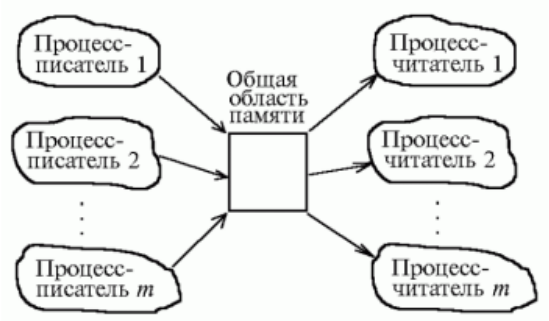
Архитектура вычислительных систем
ИДЗ 4. Отчёт
Вариант 7
Работа на 10 баллов

Фролов-Буканов Виктор Дмитриевич БПИ-228

17 декабря 2023

1 Условие задачи

7. Задача о читателях и писателях («подтвержденное чтение»). Базу данных разделяют два типа процессов — читатели и писатели. Читатели периодически просматривают случайные записи базы данных и выводя номер свой номер, индекс записи и ее значение. Писатели изменяют случайные записи на случайное число и также выводят информацию о своем номере, индексе записи, старом значении



и новом значении. Предполагается, что в начале БД находится в непротиворечивом состоянии (все числа отсортированы). Каждая отдельная новая запись переводит БД из одного непротиворечивого состояния в другое (то есть, новая сортировка может поменять индексы записей). Транзакции выполняются в режиме «подтвержденного чтения», то есть процесс-писатель не может получить доступ к БД в том случае, если ее занял другой процесс–писатель или процесс–читатель. К БД может обратиться одновременно сколько угодно процессов–читателей. Процесс читатель получает доступ к БД, даже если ее уже занял процесс–писатель. Создать многопоточное приложение с потоками–писателями и потоками–читателями.

2 Модель параллельных вычислений, используемая при разработке многопоточной программы

В соответствии с условием задачи логичнее всего будет выбрать модель "Производители и потребители". Потоками-производителями тут являются потоки-писатели, потоками-потребителями - потоки-читатели

Производители и потребители. Парадигма взаимодействующих неравноправных потоков. Одни потоки «производят» данные, другие их «потребляют». Часто такие потоки организуются в конвейер, через который проходит информация. Каждый поток конвейера потребляет выход своего предшественника и производит входные данные для своего последователя.

3 Описание решения задачи

Весь код я декомпозировал на несколько header-файлов, содержащих необходимый функционал. По названию в целом понятно, какие методы содержит в себе тот или иной header-файл, но поясню дополнительно:

- *main.cpp* - содержит функцию main, а также методы обработки введенных данных в зависимости от того, вводятся они с консоли или из командной строки
- *multithread.h* - файл с непосредственный решением задачи через мьютексы
- *multithread_alt.h* - файл с непосредственным решением задачи через другой синхропримитив (блокировщик чтения-записи)
- *multithread_omp.h* - файл с непосредственным решением задачи с использованием OpenMP
- $uniform\_random.h$ - файл, где хранится генератор чисел из равномерного распределения с вариативными диапазонами при многократных запусках (вариативность достигается через привязку seed к текущему времени с использованием библиотеки *chrono*, а необходима она для удовлетворения 4 по счёту критерия на 4-5 баллов)
- $ConsoleIOConroller.h$ - файл, содержащий методы для чтения/записи векторов по передаваемым потокам, а также методы для валидации вводимых пользователем данных (повторный запрос некорректного ввода)

4 Описание тестовых прогонов

По условию необходимо предоставить не менее 3 файлов. Я создал ровно 3 файла, на которых запускаю свою программу. Их содержимое:

input1.txt

```
5
1
2
3
4
5
```

input2.txt

```
3
−10
100
42
```

input3.txt

```
8
90
−10
56
31
7
1
67
−12
```

Первое число в файле - n - число элементов массива, а далее следуют n чисел. Сами файлы (как входные, так и выходные) лежат в директории *cmake−build−debug*

