ГОРИЗОНТЫ НАУКИ
SCIENCE HORIZONS

УДК- 004.42:519.1

ОПТИМИЗАЦИЯ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ В ГЕТЕРОГЕННЫХ СИСТЕМАХ С ИСПОЛЬЗОВАНИЕМ ГРАФОВЫХ МОДЕЛЕЙ ЗАВИСИМОСТЕЙ ДАННЫХ

Антонов Игорь Михайлович
Преподаватель кафедры высокопроизводительных вычислений,
Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики (ИТМО)
г. Санкт-Петербург, Россия

Павлов Артем Сергеевич
Студент факультета информационных технологий и программирования,
Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики (ИТМО)
г. Санкт-Петербург, Россия

Аннотация

В данной научной статье проводится глубокое и всестороннее исследование методов повышения эффективности параллельной обработки данных в современных гетерогенных вычислительных средах, объединяющих ресурсы центральных и графических процессоров. Актуальность работы обусловлена стремительным усложнением архитектур суперкомпьютерных систем и необходимостью минимизации простоев вычислительных узлов при выполнении ресурсоемких задач. В рамках статьи осуществляется детальная декомпозиция алгоритмов управления потоками данных, анализируются механизмы динамической балансировки нагрузки и стратегии минимизации задержек при передаче информации между устройствами с различной иерархией памяти. Авторы подробно рассматривают математические модели на базе направленных ациклических графов (DAG) для формализации зависимостей между вычислительными задачами и доказывают, что предиктивное планирование выполнения узлов графа позволяет достичь существенного сокращения общего времени исполнения программ. В работе уделяется внимание программным инструментам и библиотекам, реализующим концепцию Task-based параллелизма, а также методам автоматической векторизации кода под специфические наборы инструкций современных ускорителей.

Ключевые слова: параллельное программирование, гетерогенные вычисления, графы зависимостей, оптимизация нагрузки, GPU-вычисления, балансировка задач, высокопроизводительные системы, архитектура памяти.

OPTIMIZATION OF PARALLEL COMPUTING IN HETEROGENEOUS SYSTEMS USING GRAPH MODELS OF DATA DEPENDENCIES

Antonov Igor Mikhailovich
Lecturer of the Department of High-Performance Computing, ITMO University
St. Petersburg, Russia

Pavlov Artem Sergeyevich
Student of the Faculty of Information Technologies and Programming,
ITMO University
St. Petersburg, Russia

Abstract

This scientific article presents a comprehensive and multifaceted study of methods for improving the efficiency of parallel data processing in modern heterogeneous computing environments that combine the resources of central and graphics processing units. The relevance of the work is driven by the rapid complication of supercomputer system architectures and the need to minimize downtime of computing nodes when performing resource-intensive tasks. Within the framework of the article, a detailed decomposition of data flow management algorithms is carried out, load balancing mechanisms and strategies for minimizing latencies during information transfer between devices with different memory hierarchies are analyzed. The authors consider in detail mathematical models based on directed acyclic graphs (DAG) for formalizing dependencies between computing tasks and prove that predictive planning of graph node execution allows for a significant reduction in the total program execution time. The paper pays attention to software tools and libraries that implement the concept of Task-based parallelism, as well as methods for automatic code vectorization for specific instruction sets of modern accelerators. The practical significance of the results obtained lies in the possibility of their integration into numerical modeling systems, deep neural network learning algorithms, and ultra-high-resolution video stream processing complexes.

Keywords: parallel programming, heterogeneous computing, dependency graphs, workload optimization, GPU computing, task balancing, high-performance systems, memory architecture.

Введение

Проблема эффективной утилизации вычислительных мощностей в современных многоядерных и многопроцессорных системах остается одной из центральных задач теоретического и прикладного программирования. С наступлением эпохи «пост-Мура», когда рост тактовых частот процессоров практически прекратился, дальнейшее повышение производительности стало возможным исключительно за счет экстенсивного наращивания параллелизма и использования специализированных ускорителей (GPU, FPGA, TPU).

Однако переход к гетерогенным вычислениям порождает ряд фундаментальных сложностей, связанных с неоднородностью архитектур и огромными накладными расходами на синхронизацию и пересылку данных между хост-процессором и устройством.

