РАЗРАБОТКА ПРОГРАММНОГО КОМПЛЕКСА ДЛЯ АНАЛИЗА ЭНЕРГОЭФФЕКТИВНОСТИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

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

Нижегородский госуниверситет им. Н.И. Лобачевского

Введение

В настоящее время тема энергосбережения является достаточно актуальной. В связи с повсеместным применением информационных технологий энергетическая эффективность стала приоритетным направлением и в IT-сфере. Так, увеличение производительности процессоров привело к нелинейному росту энергопотребления в рабочих станциях и серверах. Производители мобильных устройств столкнулись с существенным падением времени автономной работы при расширении функциональности. Таким образом, необходимость внедрения энергосберегающих технологий продиктована не только желанием сэкономить ресурсы, но и невозможностью обеспечить приемлемое время автономной работы.

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 работе предлагается новый программный инструмент для сбора и анализа данных об энергопотреблении в программах. Описывается подход к организации инструмента, приводятся примеры его использования на модельных задачах.

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 Top-500 самых высокопроизводительных систем в мире (http://www.top500.org/) был создан список наиболее энергоэффективных систем – Green 500 (www.green500.org). В 2010 году была предложена альтернативная метрика [2] – FTTSE (*f(time to solution) * energy*), одновременно учитывающая время работы программы (*f(time to solution)*) 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 про-

цессора. Центральный процессор 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 подход, 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.

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, 9–13]. Выделены типовые подходы к оптимизации:

1. преобразование циклов (loop transformations);
2. преобразование структур данных (data structure transformations);
3. межпроцедурная оптимизация (inter-procedural transformations);
4. преобразование управляющих конструкций (operators and control structure transformations).

Оценка энергоэффективности программного обеспечения

Целью данной работы является разработка методов и программных средств анализа и оптимизации энергопотребления вычислительных устройств. 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. количество переходов между разными C-state (переходов/с);
4. процент времени загруженности процессора;
5. процент времени бездействия (простоя) процессора.

Существует множество инструментов, которые позволяют определять данные метрики. Сравнительный анализ таких инструментов представлен в таблице.

	Intel(R) Power Checker	**Perfmon**	**Intel(R) Power Informer**	**Powertop**	**Joulemeter**
ОС	Windows	Windows	Windows	Linux	Windows
Потребляемая мощность	Да	Да	Да	Да	Да
Анализ C-State	Да	Да	Да	Да	Нет
Информация о центральном процессоре	Нет	Да	Да	Да	Нет
Отображение результатов в реальном времени	Нет	Да	Нет	Да	Да
Проведение серий экспериментов	Нет	Нет	Нет	Нет	Нет
Обновление данных (сек.)	1.0	1.0	неизвестно	15.0	1.0
Вывод результатов в файл	Да	Да	Да	Нет	Нет
Хранение результатов	Нет	Нет	Нет	Нет	Нет

Все инструменты обладают теми или иными недостатками. Более того, ни один из них не предоставляет разработчику полнофункциональный исследовательский комплекс, который был бы удобен при оптимизации программных продуктов. В связи с

этим возникла необходимость разработать систему, которая могла бы решить данную проблему. Были предъявлены следующие требования к системе:

Требование	Описание
Кроссплатформенность	Поддержка ОС: Windows XP, Windows 7 и Linux. Предусмотреть возможность портирования на другие архитектуры процессоров (ARM) и мобильные ОС (Android, iOS, Windows Phone).
Нагрузка	Система должна добавлять минимальную погрешность в проводимые эксперименты (не более 5%).
Автоматизация запуска экспериментов	Ручной эксперимент: приложение должно работать в интерактивном режиме с настройкой параметров эксперимента. Серийный эксперимент: приложение должно работать в автоматизированном режиме, используя конфигурационный файл. Система должна иметь открытую архитектуру и поддерживать расширение списка используемых параметров. Система должна поддерживать функцию калибровки. Основываясь на полученных параметрах, система должна следить за корректностью условий проведения экспериментов.
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 данных должен быть взаимозаменяемым и не влиять на другую функциональность системы.
Пользовательский интерфейс	Взаимодействие с пользователем должно происходить с использованием графического пользовательского интерфейса. Должна поддерживаться следующая функциональность: задача параметров эксперимента (графическая форма или выбор конфигурационного файла на файловой системе), манипуляция с очередью экспериментов, запуск экспериментов, сбор результатов, представление результатов экспериментов (графический, конвертация в текстовый формат).
Представление данных	Пользователь должен иметь возможность загрузить результаты любого проведенного эксперимента. Результаты должны быть представлены в удобном и наглядном виде для пользователя с использованием графиков, таблиц, гистограмм и других вариантов представления данных.

