УДК 004.45

РАЗРАБОТКА ПРОГРАММНОГО КОМПЛЕКСА ДЛЯ АНАЛИЗА ЭНЕРГОЭФФЕКТИВНОСТИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ[1]

А.В. Калачев, А.С. Карсаков, И.Б. Мееров, Я.А. Напыльникова, А.Ю. Овсюхно

В работе рассмотрена задача разработки энергоэффективного программного обеспечения. Основное внимание уделено программированию для мобильных устройств. Предложен новый программный инструмент для анализа энергоэффективности приложений для мобильных устройств. Инструмент позволяет собирать и обрабатывать экспериментальную информацию, характеризующую энергоэффективность приложений на используемой платформе. Приведено сравнение аналогичных инструментов. Описаны основная функциональность, метод использования и результаты применения на примере задачи матричного умножения. Проведен анализ энергоэффективности реализаций алгоритма с использованием набора команд SSE, а также технологий параллельного программирования OpenMP и Cilk Plus. Исследованы зависимость энергоэффективности от числа использованных потоков, количества кэш-промахов, количества переходов между C-State-состояниями процессора. Сформулированы выводы по результатам проведенного анализа. Приведены планы по дальнейшему развитию функциональности инструмента.

Ключевые слова: мобильные системы, оптимизация энергопотребления, инструменты для анализа энергоэффективности, метрики для оценки энергопотребления, C-State-состояния.

Введение

В настоящее время проблема оптимизации энергосбережения является достаточно актуальной в разных областях науки и техники. В связи с повсеместным применением информационных технологий (ИТ) энергетическая эффективность (энергоэффективность) стала приоритетным направлением и в ИТ-сфере. Так, увеличение производительности процессоров привело к нелинейному росту энергопотребления в рабочих станциях и серверах. Особенно сильно эта проблема коснулась центров обработки данных, где используется большое количество вычислительных устройств и сетевого оборудования. Так, в период с 2011 по 2012 год общее энергопотребление всех таких центров в мире существенно возросло.

С другой стороны производители мобильных устройств столкнулись с существенным падением времени автономной работы при расширении функциональности. Если первые модели процессоров потребляли лишь доли Ватта, то современные могут требовать до 130 и более Ватт. К сожалению, емкость батарей увеличивается медленнее и не успевает удовлетворить существующие потребности пользователей.

Таким образом, основная задача, которая ставится при конструировании современных процессоров, — достижение не максимально возможной производительности, а вы-

[1] Статья рекомендована к публикации программным комитетом научной конференции «Высокопроизводительные параллельные вычисления на кластерных системах» (Нижний Новгород, 2012).

сокого уровня производительности при обеспечении энергопотребления на приемлемом уровне. Необходимость внедрения энергосберегающих технологий продиктована не только желанием сэкономить ресурсы, но и невозможностью обеспечить приемлемое время автономной работы мобильных устройств. Сейчас это является одной из движущих сил совершенствования архитектур и технологий, как мобильных процессоров, так и мощных суперкомпьютеров и серверов.

Полностью справиться с проблемой, используя только аппаратные решения (увеличение емкости батареи, оптимизация устройства процессора и др.), не представляется возможным, поэтому необходимо использовать и программные решения.

В данной работе проводится обзор текущих достижений в данной области, проведен обзор существующих программных инструментов для анализа энергоэффективности и предлагается новый инструмент для сбора и анализа данных об энергопотреблении в программах, описывается подход к организации инструмента, приводятся примеры его использования на модельных задачах, анализируются результаты экспериментов, формулируются выводы и планы по дальнейшему развитию.

1. Текущее состояние предметной области

Метрики. Вопрос о введении согласованной системы метрик для оценки энергопотребления активно изучается в литературе. Для оценки энергопотребления могут использоваться классические физические характеристики: Ватты, Ватт*часы и Джоули. В работе [1] предлагается ввести новую физическую величину: MIPJ (millions of instructions per joule), миллион операций на джоуль. В некоторых других источниках эту величину называют Gflop/s per Watt, что с точки зрения физики является эквивалентным. Чем больше эта величина, тем более энергоэффективной является вычислительная система. Долгое время эта метрика использовалась для сравнения энергоэффективности вычислительной техники. Аналогично списку Top500 самых высокопроизводительных систем в мире[2] был создан список наиболее энергоэффективных систем – Green500[3]. В 2010 была предложена альтернативная метрика [2] – FTTSE (*f(timetosolution) * energy*), одновременно учитывающая время работы программы (*f(timetosolution)*) и потраченную энергию (*energy*). Вопрос формирования и применения системы метрик для оценки энергопотребления представляет большой практический интерес.

Оптимизация энергопотребления. Оптимизация энергопотребления в мобильных устройствах может выполняться как на аппаратном, так и на программном уровне. Развитие вопроса протекало в двух основных направлениях: Dynamic Power Management (DPM) [3–6] и Dynamic Voltage Scaling (DVS) [6–8]. Суть первого направления заключается в том, чтобы операции, не требующие всей вычислительной мощи центрального процессора, выполнялись в более энергоэффективных состояниях центрального процессора. Центральный процессор

[2] Список самых высокопроизводительных систем – http://www.top500.org

[3] Список самых энергоэффективных систем – http://www.green500.org

может переходить в состояния, при которых производительность понижена, что в свою очередь уменьшает энергопотребление. Основная сложность заключается в том, чтобы правильно предсказывать момент перехода в режим экономии энергии. Направление Dynamic Voltage Scalling предлагает другой поход, суть которого заключается в динамическом изменении питания, подаваемого на разные компоненты аппаратного обеспечения (такие как центральный процессор, оперативная память и др.) в зависимости от потребностей исполняемого в данный момент программного обеспечения.

