УДК 004.021

СРАВНЕНИЕ ЭФФЕКТИВНОСТИ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ С ПОМОЩЬЮ БИБЛИОТЕК PPL И TPL ДЛЯ РЕШЕНИЯ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ

Русецкий А.Д.

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Научный руководитель: Логинова И.П. – к. т. н, доцент кафедры ЭИ

Аннотация. В статье проводится сравнительный анализ эффективности двух библиотек параллельного программирования - Parallel Patterns Library (PPL) для C++ и Task Parallel Library (TPL) для C#/.NET - при решении систем линейных алгебраических уравнений. Исследование включает тестирование на системах размерностью 500×500, 1000×1000 и 1500×1500 с использованием метода Гаусса, метода Гаусса-Жордана и метода Якоби.

Ключевые слова. Параллельные вычисления, системы линейных уравнений, метод Гаусса, PPL, TPL, сравнительный анализ производительности, масштабируемость алгоритмов, многопоточное программирование.

Введение. Развитие многопроцессорных вычислительных систем и многоядерных процессоров обусловило необходимость разработки эффективных параллельных алгоритмов для решения задач вычислительной математики. В контексте решения систем линейных алгебраических уравнений (СЛАУ), которые являются фундаментальной проблемой в научных вычислениях, инженерном анализе и машинном обучении, параллелизация представляет особый интерес ввиду высокой вычислительной сложности традиционных алгоритмов.

Современные технологии параллельного программирования предлагают различные подходы к реализации параллельных вычислений. В экосистеме Microsoft разработчикам доступны две ключевые технологии: Parallel Patterns Library (PPL) для C++ и Task Parallel Library (TPL) для C#/.NET. Несмотря на схожие концептуальные основы, эти библиотеки существенно различаются по своей архитектуре, уровню абстракции и механизмам управления потоками, что может приводить к различиям в производительности при решении однотипных вычислительных задач.

Основная часть. В настоящей статье проводится сравнительный анализ эффективности PPL и TPL на примере реализации параллельных алгоритмов решения СЛАУ больших размерностей (500×500, 1000×1000 и 1500×1500). Основное внимание уделяется исследованию масштабируемости алгоритмов при различном количестве вычислительных потоков (1, 2, 4, 8, 12), анализу ускорения вычислений и оценке эффективности использования вычислительных ресурсов.

Для обеспечения корректности сравнения были разработаны идентичные алгоритмические реализации метода Гаусса с выбором главного элемента и методом Якоби на обеих платформах [1]. Алгоритмы включают как последовательные, так и параллельные версии с использованием технологии многопоточных вычислений, предоставляемых соответствующими библиотеками.

Перед тем, как подробно приступить к исследованию, рассмотрим по отдельности каждую из библиотек: PPL и TPL.

Parallel Patterns Library (PPL) для C++. Parallel Patterns Library (PPL) – это высокоуровневая библиотека параллельного программирования, разработанная Microsoft для языка C++ и интегрированная в среду разработки Visual Studio. PPL предоставляет модель параллелизма, основанную на задачах (task-based), и поддерживает различные параллельные шаблоны для упрощения разработки многопоточных приложений [2].

Архитектурные особенности:

1 *Интеграция с C++ Standard Library*: PPL тесно интегрирована со стандартной библиотекой C++ и использует идиомы современного C++ (лямбда-выражения, шаблоны).

2 *Модель выполнения на основе задач*: основная абстракция – задача (task), которая представляет собой единицу работы, планируемую для выполнения в пуле потоков.

3 *Автоматическое управление потоками*: библиотека автоматически управляет созданием и уничтожением потоков через внутренний планировщик задач.

4 *Поддержка различных параллельных шаблонов*:

- parallel_for – параллельное выполнение циклов;
- parallel_for_each – параллельная обработка контейнеров;
- parallel_invoke – параллельное выполнение нескольких функций;
- task_group – группировка связанных задач.

Task Parallel Library (TPL) для C#/.NET. Task Parallel Library (TPL) – это набор компонентов .NET Framework и .NET Core, предоставляющих модели параллелизма и распараллеливания для языков семейства .NET. TPL абстрагирует низкоуровневые детали управления потоками, позволяя разработчикам сосредоточиться на логике приложения [3].

