Архитектура вычислительных систем
ДЗ 8. Отчёт
Работа на 10 баллов

Фролов-Буканов Виктор Дмитриевич БПИ-228

06 декабря 2023

1 Составление таблицы из условия

Время работы программы для разных входных данных (в секундах)

Число элементов в массиве	1 поток	2 потока	4 потока	8 потоков	1000 потоков
100 000 000	0.631	0.756	0.948	0.615	1.073
500 000 000	3.55	3.9	4.12	3.07	5.69
800 000 000	39.48	33.98	30.6	34.6	50

2 Пару скриншотов, подтверждающую верность составления таблицы

```
Number of elements is 100000000
Scalar product (single) = 1.6666671666618179781e+23
Time of program execution (single) = 0.63100000000000000533 seconds
Number of threads is 2
Scalar product (async) = 1.6666671666668209439e+23
Time of program execution (async) = 0.75600000000000000533 seconds
Process finished with exit code 0
```

Figure 1: 1 скрин

```
Number of elements is 100000000
Scalar product (single) = 1.6666671666618179781e+23
Time of program execution (single) = 0.63800000000000001155 seconds
Number of threads is 4
Scalar product (async) = 1.6666671666668813418e+23
Time of program execution (async) = 0.94799999999999995381 seconds
Process finished with exit code 0
```

Figure 2: 2 скрин

```
Number of elements is 100000000
Scalar product (single) = 1.6666671666618179781e+23
Time of program execution (single) = 0.62700000000000000178 seconds
Number of threads is 8
Scalar product (async) = 1.6666671666665860628e+23
Time of program execution (async) = 0.61499999999999999112 seconds
Process finished with exit code 0
```

Figure 3: 3 скрин

```
Number of elements is 100000000
Scalar product (single) = 1.6666667166661817978e+23
Time of program execution (single) = 0.62800000000000000266 seconds
Number of threads is 1000
Scalar product (async) = 1.6666667166666867261e+23
Time of program execution (async) = 1.0729999999999999538 seconds
Process finished with exit code 0
```

Figure 4: 4 скрин

```
Number of elements is 100000000
Scalar product (single) = 1.6666667166661817978e+23
Time of program execution (single) = 0.62800000000000000266 seconds
Number of threads is 1000
Scalar product (async) = 1.6666667166666867261e+23
Time of program execution (async) = 1.0729999999999999538 seconds
Process finished with exit code 0
```