На основе технологий DPM и DVS в современных процессорах реализованы P-state- и C-state-состояния процессора. P-state-состояния – активные режимы работы процессора, характеризуемые комбинацией тактовой частоты и рабочего напряжения. Для различных моделей процессоров эти режимы могут отличаться. При снижении вычислительной нагрузки процессор может регулировать свою тактовую частоту и напряжения питания. Этот процесс незначительно сказывается на производительности, но дает ощутимый выигрыш в энергоэффективности. В отличие от P-state, C-State-состояния показывают степень засыпания процессора во время «простоя», т.е. когда он не исполняет инструкции. У каждого состояния есть номер, чем он больше, тем ниже энергопотребление процессора, но тем больше времени и энергии понадобится для перехода в активное состояние. Далее приведена краткая характеристика основных C-state-состояний [9]:

- C0 – активное состояние процессора, в котором он выполняет вычисления и исполняет инструкции.
- C1 – состояние ожидания, процессор не исполняет инструкции, но готов с минимальной задержкой приступить к их исполнению. В данном режиме процессор сохраняет состояние системного кэша.
- C2 – данное состояние ожидания процессора более энергоэффективное, чем C1, реализованное за счет аппаратных возможностей процессора.
- C3 – состояние ожидания процессора, при котором выключаются тактовый генератор и механизмы поддержания когерентности кэшей.

В некоторых процессорах могут быть реализованы более энергоэффективные состояния ожидания, вплоть до C6.

В имеющихся статьях по написанию энергоэффективного кода, существуют рекомендации снизить количество переходов из одного C-state-состояния в другое, т.к. эти переходы зачастую ведут к потере энергии [10].

В настоящее время указанные подходы достаточно хорошо изучены и активно развиваются с появлением новой вычислительной техники. Вместе с тем, все больший интерес вызывает оптимизация на уровне прикладных программ – написание энергоэффективного программного обеспечения [5, 11–15].

2. Оценка энергоэффективности программного обеспечения

Целью данной работы является разработка методов и программных инструментов для анализа энергоэффективности программного обеспечения. Для ее достижения необходимо определить набор величин, основываясь на которых можно будет оценивать энергоэффективность. Нами были выбраны следующие метрики:

1. Текущая потребляемая мощность (Вт).
2. Процент времени, в течение которого процессор находится в том или ином C-state-состоянии.
3. Кол-во переходов между разными C-state-состояниями (переходов/с).
4. Процент времени загруженности процессора.
5. Процент времени бездействия (простоя) процессора.
6. Существуют множество инструментов, которые позволяют определять данные метрики.

Сравнительный анализ таких инструментов представлен в табл. 1. Для сравнения были взяты следующие инструменты. Power Checker [16], Power Informer [17] продукты корпорации Intel, Joulemeter [18], Perfmon [19] корпорации Microsoft и продукт с открытым кодом Powertop [20]. В качестве критериев сравнения были использованы основные функции, необходимые для оценки энергоэффективности программ.

Все инструменты обладают теми или иными достоинствами и недостатками. Ни один из них не реализует полный набор функций, перечисленных в первом столбце табл. 1. Особо стоит обратить внимание на возможность инструмента автоматически следить за условиями проведения экспериментов, т.к. на энергопотребление приложения влияет большое количество факторов: загруженность центрального процессора, текущий уровень потребляемой мощности системы и отельных ее компонент, работы системы охлаждения мобильного устройства и т.д. Исследователю необходимо учитывать эти факторы для того, чтобы сделать корректные выводы из результатов эксперимента. К сожалению, перечисленные в табл. 1 инструменты не предоставляют такой возможности.

В связи с вышесказанным возникла необходимость разработать новую систему для анализа энергоэффективности программного обеспечения.

Были предъявлены следующие требования к системе:

• **Кроссплатформенность**. Предусмотреть поддержку ОС WindowsXP, Windows 7 и Linux, процессоров на базе архитектуры x86. Обеспечить возможность портирования на мобильные ОС (Android, iOS, WindowsPhone) и другие архитектуры процессоров (ARM).

• **Нагрузка.** Система должна добавлять минимальную погрешность в проводимые эксперименты (не более 5%).

Таблица 1

Инструменты для анализа энергоэффективности

	Intel Power Checker	**Intel Power Informer**	**Joulemeter**	**Perfmon**	**Powertop**
ОС	Windows	Windows	Windows	Windows	Linux
Измерение текущей потребляемой мощности	Да	Да	Да	Да	Да
Анализ C-State-состояний	Да	Да	Нет	Да	Да
Сбор информация о центральном процессоре	Нет	Да	Нет	Да	Да
Отображение результатов в реальном времени	Нет	Нет	Да	Да	Да
Проведение серий экспериментов	Нет	Нет	Нет	Нет	Нет
Слежение за условиями проведения эксперимента	Нет	Нет	Нет	Нет	Нет
Период обновления данных (с)	1,0	неизвестно	1,0	1,0	15,0
Вывод результатов в файл	Да	Да	Нет	Да	Нет
Возможность хранения результатов	Нет	Нет	Нет	Нет	Нет

- **Автоматизация запуска экспериментов**. Система должна иметь открытую архитектуру, поддерживать расширение списка используемых параметров, функцию калибровки. Основываясь на полученных параметрах, система должна следить за корректностью условий проведения экспериментов. В режиме ручного эксперимента приложение должно позволять настраивать параметры эксперимента. В режиме серийного эксперимента приложение должно работать, используя конфигурационный файл.
- **Формат хранения данных**. Формат хранения должен быть расширяемым, самодостаточным, ориентированным на хранение параметров и результатов экспериментов, поддерживать быстрые операции чтения/записи в файл.
- **Источник данных**. Исследовательский комплекс должен иметь возможность получать информацию об энергопотреблении из различных источников (измерительный прибор, API ОС, драйвер батареи и др.) Каждый источник дан-

ных должен быть взаимозаменяемым и не влиять на другую функциональность системы.

