УДК 004.421

Аль Аззави Аус, М. Ю. Перухин, Т. Н. Мустафин

АНАЛИЗ ВЫЧИСЛИТЕЛЬНЫХ ВОЗМОЖНОСТЕЙ РАЗЛИЧНЫХ АЛГОРИТМОВ ПРИМЕНИТЕЛЬНО К ПОСТРОЕНИЮ РАСЧЕТНЫХ СЕТОК

Ключевые слова: модель, метод расчета, сетка.

В статье рассматриваются различные методы вычислений для построения расчетных сеток. Проводится анализ существующих методик.

Keywords: model, calculation method, net.

The article presents various methods for computing grid generation. The analysis of existing methods for computing grid generation is described.

Часто современные математические задачи моделирования физических процессов связаны с обработкой большого количества аналогичных данных. Такие расчеты при последовательном их решении могут занимать много времени. Решением этой проблемы может стать использование параллельного метода обработки данных. В этом случае задача делиться на сотни и тысячи частей и решается параллельно. Для увеличения скорости расчетов также применяются ГПУ вычисления.

С постоянно растущим спросом на производительность компьютерных систем, HPC индустрия движется в сторону гибридных моделей вычислений, где графические процессоры и ЦПУ работают вместе для выполнения универсальных вычислительных задач [1]. В случае параллельных расчетов графические процессоры преуспели в решении большого количества аналогичных данных, так как задачу можно разделить на сотни и тысячи частей и рассчитывать одновременно. ЦПУ не предназначены для такого вида вычислений, но они применяются в более последовательных задачах, таких как работа операционных систем и организация данных. ГПУ решения от NVIDIA опережают другие, где применяются наиболее актуальные процессы с конкретной задачей.

ГПУ вычисления используют графический процессор, для научно-технических вычислений общего назначения.

Модель для ГПУ вычислений использует процессор и ГПУ вместе в гетерогенной совместной обработке вычислительной модели. Последовательная часть приложения работает на процессоре, а вычислительно - интенсивная часть ускоряется в ГПУ. С точки зрения пользователя, приложение просто работает быстрее, потому что он использует высокую производительность ГПУ для повышения общей производительности.

Обработка в ГПУ применяется при параллельных операциях - в частности операциях линейной алгебры и матриц. Вначале, в GPGPU использовались обычные графические интерфейсы API для выполнения программ. Однако для адаптации интерфейсов к NVIDIA и AMD были написаны построены несколько новых языков программирования и платформ, которые представляют собой как интегрируемые в существующие среды дополнительные компиляторы со встроенными дополнительными расширениями среды, тем самым фактически подменяя отчасти возможности стандартной среды разработки, так и полностью отдельные среды разработки. К первым относится NVIDIA CUDA который распространяется как дополнительный компилятор к имеющимся IDE в частности может быть подключен и к Microsoft Visual Studio 12. CUDA - это архитектура параллельных вычислений NVIDIA, которая позволяет резко увеличить вычислительную производительность при использовании ГПУ [2].

Существует два основных метода расчетов в GPU: метод линейной алгебры и параллельный метод.

Параллельные вычисления одновременно используют более одного процессора или процессорного ядра для выполнения программы или нескольких вычислительных потоков. В идеале, параллельная обработка позволяет программе работать быстрее, вследствие использования нескольких двигателей (процессоров или ядер), управляющих им. На практике часто бывает трудно разделить программу таким образом, чтобы отдельные процессоры или ядра могли выполнять различные части, не мешая друг с другу. Большинство компьютеров имеют только один процессор, но некоторые модели имеют несколько, и чипы многоядерных процессоров становятся нормой. Есть даже компьютеры с тысячами процессоров. В однопроцессорных, одноядерных компьютерах, можно выполнять параллельную обработку путем соединения компьютеров в сети. Тем не менее, этот тип параллельной обработки требует очень сложное программное обеспечение: распределенное программное обеспечение обработки.

Стоит отметить, что параллелизм отличается от параллелизма. Термин параллельное, используемый в операционных системах и групп

база данных, относится к свойству системы, в которой несколько задач остаются логически активными и добиться прогресса в то же время, чередуя порядок выполнения задач и тем самым создавая иллюзию одновременного выполнения инструкции [3].