Архитектурные особенности:

1 *Интеграция с языком C#*: глубокая интеграция с языковыми конструкциями C#, включая ключевые слова async/await и LINQ.

2 *Иерархия абстракций*:

- Task – базовая единица асинхронной работы;
- Parallel – высокоуровневые параллельные конструкции;
- PLINQ – параллельные LINQ-запросы.

3 *Work-stealing алгоритмы*: использует расширенные алгоритмы кражи работы для балансировки нагрузки между потоками.

4 *Поддержка отмены и продолжений*: встроенные механизмы для отмены задач и определения зависимостей между ними.

Сравнение эффективности вычислений библиотек. Для сравнения эффективности параллельных вычислений потребуется единый набор входных данных – коэффициентов для систем линейных алгебраических уравнений. Для этого на языке C++ была написана программа, генерирующая матрицу случайных коэффициентов и вектор правых частей уравнений для СЛАУ размерностями 500×500, 1000×1000 и 1500×1500. Файлы с этими параметрами в последующем считываются и обрабатываются двумя программами, использующими методы библиотек PPL и TPL для параллельных вычислений [4]. В ходе сравнения эффективности параллельных вычислений этих библиотек для решения СЛАУ был использован ПК с конкретными характеристиками (таблица 1).

Таблица 1 – Технические характеристики ПК

Операционная система	Windows 10.0.19045
Версия C++	14
Версия .Net C#	5.0.201
Размер ОЗУ	16 ГБ
CPU	11th Gen Intel(R) Core(TM) i5-1135G7 @2.40GHz
Количество ядер	4
Количество логических процессоров	8

Для решения СЛАУ каждая из программ с библиотеками PPL и TPL использует одинаковые математические методы и конфигурации потоков:

1 Последовательное решение СЛАУ методом Гаусса [5].

Метод состоит из прямого и обратного ходов. Прямой ход заключается в сведении расширенной матрицы системы уравнений к треугольному виду путем применения эквивалентных преобразований.

Обратный ход заключается в вычислении суммы a[i][j]*x[j] для каждой из n строк, начиная с последней. При этом обрабатываются j только большие чем i (элементы выше главной диагонали). Асимптотическая сложность такого алгоритма – O(n*n).

Таким образом, распараллеливать целесообразно только прямой ход. Даже если параллельно выполнять вычисления обратного хода с нулевыми издержками (на создание потоков и синхронизацию), например, на 4 ядрах – то теоретическое ускорение составит 0,4%. Для эффективного распараллеливания прямого хода, достаточно использовать в нем параллельную функцию триангуляции.

2 Параллельное решение СЛАУ методом Гаусса-Жордана [6] в конфигурациях на 1, 2, 4, 8 и 12 потоков.

Метод Гаусса – Жордана является модификацией метода Гаусса. В отличие от метода Гаусса, который состоит из двух этапов, метод Гаусса – Жордана выполняется за один проход по столбцам. Как и в методе Гаусса для решения СЛАУ расширенная матрица системы преобразуется с помощью таких операций: смена мест двух строк; умножение всех элементов строки на некоторое число, не равное нулю; прибавление к элементам одной строки соответствующих элементов другой строки, умноженных на любой множитель.

Как и в методе Гаусса, можно менять местами и столбцы матрицы системы, но нужно запоминать новый порядок переменных в уравнениях.

3 Параллельное решение СЛАУ методом Якоби [7] в конфигурации на 8 потоков.

Метод Якоби сильно отличается от таких методов как Гаусса или Камера, т.к. является итеративным. Если для метода Гаусса можно заранее определить примерное количество выполняемых операций, то для итеративных методов сделать это нельзя. Итеративный процесс поиска решения предполагает повышение точности с каждой последующей итерацией.

СЛАУ задается матрицей коэффициентов A и векторами X (неизвестных) и B (свободных членов).

