2. Cha Zhang and Zhengyou Zhang. A Survey of Recent Advances in Face Detection / Microsoft Research Technical Report, 2010.
3. http://opencv.willowgarage.com/wiki.
4. http://openmp.org/wp.

А.В. Русаков

Нижегородский государственный университет им. Н.И. Лобачевского

ОБ ОДНОМ ПОДХОДЕ К СРАВНИТЕЛЬНОМУ АНАЛИЗУ ТЕХНОЛОГИЙ ПАРАЛЛЕЛЬНОГО ПРОГРАММИРОВАНИЯ В СИСТЕМАХ С ОБЩЕЙ ПАМЯТЬЮ

В настоящее время наблюдается неуклонный рост производительности вычислительной техники, что открывает все новые возможности для решения ресурсоемких задач, имеющих большое практическое значение. Такие задачи можно встретить в физике, химии, биологии, экономике, финансах и т.д. При этом одно из основных направлений развития состоит в частичном дублировании устройств, интегрированных на кристалле центрального процессора. Наличие и распространение таких вычислительных систем потребовало новых технологий, моделей, методов и программных средств системного уровня, ориентированных на их эффективное использование. Так, в научной периодике наблюдается неуклонный рост числа публикаций, связанных с новыми технологиями для параллельного программирования. Разрабатываются новые подходы (OpenMP, MPI), фреймворки (OpenCL, CUDA), библиотеки (TBB, CCR), языки (Cilk, Erlang, APL, Sisal) параллельного программирования. Одна из основных проблем заключается в сравнительной сложности разработки параллельных программ с нуля. Переход от обычного последовательного кода к параллельному требует определенных навыков системного программирования, не всегда присутствующих у прикладных программистов – ключевых потребителей новых технологий параллельных вычислений. Таким образом, на первый план выходят два вопроса: производительность получаемых параллельных программ и трудоемкость их разработки.

Используемые критерии. Очевидно, параллельные программы, созданные с применением высокоуровневых надстроек, в большинстве своем не могут опередить по производительности код, написанный с использованием низкоуровневых средств параллелизма. Однако в области прикладного программирования часто оправданна попытка пожертвовать незначительной долей производительности ради удобства разработки.

На сегодняшний день существует множество публикаций, в которых дается сравнение производительности программ с использованием различных технологий параллельного программирования, однако вопрос времени обучения и трудоемкости разработки программ с их использованием мало изучен. Одним из немногих примеров может служить работа [3].

В данной работе приводится сравнение особенностей выбранного набора технологий параллельного программирования для систем с общей памятью. Первая часть работы описывает анализ производительности каждой из технологий на примере тестовой задачи, во второй же части предлагается набор метрик для определения трудоемкости разработки программ с их использованием.

Класс задач. Сравнение выбранных технологий параллельного программирования производится на примере задач финансовой математики. Финансовая математика – раздел прикладной математики, связанный с финансовыми расчетами. Основу задач данной области составляют вычисления стоимостей производных ценных бумаг (опционов, акций, облигаций и других). Модель финансового рынка в данных задачах описывается с использованием систем стохастических дифференциальных уравнений, а алгоритмы решения обычно задействуют Монте-Карло-моделирование, методы оптимизации и численное интегрирование. Исходя из этого, решаемые задачи относятся к классу высокопроизводительных вычислений, однако имеют ясное прикладное значение и конкретный экономический смысл. Выбор данной области также обусловлен наличием у автора опыта решения подобных задач.

Предлагаемый подход для анализа производительности. Оценка производительности программ, написанных с использованием различных технологий параллельного программирования, заключается в стандартной процедуре сравнения времени выполнения параллельной программы со временем выполнения последовательной версии. Таким образом, при работе параллельного кода на системе с p исполняющими устройствами во внимание берется величина $S_p = \frac{T_1}{T_p}$, называемая ускорением параллельного алгоритма, где T_1 – время вычислений на одном исполняющем устройстве, T_p – время вычислений с использованием p исполняющих устройств. Кроме того, для анализа эффективности той или иной технологии, применимой к конкретной задаче, рассматривается масштабируемость реализованного алгоритма при увеличении количества исполняющих устройств системы. В данном случае учитывается величина эффективности параллельного алгоритма $E_p = \frac{S_p}{p}$, равная отношению ускорения параллельной программы к количеству исполняющих устройств, на котором это ускорение было зафиксировано.