Методы линейного, параллельного и GPGPU программирования были применены при построении расчетной сетки ведущего и ведомого роторов винтового компрессора, необходимой для дальнейшего решения задач описанных Мустафиным Т.Н. в [4,5]. Цель программы - создание сетки из точек и дальнейшее ее компьютерное преобразование алгебраическим методом. Посторенние расчетной сетки основано на решении больших систем линейных алгебраических уравнений (СЛАУ), которые подразумевают присутствие большого количества схожих расчетов. Для определения оптимального метода расчета для данной определенной задачи были смоделированы все три метода и в дальнейшем определено время их работы. На рисунке 1 и 2 представлены зависимости времени от количества расчетных точек для всех методов. Из рисунка видно, что наиболее оптимальным по времени работы программы является классический параллельный метод.

При осуществлении расчета линейным методом задействуется только одно ядро многоядерного процессора, и программа занимает большее время, так как рассчитывается каждая точка последовательно. Параллельный метод вычисляет сразу несколько потоков в одно и то же время, такой расчет будет наоборот требовать не одно ядро, но займет меньше времени. CUDA – это параллельный метод с использованием ГПУ.

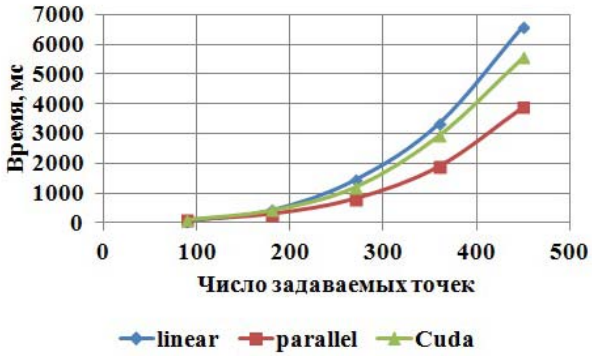

Рис. 1 - График зависимости времени работы программы от количества расчетных точек для ведущего вала (-◊- линейный метод; -■- параллельный метод; -▲- метод CUDA)

Это наиболее эффективный метод, однако, в нашем случае он показал средний результат. Такой результат объясняется тем, что операция копирования и вставки информации в памяти и файлы занимает больше времени. При более загруженном расчете, скорее всего он показал бы лучший результат.

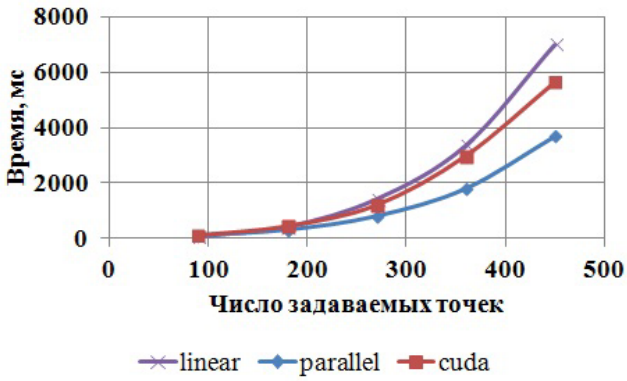

Рис. 2 - График зависимости времени работы программы от количества расчетных точек для ведомого вала (-x- линейный метод, -◊- параллельный метод, -■- метод CUDA).

Литература

1. Steven A Leduc. *Linear Algebra*. Cliffs Notes, Wiley Publishing. Inc. NY. 1996. 327 p.
2. http://www.nvidia.com/object/cuda_home_new.html
3. Линк., Снайдер Л. *Принципы параллельного программирования*: Учеб. пособие / Предисл.: В. А. Садовничий. Издательство Московского университета. Москва, 2013, 408 с.
4. Мустафин Т.Н., Якупов Р.Р., Акшинская В.В., Хамидуллин М.С., Хисамеев И.Г. Критерий оценки возможности потери контакта зацепления роторов винтового компрессора. Вестник Казанского технологического университета. 2013. Т. 16. №21. С. 249-252.
5. Мустафин Т.Н., Якупов Р.Р., Акшинская В.В., Хамидуллин М.С., Хисамеев И.Г. Геометрический анализ зацепления роторов винтового компрессора. вестник Казанского технологического университета. 2013. Т. 16. №19. С. 273-277.

© **Аль Аззави Аус** – магистрант кафедры АССОИ КНИТУ, awsawskee@yahoo.com; **М. Ю. Перухин** – доцент той же кафедры, perukhin@inbox.ru; **Т. Н. Мустафин** – к.т.н., доц. каф. холодильной техники КНИТУ, mustimur@rambler.ru.