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).

Высокоуровневая архитектура системы выглядит следующим образом (рис. 1).

Центральным управляющим модулем системы является *Controller*. Данная компонента предоставляет API, посредством которого другие компоненты системы могут взаимодействовать с внутренней машиной состояний и очередью экспериментов. Также, данная компонента имеет набор функций, с помощью которых другие компоненты могут получить доступ к модулю *Data Representation*, который отвечает за программное представление данных об эксперименте. Модуль *Experiment Executer* реализует всю логику, связанную с проведением экспериментов и сбором метрик. Для получения необработанных данных используются различные источники данных, реализации которых представлены в модуле *Data Source*. Все инструменты визуализации, а также компонента предварительной обработки данных находятся в модуле *Visualization*.

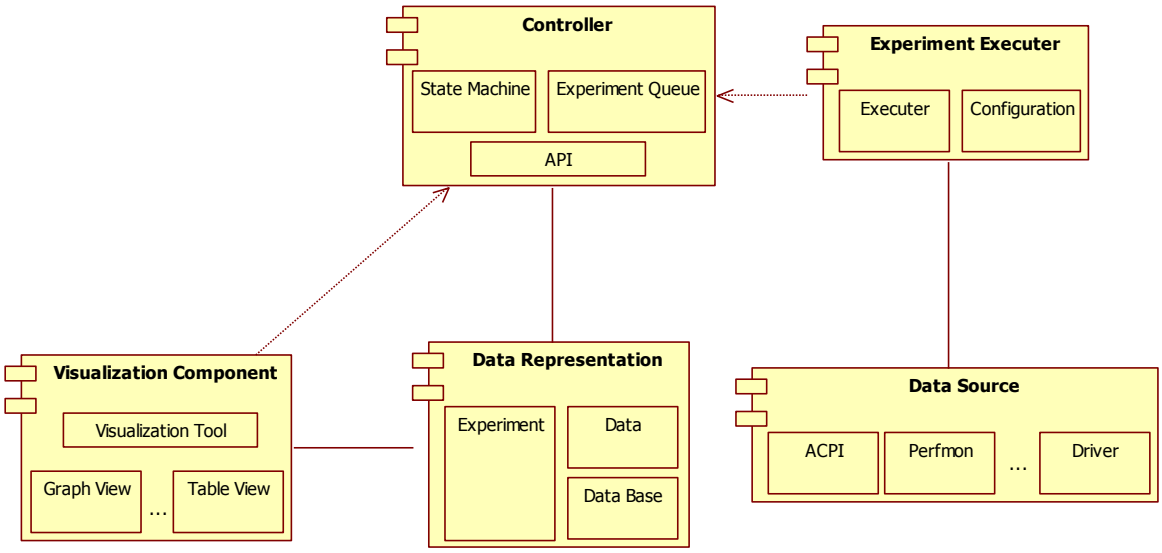

Рис. 1. Архитектура системы

Результаты проведенных экспериментов

Тестовая система

- Processor: Mobile QuadCore Intel Core i7 2630QM @ 2GHz (2.9 GHz Turbo Boost).
- Cache: L1 – 32 Kb, L2 – 256 Kb, L3 – 6Mb.
- RAM: 4Gb, DDR3 1333 MHz.
- Operating System: Windows 7 Ultimate (x64).

В первой серии экспериментов (рис. 2-4) производится сравнение потребляемой мощности при решении задачи матричного умножения в следующих конфигурациях: последовательная версия, версии с использованием SSE, OpenMP и Cilk+.

Заметим, что пиковое значение потребляемой мощности при работе параллельных версий больше, чем у последовательной версии. OpenMP и Cilk+ задействуют большие ресурсы, поэтому время работы приложений значительно меньше, что приводит к лучшим показателям энергопотребления, чем у последовательной версии. Реализация на SSE использует дополнительные возможности процессора, что позволяет решить задачу наиболее энергоэффективно.

На гистограммах видны следующие результаты: с ростом размеров матрицы увеличивается и разница в количестве энергии, потребляемой последовательной и параллельными версиями и версии с использованием SSE. Сравнивая версии на OpenMP и Cilk+ между собой, можно выделить преимущество второй версии при достаточно больших размерах задачи.

