УДК 004.432

ПАРАЛЛЕЛЬНАЯ РЕАЛИЗАЦИЯ ПОИСКА ОПРЕДЕЛИТЕЛЯ МАТРИЦЫ НА ЯЗЫКЕ C#

Казачёнок М.С.

Научный руководитель – Прихожий А.А., профессор, д.т.н.

Для начала хочется сказать, что данная работа является небольшой частью моей магистерской диссертации. В магистерском исследовании я оптимизирую время поиска обратной матрицы разными методами с использованием разных инструментов. И нахождение определителя матрицы является первым шагом для поиска обратной матрицы. Если матрица является вырожденной, то есть определитель матрицы равен нулю, то обратной матрицы не существует.

Определитель матрицы высокого порядка[1-2] можно вычислить, применив рекурсивную формулу, представленную на рисунке 1, где M является дополнительным минором к элементу a_{1j}. Минор, в свою очередь это определитель матрицы без 1-ой строки и j-го столбца, вот здесь и появится рекурсия. Такая формула называется разложением по строке.

$$\Delta = \sum_{j=1}^{n} (-1)^{1+j} a_{1j} \bar{M}_j^1$$

Рисунок 1 - Нахождение определителя матрицы n-го порядка

Сравнениеалгоритмов вычисления определителя матрицы по времени выполнения [3] проведено при рассмотрении трех способов реализации алгоритмов:

1. последовательное выполнение;
2. параллельное выполнение с использованием библиотеки TPL [4-6] языка программирования C#;
3. параллельное выполнение с использованием класса Thread [4-6], представляющего потокимногопоточного приложения.

При проведении экспериментов двумерные массивы заполнялись случайными числами в диапазоне от -20 до 20.

В параллельных реализациях, распараллеливание кода выполнено в одинаковых местах, а именно при нахождении дополнительного минора к элементу a_{1j}.

Во втором подходе,где использовалась библиотека TPL, был использован метод Parallel.For, который позволяет выполнять итерации цикла параллельно. Он имеет следующее определение: For(int, int, Action<int>), где первый параметр задает начальный индекс элемента в цикле, а второй параметр - конечный индекс. Третий параметр - делегат Action - указывает метод, который будет выполняться на каждой итерации.

В третьем случае, при применении класса Thread для создания нового потока, сначала был использован делегат ThreadStart, получающий в качестве параметра метод, который необходимо вызывать. При этом делегат ThreadStart запускает метод, который не содержит параметры.

А что, если необходимо передать какие-нибудь параметры в метод, который выполняется в потоке?Для этой цели используется делегат ParameterizedThreadStart. Данный делегат идентичен делегату ThreadStart, но при запуске потока необходимо передать переменную, значение которой хотим транспортировать в поток. Такой подход имеет ограничение: можно запускать в потоке только такой метод, который в качестве единственного параметра принимает объект типа object. Из-за этого ограничения, в методе требуется дополнительно привести переданное значение к целевому типу, например, к int, которое будет использоваться в вычислениях.

