**Отчет**

В данной работе для оценки производительности работы использовалось среднеарифметическое значение времени выполнения для каждого задания, число экспериментов принималось равным expCount = 10. При оценке среднего значения исключался первый разгадочный прогон.

**Задание №1:**

Реализуйте последовательную обработку элементов вектора, например, умножение элементов вектора на число. Число элементов вектора задается параметром N(vectorSize).

Результат: эксперимент - 1, число потоков - 1, число элементов - 1000, тип функции - простая, время выполнения - 13,5339 ms

Результат: эксперимент - 2, число потоков - 1, число элементов - 1000, тип функции - простая, время выполнения - 12,5359 ms

Результат: эксперимент - 3, число потоков - 1, число элементов - 1000, тип функции - простая, время выполнения - 8,8067 ms

Результат: эксперимент - 4, число потоков - 1, число элементов - 1000, тип функции - простая, время выполнения - 5,4874 ms

Результат: эксперимент - 5, число потоков - 1, число элементов - 1000, тип функции - простая, время выполнения - 7,6774 ms

Результат: эксперимент - 6, число потоков - 1, число элементов - 1000, тип функции - простая, время выполнения - 7,536 ms

Результат: эксперимент - 7, число потоков - 1, число элементов - 1000, тип функции - простая, время выполнения - 4,3163 ms

Результат: эксперимент - 8, число потоков - 1, число элементов - 1000, тип функции - простая, время выполнения - 6,4828 ms

Результат: эксперимент - 9, число потоков - 1, число элементов - 1000, тип функции - простая, время выполнения - 5,2296 ms

Результат: эксперимент - 10, число потоков - 1, число элементов - 1000, тип функции - простая, время выполнения - 5,5317 ms

Среднее: число потоков - 1, число элементов - 1000, тип функции - простая, время выполнения - 6,36038 ms

В дальнейших примерах для оценки производительности будут приводиться только среднее значение выполнения.

**Задание №2:**

Реализуйте многопоточную обработку элементов вектора, используя разделение вектора на равное число элементов. Число потоков задается параметром M(thredsCounts).

Среднее: число потоков - 1, число элементов - 1000, тип функции - простая, время выполнения - 6,36038 ms

Среднее: число потоков - 2, число элементов - 1000, тип функции - простая, время выполнения - 24,71346 ms

Среднее: число потоков - 4, число элементов - 1000, тип функции - простая, время выполнения - 34,01442 ms

Среднее: число потоков - 8, число элементов - 1000, тип функции - простая, время выполнения - 96,43132 ms

**Задание №3:**

Выполните анализ эффективности многопоточной обработки при разных параметрах N (10, 100, 1000, 100000) и M (1, 2, 4, 8). Результаты представьте в табличной форме.

Среднее: число потоков - 1, число элементов - 10, тип функции - простая, время выполнения - 7,93996 ms

Среднее: число потоков - 2, число элементов - 10, тип функции - простая, время выполнения - 19,89064 ms

Среднее: число потоков - 4, число элементов - 10, тип функции - простая, время выполнения - 59,37617 ms

Среднее: число потоков - 8, число элементов - 10, тип функции - простая, время выполнения - 96,13029 ms

Среднее: число потоков - 1, число элементов - 100, тип функции - простая, время выполнения - 7,05042 ms

Среднее: число потоков - 2, число элементов - 100, тип функции - простая, время выполнения - 17,34901 ms

Среднее: число потоков - 4, число элементов - 100, тип функции - простая, время выполнения - 46,69843 ms

Среднее: число потоков - 8, число элементов - 100, тип функции - простая, время выполнения - 80,69505 ms

Среднее: число потоков - 1, число элементов - 1000, тип функции - простая, время выполнения - 6,36038 ms

Среднее: число потоков - 2, число элементов - 1000, тип функции - простая, время выполнения - 24,71346 ms

Среднее: число потоков - 4, число элементов - 1000, тип функции - простая, время выполнения - 34,01442 ms

Среднее: число потоков - 8, число элементов - 1000, тип функции - простая, время выполнения - 96,43132 ms

Среднее: число потоков - 1, число элементов - 100000, тип функции - простая, время выполнения - 14,83994 ms

Среднее: число потоков - 2, число элементов - 100000, тип функции - простая, время выполнения - 26,64518 ms

Среднее: число потоков - 4, число элементов - 100000, тип функции - простая, время выполнения - 68,77899 ms

Среднее: число потоков - 8, число элементов - 100000, тип функции - простая, время выполнения - 123,60953 ms

**Задание №4:**

Выполните анализ эффективности при усложнении обработки каждого элемента вектора. Масштабный коэффициент k при выполнении усложненной обработки задается случайно при каждом запуске программы new Random().Next(2, 25).