5 Выводы программ

Один из выводов программы, реализованной без использования OpenMP

```
[write] 1. Thread number 1 changed the value of record with
   id 2 from 3 to 1863
 [read] 2. Thread number 1 read the record with id 3 and
    value 5
 [read] 3. Thread number 2 read the record with id 1 and
    value 2
 [read] 4. Thread number 10 read the record with id 2 and
    value 4
 [write] 5. Thread number 4 changed the value of record with
    id 3 from 5 to 1224
 [read] 6. Thread number 4 read the record with id 4 and
    value 1863
```

```
[read] 7. Thread number 17 read the record with id 1 and
   value 2
[read] 8. Thread number 19 read the record with id 0 and
   value 1
[read] 9. Thread number 6 read the record with id 0 and
   value 1
[read] 10. Thread number 7 read the record with id 0 and
   value 1
[write] 11. Thread number 2 changed the value of record with
    id 3 from 1224 to 1554
[read] 12. Thread number 8 read the record with id 0 and
   value 1
[read] 13. Thread number 9 read the record with id 0 and
   value 1
[read] 14. Thread number 11 read the record with id 4 and
   value 1863
[read] 15. Thread number 3 read the record with id 2 and
   value 4
[read] 16. Thread number 12 read the record with id 2 and
   value 4
[read] 17. Thread number 13 read the record with id 1 and
   value 2
[read] 18. Thread number 15 read the record with id 2 and
   value 4
[read] 19. Thread number 16 read the record with id 4 and
   value 1863
[write] 20. Thread number 12 changed the value of record
   with id 3 from 1554 to 938
[read] 21. Thread number 18 read the record with id 4 and
   value 1863
[read] 22. Thread number 20 read the record with id 4 and
   value 1756
[read] 23. Thread number 14 read the record with id 3 and
   value 938
[read] 24. Thread number 5 read the record with id 1 and
   value 2
[write] 25. Thread number 7 changed the value of record with
    id 4 from 1863 to 1756
[write] 26. Thread number 9 changed the value of record with
    id 0 from 1 to 909
[write] 27. Thread number 3 changed the value of record with
    id 4 from 1756 to 1232
[write] 28. Thread number 11 changed the value of record
   with id 1 from 4 to 1423
[write] 29. Thread number 14 changed the value of record
   with id 2 from 938 to 1790
```

```
 [write] 30. Thread number 15 changed the value of record
    with id 4 from 1790 to 1388
 [write] 31. Thread number 16 changed the value of record
    with id 0 from 2 to 966
 [write] 32. Thread number 17 changed the value of record
    with id 4 from 1423 to 948
 [write] 33. Thread number 18 changed the value of record
    with id 3 from 1232 to 1492
 [write] 34. Thread number 5 changed the value of record with
     id 2 from 966 to 1303
 [write] 35. Thread number 19 changed the value of record
    with id 4 from 1492 to 1607
 [write] 36. Thread number 6 changed the value of record with
     id 0 from 909 to 1482
 [write] 37. Thread number 8 changed the value of record with
     id 1 from 1303 to 1372
 [write] 38. Thread number 10 changed the value of record
    with id 1 from 1372 to 1166
 [write] 39. Thread number 13 changed the value of record
    with id 2 from 1388 to 1532
 [write] 40. Thread number 20 changed the value of record
    with id 3 from 1532 to 1188
 Final vector:  948  1166  1188  1482  1607
Used distribution: Uniform(891, 1891)
```

Один из выводов программы, реализованной с использованием OpenMP

```
[read] 1. Thread number 5 read the record with id 3 and
   value 4
 [read] 2. Thread number 1 read the record with id 4 and
    value 5
 [read] 3. Thread number 17 read the record with id 1 and
    value 2
 [write] 4. Thread number 0 changed the value of record with
    id 0 from 1 to 1467
 [write] 5. Thread number 14 changed the value of record with
     id 1 from 3 to 1419
 [read] 6. Thread number 13 read the record with id 0 and
    value 2
 [read] 7. Thread number 7 read the record with id 1 and
    value 4
 [write] 8. Thread number 8 changed the value of record with
    id 1 from 4 to 1368
 [read] 9. Thread number 9 read the record with id 2 and
    value 1368
 [write] 10. Thread number 10 changed the value of record
    with id 2 from 1368 to 1389
```

```
 [write] 11. Thread number 2 changed the value of record with
     id 1 from 5 to 1012
 [read] 12. Thread number 11 read the record with id 1 and
    value 1012
 [write] 13. Thread number 12 changed the value of record
    with id 3 from 1419 to 937
 [read] 14. Thread number 3 read the record with id 4 and
    value 1467
 [write] 15. Thread number 16 changed the value of record
    with id 4 from 1467 to 973
 [write] 16. Thread number 4 changed the value of record with
     id 1 from 937 to 829
 [read] 17. Thread number 15 read the record with id 1 and
    value 829
 [write] 18. Thread number 6 changed the value of record with
     id 3 from 1012 to 1325
 [write] 19. Thread number 18 changed the value of record
    with id 0 from 2 to 1589
 [read] 20. Thread number 19 read the record with id 0 and
    value 829
 Final vector:  829  973  1325  1389  1589
Used distribution: Uniform(784, 1784)
```