Figure 5: 5 скрин

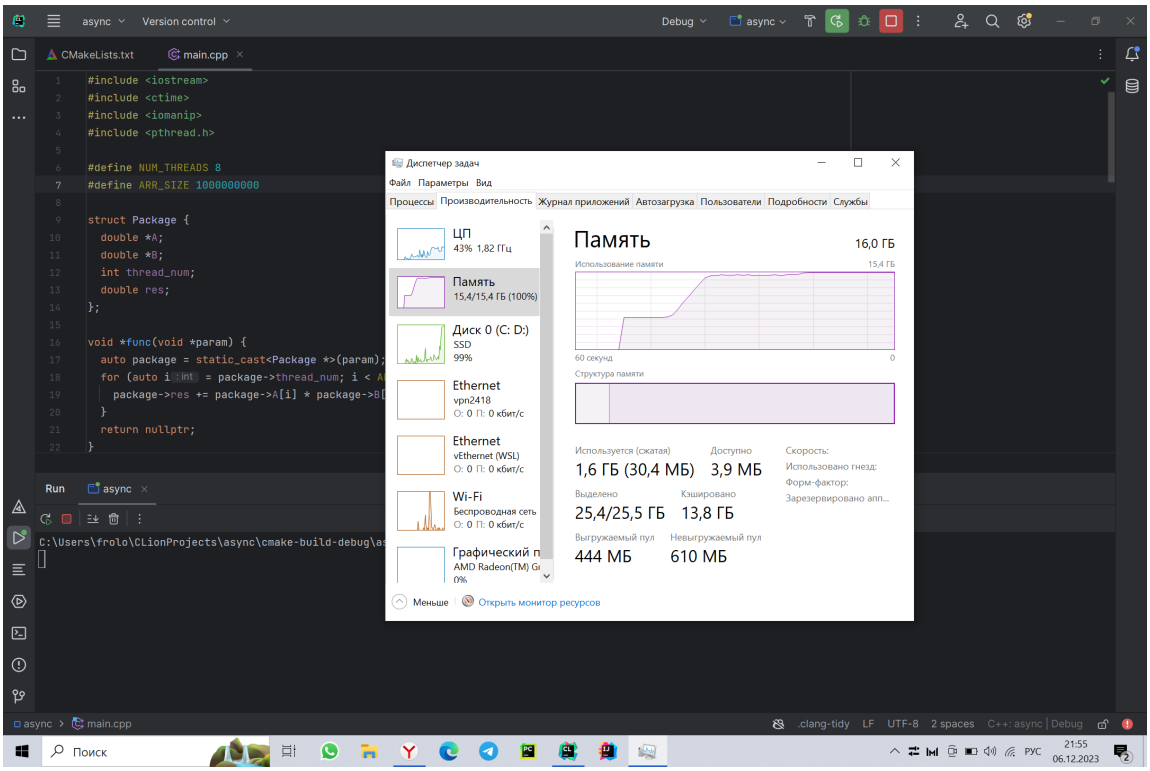

Figure 6: 6 скрин

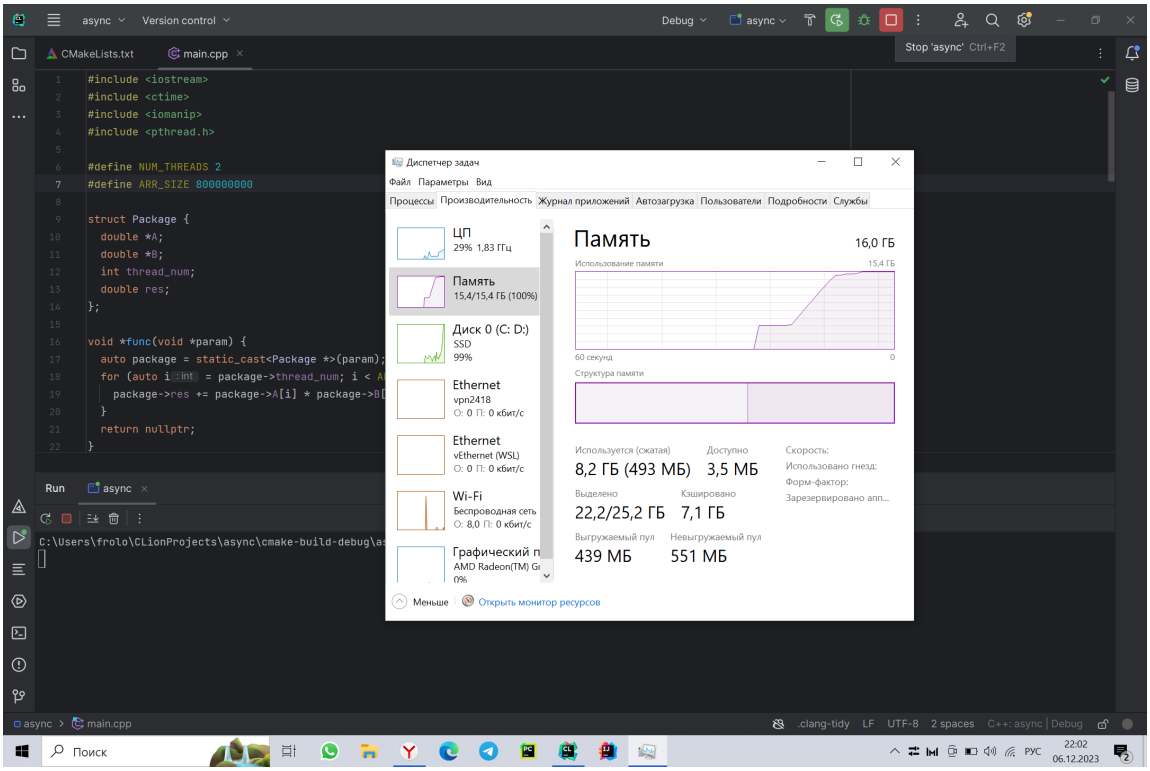

Figure 7: 7 скрин

Для остальных размеров массива скриншоты не привожу, так как там все то же самое, но полученные значения честно занес в таблицу. Также отмечу, что в приведенных скриншотах меняется время выполнения однопоточной части программы, но при этом они радикально не меняются, так что будем считать, что все значения в таблице приближенные, так как от запуска программы к запуску значения могут меняться, но существенных приростов/падений не происходит

Также отмечу, что запустить программу на 1 000 000 000 элементов в массиве не получается, так как память моего компьютера полностью заполняется даже с учетом свопинга (оперативная память и SSD) (скрин 6). В таком случае понижу число элементов с 1 000 000 000 до 700 000 000. И посмотрю, что будет в таком случае. Цель найти как можно более минимальную границу, при которой будет происходить свопинг ... Получилось так, что при 700 000 000 ещё не совсем происходит свопинг, а при 800 000 000 он уже происходит, причем заполняя всю память моего компьютера (см скрин 7), однако не происходит того, что происходит в случае с 1 000 000 000 элементов (компьютер тотально виснет и памяти вообще не хватает). Случай с 800 000 000, видимо, является примерно предельным, так что оставлю его в качестве последнего теста

3 Выводы и замечания

Отмечу, что результаты работы приведенной программы (*main.cpp*). Могут отличаться в зависимости от свойств компьютера, на котором она запускается, в особенности важен показатель оперативной памяти, а также объём SSD-накопителя. Однако на моей машине результат вышел такой:

1. 1000 потоков всегда замедляют работу программы
2. На предельных входных данных (800 000 000) любая асинхронность дает прирост (оптимальная на 4 потока)

3. На остальных тестах 8 потоков дают наибольший прирост производительности
4. В целом, 8-поточная конфигурация является самой оптимальной из всех предложенных