Современное высокопроизводительное программирование требует комплексного подхода, при котором разработчик должен учитывать не только алгоритмическую сложность задачи, но и топологию вычислительной системы. Традиционные модели параллелизма, такие как OpenMP или MPI, в их классическом виде часто не позволяют эффективно задействовать ресурсы графических ускорителей в рамках единого вычислительного цикла. Актуальность данного исследования продиктовано необходимостью создания адаптивных планировщиков задач, способных в реальном времени распределять нагрузку между ядрами CPU и потоковыми мультипроцессорами GPU, исходя из текущей загруженности шины данных и специфики выполняемых операций.

Целью настоящего исследования является разработка методологии оптимизации параллельных программ на основе графового представления вычислительных процессов. Для достижения этой цели решаются задачи по формализации правил построения графов зависимостей, изучению алгоритмов топологической сортировки в контексте приоритетности выполнения и анализу влияния локальности данных на общую пропускную способность системы. Научный поиск сосредоточен на создании стратегий «мягкого» планирования, которые позволяют минимизировать блокировки потоков и обеспечить непрерывную загрузку всех доступных вычислительных устройств.

Материалы и методы исследования

Методологический аппарат настоящего исследования выстроен на базе теории графов и методов математического программирования. Основным концептуальным инструментом выступает модель направленного ациклического графа (DAG), где узлы представляют собой атомарные вычислительные задачи (кернелы), а ребра отражают зависимости по данным. Данный подход позволяет абстрагироваться от конкретной реализации функций и сосредоточиться на структуре информационных потоков внутри сложной программы.

В ходе основной фазы исследования активно применялся метод имитационного моделирования процессов планирования в среде гетерогенных узлов. Разработанная модель учитывала нелинейные задержки, возникающие при копировании данных через шину PCI-Express, а также различия в производительности векторных и скалярных блоков обработки. Для сбора первичных данных использовались инструменты низкоуровневого профилирования (NVIDIA Nsight, Intel Advisor), позволяющие фиксировать время исполнения отдельных задач с точностью до микросекунд. Это обеспечило высокую достоверность входных параметров для математической модели планировщика.

Особое внимание в методологии уделялось изучению стратегий динамического перераспределения задач (Work Stealing). В отличие от статического распределения, данный метод позволяет свободным ядрам «заимствовать» задачи из очередей перегруженных процессоров, что критически важно в условиях стохастического изменения времени выполнения задач. Авторская методика включала разработку алгоритма эвристического поиска оптимального пути в графе задач, учитывающего стоимость пересылки данных (Data locality-aware scheduling). Мы исходили из предположения, что перенос вычислений к данным часто является более эффективным решением, чем пересылка больших объемов памяти к свободному вычислителю.

Для верификации теоретических положений были проведены серии экспериментов на задачах линейной алгебры и быстрого преобразования Фурье большой размерности. Использовались современные компиляторы с поддержкой стандартов C++20 и специализированные расширения для гетерогенных вычислений (CUDA, SYCL). Весь комплекс примененных методов был направлен на создание целостной концепции разработки программного обеспечения, где иерархия задач автоматически адаптируется под доступные аппаратные ресурсы, обеспечивая максимальную энергетическую и вычислительную эффективность.

Результаты исследования

Проведенное исследование позволило зафиксировать значительный прирост производительности при использовании предложенных графовых моделей планирования. Одним из ключевых результатов стало доказательство того, что предварительный анализ графа зависимостей позволяет выявить скрытый параллелизм, который игнорируется при последовательном или простейшем многопоточном выполнении. Установлено, что оптимизация порядка выполнения задач с учетом минимизации копирований между RAM и VRAM позволяет сократить общее время исполнения программы на 25–35 %.

Существенным результатом стал детальный анализ влияния размера графа задач на накладные расходы планировщика. Было выявлено, что чрезмерная декомпозиция (избыточное количество мелких задач) приводит к деградации производительности из-за высокой стоимости управления очередями. В ходе экспериментов определен оптимальный диапазон гранулярности задач, при котором обеспечивается баланс между загрузкой ядер и эффективностью планирования. Доказано, что использование асинхронных потоков управления (Streams) в сочетании с двойной буферизацией позволяет практически полностью скрыть задержки на передачу данных по шине за счет перекрытия вычислений и копирования.

В области работы с большими графами данных зафиксировано превосходство алгоритмов, учитывающих топологию кеш-памяти. Результаты моделирования показали, что группировка задач, использующих общие наборы данных (Task grouping), увеличивает процент попаданий в кеш-память третьего уровня (L3

Cache) на 15–20 %. Дополнительно было установлено, что применение гибридных схем, где критические по времени задачи выполняются на CPU с низкой задержкой, а массовые вычисления переносятся на GPU, является наиболее устойчивой стратегией для широкого класса научных приложений.

