**Дополнительные вопросы и упражнения**

1. Какими достоинствами и недостатками обладает каждый вариант распараллеливания?

Замеры делались для поиска простых чисел от 2 до 400.000.000.

|  |  |  |
| --- | --- | --- |
| Вариант распраллеливания | Достоинства | Недостатки |
| Декомпозиция по данным | 7.8 сек  Сбалансированное распраллеливание в нашем случае.  Не требуется синхронизация | В других случаях распараллеливание может быть несбалансированным. |
| Декомпозиция набора простых чисел | 11.6 сек  Не требуется синхронизация | Несбалансированное распараллеливание. |
| Применение пула потоков | 7.5 сек  Динамическая декомпозиция, следовательно, сбалансированное распараллеливание | Требуется синхронизация счетчика потоков или ожидание событий завершения от потоков. |
| Последовательный перебор простых чисел | 7.2 сек  Динамическая декомпозиция, следовательно, сбалансированное распараллеливание | Требуется синхронизация для доступа к общему ресурсу |

2. \*Какие средства синхронизации можно использовать вместо конструкции lock? Какой вариант будет более эффективным?

Monitor, Mutex, Interlocked. Наиболее эффективные: Interlocked , Monitor, lock.

3. \*Какой вариант ожидания завершения работ, запущенных пулом потоков, более эффективный и почему?

C помощью событий – эффективнее, т.к. в этом случае ожидающий главный поток фактически не работает. Но количество event – ограниченно 64.

С помощью счетчиков потоков – главному потоку нужно в цикле проверять счетчик.

4. \*Реализуйте один или несколько вариантов распараллеливания с помощью объектов Task и с помощью метода Parallel.For. Выполните эффективность алгоритмов.

Parallel.For показал лучшие результаты 5.9 сек. Синхронизация и потокобезопасная коллекция не применялась.

Task – 7.7 сек. – примерно как при Декомпозиции по данным, т.к. этот же алгоритм применялся и как у ThreadPool.

5. \*Реализуйте алгоритм поиска простых чисел как LINQ-запрос к массиву чисел.

//