Однако лучше вывод смотреть в файлах на гитхабе, так как TeX переносит строчки там, где не следует. Выводы отличаются в силу немного разной логики работы программ (а именно разного количества потоков)

6 Общие замечания по реализации задачи

1. Так как в условии не сказано, сколько создавать потоков читателей, а сколько потоков-писателей, то я в 1 и 2 решениях использовал 20 читателей и 20 писателей

2. В решении через OpenMP я использовал 20 потоков всего, причем все нечетные являются потоками чтения, четные - потоками-писателями

3. 1 и 2 решения отличаются использованиями разных синхропримитивов. В 1 для основного решения задачи я использовал мьютексы, во 2 - блокировщики чтения-записи (на деле использовался только блокировщик записи в методе $write\_nums$), при этом наряду с ними также использовались мьютексы, так как это не противоречит условию (мьютексы используются для сихронизации вывода, а также работы с условной переменной)

4. При работе программы вывод происходит как сразу на консоль, так и в вектор, чтобы потом его из этого вектора записать в файл

5. В исходнике лежит 2 исполняемых файла: a.exe - это файл с решением через OpenMP и main.exe - это файл с одним из решений не через OpenMP

6. Каждый вывод в конце сопровождается описанием распределения, которое использовалось в данном конкретном запуске программы (распределение всегда равномерное, отличаются лишь границы)

7. Для лучшей ориентации в коде я оставил комментарии, где они уместны и необходимы

8. **При изменении решения задачи нужно в main.cpp подключить header-файл с соответствующим решением**

7 Как работать с вызовом программы из командной строки?

Для начала надо убедиться, что в системе прописан путь к g++.exe. Если нет, то это необходимо сделать через переменные среды (проверить можно, введя g++ –version в консоли, если все хорошо, то ошибок быть не должно). Далее надо перейти в директорию, где расположен файл main.cpp, и в ней прописать команду "g++ main.cpp -o main". После этого должен создаться исполняемый файл main.exe. Командой "main.exe 2 'полное имя входного файла', 'полное имя выходного файла' " запускаем программу и наслаждаемся результатом работы. Замечу, что важно указывать именно полный путь до файлов, иначе программа будет заново запрашивать ввод, так как она не будет находить запраишваемый файл. При этом при запуске из консоли какой-либо IDE полный путь можно не указывать, так как IDE автоматически унаследует текущую директорию и все названия будет воспринимать относительно неё. Также есть опция вести такую команду: "main.exe 1 3 1 2 3 'полное имя выходного файла' ", которая использует ручной ввод, а не ввод с файла (число после main.exe указывает на тип ввода: 1 - ручной, 2 - из файла). Если мы вводим 1 после main.exe, то дальше передаем число элементов в массиве и дальше сам массив, а в конце имя выходного файла. Программа также корректно работает, выводя результат как в консоль, так и в файл, путь до которого мы указали (важно также указывать полный путь до файла).

Запуск программы, написанной с использованием OpenMP происходит несколько иначе. Нужно также перейти в директорию с файлом main.cpp, но уже прописать команду "g++ main.cpp -fopenmp". Последний аргумент - это флаг, который позволит программе отработать, используя библиотеку <omp.h>. У меня не получилось запустить эту программу из IDE, так как компилятор в моей среде не настроен на работу с этой библиотекой. Таким образом, запуск программы на 10 баллов возможен только из командной строки с передачей соответствующего флага (либо нужно прописывать какие-то настройки в CLion). В результате работы создастся исполняемый файл a.exe. Дальнейший запуск программы с передачей параметров аналогичен тому, как я описывал в предыдущем параграфе