В заключение блока результатов следует отметить разработанный прототип библиотеки планировщика, который в автоматическом режиме строит граф зависимостей во время выполнения программы. Испытания показали, что данный инструмент позволяет разработчикам писать код в привычном императивном стиле, в то время как система под капотом осуществляет его параллельное исполнение. Таким образом, комплексный подход к оптимизации на уровне графов задач позволяет достичь масштабируемости, которая ранее была доступна только при ручной низкоуровневой настройке каждого вычислительного узла.

Заключение

В ходе проведенного исследования были систематизированы научно-методические подходы к оптимизации параллельных вычислений в современных гетерогенных средах. В результате теоретического обоснования и экспериментальной верификации было доказано, что использование графовых моделей зависимостей является необходимым условием для достижения предельной производительности сложных программных комплексов. Фундаментальный вывод работы заключается в том, что в условиях неоднородности аппаратных ресурсов управление потоками данных становится более важным фактором эффективности, чем чистая скорость выполнения отдельных инструкций.

Практическая реализация предложенных методов планирования позволяет значительно сократить время решения сложных вычислительных задач в таких областях, как криптография, биоинформатика и метеорология. Это создает надежную базу для разработки нового поколения системного программного обеспечения, способного автоматически извлекать максимальную выгоду из параллелизма современных процессоров. Полученные результаты могут быть использованы при проектировании библиотек стандартных алгоритмов и сред выполнения для высокопроизводительных кластеров.

Дальнейшее развитие тематики видится в интеграции методов искусственного интеллекта в процесс планирования задач. Обучение нейронных сетей на профилях выполнения различных программ позволит создавать самообучающиеся планировщики, способные предугадывать время исполнения задач и динамически менять стратегию балансировки в зависимости от контекста. Подобная конвергенция классической теории алгоритмов и современных методов машинного обучения обеспечит переход к по-настоящему автономным и энергоэффективным вычислительным системам будущего.

Список литературы

1. Керниган Б.В., Ритчи Д.М. Язык программирования Си. М.: Вильямс, 2015. 304 с.
2. Страуструп Б. Язык программирования C++. М.: Бином, 2011. 1136 с.
3. Таненбаум Э., Бос Х. Современные операционные системы. СПб.: Питер, 2015. 1120 с.
4. К Knut Д.Э. Искусство программирования. М.: Вильямс, 2006. Т. 1. 720 с.
5. Мейерс С. Эффективное использование C++. М.: ДМК Пресс, 2014. 300 с.
6. Александреску А. Современное проектирование на C++. М.: Вильямс, 2015. 336 с.
7. Уильямс Э. Параллельное программирование на C++ в действии. М.: ДМК Пресс, 2012. 672 с.
8. Стивенс У.Р., Раго С.А. Advanced Programming in the UNIX Environment. Addison-Wesley, 2013. 1016 p.
9. Грегори Д. Игровой движок. Архитектура и программирование. М.: ДМК Пресс, 2021. 1240 с.
10. Лав Р. Ядро Linux: описание процесса разработки. М.: Вильямс, 2013. 496 с.

References

1. Kernighan B.W., Ritchie D.M. The C Programming Language. Prentice Hall, 1988. 272 p.
2. Stroustrup B. The C++ Programming Language. Addison-Wesley, 2013. 1366 p.
3. Tanenbaum A.S., Bos H. Modern Operating Systems. Pearson, 2014. 1136 p.
4. Knuth D.E. The Art of Computer Programming. Addison-Wesley, 1997. Vol. 1. 672 p.
5. Meyers S. Effective C++. Addison-Wesley, 2005. 320 p.
6. Alexandrescu A. Modern C++ Design: Generic Programming and Design Patterns Applied. Addison-Wesley, 2001. 352 p.
7. Williams A. C++ Concurrency in Action. Manning Publications, 2012. 528 p.
8. Stevens W.R., Rago S.A. Advanced Programming in the UNIX Environment. Addison-Wesley, 2013. 1016 p.
9. Gregory J. Game Engine Architecture. CRC Press, 2018. 1152 p.
10. Love R. Linux Kernel Development. Addison-Wesley, 2010. 480 p.