Министерство науки высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

(Университет ИТМО)

Факультет программной инженерии и компьютерной техники

ЛАБОРАТОРНАЯ РАБОТА №2

По курсу «Реактивная Java»

Предметная область «Матчи, турниры, команды»

Выполнили:

Голоскок Дмитрий Сергеевич P4119

Симовин Кирилл Константинович P4116

Преподаватель:

Гаврилов Антон Валерьевич

Санкт-Петербург

2024

Задание

1. В один из методов, использовавшийся для сбора статистики, добавить возможность задать задержку, имитирующую задержку получения результата, например, из базы данных. К примеру, был метод getName(), в который нужно добавить параметр getName(long delay).

2. Заменить последовательный стрим, собирающий статистику из лабораторной №1, на параллельный. Измерить производительность для разного количества элементов с дополнительной задержкой и без задержки. Для случаев с задержкой и без найти количество элементов, при котором сбор статистики последовательным и параллельным стримами дают одинаковую скорость выполнения. Поменять итоговую коллекцию, где собирается результат, на соответствующую потокобезопасную.

3. Оптимизировать параллельный сбор статистики, реализовав собственный ForkJoinPool или Spliterator. Измерить производительность своего варианта.

4. Измерения производительности выполнять с помощью фрейворка JMH.

Ход выполнения работы

В таблице 1 представлено время выполнения программ в зависимости от размера обрабатываемых коллекций и задержки. Время задержки устанавливалось равное 5 миллисекундам.

Таблица 1 – Сравнительная таблица методов обработки

N	Время выполнения программ, с					
	Цикл		Стандартный коллектор		Собственный коллектор	
	С задержкой	Без задержки	С задержкой	Без задержки	С задержкой	Без задержки
5000	0,02317	0,01417	0,01467	0,00717	0,01317	0,00450
50000	0,03242	0,00908	0,01917	0,00767	0,01742	0,00800
250000	0,03625	0,02783	0,08392	0,05000	0,07942	0,05375

Как можно заметить, в отличие от первой лабораторной работы, программа с циклом работает медленнее решений с использованием коллекторов в двух из трех случаев.

На рисунке 1 представлен график зависимости времени выполнения программы от количества элементов в массиве для каждого из вариантов обработки.

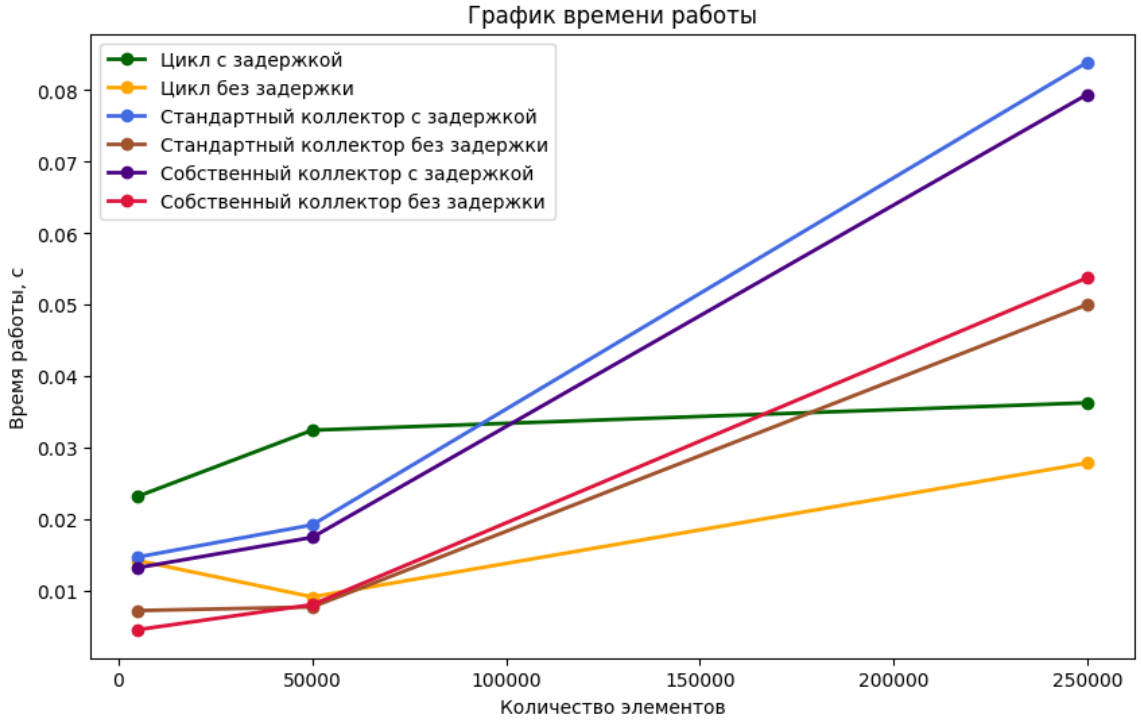

Рисунок 1 – График времени выполнения методов обработки

В таблице 2 представлено общее число элементов, при котором параллельный поток и последовательная программа работают за, примерно, одинаковое время. Также, в данной таблице представлено время выполнения, соответствующее параллельному потоку и последовательной программе при соответствующем количестве элементов.

