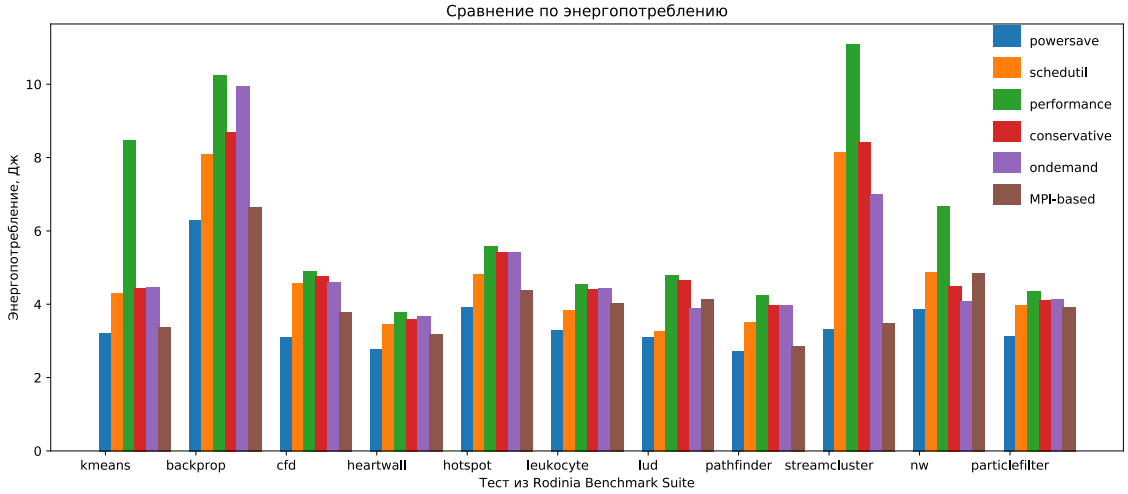
4 Полученные результаты

В ходе работы была достигнута поставленная цель: был разработан метод управления частотой процессора для решения задачи минимизации энергопотребления. Для этого были решены следующие задачи:

1. Была разработана математическая модель для двухкритериальной дискретной оптимизации частоты процессора;
2. Был разработан метод управления частотой процессора в соответствии с описанной концепцией и математической моделью;
3. На основе предложенного метода было создано программное обеспечение (ПО);
4. Было проведено исследование ПО с целью определения его эффективности.

Исследование проводилось на 11 тестах из *Rodinia Benchmark Suite* [25]. *Rodinia Benchmark Suite* – это набор тестов, который используется для оценки производительности компьютеров на различных типах задач, таких как вычисления общего назначения, обработка изображений, обработка сигналов и многие другие. Каждый тест включает в себя набор данных и реализацию алгоритма для решения задачи.

Результаты сравнения по энергопотреблению со стандартными подходами, входящими в ядро Linux, к управлению частотой представлены на рисунке 4.



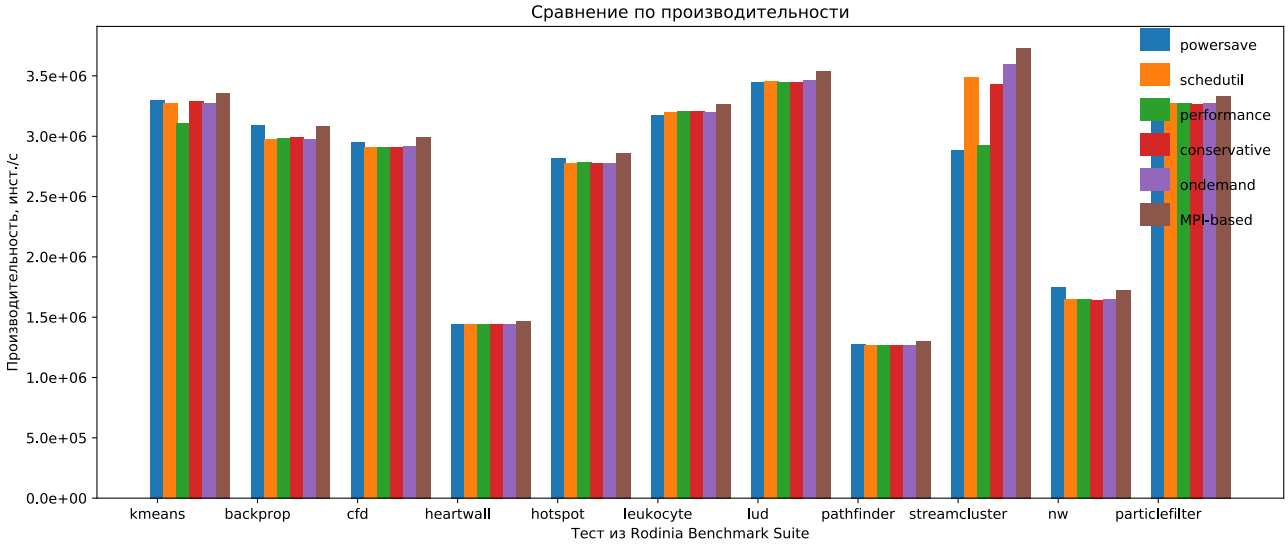
Р и с. 4. Сравнение по энергопотреблению
F i g. 4. Power consumption comparison