- **Пользовательский интерфейс**. Взаимодействие с пользователем должно происходить с использованием графического интерфейса. Должна поддерживаться следующая функциональность: задание параметров эксперимента (графическая форма или выбор конфигурационного файла на файловой системе), работа с очередью экспериментов, запуск экспериментов, сбор и представление результатов экспериментов (графический, конвертация в текстовый формат).
- **Представление данных**. Пользователь должен иметь возможность загружать результаты проведенных экспериментов. Результаты должны быть представлены в удобном и наглядном виде с использованием графиков, таблиц и гистограмм.

3. Архитектура системы

Для минимизации накладных расходов при сборе метрик для разработки инструмента был использован язык программирования C++. Для обеспечения кроссплатформенности реализация пользовательского интерфейса была выполнена с помощью QT framework. Использование объектного подхода позволило предусмотреть возможность расширения системы как с точки зрения поддержки новых метрик, так и с точки зрения подключения новых измерителей (в текущей версии поддерживается Perfmon).

Архитектура системы представлена на рис. 1.

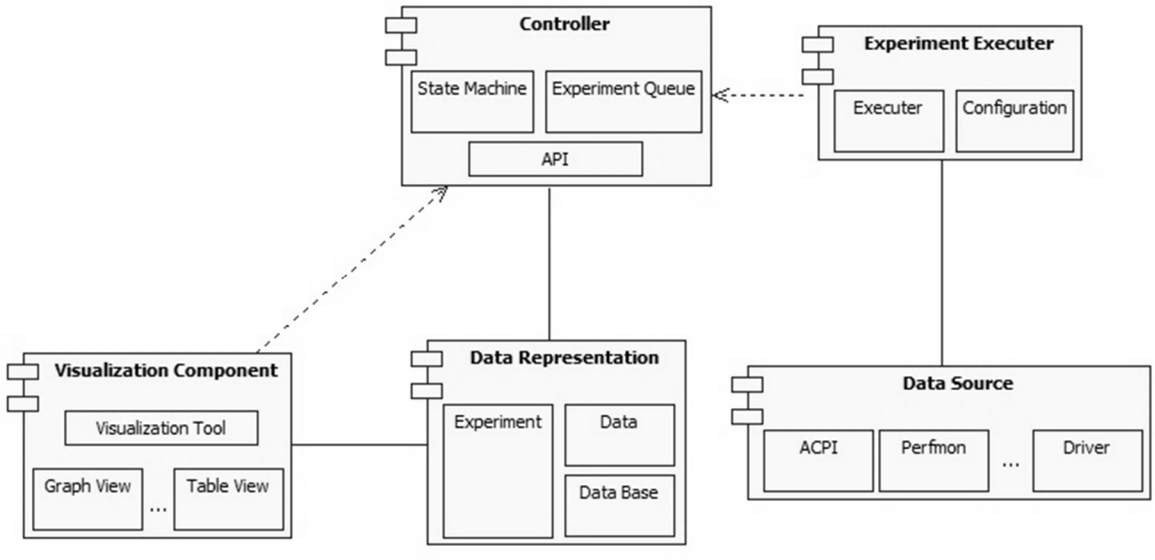

Рис. 1. Архитектура системы

Центральным управляющим модулем системы является *Controller*. Данная компонента предоставляет API, посредством которого другие компоненты системы могут взаимодействовать с внутренней машиной состояний и очередью экс-

периментов. Также данный компонент имеет набор функций, с помощью которых другие компоненты могут получить доступ к модулю *DataRepresentation*, который отвечает за программное представление данных об эксперименте. Модуль *ExperimentExecuter* реализуют всю логику, связанную с проведение экспериментов и сбором метрик. Для получения необработанных данных используются различные источники данных, реализации которых представлены в модуле *DataSource*. Все инструменты визуализации, а также компонента предварительной обработки данных находятся в модуле *Visualization*.

4. Результаты проведенных экспериментов

Использовалось следующее программно-аппаратное окружение:

- Процессор: Mobile Quad Core Intel Core i7 2630QM @ 2GHz (2.9 GHz Turbo Boost).
- Кэш процессора: L1 – 32 KB, L2 – 256 KB, L3 – 6MB.
- Оперативная память: 4GB, DDR3 1333 MHz.
- Операционная система: Windows 7 Ultimate (x64).

В первой серии экспериментов (рис. 2) проводится сравнение потребляемой мощности при решении задачи матричного умножения в следующих конфигурациях: последовательная версия, версии с использованием SSE, OpenMP и Cilk Plus. Здесь и далее за основу принят классический алгоритм матричного умножения, осуществляющий вычисление элементов по определению.

На рис. 2a, 2b, 2c слева, по оси абсцисс отложено время в секундах, по оси ординат текущая потребляемая мощность, измеряемая в мВт. Таким образом, данная серия графиков отображает количество потребляемой энергии в каждый момент времени работы алгоритма. Заметим, что пиковое значение потребляемой мощности при работе параллельных версий больше, чем у последовательной версии (фиолетовый). Так, при размерности матриц 800 на 800 элементов (см. рис. 2a) пиковая потребляемая мощность у реализаций на OpenMP (красный) и Cilk Plus (синий) на 35% больше. При увеличении размерности задачи, например, до 1600 на 1600 элементов (см. рис. 2c) разница составляет уже 45 %. Данное явление объясняется тем, что параллельные реализации задействуют большие вычислительные ресурсы системы. Этим же можно объяснить, что время работы параллельных реализаций меньше по сравнению с последовательной. Действительно, при небольшой размерности задачи (см. рис. 2a) время работы параллельных реализаций в 2 раза меньше, чем у последовательной, но при увеличении размерности задачи (см. рис. 2c) параллельная версия работает более чем в 3 раза быстрее. Реализация с использованием SSE (оранжевый) задействует дополнительные возможности процессора, что позволяет решить задачу с минимальным временем и минимальной пиковой потребляемой мощностью.

