ИНФОРМАТИКА И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ
INFORMATICS AND INFORMATION TECHNOLOGIES

УДК 681.325.518.5

ВОЗМОЖНОСТИ РАСПАРАЛЛЕЛИВАНИЯ ВЫЧИСЛЕНИЙ ПРИ СГЛАЖИВАНИИ ЭКСПЕРИМЕНТАЛНЫЕ ДАННЫХ СПЛАЙНА

Зайниддинов Х.Н., Маллаев О.У., Азимов Б.Р.

В работе исследованы возможности при сглаживание данных распараллеливания вычислений методом кубического сплайна. В качестве примера распараллеливание вычисление при обработки сейсмических сигналов. Были вычислены основные показатели эффективности и ускорениях параллельного алгоритма по сравнению с последовательным алгоритмом.

Ключевые слова: MPI, PFX, TPL, параллельные вычисления, интерполяция.

In this paper, we investigate the possibility of smoothing data parallelizing the computations by the cubic spline method. As an example, parallelization is the calculation when processing seismic signals. The main performance indicators and accelerations of the parallel algorithm were calculated in comparison with the sequential algorithm.

Key words: The MPI, PFX, TPL, parallel computing, interpolation.

I. ВВЕДЕНИЕ

Параллельные вычисления – одна из наиболее актуальных областей вычислительных наук в последнее десятилетие. Актуальность данной области связана прежде всего с бурным развитием численного моделирования. Численное моделирование является промежуточным элементом между аналитическими методами изучения и физическими экспериментами [1]. Рост количества задач, для решения которых необходимы параллельные вычисления обусловлен:

- возможностью изучать явления, которые затруднительно, по некоторым причинам, изучить экспериментально.
- необходимостью управления сложными промышленными и технологическими процессами в режиме реального времени, в условиях неопределенности [2].
- ростом числа задач, для решения которых необходимо обрабатывать большие объемы информации.

Распараллеливание алгоритмов требовало управления потоками и блокировками на низком уровне. Visual Studio 2012 и .NET Framework 4 улучшают поддержку параллельного программирования путем предоставления новой среды выполнения, новых типов библиотек класса (TPL) и новых средств диагностики (Concurrency Visualizer)[10-19]. Эти возможности упрощают параллельную разработку, что позволяет разработчикам писать эффективный, детализированный и масштабируемый параллельный код с помощью естественных выразительных средств без необходимости непосредственной работы с потоками или пулом потоков. На Рис. 1 представлен общая архитектура параллельного программирования в .NET Framework 4.