Предлагаемый подход для анализа трудоемкости разработки. Современная литература, посвященная вопросам управления качеством, содержит большое количество различных метрик, использующихся для оценки качества и сложности программного обеспечения и программного кода [6, 7]. Автором данной работы предлагается набор метрик, способных ярче в количественных значениях представить трудоемкость различных подходов к написанию параллельных программ. Далее приведен список метрик, сгруппированных по единицам измерения.

Временные:

- среднее время обучения технологии программирования;
- время разворачивания и настройки среды программирования;
- общее время разработки продукта;

- время разработки для каждой стадии;
- общие трудозатраты (в человеко-месяцах, человеко-часах).

Элементы кода:

- LOC'и (количество строк исходного кода – с комментариями и без);
- количество функций, операторов, операндов;
- метрики классовой сложности (количество классов, насыщенность классов, глубина наследования, связность, количество детей).

Тестирование и отладка

- количество ошибок при разработке и только на этапе тестирования;
- группы ошибок по типам;
- плотность ошибок на единицу кода;
- количество сборок кода.

Методом сбора временных метрик является фиксация самим программистом данных в процессе обучения и разработки. Метрики, связанные с элементами кода, собираются после написания программы при использовании вспомогательных средств интегрированных сред разработки. Данные о тестировании и отладке фиксируются средой разработки во время написания программы, а после группируются и усредняются программистом.

По результатам исследований состав метрик и методика могут быть модифицированы.

Пример применения. В качестве тестовой задачи в работе используется одна из задач финансовой математики – нахождение справедливой цены опциона Бермудского типа. Для анализа производительности были выбраны следующие технологии параллельного программирования для систем с общей памятью: OpenMP, Intel® Threading Building Blocks (TBB) и OpenCL.

Постановка задачи. Рассмотрим N-мерный финансовый рынок в непрерывном времени. Будем использовать модель Блэка-Шоулса [1]

$$dB_t = rB_t dt,\ B_0 > 0,$$

$$dS_t^i = S_t^i \left((r^i - \delta^i)dt + \sigma^i dW_t^i \right),\ S_0^i > 0,$$

где процессы $B = (B_t)_{t \geq 0}$ и $S^i = (S_t^i)_{t \geq 0}$, $i = 1,...,N$ описывает поведение облигаций B и акций S^i в момент времени $t \geq 0$ соответственно. Цена облигаций B зависит от процентной ставки r, цена акций S^i определяется волатильностью σ^i, процентной r^i и дивидендной δ^i ставками. Случайная стохастическая составляющая цены акций S^i описывается i-й компонентой Винеровского процесса $W^i = (W_t^i)_{t \geq 0}$ ($W = (W^1,...,W^N)$). Пусть матрица ковариации $COV = (\mathrm{cov}_{ij})_{i,j=1..N}$ описывает зависимости между ценами акций в каждый фиксированный момент времени.

Рассмотрим одну из часто встречающихся разновидностей опциона – max-call-опцион, для которого функция выплаты $h(t, S_t)$ имеет следующий вид:

$$h(t, S_t) = (\max_{i=1...N}(S_t^i) - P)^+,$$

где положительная константа P определяется опционным контрактом, $x^+ = \max(x, 0)$. Решим задачу вычисления стоимости опциона Бермудского типа, который может быть предъявлен к исполнению в любой из заранее фиксированных моментов $t \in Time = \{t_0 = 0, t_1, t_2, ..., t_d = T\}$, где момент T задает истечение срока действия опциона.

Метод решения задачи. Для решения данной задачи определения автором был реализован популярный алгоритм имитационных деревьев для определения истинной цены многомерных опционов Бермудского типа Broadie–Glasserman Random Trees 1997 [5], описанный в [2]. Программная реализация основана на работе [4].