Чтобы сделать выводы об энергоэффективности, проинтегрируем полученные результаты по времени. На гистограммах (см. рис. 2a, 2b, 2c, справа) отражено суммарное количество энергии, потраченное на решение задачи. Несмотря на то, что пиковая потребляемая мощность в параллельных реализациях больше,

чем в последовательной реализации, энергия, потраченная на решение задачи меньше. Если для небольших задач (см. рис. 2a, справа) разница составляет 20%, то для задач большей размерности (см. рис. 2c, справа) параллельным реализациям требуется в 2 раза меньше энергии, чем последовательной. Отметим, что версия на OpenMP (красный) и версия на Cilk Plus (синий) показывают примерно одинаковые результаты. Версия на SSE для этой же задачи (см. рис. 2c) потребляет в 6 раз меньше энергии, чем последовательная.

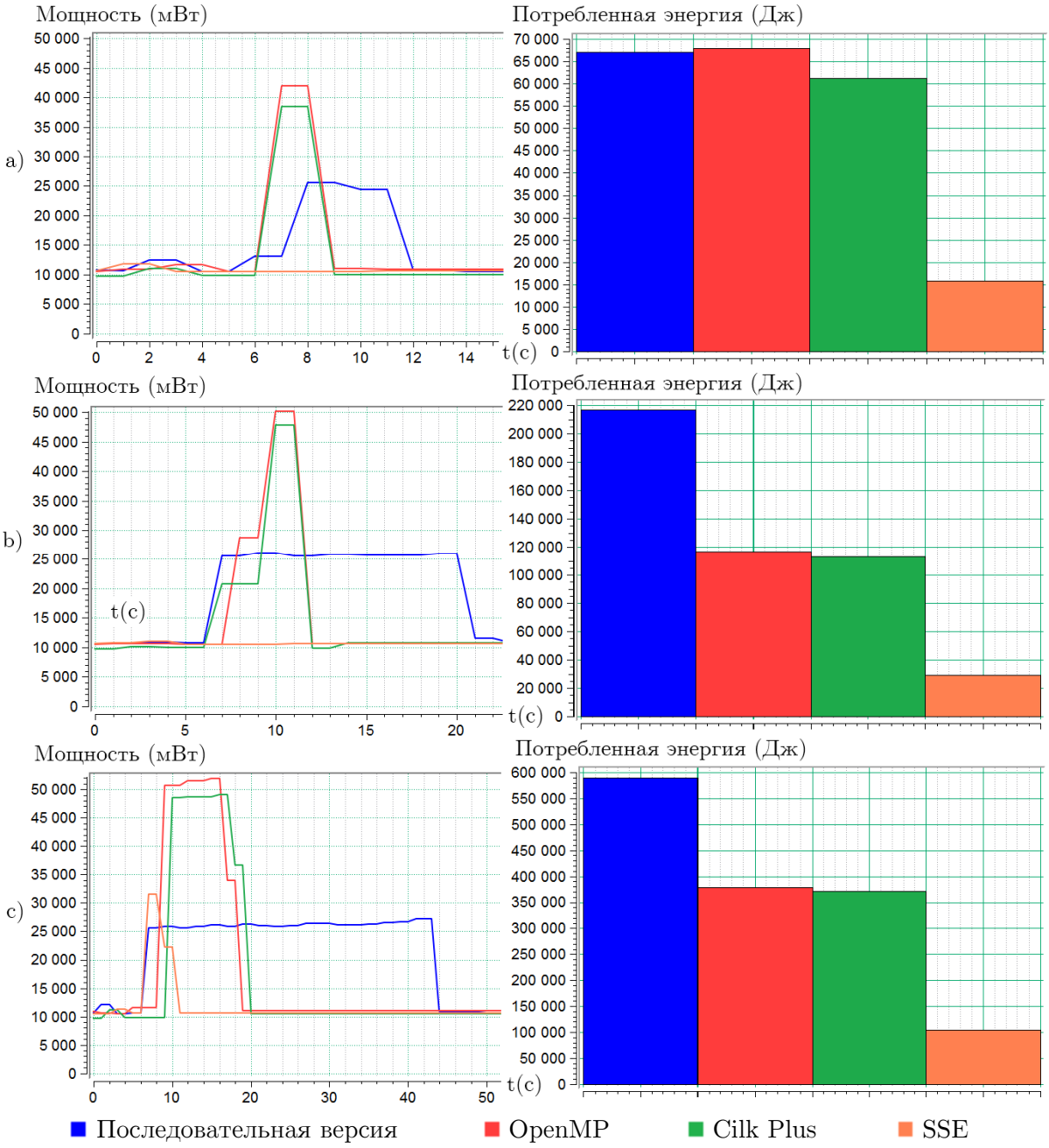

Рис. 2. Сравнение энергоэффективности реализаций матричного умножения при следующих размерах матриц: a) 800*800 b) 1200*1200 c) 1800*1800. *Слева:* измерение текущей потребляемой мощности во время эксперимента. *Справа:* суммарная потребленная энергия каждой из реализаций

Таким образом, параллельные реализации являются более энергоэффективными, чем последовательная реализация, несмотря на то, что пиковая потребляемая мощность у параллельных версий больше. Причем с ростом размеров матриц увеличивается и разница в количестве энергии, потребляемой последовательной и параллельными версиями.