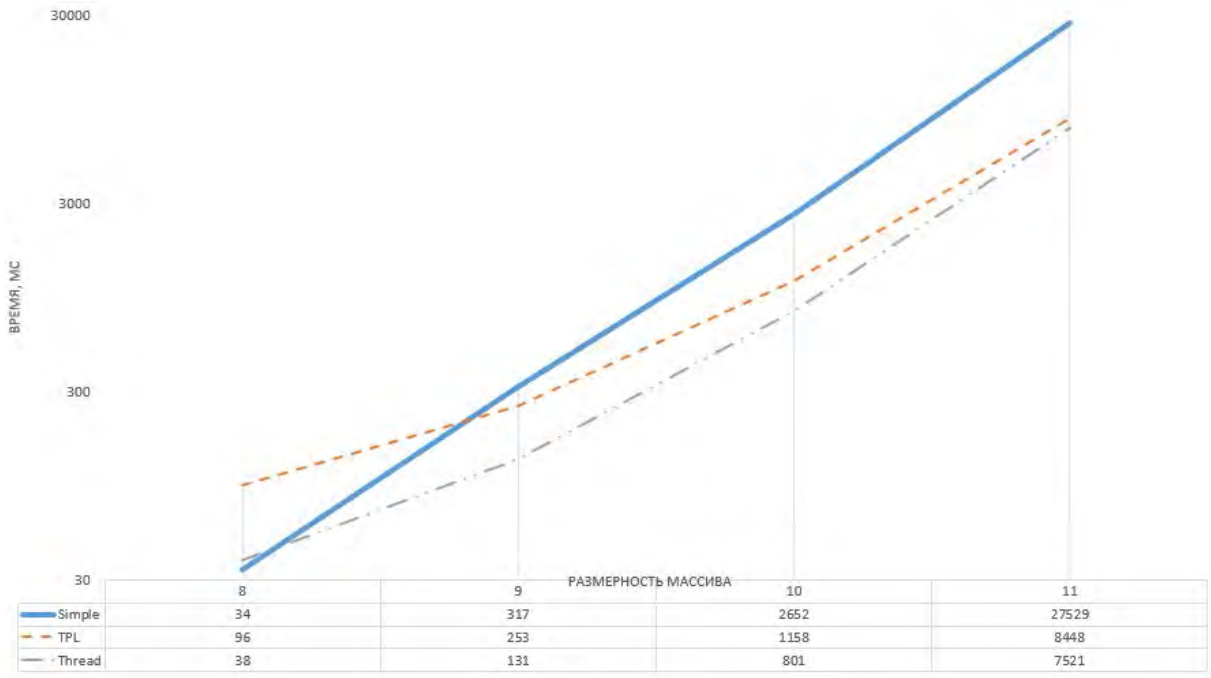
Но что делать, если нам надо передать не один, а несколько параметров различного типа? В этом случае необходимо передавать объект, который будет содержать все необходимые параметры. Но тут есть еще одно ограничение: метод Thread.Start не является типо-безопасным, то есть мы можем передать в него любой тип, и потом придется приводить переданный объект к нужному нам типу. Для решения данной проблемы рекомендуется объявлять все используемые методы и переменные в специальном классе, а в основной программе запускать поток через ThreadStart.

Зависимость времени выполнения трех способов вычисления определителя от размера матрицы представлена в виде графика на рисунке 2. На горизонтальной оси указана порядок (размерность) матрицы, на вертикальной оси указано время выполнения в мс.

Рисунок 2 показывает, что при небольшом порядке (8-ой порядок и ниже) последовательный код работает быстрее, чем параллельные реализации (с помощью TPLи Thread). При размерности матрицы более чем 9, параллельный код выполняется быстрее.

Если сравнивать время выполнения параллельных реализаций, то в самом начале параллельное выполнение с использованием Threadпроисходит быстрее по сравнению с использованием средств TPL. При постепенном увеличении порядка матрицы, разница во времени выполнения уменьшается.

В итоге, можно сделать вывод о том, что задача вычисления определителя матрицы является вычислительно емкой, требующей применения параллельных вычислений на высокопроизводительных многопроцессорных системах.



	8	9	10	11
Simple	34	317	2652	27529
TPL	96	253	1158	8448
Thread	38	131	801	7521

Рисунок 2 – Графики времени выполнения трех вариантов программы

Литература

1. Матрицы и определители. Аналитическая геометрия на плоскости и в пространстве: пособие / В.К. Пчельник, Е.А. Сетько, И.Н. Ревчук. – Гродно: ГрГУ, 2007. — 164 с.

2. Конев, В.В. Линейная алгебра. Учебное пособие. – Томск. Изд. ТПУ. 2008 – 65стр.

3. Prihozhy, A.A. Analysis, transformation and optimization for high performance parallel computing / A.A. Prihozhy // Minsk: BNTU. – 2019. – 229 p.

4. Троелсен, Э., Джепикс, Ф. Язык программирования C# 7 и платформы .NET и .NET Core, 8-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2018 — 1328 с.

5. Голдштейн, С., Зурбалев, Д., Флатов, И. и др. Оптимизация приложений на платформе .NET. – Пер. с англ. Киселев А.Н. – М: ДМК Пресс, 2014. – 524 с.

6. Скит, Д. C# для профессионалов: тонкости программирования, 3-е изд.: Пер. с англ. — М.: ООО “И.Д. Вильямс”, 2014. – 608 с.