Среднее: число потоков - 1, число элементов - 10, тип функции - параметрируемая (k=8), время выполнения - 3,88332 ms

Среднее: число потоков - 2, число элементов - 10, тип функции - параметрируемая (k=8), время выполнения - 14,38632 ms

Среднее: число потоков - 4, число элементов - 10, тип функции - параметрируемая (k=8), время выполнения - 37,97285 ms

Среднее: число потоков - 8, число элементов - 10, тип функции - параметрируемая (k=8), время выполнения - 95,96129 ms

Среднее: число потоков - 1, число элементов - 100, тип функции - параметрируемая (k=8), время выполнения - 5,11633 ms

Среднее: число потоков - 2, число элементов - 100, тип функции - параметрируемая (k=8), время выполнения - 19,40916 ms

Среднее: число потоков - 4, число элементов - 100, тип функции - параметрируемая (k=8), время выполнения - 40,89459 ms

Среднее: число потоков - 8, число элементов - 100, тип функции - параметрируемая (k=8), время выполнения - 96,68872 ms

Среднее: число потоков - 1, число элементов - 1000, тип функции - параметрируемая (k=8), время выполнения - 32,67438 ms

Среднее: число потоков - 2, число элементов - 1000, тип функции - параметрируемая (k=8), время выполнения - 41,4002 ms

Среднее: число потоков - 4, число элементов - 1000, тип функции - параметрируемая (k=8), время выполнения - 77,0966 ms

Среднее: число потоков - 8, число элементов - 1000, тип функции - параметрируемая (k=8), время выполнения - 134,05067 ms

Среднее: число потоков - 1, число элементов - 100000, тип функции - параметрируемая (k=8), время выполнения - 281190,9944 ms

Среднее: число потоков - 2, число элементов - 100000, тип функции - параметрируемая (k=8), время выполнения - 69897,28988 ms

Среднее: число потоков - 4, число элементов - 100000, тип функции - параметрируемая (k=8), время выполнения - 18100,32554 ms

Среднее: число потоков - 8, число элементов - 100000, тип функции - параметрируемая (k=8), время выполнения - 10367,53397 ms

**Задание №5:**

Исследуйте эффективность разделения по диапазону при неравномерной вычислительной сложности обработки элементов вектора.

Среднее: число потоков - 1, число элементов - 10, тип функции - неравномерная, время выполнения - 5,08224 ms

Среднее: число потоков - 2, число элементов - 10, тип функции - неравномерная, время выполнения - 20,35295 ms

Среднее: число потоков - 4, число элементов - 10, тип функции - неравномерная, время выполнения - 39,19057 ms

Среднее: число потоков - 8, число элементов - 10, тип функции - неравномерная, время выполнения - 97,15441 ms

Среднее: число потоков - 1, число элементов - 100, тип функции - неравномерная, время выполнения - 5,7622 ms

Среднее: число потоков - 2, число элементов - 100, тип функции - неравномерная, время выполнения - 25,59466 ms

Среднее: число потоков - 4, число элементов - 100, тип функции - неравномерная, время выполнения - 41,26687 ms

Среднее: число потоков - 8, число элементов - 100, тип функции - неравномерная, время выполнения - 104,82241 ms

Среднее: число потоков - 1, число элементов - 1000, тип функции - неравномерная, время выполнения - 33,21412 ms

Среднее: число потоков - 2, число элементов - 1000, тип функции - неравномерная, время выполнения - 53,78376 ms

Среднее: число потоков - 4, число элементов - 1000, тип функции - неравномерная, время выполнения - 84,67228 ms

Среднее: число потоков - 8, число элементов - 1000, тип функции - неравномерная, время выполнения - 157,89037 ms

Среднее: число потоков - 1, число элементов - 100000, тип функции - неравномерная, время выполнения - 273156,31055 ms

Среднее: число потоков - 2, число элементов - 100000, тип функции - неравномерная, время выполнения - 192717,49532 ms

Среднее: число потоков - 4, число элементов - 100000, тип функции - неравномерная, время выполнения - 116671,69026 ms

Среднее: число потоков - 8, число элементов - 100000, тип функции - неравномерная, время выполнения - 77828,21295 ms

**Задание №6:**

Исследуйте эффективность параллелизма при круговом разделении элементов вектора. Сравните с эффективностью разделения по диапазону.