Вторая серия экспериментов (рис. 3) иллюстрирует, как использование различного числа потоков при решении задачи влияет на энергоэффективность. Для анализа использовалась реализация матричного умножения с использованием технологии OpenMP. Серия экспериментов подразумевала запуск программы в одно- (синий), двух- (красный), четырех- (оранжевый) и восьмипоточной (фиолетовый) конфигурации. Отметим, что тестовая система имеет четыре физических ядра.

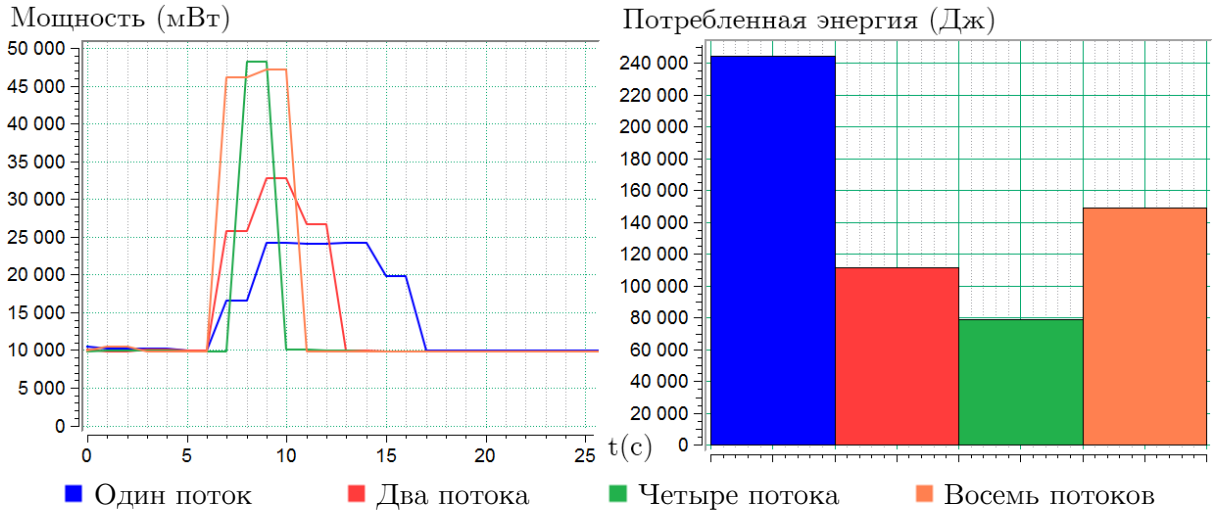

Рис. 3. Сравнение энергоэффективности параллельной реализации матричного умножения при различном числе потоков. *Слева:* измерение текущей потребляемой мощности во время эксперимента. *Справа:* суммарная потребленная энергия при разном числе потоков

Из результатов экспериментов видно, что при увеличении количества потоков время работы алгоритма уменьшается, однако потребляемая мощность в каждый момент времени увеличивается (см. рис. 3, слева). Это объясняется тем, что программа использует для решения задачи все доступные ей ресурсы центрального процессора. Архитектура современных процессоров устроена таким образом, что распределение работы между ядрами процессора является более энергоэффективным, чем выполнение этой работы при использовании одного ядра. Данный эффект подтверждается гистограммой (см. рис. 3, справа). Отметим, что в случае, когда число потоков превышает число ядер процессора (восьмипоточная конфигурация), наблюдается увеличение потребляемой энергии из-за дополнительных накладных расходов. Таким образом, для каждой вычислительной системы можно подобрать оптимальное, с точки зрения энергоэффективности, количество потоков для решения задачи.

Третья серия экспериментов (рис. 4) демонстрирует влияние количества кэш-промахов на энергоэффективность программы. В данном эксперименте для уменьшения количества кэш-промахов правая матрица была предварительно транспонирована.

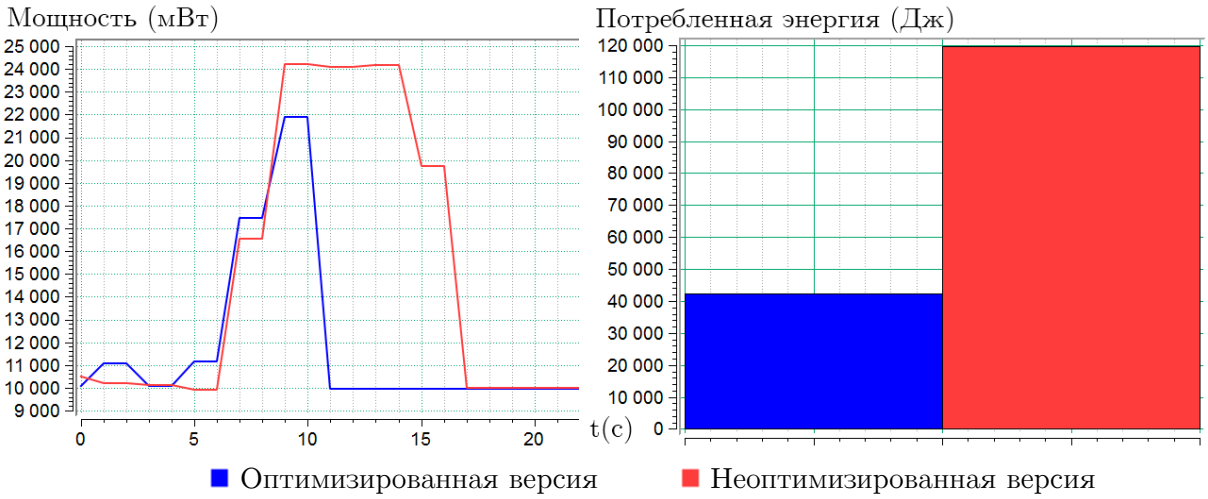

Рис. 4. Анализ влияния количества кэш-промахов на энергопотребление на примере матричного умножения. *Слева:* измерение текущей потребляемой мощности во время эксперимента. *Справа:* суммарная потребленная энергия каждой из реализаций