Алгоритм поиска корней можно описать следующим образом:

– ввод размерности (n), точности (eps), матриц системы, приближенных значений неизвестных (элементы начального приближения могут быть равно нулю) и свободных членов;

– расчет приближения – вектора (формула 1):

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{pmatrix} \quad (1)$$

где $y_i = \frac{1}{a_{i,i}} * \left(b_i - \sum_{j=0, j \neq i}^{n} \left(a_{i,j} * x_i\right)\right)$, для $i = j = 1 \dots n$;

– вычисление погрешности текущего приближения (формула 2):

$$eact = \max(|y_i - x_i|), i = 1 \dots n; \quad (2)$$

– если eact > e, то X = Y и переход на п.2

– конец – результат работы алгоритма – вектор X.

Результаты сравнения скорости параллельных вычислений, а значит и эффективности указанных библиотек, представлены в таблице 2.

Таблица 2 – Результаты сравнения скорости решения СЛАУ библиотеками PPL и TPL

Размерность СЛАУ	Метод решения	Конфигурация потоков	Время решения, с.	
			PPL C++	TPL .Net C#
1	*2*	*3*	*4*	*5*
500×500	Гаусса	1	10,322	0,694
	Гаусса-Жордана	1	2,289	1,047
		2	2,279	0,634
		4	2,231	0,516
		8	2,236	0,517
		12	2,297	0,475
	Якоби	8	0,051	0,016
1000×1000	Гаусса	1	73,41	4,756
	Гаусса-Жордана	1	16,607	7,346
		2	16,556	4,166
		4	16,567	3,018
		8	16,571	2,566
		12	16,558	2,685
	Якоби	8	0,196	0,039
1500×1500	Гаусса	1	247,639	15,758
	Гаусса-Жордана	1	54,847	23,277
		2	55,374	11,981
		4	55,390	8,595
		8	55,243	7,494
		12	54,933	7,395
	Якоби	8	0,394	0,065

Для удобства восприятия, отобразим полученные результаты замеров времени вычислений СЛАУ с помощью графика (рисунок 1). Для большей читаемости, линии графиков одной и той же технологии, но с разным числом используемых потоков и минимальным отличием в скорости вычислений, были объединены в одну линию.

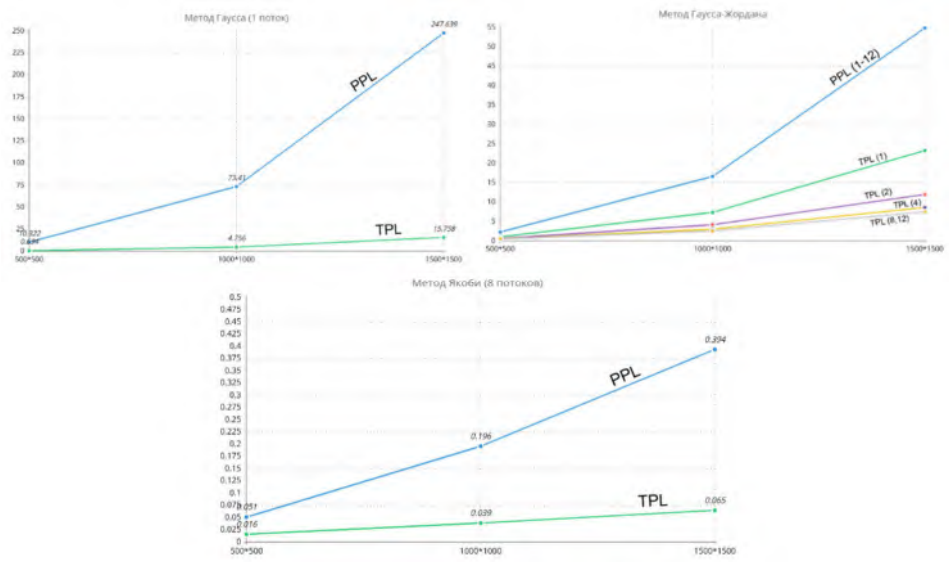

Рисунок 1 – Графики времени вычислений СЛАУ с помощью разных математических методов и конфигураций потоков

Заключение. Проведенное сравнительное исследование эффективности библиотек PPL для C++ и TPL для C#/.NET при решении систем линейных алгебраических уравнений позволило получить значимые результаты, демонстрирующие различные аспекты производительности этих технологий параллельного программирования.