На рисунке 4 предлагаемый в данной работе метод обозначен как «MPI-based». Видно, что в большинстве сценариев тестового набора (отложенных по оси абсцисс) наблюдается выигрыш по энергопотреблению предлагаемого метода относительно стандартных. Наибольший выигрыш разработанного метода наблюдается на тесте «streamcluster», что ожидаемо, поскольку тест обрабатывает большое количество данных, а, значит, интенсивно обращается в память.

Результаты по производительности в сравнении со стандартными методами, входящими в ядро Linux, представлены на рисунке 5.

Т а б л и ц а 1. **Выигрыш предлагаемого метода относительно стандартных**
T a b l e 1. **The advantage of the proposed method over the standard ones**

Методы для сравнения	Выигрыш предлагаемого метода по энергопотреблению, %	Выигрыш предлагаемого метода по производительности, %
powersave	-0.76	3.96
schedutil	6.91	3.06
performance	7.87	5.44
conservative	14.57	3.19
ondemand	11.38	2.79



Р и с. 5. Сравнение по производительности
F i g. 5. Performance comparison

На рисунке 5 предлагаемый в данной работе метод обозначен как «MPI-based». Видно, что в большинстве сценариев тестового набора (отложенных по оси абсцисс) наблюдается выигрыш по производительности разработанного метода управления частотой относительно стандартных (входящих в ядро Linux).

В таблице 1 **приведены усредненные по всем тестам** результаты исследования **в виде выигрыша разработанного метода (в процентах) относительно существующих решений.** В качестве существующих решений для сравнения были выбраны методы из ядра Linux, описанные в разделе 1. Сравнение с методами из статей не представляется возможным, поскольку авторами статей не предоставлены ссылки на исходные коды программ и инструкции к запуску. Из таблицы видно, что выигрыш разработанного метода относительно стандартных по энергопотреблению составляет 7-15%, а по производительности – 3-5%.

Обсуждение и заключение

В ходе работы был проведен анализ существующих методов управления частотой процессора. Определены критерии для сравнения методов и проведено соответствующее сравнение. При описании концепции предлагаемого метода проведено теоретические обоснования, состоящие в том, что повышение частоты процессора приводит к уменьшению производительности и энергоэффективности системы при выполнении нагрузки, характеризующейся большим количеством обращений в память (за счёт потраченных впустую циклов). Эти теоретические обоснования были подтверждены результатами работы ПО, созданного на основе разработанного метода. В результате исследования ПО на тестах из *Rodinia Benchmark Suite* было выяснено, что разработанный метод управления частотой в среднем на 3-5% лучше стандартных подходов (входящих в ядро Linux) по производительности и в среднем на 7-15% по энергоэффективности.