Как и ожидалось, эксперимент показывает сокращение времени работы программы более чем на 40% (см. рис. 4, слева), что отражается в соответствующем уменьшении суммарной потраченной энергии (см. рис. 4, справа). Однако суммарная потраченная энергия уменьшается более чем на 60%. Дополнительное уменьшение потраченной энергии на 20% объясняется тем, что в оптимизированной версии алгоритма из-за уменьшения количества кэш-промахов понижается нагрузка на аппаратное обеспечение, в частности, на оперативную память, что, в свою очередь, снижает общие затраты энергии.

Четвертая серия экспериментов (рис. 5) иллюстрирует влияние количества переходов между C-state-состояниями процессора на энергопотребление. Рассмотрим задачу, состоящую из нескольких независимых частей, которые можно запускать либо непрерывно друг за другом (красный), либо с некоторыми паузами (синий), что приводит к увеличению числа C-State-переходов.

В результате видим, что время обработки данных примерно одинаковое, но в случае, когда задачи запускаются непрерывно друг за другом, энергопотребление уменьшается приблизительно на 15% (см. рис. 5, справа). Это показывает, что увеличение числа переходов между C-state-состояниями процессора приводит к росту энергопотребления. Таким образом, группировка при обработке данных может привести к существенному выигрышу в энергоэффективности приложения.

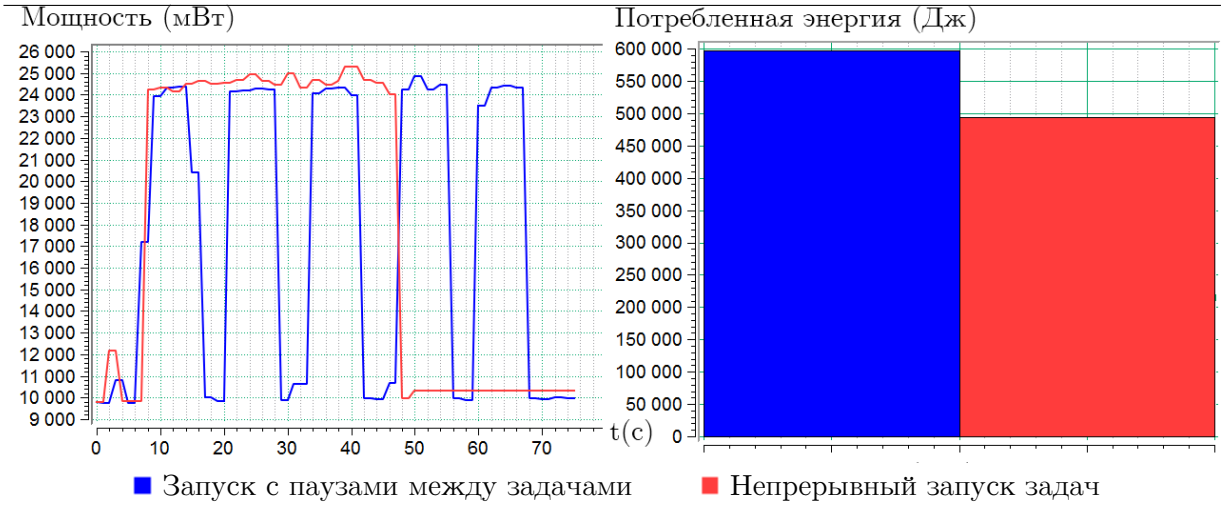

Рис. 5. Анализ влияния количества переходов между C-State-состояниями на энергопотребление на примере матричного умножения. *Слева:* измерение текущей потребляемой мощности во время эксперимента. *Справа:* суммарная потребленная энергия каждой из реализаций

Заключение

В работе представлен инструмент, предназначенный для помощи разработчику в оптимизации энергоэффективности программного обеспечения. Инструмент предоставляет широкий спектр возможностей для запуска экспериментов, сбора необходимых данных, хранения, обработки и визуализации результатов.

Работоспособность системы продемонстрирована на примере анализа энергоэффективности разных программных реализаций алгоритма матричного умножения. Проведены следующие эксперименты:

1. Сравнение энергоэффективности реализаций, выполненных с использованием набора команд SSE и технологий параллельного программирования OpenMP и Cilk Plus. В ходе эксперимента было показано, что при анализе энергоэффективности нужно обращать внимание не только на потребляемую мощность в каждый момент времени работы алгоритма, но и на общее время работы алгоритма.
2. Исследование влияния числа потоков в параллельной реализации на энергоэффективность. Было продемонстрировано, что можно подобрать оптимальное, с точки зрения энергопотребления, количество потоков для решения задачи.
3. Анализ влияния количества кэш-промахов на энергопотребление. Результаты экспериментов продемонстрировали, что на энергоэффективность помимо использования центрального процессора также влияет и использование других компонентов вычислительной системы, например, оперативной памяти.

4. Анализ влияния количества переходов между C-State-состояниями на энергопотребление. Эксперимент показал, что для достижение максимальной энергоэффективности разработчику необходимо уметь работать с C-State-состояниями центрального процессора.

Архитектура системы обеспечивает ее расширяемость (добавление новых метрик, источников данных, способов визуализаций и др.). Расширение функциональности системы, всесторонняя апробация, выработка подходов к оптимизации энергопотребления являются темами дальнейшей исследований.

Работа выполнена в лаборатории Intel-ННГУ «Информационные технологии» (ITLab).