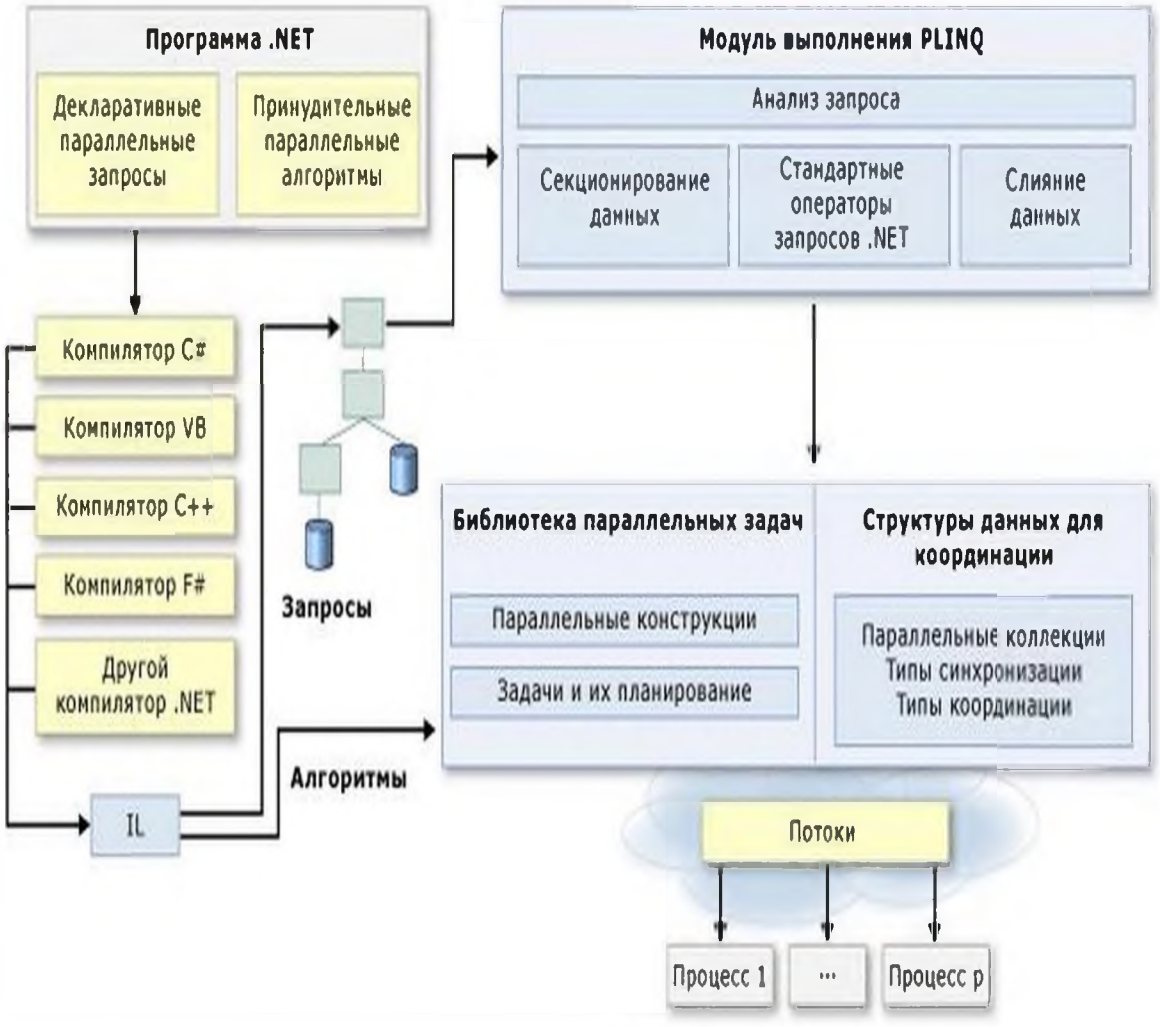

Рис. 1. Архитектура параллельного программирования в .NET Framework 4

II. ОСНОВНАЯ ЧАСТЬ

Библиотека PFX (Parallel Extensions to the .NET Framework) была разработана фирмой Microsoft. Данная библиотека позволяет распараллеливать задачи, в которых могут использоваться специальные координирующие структуры данных, таким образом, упрощая написание параллельных программ и обеспечивая увеличение производительности при увеличении числа ядер или числа процессоров. Существует две стратегии разделения работы между потоками[3-6]:

-параллелизм данных (data parallelism). Параллелизм данных используется, если необходимо над большим объемом данных выполнить некий набор задач. Поэтому этот подход называется параллелизмом данных, поскольку мы разбиваем данные между потоками;

-параллелизм задач (task parallelism). В противоположность параллелизму данных, используется параллелизме задач с распараллеливанием задачи, в таком случае каждый поток выполняет разную задачу.

Примечание. Параллелизм данных проще и лучше масштабируется на высокопроизводительном оборудовании, поскольку уменьшает или полностью устраняет совместное использование. Кроме того, параллелизм данных основывается на том факте, что данных обычно значительно больше, чем отдельных задач, что увеличивает возможности параллельной обработки.

Библиотека параллелизма задач (TPL).

Одним из самых главных среди нововведений, внедренных в среду .NET Framework 4.0, является библиотека распараллеливания задач (TPL). Эта библиотека усовершенствует многопоточное программирование двумя основными способами[16-22]. Во-первых, упрощает создание и применение многих потоков, во-вторых, позволяет автоматически использовать несколько процессоров.

Проще говоря, TPL предоставляет возможности для автоматического масштабирования приложений с целью эффективного использования ряда доступных процессоров.

Главной причиной появления нововведений, таких как TPL и PLINQ, служит возросшее значение параллелизма в современном программировании. В настоящее время многоядерные процессоры уже стали обычным явлением. Кроме того, постоянно растет потребность в повышении производительности программ. Все это, в свою очередь, вызвало растущую потребность в механизме, который позволял бы с выгодой использовать несколько процессов для повышения производительности программного обеспечения.