Результаты экспериментов. В этом разделе приведены результаты вычислительных экспериментов, полученные с использованием следующего программного и аппаратного обеспечения:

- CPUs:
  - Intel(R) Core(TM)2 Duo 2.34GHz, Cache L1 2*64KB, Cache L2 4MB, Mem 3328 MB

- 4x Intel Xeon X7560 (8 cores, 2.27 GHz), Mem 64 GB
- GPU: NVidia 9800GT, (Intel Core2Duo E8500@3, 8GHz, Mem 2GB)
- Intel® Math Kernel Library (Intel® MKL) 10.2.2
- Intel® Thread Building Blocks (Intel® TBB) 3.0
- Intel® C/C++ Compiler 11.1, Microsoft ® Visual Studio 9.0

Ускорение программы на многоядерных системах для каждой из параллельных реализаций наглядно отображено в диаграммах (рис. 1, 2)

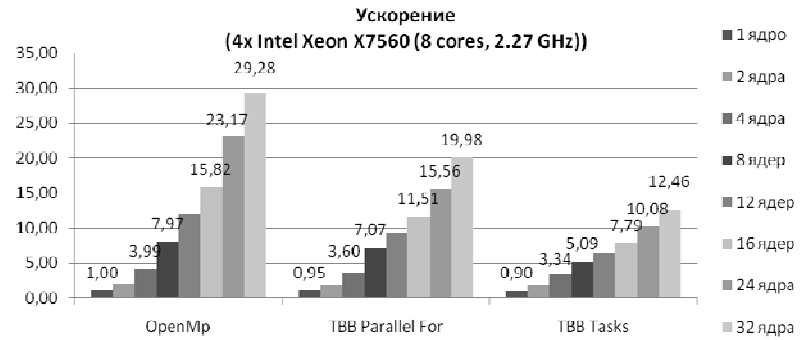

Рис. 1. Ускорение программы на 32-ядерной системе

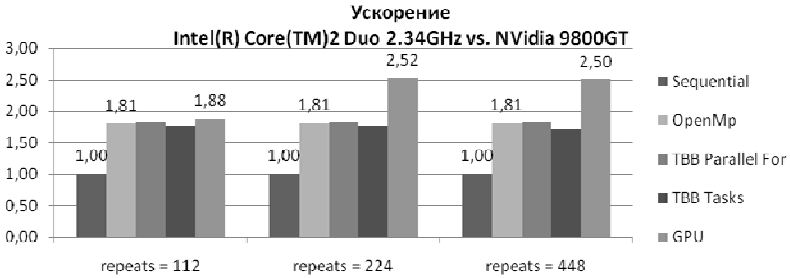

Рис. 2. Ускорение программы на 2-ядерной системе и графической карте

Анализируя полученные результаты, стоит отметить, что на настоящий момент автору не удалось добиться эффективного ис-

пользования высокоуровневой библиотеки TBB, предоставляющей гибкие возможности распараллеливания, по сравнению с технологией OpenMP. Это отчетливо видно на системах с большим количеством исполняющих устройств. Так, эффективность параллельного кода, созданного при помощи подхода TBB Tasks, на системе с 32 ядрами составляет всего около 39 %, при 63 % у TBB Parallel for и 91 % у OpenMP на аналогичном оборудовании. При этом необходимо отметить, что автор ранее не использовал Intel TBB. В дальнейшем планируется продолжить работу над оптимизацией этой версии. Для разработки GPU-версии использовалась технология OpenCL. Использование гетерогенных систем для исполнения параллельных программ на OpenCL по производительности оказалось чуть лучше остальных подходов на системе с двумя ядрами, но заметно проигрывает архитектурам с большим количеством ядер. Дальнейшая оптимизация OpenCL-версии также выглядит необходимой. Обращая внимание не только на производительность, но и на трудоемкость разработки, можно сделать следующие выводы.

Среди всех рассмотренных технологий для решения данной прикладной задачи, написание параллельного кода на OpenCL является самым трудоемким процессом. Это объясняется сложностью изучения архитектуры стандарта, большим количеством ошибок в коде программы даже у опытных программистов ввиду отсутствия отладчика ядер. Отдельные проблемы возникли в процессе разворачивания системы программирования для GPU.