Литература

1. Scheduling for Reduced CPU Energy / M. Weiser, B. Welch, A. Demers et al. // Proceedings of the 1st USENICS Symposium on Operating Systems Design and Implementation (Monterey, CA). – Nov. 1994. – P. 13–23.
2. Bekas, C. A new energy aware performance metric / C. Bekas, A. Curioni // Computer Science – R&D. – 2010. –Vol. 25. – P. 187–195.
3. Benini, L. Dynamic Power Management: Design Techniques and Cad Tools / L. Benini, G. De Micheli – Kluwer Academic Publishers, 1998. – 231 p.
4. Dynamic Voltage Scaling and Power Management for Portable Systems / T. Simunic, L. Benini, A. Acquaviva et al. // Proceedings of the 38th conference on Design automation (Las Vegas, Nevada). – Jun. 2001. – P. 524–529
5. Simunic, T. Energy efficient system design and utilization: PhD Thesis / T. Simunic – Stanford University (Stanford, CA), 2001. – 128 p.
6. Snowdon, D. Power Management and Dynamic Voltage Scaling: Myths and Facts / D. Snowdon, S. Ruocco, G. Heiser // Proceedings of the 7th ACM & IEEE International conference on Embedded software (Salzburg). – Sep. 2007. – P. 84–93.
7. Kappiah, N. Just In Time Dynamic Voltage Scaling: Exploiting Inter–Node Slack to Save Energy in MPI Programs / N. Kappiah, V.W. Freeh, D.K. Lowenthal // Proceedings of the ACM/IEEE SC 2005 Conference (Seattle, Washington). – Nov. 2005. – P. 33.
8. Reducing power with performance constraints for parallel sparse applications / G. Chen, K. Malkowski, M.T. Kandemir et al. // Proceedings of the 19th International Parallel & Distributed Processing Symposium (Denver, Colorado). – Apr. 2005. – P. 8.
9. Intel Corp., Intel 64 and IA-32 Architectures Optimization Reference Manual / URL: http://www.intel.com/content/dam/doc/manual/64-ia-32-architectures-optimization-manual.pdf (дата обращения: 10.1.2013), Chapter 11. P. 423–430.
10. Energy-Efficient Platforms – Considerations for Application Software and Services / URL: http://download.intel.com/technology/pdf/322304.pdf (дата обращения: 05.09.2012).
11. Tiwari, V. Power Analysis of Embedded Software: A First Step / V. Tiwari, S. Malik, A. Wolfe // IEEE Transactions on VLSI Systems. – 1994. –Vol. 2, No. 4. – P. 437–445.

12. Instruction Level Power Analysis and Optimization of Software / V. Tiwari, S. Malik, A. Wolfe et al. // Proceedings of the 9th International Conference VLSI Design (Bangalore, India). – Jan. 1996. – P. 326–328.
13. Instruction scheduling for power reduction in processor–based system design / H.H. Tomiyama, T. Ishihara, A. Inoue et al. // Proceedings of the Conference Design, Automation ant Test in Europe (Paris, France). – Feb. 1998. – P. 855–860.
14. Simunic, T. Energy–Efficient Design of Battery–Powered Embedded Systems / T. Simunic, L. Benini, G. De Micheli // IEEE Transactions on VLSI Systems. – 2001. –Vol. 9, No. 1. – P. 15–28.
15. The impact of source code transformations on software power and energy consumption / C. Brandoles, W. Fornaciari, F. Salice et al. // World Scientific Journal of Circuits Systems and Computers. – 2002. – Vol. 11, No. 5 – P. 477–502.
16. Power Checker Web Site / URL: http://software.intel.com/en-us/blogs/2011/06/27/intel-power-checker/ (дата обращения: 10.10.2012).
17. Power Informer Web Site / URL: http://software.intel.com/en-us/articles/intel-powerinformer/ (дата обращения: 10.10.2012).
18. Joulemeter Web Site / URL: http://research.microsoft.com/en-us/downloads/fe9e10c5-5c5b-450c-a674-daf55565f794/ (дата обращения: 10.10.2012).
19. Perfmon Web Site / URL: http://technet.microsoft.com/en-us/library/bb490957.aspx (дата обращения: 10.10.2010).
20. Power Checker Web Site / URL: https://01.org/powertop/ (дата обращения: 10.10.2012).

Калачев Артем Валерьевич, студент 2 курса магистратуры факультета Вычислительной математики и кибернетики, Нижегородский государственный университет им. Н.И. Лобачевского, artem.kalachev@me.com.

Карсаков Александр Сергеевич, студент 4 курса факультета Вычислительной математики и кибернетики, Нижегородский государственный университет им. Н.И. Лобачевского, karsakov.a.s@gmail.com.

Мееров Иосиф Борисович, к.т.н., доцент, зам. зав. каф, математического обеспечения ЭВМ, факультет Вычислительной математики и кибернетики, Нижегородский государственный университет им. Н.И. Лобачевского, meerov@vmk.unn.ru.

Напыльникова Яна Александровна, студентка 4 курса факультета Вычислительной математики и кибернетики Нижегородский государственный университет им. Н.И. Лобачевского, napylnikova.ja@gmail.com.

Овсюхно Андрей Ювенальевич, студент 4 курса факультета Вычислительной математики и кибернетики, Нижегородский государственный университет им. Н.И. Лобачевского, kselar@gmail.com.