Библиотека TPL определена в пространстве имен System.Threading.Tasks (Рис. 2.). Но для работы с ней обычно требуется также включать в программу класс System.Threading, поскольку он поддерживает синхронизацию и другие средства многопоточной обработки, в том числе и те, что входят в класс Interlocked.

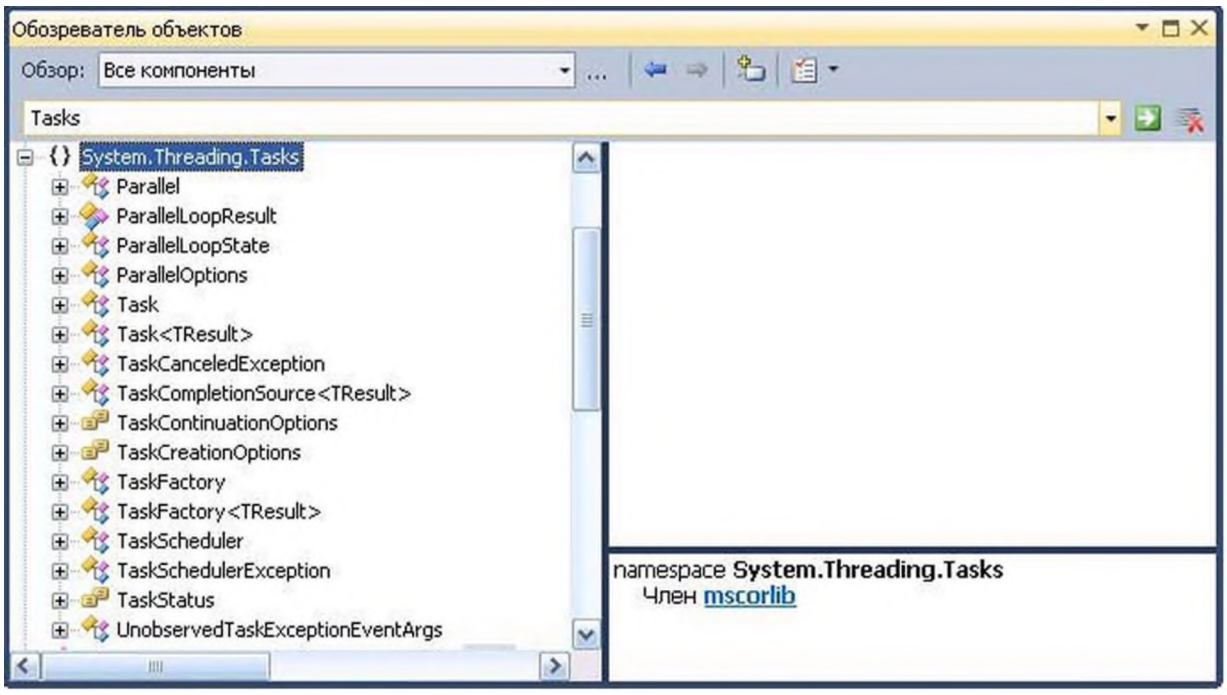

Рис. 2. Пространство имен System.Threading.Tasks

Библиотека TPL позволяет автоматически распределять нагрузку приложений между доступными процессорами в динамическом режиме, используя пул потоков CLR. Библиотека TPL занимается распределением работы, планированием потоков, управлением состоянием и прочими низкоуровневыми деталями. В результате появляется возможность максимизировать производительность приложений .NET. не имея дела со сложностями прямой работы с потоками.

Помимо всего прочего процессоры приближаются к пределу тактовой частоты. Исторически сложилось, что выполняется закон Мура: каждые 18 месяцев производительность вычислительных систем увеличивается в 2 раза. До XXI века производительность процессов увеличивалась за счет повышения тактовой частоты. Главным тормозом здесь служило энергопотребление: при увеличении тактовой частоты на 20 %, энергопотребление возрастало на 73 % [3]. Поэтому было решено пойти другим путем. Производители процессоров стали уделять больше внимания реализации параллелизма, нежели тактовой частоте процессоров.