Среднее: число потоков - 1, число элементов - 10, тип функции - неравномерная, время выполнения - 3,65208 ms

Среднее: число потоков - 2, число элементов - 10, тип функции - неравномерная, время выполнения - 23,53717 ms

Среднее: число потоков - 4, число элементов - 10, тип функции - неравномерная, время выполнения - 52,28115 ms

Среднее: число потоков - 8, число элементов - 10, тип функции - неравномерная, время выполнения - 132,6772 ms

Среднее: число потоков - 1, число элементов - 100, тип функции - неравномерная, время выполнения - 6,55118 ms

Среднее: число потоков - 2, число элементов - 100, тип функции - неравномерная, время выполнения - 16,15716 ms

Среднее: число потоков - 4, число элементов - 100, тип функции - неравномерная, время выполнения - 41,46519 ms

Среднее: число потоков - 8, число элементов - 100, тип функции - неравномерная, время выполнения - 100,59182 ms

Среднее: число потоков - 1, число элементов - 1000, тип функции - неравномерная, время выполнения - 28,10518 ms

Среднее: число потоков - 2, число элементов - 1000, тип функции - неравномерная, время выполнения - 57,09261 ms

Среднее: число потоков - 4, число элементов - 1000, тип функции - неравномерная, время выполнения - 88,29794 ms

Среднее: число потоков - 8, число элементов - 1000, тип функции - неравномерная, время выполнения - 177,3409 ms

Среднее: число потоков - 1, число элементов - 100000, тип функции - неравномерная, время выполнения - 206656,79501 ms

Среднее: число потоков - 2, число элементов - 100000, тип функции - неравномерная, время выполнения - 107345,17068 ms

Среднее: число потоков - 4, число элементов - 100000, тип функции - неравномерная, время выполнения - 56917,04747 ms

Среднее: число потоков - 8, число элементов - 100000, тип функции - неравномерная, время выполнения - 59223,71671 ms

При небольшом числе элементов массива эффект при круговом разделении массива между патоками нет, время выполнения вычислений примерно остается одинаковым, а однопоточная обработка элементов массива остается более эффективной.

Значительный выигрыш в скорости при круговой обработке наблюдается при большом числе элементов исследуемого массива ~100000, увеличение производительности связано с более сбалансированной вычислительной нагрузкой между отдельными потоками, по сравнению с разделением массива по диапазону между отдельными потоками.

**Ответы на вопросы**

**Вопрос №1: почему эффект от распараллеливания наблюдается только при большем числе элементов?**

При небольшом числе исследуемых элементов и простоте вычислительной функции однопоточная обработка проявляет себя эффективней за счет оптимизации на аппаратном уровне и оптимизационных свойств компилятора (распределение элементов в кэш памяти, специальные векторные инструкции для процессора и тд). В таких условиях многопоточное обработка проигрывает за счет сбора информации о системе подготовки ресурсов и формировании отдельных потоков для расчетов. Эффективность будет заметна тогда, когда время на подготовку отдельных потоков будет намного меньше времени вычислении поставленной задачи одним потоком.

**Вопрос №2: почему увеличение сложности обработки повышает эффективность многопоточной обработки?**

Потому что при увеличенной сложности обработки данных задача разделяется между несколькими потоками, которые могут синхронно выполняют параллельно обработку на нескольких ядрах процессора. В то время как обычный метод (однопоточный) выполнять эту задачу в одиночку последовательно элемент за элементом.

**Вопрос №3: какое число потоков является оптимальным для конкретной вычислительной системы?**

При небольшом числе объектов и простоте вычислительной функции (где можно применить специальные векторные инструкции процессора) – 1 поточная обработка;

При больших объемах данных и сложных вычислительных функциях эффективней всего применять число потоков равное числу физических ядер процессора (для исключения конкуренции большого числа потоков за вычислительный ресурс) + применять специальные алгоритмы, выравнивающие нагрузку на отдельные потоки.

**Вопрос №4: почему неравномерность загрузки потоков приводит к снижению эффективности многопоточной обработки?**

Потому что данные могут быть обработаны с разной вычислительной сложностью (из-за специфики вычислительной функции), при разбиении на равные участки (интервалы) может сложиться такая ситуация, что один участок может быть вычислен быстрее другого. Данная проблема носит название - проблема статической декомпозиции.

**Вопрос №5: какие другие варианты декомпозиции позволяют увеличить равномерность загрузки потоков?**

Сбалансированность (равномерность) загрузки потоков при неизвестной вычислительной сложности элементов можно осуществить при помощи динамической декомпозиции. В этом случае если какой-то поток обрабатывает выделенный массив быстрее, чем другой, то он может увеличить размер обрабатываемого массива. Следующий подход — это применение круговой декомпозиции, которая выравнивает загруженность отдельных потоков выполнения. К примеру, при обработке массива объектов можно поступить следующим образом разбить задачу на два потока один поток обрабатывает все четные элементы второй все нечетные элементы, что приведет к сбалансированию нагрузки на отдельные потоки.

**Вопрос №6: в какой ситуации круговая декомпозиция не обеспечивает равномерную загрузку потоков?**

В случае независимости вычислительной сложности отдельных объектов массива от его индекса.