References

1. Wenlei B., Changwan H., Sudheer C., Sriram K., Louis-Noël P., Fabrice R., Ponnuswamy S. Static and Dynamic Frequency Scaling on Multicore CPUs. *ACM Transactions on Architecture and Code Optimization*. 2009;13:1-26. https://doi.org/10.1145/3011017
2. Reddy B.K., Merrett G.V., Al-Hashimi B.M., Singh A.K. Online concurrent workload classification for multi-core energy management. In: 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE). Dresden, Germany: IEEE Press; 2018. p. 621-624. https://doi.org/10.23919/DATE.2018.8342084
3. Basireddy K.R., Singh A.K., Al-Hashimi B.M., Merrett G.V. AdaMD: Adaptive Mapping and DVFS for Energy-Efficient Heterogeneous Multicores. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.* 2020;39(10):2206-2217. https://doi.org/10.1109/TCAD.2019.2935065
4. Basireddy K.R., Wachter E.W., Al-Hashimi B.M., Merrett G. Workload-Aware Runtime Energy Management for HPC Systems. In: 2018 International Conference on High Performance Computing & Simulation (HPCS). Orleans, France: IEEE Press; 2018. p. 292-299. https://doi.org/10.1109/HPCS.2018.00057
5. Bogdanov E., Bozhnyuk A., Sartasov S., Granichin O. On Application of Simultaneous Perturbation Stochastic Approximation for Dynamic Voltage-Frequency Scaling in Android OS. In: 2021 7th International Conference on Event-Based Control, Communication, and Signal Processing (EBCCSP). Krakow, Poland: IEEE Press; 2021. p. 1-7. https://doi.org/10.1109/EBCCSP53293.2021.9502396
6. Reddy B. K., Singh A. K., Biswas D., Merrett G.V., Al-Hashimi B.M. Inter-Cluster Thread-to-Core Mapping and

DVFS on Heterogeneous Multi-Cores. *IEEE Transactions on Multi-Scale Computing Systems*. 2018;4(3):369-382. https://doi.org/10.1109/TMSCS.2017.2755619
7. Quan W., Pimentel A.D. A scenario-based run-time task mapping algorithm for MPSoCs. In: Proceedings of the 50th Annual Design Automation Conference (DAC '13). New York, NY, USA: Association for Computing Machinery; 2013. Vol. 131. p. 1-6. https://doi.org/10.1145/2463209.2488895
8. Van Craeynest K., Jaleel A., Eeckhout L., Narvaez P., Emer J. Scheduling heterogeneous multi-cores through Performance Impact Estimation (PIE). *ACM SIGARCH Computer Architecture News*. 2012;40(3):213-224. https://doi.org/10.1145/2366231.2337184
9. Gupta U., Patil C.A., Bhat G., Mishra P., Ogras U. DyPO: Dynamic Pareto-optimal configuration selection for heterogeneous MpSoCs. *ACM Transactions on Embedded Computing Systems*. 2017;16(5s):123. https://doi.org/10.1145/3126530
10. Shamsa E., Kanduri A., Liljeberg P., Rahmani A.M. Concurrent Application Bias Scheduling for Energy Efficiency of Heterogeneous Multi-Core Platforms. *IEEE Transactions on Computers*. 2022;71(4);743-755. https://doi.org/10.1109/TC.2021.3061558
11. Zhu Y., Reddi V.J. High-performance and energy-efficient mobile web browsing on big/little systems. In: 2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA). Shenzhen, China: IEEE Press; 2013. p. 13-24. https://doi.org/10.1109/HPCA.2013.6522303
12. Moolchandani D., Kumar A., Martínez J.F., Sarangi S.R. VisSched: An Auction-Based Scheduler for Vision Workloads on Heterogeneous Processors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.* 2020;39(11):4252-4265. https://doi.org/10.1109/TCAD.2020.3013076
13. Singh A.K., Dey S., McDonald-Maier K., Basireddy K.R., Merrett G.V., Al-Hashimi B.M. Dynamic Energy and Thermal Management of Multi-core Mobile Platforms: A Survey. *IEEE Design & Test.* 2020;37(5):25-33. https://doi.org/10.1109/MDAT.2020.2982629
14. Ranjbar B., Nguyen T.D.A., Ejlali A., Kumar A. Power-Aware Runtime Scheduler for Mixed-Criticality Systems on Multicore Platform. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.* 2021;40(10):2009-2023. https://doi.org/10.1109/TCAD.2020.3033374
15. Han S., Yun Y., Kim Y.H., Kang S. Proactive Scenario Characteristic-Aware Online Power Management on Mobile Systems. *IEEE Access*. 2020;8:69695-69711. https://doi.org/10.1109/ACCESS.2020.2986214
16. Siddesha K., Jayaramaiah G.V. Energy Efficient Greedy Scheduling of Tasks for DVFS Enabled Heterogeneous Multicore Processors. In: 2021 International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT). Bangalore, India: IEEE Press; 2021. p. 538-542. https://doi.org/10.1109/RTEICT52294.2021.9573873
17. Pelogeiko M., Sartasov S., Granichin O. On Stochastic Optimization for Smartphone CPU Energy Consumption Decrease. *Informatics and Automation*. 2023;22(5):1004-1033. https://doi.org/10.15622/ia.22.5.3
18. Lee J., Nam S., Park S. Energy-Efficient Control of Mobile Processors Based on LongShort-Term Memory. *IEEE Access.* 2019;7:80552-80560. https://doi.org/10.1109/ACCESS.2019.2923334
19. Song S., Kim J., Chung J.-M. Energy Consumption Minimization Control for Augmented Reality Applications based on Multi-core Smart Devices. In: 2019 IEEE International Conference on Consumer Electronics (ICCE). Las Vegas, NV, USA: IEEE Press; 2019. p. 1-4. https://doi.org/10.1109/ICCE.2019.8661917
20. Pricopi M., Muthukaruppan T.S., Venkataramani V., Mitra T., Vishin S. Power-performance modeling on asymmetric multi-cores. In: 2013 International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES). Montreal, QC, Canada: IEEE Press; 2013. p. 1-10. https://doi.org/10.1109/CASES.2013.6662519
21. Goraczko M., Liu J., Lymberopoulos D., Matic S., Priyantha B., Zhao F. Energy-optimal software partitioning in heterogeneous multiprocessor embedded systems. In: 2008 45th ACM/IEEE Design Automation Conference. Anaheim, CA: IEEE Press; 2008. p. 191-196. https://doi.org/10.1145/1391469.1391518
22. Liu G., Park J., Marculescu D. Dynamic thread mapping for high-performance, power-efficient heterogeneous many-core systems. In; 2013 IEEE 31st International Conference on Computer Design (ICCD). Asheville, NC, USA: IEEE Press; 2013. p. 54-61. https://doi.org/10.1109/ICCD.2013.6657025
23. Mukherjee A., Chantem T. An Integrated Energy Management Framework for Multiple Side-by-Side Applications on Smartphones. In: IEEE International Conference on Embedded Software and Systems (ICESS). Shanghai, China: IEEE Press; 2020. p. 1-8. https://doi.org/10.1109/ICESS49830.2020.9301538
24. Vaibhav S., Masha S. Joint frequency scaling of processor and DRAM. *The Journal of Supercomputing*. 2016;72(4):1549-1569. https://doi.org/10.1007/s11227-016-1680-4
25. Che S., et al. Rodinia: A benchmark suite for heterogeneous computing. In: Proceedings of the 2009 IEEE International Symposium on Workload Characterization (IISWC) (IISWC '09). IEEE Computer Society, USA; 2009. p. 44–54. https://doi.org/10.1109/IISWC.2009.5306797