Это достигается за счет реализации многоядерной архитектуры. Разместить на одном кристалле процессора несколько упрощенных процессоров, у которых тактовая частота была занижена, а функции АЛУ упрощены. Практически все современные устройства имеют многопроцессорную архитектуру (суперкомпьютеры, ПК, мобильные устройства)[15].

В связи с тем, что многопроцессорные системы, такие как суперкомпьютеры, стали широко распространены, а также существует множество задач, требующих больших вычислительных мощностей, возросла и роль методов параллельного программирования.

Сплайны имеют многочисленные применения как в математической теории, так и в прикладной математике (в частности, в разнообразных вычислительных программах). В частности, сплайны двух переменных интенсивно используются для задания поверхностей в различных системах компьютерного моделирования. Сплайны двух аргументов называют би-сплайнами (например, бикубический сплайн), которые являются двумерными сплайнами, моделирующими поверхности. Их часто путают с B-сплайнами (базисными сплайнами), которые являются одномерными и в линейной комбинации составляют кривые — каркас для «натягивания» поверхностей. Также из базисных сплайнов возможно составить трёхмерную конструкцию для моделирования объёмных тел[7,8,9].

Алгоритм вычисления можно разделить на две части:

- вычисление коэффициентов в узлах интерполяции.
- вычисление значений сплайна в заданной точке.

Первый этап не является таким трудоемким, по сравнению со вторым. Второй этап занимает гораздо большую часть вычислительного времени, поэтому выполнение второго этапа алгоритма распределяется между процессами.

Прежде всего пользователем производится ввод значений узлов входных данных. Затем, с помощью специальных директив MPI, специальной функции библиотеки PFX и TPL создается параллельная секция, которая будет выполняться на нескольких процессах, количество которых указывается пользователем при запуске приложения. После этого производится распределение данных между процессами. Пример распределения массива, состоящего из 100 элементов представлен на рисунке 3.

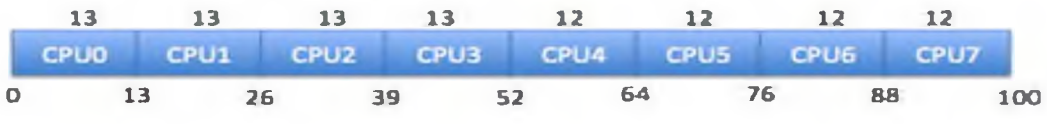

Рис. 3. Пример распределения данных между процессами

После того, как произведено распределение данных, на каждом процессе начинается выполнение вычисления значений между входными узлами интерполяции. Для этого, на каждом из узлов производится вызов последовательной функции, реализующей метод кубических сплайнов. После выполнения вычислений каждым из процессов производится барьерная блокировка, чтобы все процессы окончили вычисление, а после этого производится сборка результатов вычислений со всех процессов в один массив на одном процессе.

После того, как все данные собраны на одном процессе выполнение алгоритма окончено. Результаты выполнения программ на разных наборах данных приведены в таблице 1.

Таблица 1

Размерность	Последовательно (1 процесс), сек	2 процесса, сек	4 процесса, сек
1024	0.00128654	0.000809977	0.000403943
263000	0.0115795	0.00671721	0.00318956
33555000	0.98814	0.528883	0,261672
134218000	6.56067	3.44094	1.69863

Ускорение при выполнении программы на 2 и 4 процессах по сравнению с последовательным выполнением представлено в таблице 2.

Таблица 2

Размерность	Ускорение по сравнению с последовательным выполнением	
	2 процесса	4 процесса
1024	1,588	3,1849
263000	1,723	3,6304
33555000	1,872	3,776
134218000	1,907	3,8654

Соотношение эффективности выполнения программы на 2 и 4 процессах по сравнению с параллельным выполнением представлено в таблице 3.

Таблица 3

Размерность	Эффективность по сравнению с последовательным выполнением	
	2 процесса	4 процесса
1024	0,794	0,796225
263000	0,8615	0,9076
33555000	0,936	0,944
134218000	0,9535	0,96635

III. ЗАКЛЮЧЕНИЕ

Как видно из таблиц 2 и 3 с увеличением объема вычислений увеличивается ускорение вычислений, а также эффективность. Это обусловлено тем, что на создание процессов и обмен сообщениями между ними необходимы некоторые издержки. С увеличением объемов данных эти издержки становятся не такими существенными, как на небольших объемах.

