Секция 1
ИНФОРМАЦИОННО-КОММУНИКАЦИОННЫЕ ТЕХНОЛОГИИ И СИСТЕМЫ ПРОГРАММИРОВАНИЯ

УДК 004.424

ЭФФЕКТИВНОСТЬ МЕТОДОВ ПАРАЛЛЕЛЬНОГО ВЫПОЛНЕНИЯ КОДА В ЯЗЫКЕ ПРОГРАММИРОВАНИЯ C#

студент Д. А. НИКИТИН, канд. пед. наук, доц. Т. А. ПАРАФИЯНОВИЧ
(Белорусский государственный университет информатики и радиоэлектроники, Минск)

Аннотация. *В статье представлена информация о параллельном программировании в языке C#, методах параллельного выполнения кода: многопоточности (класс Thread) и библиотеке TPL (класс Task), проведён анализ и сравнение методов, в качестве критерия для сравнения определено количество параллельных задач и производительность устройства, рассмотрены временные затраты и определён наиболее эффективный метод параллельного выполнения кода в языке программирования C#.*

Ключевые слова. *Параллельное программирование, многопоточность, библиотека TPL, производительность устройства, временные затраты, эффективный метод.*

При разработке программ, насыщенных большим функционалом, возникает вопрос о качестве и скорости работы программы. В настоящее время для программ различного вида и назначения критически важно время выполнения запросов, скорость работы системы и отзывчивость пользовательского интерфейса. Внедрение в программное средство параллельности выполнения задач может повысить быстродействие.

Для решения алгоритмов с большой временной сложностью в языке программирования C# существует проблема блокирования графического интерфейса, связанная с использованием основного потока, отвечающего за работу графического интерфейса. Параллельность в программировании – способ организации компьютерных вычислений, при котором для выполнения задачи выделяется свободный ресурс, не затрагивающий выполняющиеся в этот момент задачи [1]. При разработке программного средства была необходимость внедрения параллельности, что обосновывалось наличием большого количества

сущностей в базе данных и подходом по выгрузке этих данных на сторону клиента. Подход заключался в единовременной выгрузке данных при запуске программы, а из-за большого объёма выгружаемой информации пользовательский интерфейс зависал.

Язык программирования C# имеет несколько методов параллельного выполнения кода, в работе рассматриваются следующие: многопоточность (класс Thread) и библиотека TPL (класс Task). Каждый из этих методов имеет уникальные особенности, но в данной ситуации рассматривается функционал, позволяющий параллельно выполнять несколько задач. Первым для решения этой проблемы был выбран метод многопоточности (класс Thread) – использования потоков – последовательность программных команд, которые могут выполняться одновременно и независимо управляться планировщиком [2]. Использование метода многопоточности решило проблему с отзывчивостью пользовательского интерфейса, но при проведении системного тестирования было выявлено, что некоторые потоки не выполняются, следовательно, происходила потеря данных, однако, при этом устройство полностью соответствовало системным требованиям. Для решения этой задачи был использован метод параллельного выполнения кода – TPL (Task Parallel Library) библиотека параллельных задач, а именно набор инструкций, позволяющий распараллелить задачи и выполнять их сразу на нескольких процессорах [3]. С использованием класса Task для распараллеливания задач проблема была устранена.

Исходя из вышесказанного: класс Task имеет преимущество над классом Thread в виде гарантированного выполнения всех распараллеленных задач. Так как оба метода направлены на реализацию параллельного выполнения кода, то есть смысл сравнить их в различных ситуациях, для определения более эффективного метода. В качестве критериев для сравнения будут выступать: количество параллельных задач и производительность устройства. Эффективность будет определяться временем выполнения кода, задачей для параллельного выполнения выбран бинарный поиск.

Для определения временных затрат был использован класс StopWatch, для измерения производительности устройства задействована виртуальная машина VirtualBox, в качестве операционной системы выбрана Windows 10. С исходным кодом тестирующего модуля можно ознакомиться в репозитории GitHub «denden1s/Efficiency-of-Tasks-and-Threads».

На основании критериев, включающих количество решаемых программой задач, мощности процессора, проведены тесты и составлена таблица временных затрат.

Таблица 1. – Временные затраты на выполнение задач

Кол-во ядер CPU	1	2	3	4	5	6	7	8	Кол-во задач
Thread	0,042	0,054	0,049	0,063	0,061	0,062	0,056	0,063	100
Task	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	
Thread	1,025	0,462	0,646	0,603	0,577	1,0481	0,597	0,645	1000
Task	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	
Thread	4,844	5,037	7,151	6,035	6,831	6,281	7,181	6,389	10 000
Task	0,006	0,006	0,006	0,006	0,005	0,005	0,005	0,005	
Thread	49,201	48,772	61,251	63,026	69,727	67,402	–	67,518	100 000
Task	0,177	0,061	0,061	0,066	0,07	0,604	0,161	0,079	

Исходя из временных затрат на выполнение задач (таблица 1) можно сделать вывод, что использование библиотеки TPL эффективнее по временным ресурсам, особенно это становится заметно при повышении количества задач для обработки. В результате тестирования выявлено, что при повышении количества задач в многопоточном подходе обрабатываются не все потоки (остаются не обработанными 5–10%).

Таким образом в ходе исследования были рассмотрены методы параллельного программирования в языке C#, рассмотрены их временные затраты и определён наиболее эффективный метод на основании этого показателя. В ходе применения метода многопоточности выявлена проблема невыполнения потоков. Наиболее эффективным методом параллельного выполнения кода в языке программирования C# определён метод реализации библиотеки параллельного программирования (TPL).

ЛИТЕРАТУРА

1. Параллельные вычисления [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Параллельные_вычисления. – Дата доступа: 20.12.2021.
2. C# What is the difference between threads and tasks [Электронный ресурс]. – Режим доступа: https://peterdaugaardrasmussen.com/2018/11/08/csharp-what-is-the-difference-between-threads-and-tasks. – Дата доступа: 05.01.2022.
3. Параллельное программирование и библиотека TPL [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/tutorial/12.1.php. – Дата доступа: 05.01.2022.