Вторая 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, либо с некоторыми паузами, что приводит к увеличению числа C-State-переходов.

В результате видим, что увеличение числа переходов между C-state-состояниями процессора приводит к росту энергопотребления.

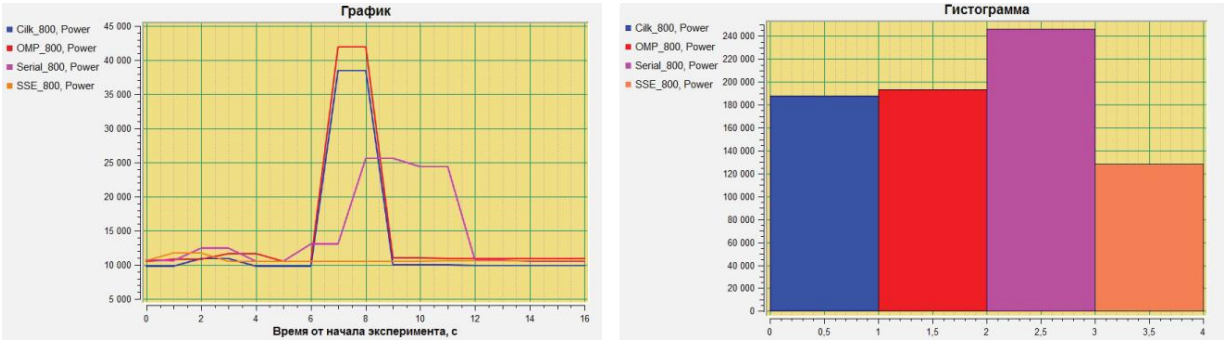

Рис. 2. Матричное умножение 600x600

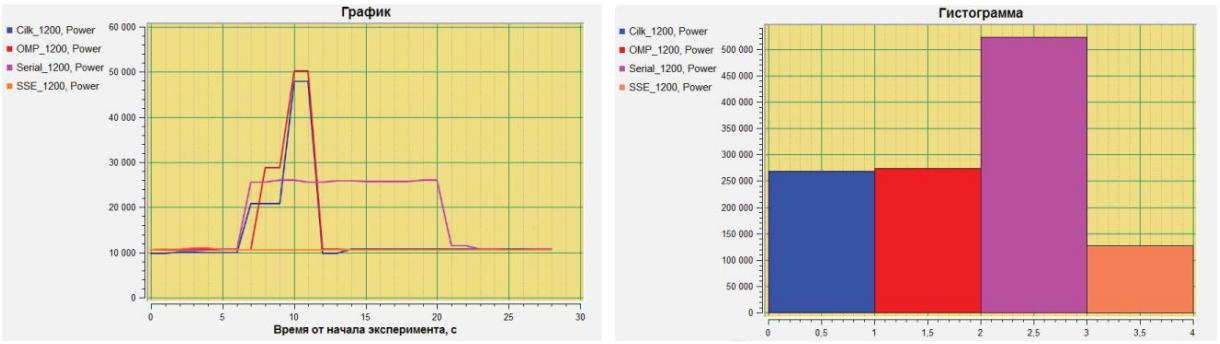

Рис. 3. Матричное умножение 1200x1200

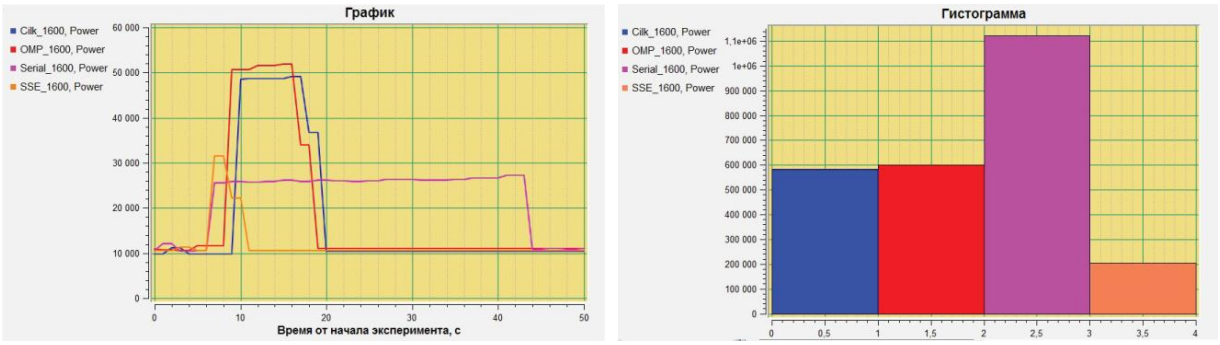

Рис. 4. Матричное умножение 1800x1800

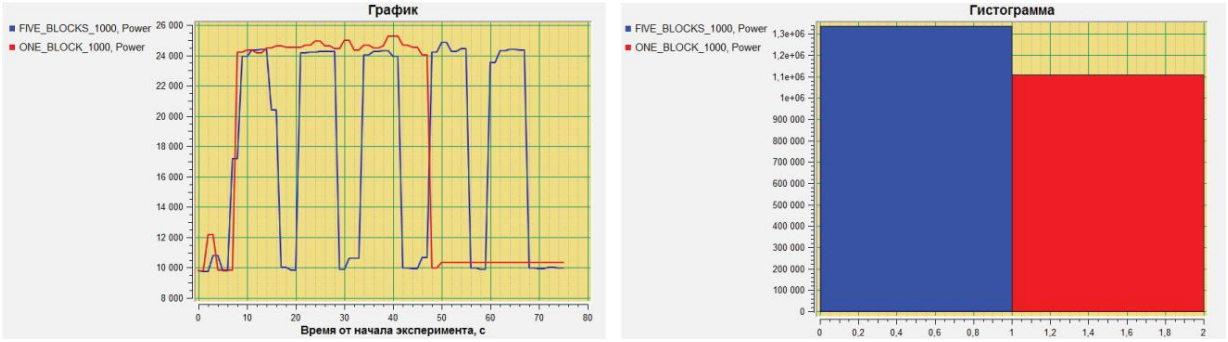

Рис. 5. Матричное умножение. C-states

Заключение

В ходе работы 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 была продемонстрирована на серии из двух демонстрационных задач:

- Сравнение энергопотребления при решении задачи матричного умножения.
- Анализ зависимости энергопотребления от количества переходов между C-State-состояниями.

Система является расширяемой (новые метрики, источники данных, способы визуализации и др.). Расширение функциональности системы, а также апробация системы (включая выработку подходов к оптимизации энергопотребления) является темой дальнейшей исследований.

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

Литература

1. Weiser M., Welch B., Demers A., Shenker S., Scheduling for Reduced CPU Energy // Proceedings of Symposium on Operating Systems Design and Implementation, 1994.
2. Bekas C., Curioni A. A new energy aware performance metric // Computer Science – R&D. 2010. P. 187-195.
3. Benini L., De Micheli G. Dynamic power management: design techniques and CAD tools // Kluwer Academic Publisher. 1997.