Применение библиотеки TBB для написания программы потребовало значительного количества времени для реализации метода распараллеливания, основанного на логических задачах. Также этот подход оказался самым плохо масштабируемым на системах с большим количеством исполняющих устройств (в текущей реализации). Распараллеливание программы с помощью технологии OpenMP позволило добиться для данной задачи практически линейного ускорения на системах с количеством ядер до 32. Данный подход занял наименьшее время на обучение технологии среди всех рассмотренных и наименьшее количество времени для реализации самой программы.

В рамках работы, выполненной в лаборатории «Информационные технологии» (ITLab) ННГУ, проведен сравнительный анализ некоторых широко распространенных технологий и библиотек для параллельного программирования в системах с общей памятью, включая гетерогенные системы с использованием графических процессоров. Анализ проводился на основе решения тестовой задачи – определения справедливой цены опционов Бермудского типа методом имитационных деревьев (Broadie-Glasserman Random Trees). Автор благодарит компанию «Интел» за возможность тестирования программ на 32-ядерной архитектуре, полученной по результатам конкурса проектов Intel® Manycore Testing Lab. Для оценки трудоемкости разработки параллельных программ предложена система метрик. В дальнейшем планируется ее апробация.

Список литературы

1. Black F., Scholes M. The pricing options and corporate liabilities // Journal of Political Economy. – 1973. – Vol. 3. – P. 637–659.

2. Broadie M., Glasserman P. Pricing American-style securities using simulation // Journal of Economic Dynamics and Control, June 1997. – Vol. 21. – P. 1323–1352.

3. A pilot study to compare programming effort for two parallel programming models / L. Hochstein [et al.]. – 28 December, 2007 (Preprint submitted to Elsevier).

4. Параллельная реализация одного алгоритма нахождения цены опционов Бермудского типа / А.С. Горбунова [и др.] // Высокопроизводительные параллельные вычисления на кластерных системах: материалы Пятого Международного науч.-практ. семинара / под ред. проф. Р.Г. Стронгина. – Н. Новгород: Изд-во Нижегород. гос. ун-та, 2005. – С. 60–67.

5. Эффективное использование вычислительных ресурсов для определения справедливых цен опционов бермудского типа / А.С. Горбунова [и др.] // Научно-технический вестник Санкт-Петербургского государственного университета информационных технологий, механики и оптики. Вып. № 54. – СПб.: Изд-во СПбГУ ИТМО, 2008. – С. 145–151.

6. Новичков А. Метрики кода и их практическая реализация в IBM Rational ClearCase. – URL: http://www.ibm.com/developerworks/ru/edu/0108novich/section2.html, 04.08.2008.

7. Петрухин В.А., Лаврищева Е.М. Метрики качества программного обеспечения. – URL: http://www.intuit.ru/ department/ se/swebok/10/3.html, 24.09.2008.

В.В.Рябов, С.В.Сидоров

Нижегородский государственный университет им. Н.И.Лобачевского

ИСПОЛЬЗОВАНИЕ КРИВЫХ ПЕАНО С РАСТУЩИМ УРОВНЕМ ДЕТАЛИЗАЦИИ В АЛГОРИТМАХ ГЛОБАЛЬНОЙ ОПТИМИЗАЦИИ[1]

Данная работа продолжает развитие известного подхода к минимизации многоэкстремальных функций при невыпуклых ограничениях, описанного в работах [1–6] и получившего название *индексного метода* глобальной оптимизации. Подход основан на раздельном учете каждого ограничения задачи и не связан с использованием штрафных функций. При этом решение многомерных задач сводится к решению эквивалентных им одномерных. Соответствующая редукция основана на использовании кривых Пеано (называемых также развертками Пеано), однозначно отображающих единичный отрезок вещественной оси на гиперкуб, а также их обобщений («вращаемые развертки»), которые можно применять при решении задачи на кластерных системах с десятками и сотнями процессоров. Реализован прототип, использующий отображение единичного отрезка на гиперкуб с растущим порядком точности. Приведены результаты экспериментов, позволяющие оценить эффективность прототипа.

Постановка задачи. Рассмотрим задачу глобальной оптимизации вида

$$\varphi^* = \varphi(y^*) = \min\{\varphi(y): y \in D,\ g_j(y) \le 0,\ 1 \le j \le m\}, \qquad (1)$$

[1] Работа выполнена при поддержке совета по грантам Президента Российской Федерации (грант № МК-1536.2009.9).