Исследовав эффективность разработанного алгоритма, можно прийти к выводу, что использование технологии MPI, специальной функции библиотек PFX и TPL для создания параллельных приложений значительно повышает скорость вычислений, а ускорение растет с ростом количества вычислений, но достигает своего предела. Так же ускорение не может быть равным количеству вычислителей из-за издержек, как было сказано выше.

ЛИТЕРАТУРА

[1] Букатов А.А., Дацюк В.Н., Жегуло А.И. Программирование многопроцессорных вычислительных систем. – Ростов-на-Дону: ООО «ЦВВР», 2003. – 208 с. – ISBN 5-94153-062-5.

[2] Гэри М., Джонсон Д. Вычислительные машины и трудно решаемые задачи. – М.: Мир, 1982. – 416 с.

[3] Quinn, M. J. Parallel Programming in C with MPI and OpenMP. – New York, NY: McGraw-Hill, 2004.

[4] Barnard, S. PMRSB: Parallel multilevel recursive spectral bisection. In Proc. Supercomputing. 1995.

[5] Andrews, G. R. (2000). Foundations of Multithreaded, Parallel, and Distributed Programming. – Reading, MA: Addison-Wesley (русский перевод Эндрюс Г.Р. Основы многопоточного, параллельного и распределенного программирования. – М.: Издательский дом "Вильямс", 2003)

[6] Chandra, R., Dagum, L., Kohr, D., Maydan, D., McDonald, J., and Melon, R. Parallel Programming in OpenMP. Morgan Kaufmann Publishers, 2000.
[7] Х.З. Зайниддинов Сплайны в задачах цифровой обработки сигналов. // «Ташкент», 2015.-200 с.
[8] Х.З. Зайниддинов. Методы и средства обработки сигналов в кусочно-полиномиальных базисах. // «Ташкент», 2015.-70 с.
[9] Бахвалов Н.С., Жидков Н.П., Кобельков Г.М. Численные методы. – М.: Наука, 1987.
[10] Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. – СПб.: БХВ-Петербург, 2002.
[11] Корнеев В.В. Параллельное программирование в MPI. Москва-Ижевск: Институт компьютерных исследований. 2003.
[12] Немнюгин С., Стесик О. Параллельное программирование для многопроцессорных вычислительных систем – СПб.: БХВ-Петербург, 2002;
[13] Blake G., Dreslinski R.G., Mudge T. «A survey of multicore processors,» Signal Processing Magazine, vol. 26, no. 6, pp. 26-37, Nov. 2009;
[14] Jain A., Shankar R. Software Decomposition for Multicore Architectures, Dept. of Computer Science and Engineering, Florida Atlantic University, Boca Raton, FL, 33431.
[15] Зайнидинов Х.Н., Маллаев О.У. Параллельные алгоритмы обработки сейсмических сигналов на многоядерных процессорах. Automatics & software enginery. – 2018. –№1(23). – pp. 89-95.
[16] Mallaev O.U. Parallel methods and technologies interpolation of signals. International conference on importance of information communication technologies in innovative development of sector of economy dedicated to the 1235th anniversary of the birth of Muhammad al-Khwarithmi. April 5-6, 2018, Tashkent, Uzbekistan. 166-169.
[17] Зайнидинов Х.Н., Артикова М.А., Маллаев О.У. Распараллеливание цифровой обработки сигналов кубическими сплайнами. International conference on importance of information communication technologies in innovative development of sector of economy dedicated to the 1235th anniversary of the birth of Muhammad al-Khwarithmi. April 5-6, 2018, Tashkent, Uzbekistan. 293-296.
[18] Web-сайт OpenMP Architecture Review Board (ARB): openmp.org
[19] Web-сайт национальные открытый университет - https://www.intuit.ru
[20] Эра многоядерных энергоэффективных процессоров – [Электронный ресурс] – режим доступа http://compress.ru/article.aspx?id=16962 (Дата обращения: 03.06.2016).
[21] https://software.intel.com/en-us/intel-system-studio) – национальные открытый университет.