Таблица 2 – Общее количество элементов для программ с задержкой и без

N	Время выполнения, с			
	Параллельная программа		Последовательная программа	
	С задержкой	Без задержки	С задержкой	Без задержки
500	0,00152	0,00081	0,00121	0,00064
2500	0,00830	0,00501	0,00664	0,00408
12500	0,04162	0,01828	0,03320	0,01462
62500	0,18573	0,10287	0,14858	0,08743
312500	0,98741	0,50721	0,87897	0,46641
1562500	4,98981	**2,49182**	4,59006	**2,49178**
…				
1830500	**5,81940**	-	**5,81934**	-

Как можно заметить, с увеличением количества элементов, нивелируется разница во времени выполнения параллельной и последовательной программы.

На рисунке 2 представлено сравнение времени выполнения программ до момента, пока не будет найдено общее количество элементов.

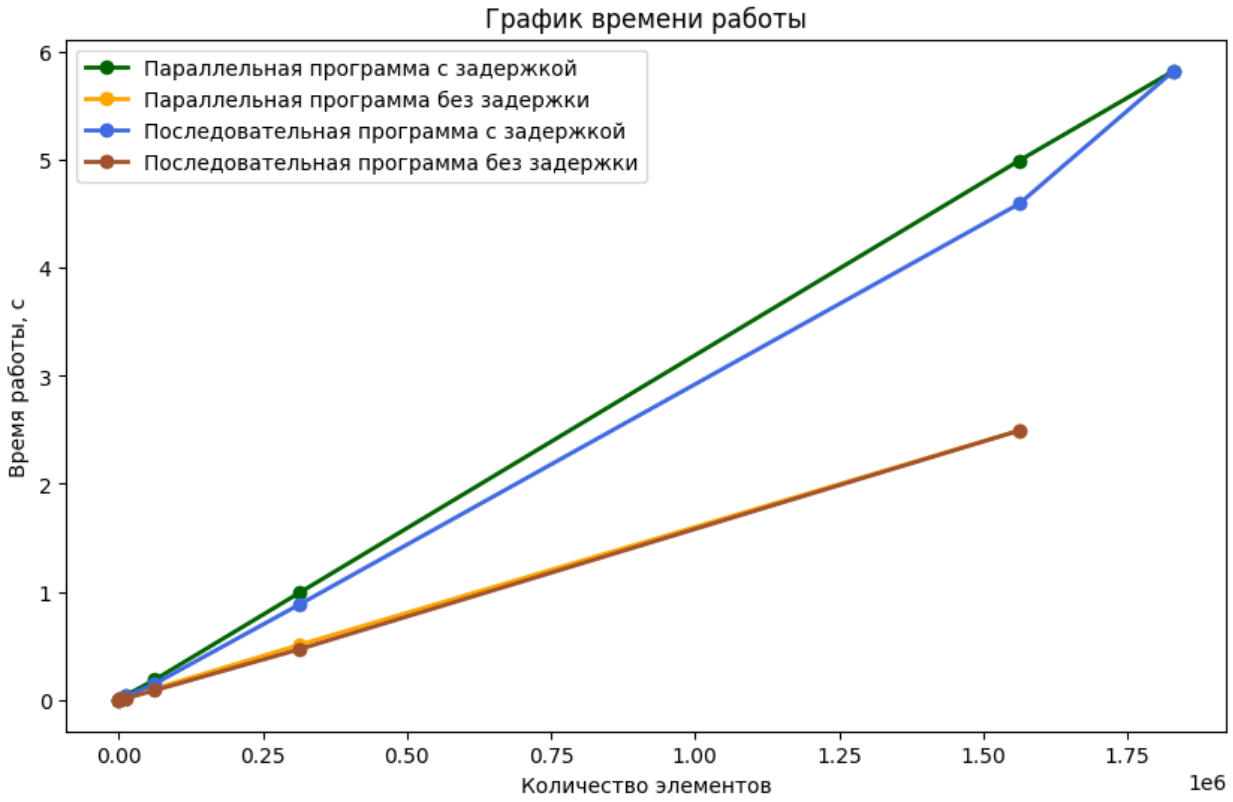

Рисунок 2 – График времени выполнения программ во время поиска общего числе элементов

В таблице 3 представлено сравнение времени выполнения программ с задержкой и без для собственного ForkPoolJoin.