Анализ экспериментальных данных выявил преимущество в скорости выполнения вычислений решений на основе TPL (.NET C#) по абсолютному времени выполнения для всех рассматриваемых размерностей СЛАУ. Наилучшие показатели быстродействия показывает

метод Гаусса-Жордана с 12 потоками: TPL демонстрирует ускорение в 4.8 раза для системы 500×500, в 6.2 раза для 1000×1000 и в 7.4 раза для 1500×1500 по сравнению с PPL. Это свидетельствует о более эффективной реализации механизмов параллельных вычислений.

Особого внимания заслуживает эффективность метода Якоби, где TPL показывает преимущество более чем в 3 раза для всех размерностей. Это указывает на оптимальную работу алгоритмов балансировки нагрузки и управления потоками в TPL, особенно для итерационных методов с высокой степенью параллелизма.

Наблюдаемое превосходство TPL можно объяснить несколькими факторами:

1 Оптимизированная работа планировщика задач в .NET, эффективно использующая work-stealing алгоритмы для балансировки нагрузки между потоками.

2 Современные JIT-оптимизации в среде выполнения .NET, которые адаптируют код под конкретную аппаратную платформу в процессе выполнения.

3 Эффективное управление памятью через сборщик мусора, минимизирующее фрагментацию памяти при работе с большими массивами данных.

4 Низкие накладные расходы на синхронизацию потоков благодаря усовершенствованным примитивам синхронизации в последних версиях .NET.

Полученные результаты показывают преимущество нативных решений на C++ над управляемым кодом C# в скорости решения задач вычислительной математики. Тем не менее, выбор языка программирования и соответствующего стека технологий должен зависеть от решаемой задачи.

Список литературы

1. Самарский, А. А. Численные методы : учебник для вузов / А. А. Самарский, А. В. Гулин. – Москва : Наука, 2019. – 432 с. – ISBN 978-5-02-040123-4.

2. Microsoft Documentation [Электронный ресурс]. – Parallel Patterns Library (PPL). – Режим доступа: https://docs.microsoft.com/en-us/cpp/parallel/concrt/parallel-patterns-library-ppl. Дата доступа: 03.01.2026.

3. Microsoft Documentation [Электронный ресурс]. – Task Parallel Library (TPL). – Режим доступа: https://docs.microsoft.com/en-us/dotnet/standard/parallel-programming/task-parallel-library-tpl. Дата доступа: 03.01.2026.

4. Гергель, В. П. Теория и практика параллельных вычислений / В. П. Гергель. – 3-е изд., перераб. и доп. – Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), 2021. – 512 с. – ISBN 978-5-9556-0152-7.

5. Фалейчик, Б. В. Вычислительные методы алгебры: базовые понятия и алгоритмы : учеб.-метод. пособие / Б. В. Фалейчик. – Минск : БГУ, 2010. – 42 с.

6. Лунгу, К. Н. Высшая математика : руководство к решению задач : учеб. пособие. В 2 ч. Ч. 1 / К. Н. Лунгу, Е. В. Макаров. – Москва : ФИЗМАТЛИТ, 2009. – 216 с.

7. Масловская, Л. В. Численные методы алгебры : учеб. пособие / Л. В. Масловская, О. М. Масловская. – Одесса : Укрполиграф, 2006. – 146 с.

UDC 004.021

COMPARISON OF THE EFFICIENCY OF PARALLEL COMPUTING USING THE PPL AND TPL LIBRARIES FOR SOLVING SYSTEMS OF LINEAR ALGORITHMIC EQUATIONS

Rusetskii A.D.

Belarusian State University of Informatics and Radioelectronics, Minsk, Republic of Belarus

Loginova I.P. – PhD in Technical Science, associate professor of the department of Economic Informatics

Annotation. This article provides a comparative analysis of the efficiency of two parallel programming libraries – Parallel Patterns Library (PPL) for C++ and Task Parallel Library (TPL) for C#/.NET – in solving systems of linear algebraic equations. The study includes testing on systems of dimensions 500×500, 1000×1000 and 1500×1500 using the Gaussian method, the Gauss-Jordan method and the Jacobi method.

Keywords. Parallel computing, systems of linear equations, Gaussian method, PPL, TPL, performance benchmarking, algorithm scalability, multithreaded programming.