4. Šimunic T., Benini L., Acquaviva A., Glynn P., De Micheli G. Dynamic Voltage Scaling and Power Management for Portable Systems // DAC 2001, Las Vegas, USA. 2001.
5. Šimunic T. Energy efficient system design and utilization // Doctoral Dissertation, Energy efficient system design and utilization. Stanford University Stanford, CA, USA, 2001.
6. Snowdon D., Ruocco S., Heiser G., Power Management and Dynamic Voltage Scaling: Myths and Facts // National ICT Australia and School of Computer Science and Engineering University of NSW. Sydney 2052, Australia, 2005.
7. Kappiah N., Freeh V.W., Lowenthal D.K. Just in time dynamic voltage scaling: exploiting inter-node slack to save energy in MPI programs // IEEE Computer Society. 2005.
8. Chen G., Malkowski K., Kandemir M.T., Raghavan P. Reducing power with performance constraints for parallel sparse applications // IPDPS. IEEE Computer Society, 2005.
9. Tiwari V., Malik S., Wolfe A. Power analysis of embedded software: A first step // IEEE Transactions on VLSI Systems, 1994.
10. Tiwari V., Malik S., Wolfe A., Lee M. Instruction level power analysis and optimization of softwar // Journal of VLSI Signal Processing Systems. 1996. P. 1-18.
11. Tomyiama H.H., Ishihara T., Inoue A., Yasuura H. Instruction scheduling for power reduction in processor-based system design // DATE, 1998.
12. Simunic T., Benini L., De Micheli G. Energy-Efficient Design of Battery-Powered Embedded Systems // International Symposium on Low Power Electronics and Design, 1999.
13. Brandoles C., Fornaciari W., Salice F., Sciuto D. The impact of source code transformations on soft-ware power and energy consumption // Journal of Circuits, Systems, and Computers. 2002. Vol. 11, No. 5. P. 477-502.