DEVELOPMENT OF SOFTWARE TOOL FOR THE ANALYSIS OF ENERGY EFFICIENCY

A.V. Kalachev, N.I. Lobachevsky State University of Nizhni Novgorod (N.Novgorod, Russian Federation),
A.S. Karsakov, N.I. Lobachevsky State University of Nizhni Novgorod (N.Novgorod, Russian Federation),
I.B. Meyerov, N.I. Lobachevsky State University of Nizhni Novgorod (N.Novgorod, Russian Federation),
Y.A. Napylnikova, N.I. Lobachevsky State University of Nizhni Novgorod (N.Novgorod, Russian Federation),
A.U. Ovsuhno, N.I. Lobachevsky State University of Nizhni Novgorod (N.Novgorod, Russian Federation)

In this paper we consider the problem of energy efficient software design. We mostly focus on programming for mobile devices. A new software tool for the analysis of energy efficiency of applications for mobile devices is proposed. The tool allows collecting and processing experimental data, which characterizes the energy efficiency of applications which are running on the platform. We compare the similar instruments. We describe the basic functionality, method of use and the results of use on the matrix multiplication example. The analysis of the energy efficient algorithm implementations is done by using the SSE instruction set, as well as parallel programming technologies OpenMP and Cilk Plus. Additionally, we investigate the dependence of energy efficiency on the number of used threads, the quantity of cache misses and the number of transitions between the C-States of CPU. We formulate conclusions for the results of the analysis and present plans to further developing.

Keywords: mobile systems, energy optimization, tools for the analysis of energy efficiency, metrics to evaluate the power, C-State.

References

1. Weiser, M. Scheduling for Reduced CPU Energy / M. Weiser, B. Welch, A. Demers, S. Shenker // Proceedings of the 1st USENICS Symposium on Operating Systems Design and Implementation (Monterey, CA). – Nov. 1994. – P. 13-23.
2. Bekas, C. A new energy aware performance metric / C. Bekas, A. Curioni // Computer Science – R&D. – 2010. –Vol. 25. – P. 187-195.
3. Benini, L. Dynamic Power Management: Design Techniques and Cad Tools / L. Benini, G. De Micheli – Kluwer Academic Publishers, 1998. – 231 P.
4. Simunic, T. Dynamic Voltage Scaling and Power Management for Portable Systems / T. Simunic, L. Benini, A. Acquaviva, P. Glynn, G. De Micheli // Proceedings of the 38th conference on Design automation (Las Vegas, Nevada). – Jun. 2001. – P. 524-529
5. Simunic, T. Energy efficient system design and utilization: PhD Thesis / T. Simunic – Stanford University (Stanford, CA), 2001. – 128 P.
6. Snowdon, D. Power Management and Dynamic Voltage Scaling: Myths and Facts / D. Snowdon, S. Ruocco, G. Heiser // Proceedings of the 7th ACM & IEEE International conference on Embedded software (Salzburg). – Sep. 2007. – P. 84-93.
7. Kappiah, N. Just In Time Dynamic Voltage Scaling: Exploiting Inter–Node Slack to Save Energy in MPI Programs / N. Kappiah, V.W. Freeh, D.K. Lowenthal // Proceedings of the ACM/IEEE SC 2005 Conference (Seattle, Washington). – Nov. 2005. – P. 33.

8. Chen, G. Reducing power with performance constraints for parallel sparse applications / G. Chen, K. Malkowski, M.T. Kandemir, P. Raghavan // Proceedings of the 19th International Parallel & Distributed Processing Symposium (Denver, Colorado). – Apr. 2005. – P. 8.
9. Intel Corp., Intel 64 and IA-32 Architectures Optimization Reference Manual / URL: http://www.intel.com/content/dam/doc/manual/64-ia-32-architectures-optimization-manual.pdf (accessed: 10.1.2013), Chapter 11. P. 423-430.
10. Energy-Efficient Platforms – Considerations for Application Software and Services / URL: http://download.intel.com/technology/pdf/322304.pdf (дата обращения: 05.09.2012).
11. Tiwari, V. Power Analysis of Embedded Software: A First Step / V. Tiwari, S. Malik, A. Wolfe// IEEE Transactions on VLSI Systems. – 1994. –Vol. 2, No. 4. – P. 437-445.
12. Tiwari, V. Instruction Level Power Analysis and Optimization of Software / V. Tiwari, S. Malik, A. Wolfe, M. Lee// Proceedings of the 9th International Conference VLSI Design (Bangalore, India). – Jan. 1996. – P. 326-328.
13. Tomiayma, H.H. Instruction scheduling for power reduction in processor–based system design / H.H. Tomiyama, T. Ishihara, A. Inoue, H. Yasuura // Proceedings of the Conference Design, Automation ant Test in Europe (Paris, France). – Feb. 1998. – P. 855-860.
14. Simunic, T. Energy–Efficient Design of Battery–Powered Embedded Systems / T. Simunic, L. Benini, G. De Micheli // IEEE Transactions on VLSI Systems. – 2001. –Vol. 9, No. 1. – P. 15-28.
15. Brandoles, C. The impact of source code transformations on software power and energy consumption / C. Brandoles, W. Fornaciari, F. Salice, D. Sciuto // World Scientific Journal of Circuits Systems and Computers. – 2002. – Vol. 11, No. 5 – P. 477-502.
16. Power Checker Web Site / URL: http://software.intel.com/en-us/blogs/2011/06/27/intel-power-checker/ (accessed: 10.10.2012).
17. Power Informer Web Site / URL: http://software.intel.com/en-us/articles/intel-powerinformer/ (accessed: 10.10.2012).
18. Joulemeter Web Site / URL: http://research.microsoft.com/en-us/downloads/fe9e10c5-5c5b-450c-a674-daf55565f794/ accessed: 10.10.2012).
19. Perfmon Web Site / URL: http://technet.microsoft.com/en-us/library/bb490957.aspx (accessed: 10.10.2010).
20. Power Checker Web Site / URL: https://01.org/powertop/ (accessed: 10.10.2012).

Поступила в редакцию 8 февраля 2013 г.