МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №3
по дисциплине «Параллельные алгоритмы»
Тема: Реализация параллельной структуры данных с тонкой блокировкой

Студент гр. 9304 _____ Арутюнян В.В.

Преподаватель _____ Сергеева Е.И.

Санкт-Петербург
2022

Цель работы

Ознакомиться с неблокирующими способами синхронизации lock-free структурой данных.

Задание

Реализовать итерационное (потенциально бесконечное) выполнение подготовки, обработки и вывода данных. Обеспечить параллельное выполнение потоков обработки готовой порции данных, подготовки следующей порции данных и вывода предыдущих полученных результатов. Реализовать на основе lock-free структуры данных.

Протестировать доступ в случае нескольких потоков-производителей и потребителей. Сравнить производительность со структурой с грубой синхронизацией (лаб. работа 2).

В отчёте сформулировать инвариант структуры данных.

Выполнение работы

Сложение матриц происходит в 3 основных этапа:

1. Первая группа потоков:
    a. Считывание двух матриц из файла *input.txt* На отдельной строке была задана размерность матрицы (высота, ширина через пробел). Затем представлена матрица, элементы которой разделены пробелами, а строки матрицы выделяются переносами строк. Аналогично представлена вторая матрица в том же файле.
    b. Передача считанных матриц в первый буфер в виде пары матриц.
2. Вторая группа потоков:
    a. Считывание из первого буфера пары матриц.
    b. Поэлементное сложение матриц с помощью заданного количества потоков (параллельно).
    c. Запись полученной матрицы во второй буфер.
3. Третья группа потоков:
    a. Считывание полученной матрицы из второго буфера.

b. Вывод матрицы в файл *output.txt* в том же виде, как представлена матрица во входном файле.

Итерационное считывание обеспечивается повторным проведением описанных выше этапов. Каждый этап выполняет отдельная группа потоков. Количество потоков в каждой группе одинаково, каждый поток выполняет одинаковое количество итерационных считываний.

Количество итерационных считываний, и потоков, выполняющих итерационное считывание, и потоков, параллельно выполняющих суммирование матриц, определяется отдельными переменными.

Элемент очереди (Node)

В качестве элемента очереди выступает класс Node. Он имеет следующие поля:

1. value – элемент, содержащий значение, которое добавляется в очередь.
2. next – указатель на следующий элемент очереди, по умолчанию указывает на nullptr.

Lock-Free очередь (LockFreeQueue)

В качестве буфера выбрана lock-free очередь (класс LockFreeQueue**)**. Для вставки элементов в очередь и для получения их из нее используются CAS операции.

Очередь имеет следующие поля:

1. head – указатель, указывающий на фиктивный элемент очереди. В head->next находится первый значимый элемент очереди.

2. tail – указатель, указывающий на последний элемент очереди. В пустой очереди указывает на фиктивный элемент, как и head.

3. name – строковое имя очереди.

И следующие методы:

1. push – добавление нового элемента в очередь.
2. pop – получение и удаление самого старого элемента очереди.

Инвариант очереди

В очереди всегда есть один фиктивный элемент, на который всегда указывает элемент head, а также элемент tail, когда очередь пуста.

Новые элементы всегда вставляются в конец очереди, путем добавления в tail->next указателя на новый элемент.

Для получения и удаления самого старого элемента очереди берется элемент head->next. Вместо. его явного удаления, он. становится новым фиктивным элементом.

Исследование производительности структур при грубой и неблокирующей синхронизациях

Исследование зависимости происходило путём перебора количества потоков. В каждом сравнении использовалось 30 итерационных повторений, а также по 8 потоков для суммирования матриц. Размер матрицы был зафиксирован и равен 17x13.

В таблице 1 представлено сравнение обработки матрицы 17x13 для разного кол-ва потоков в группах. По приведенным результатам видно, что наиболее быстрое выполнение происходит при грубой синхронизации с помощью BlockingQueue.

Таблица 1 – Сравнение полученных результатов для матрицы 17x13 при 30 итерационных повторениях

Количество потоков группы	Затраченное время, сек.	
	BlockingQueue	LockFreeQueue
1	0.012835	0.014676
2	0.020663	0.020981
4	0.038046	0.045969
8	0.062943	0.088200
16	0.126411	0.176503
32	0.254582	0.311690
64	0.547995	0.715104
128	1.482854	1.708618

Выводы

В ходе выполнения лабораторной работы была изучена работа с неблокирующими способами синхронизации (CAS).

Также реализована lock-free очередь и произведено исследование производительности при грубой и неблокирующей синхронизациях. Очередь с грубой синхронизацией оказалась быстрее, чем с неблокирующей (lock-free).