Таблица 3 – Сравнительная таблица методов обработки с использованием ForkPoolJoin

N	Время выполнения программы, с	
	Без задержки	С задержкой
5000	0,01442	0,03981
50000	0,092742	0,12812
250000	0,10319	0,20714

На рисунке 3 приведен график зависимости времени работы программ с использованием ForkPoolJoin от размера коллекций.

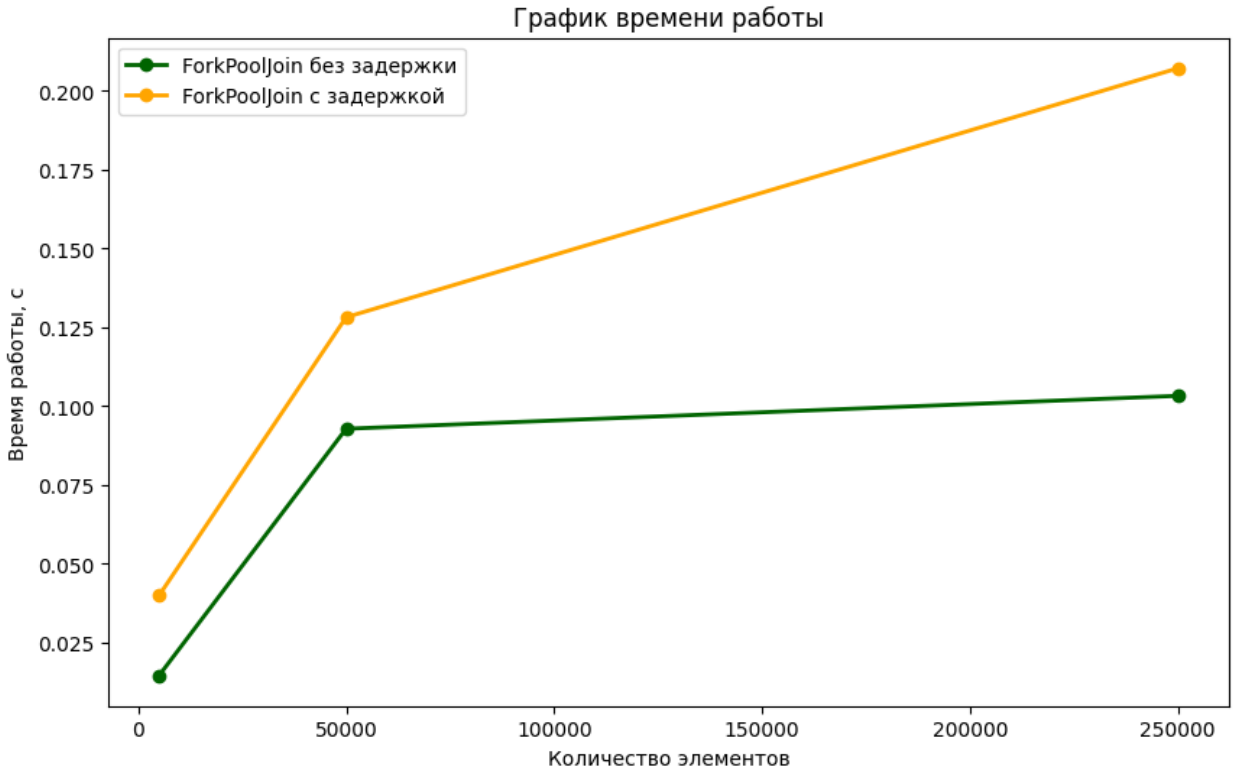

Рисунок 3 – График времени работы программ с использованием ForkPoolJoin

Заключение

Цель лабораторной работы – сравнить скорость работы решений для расчета статистический данных с наличием задержки и без нее. Показано, что на предоставленном размере коллекций решение с использованием коллекций является более предпочтительным.

Однако, для 250000 элементов итеративный способ является предпочтительным, по сравнению с коллекторами: 0,03625 секунд против 0,08392 секунд для стандартного коллектора и 0,07942 секунд с использованием собственного коллектора – решение с задержкой. Для решения без задержки: 0,02783 секунд против 0,05000 секунд для стандартного коллектора и 0,05375 секунд с использованием собственного коллектора.

Также, необходимо было определить общее число элементов для последовательного и параллельного решений. Для решений с задержкой общее число элементов составило 1830500 с примерным временем выполнения в 5,81937 секунд. Для решения без задержки: 1562500 элементов при времени выполнения 2,49180 секунд.

Заключительным этапом лабораторной работы стала собственная реализация ForkJoinPool, которая работает медленнее встроенных параллельных потоков: например, для 250000 элементов время выполнения собственной реализации ForkJoinPool без задержки составило 0,10319 секунд против 0,05375 секунд для параллельной программы с использованием собственного коллектора.

В ходе выполнения лабораторной работы были получены навыки по работе с параллельными потоками.