Поступила 16.07.2023; одобрена после рецензирования 18.09.2023; принята к публикации 22.10.2023.

Submitted 16.07.2023; approved after reviewing 18.09.2023; accepted for publication 22.10.2023.

Об авторах:

Романова Татьяна Николаевна, доцент факультета информатики и систем управления, ФГАОУ ВО «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет) (105005, Российская Федерация, г. Москва, 2-я Бауманская ул., д. 5, стр. 1), кандидат физико-математических наук, доцент, **ORCID: https://orcid.org/0009-0001-9189-8180,** rtn@bmstu.ru

Варламова Екатерина Алексеевна, магистрант факультета информатики и систем управления, ФГАОУ ВО «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет) (105005, Российская Федерация, г. Москва, 2-я Бауманская ул., д. 5, стр. 1), **ORCID: https://orcid.org/0009-0003-9491-1891,** katy1781@inbox.ru

Все авторы прочитали и одобрили окончательный вариант рукописи.

About the authors:

Tatyana N. Romanova, Associate Professor of the Faculty of Computer Science and Control Systems, Bauman Moscow State Technical University (5, 2nd Baumanskaya St., building 2, Moscow 105005, Russian Federation), Cand. Sci. (Phys.-Math.), Associate Professor, **ORCID: https://orcid.org/0009-0001-9189-8180,** rtn@bmstu.ru

Ekaterina A. Varlamova, Master degree student of the Faculty of Computer Science and Control Systems, Bauman Moscow State Technical University (5, 2nd Baumanskaya St., building 2, Moscow 105005, Russian Federation), **ORCID: https://orcid.org/0009-0003-9491-1891,** katy1781@inbox.ru

All authors have read and approved the final manuscript.