

Лабораторная работа №4

UNIX [C++ & UNIX]: C++ PROCESSES / THREADS

Выполнил: Макаренко Александр, группа Z33434, 3 курс, 2024 г.

Цель

Познакомить студента с принципами параллельных вычислений. Составить несколько программ в простейшими вычислительными действиями, чтобы освоить принципы параллельных вычислений (когда одни алгоритмы зависят / не зависят от других).

Задачи

1 [C++ SEQUENCE] Последовательные вычисления

Требуется последовательно выполнить вычисления по формуле 1, вычисления по формуле 2, после чего выполнить вычисления по формуле 3, которые выглядят следующим образом: результат вычислений 1 + результат вычислений 2 - результат вычислений 1

Выполнить последовательно на 10 000 итераций и 100 000 итераций

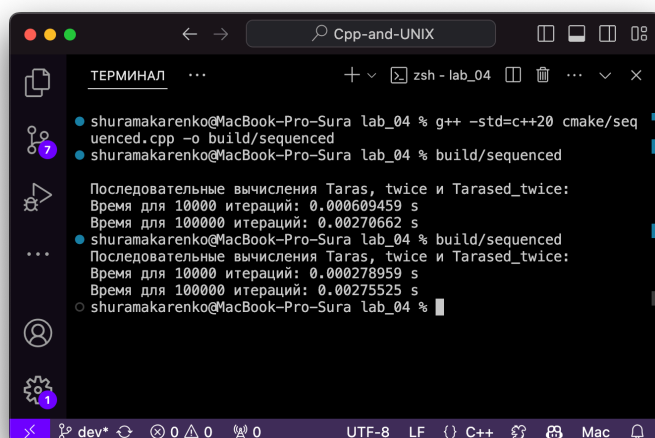
Формула 1: $f(x) = x^2 - x^2 + x^4 - x^5 + x + x$

Формула 2: $f(x) = x + x$

Вывести длительность выполнения всех 10 000 итераций и 100 000 итераций в сек.

Решение

Ожидаемая разница в 10 раз, но иногда бывают сильные скачки



```
shuramakarenko@MacBook-Pro-Sura lab_04 % g++ -std=c++20 cmake/sequenced.cpp -o build/sequenced
shuramakarenko@MacBook-Pro-Sura lab_04 % build/sequenced

Последовательные вычисления Taras, twice и Tarased_twice:
Время для 10000 итераций: 0.000609459 s
Время для 100000 итераций: 0.00270662 s
shuramakarenko@MacBook-Pro-Sura lab_04 % build/sequenced
Последовательные вычисления Taras, twice и Tarased_twice:
Время для 10000 итераций: 0.000278959 s
Время для 100000 итераций: 0.00275525 s
shuramakarenko@MacBook-Pro-Sura lab_04 %
```

2. [C++ THREADS] Параллельные вычисления через потоки

Требуется параллельно (насколько возможно с помощью потоков) выполнить вычисления по формуле 1, вычисления по формуле 2, после чего выполнить вычисления по формуле 3, которые выглядят следующим образом: результат вычислений 1 + результат вычислений 2 - результат вычислений 1.

Выполнить последовательно на 10 000 итераций и 100 000 итераций

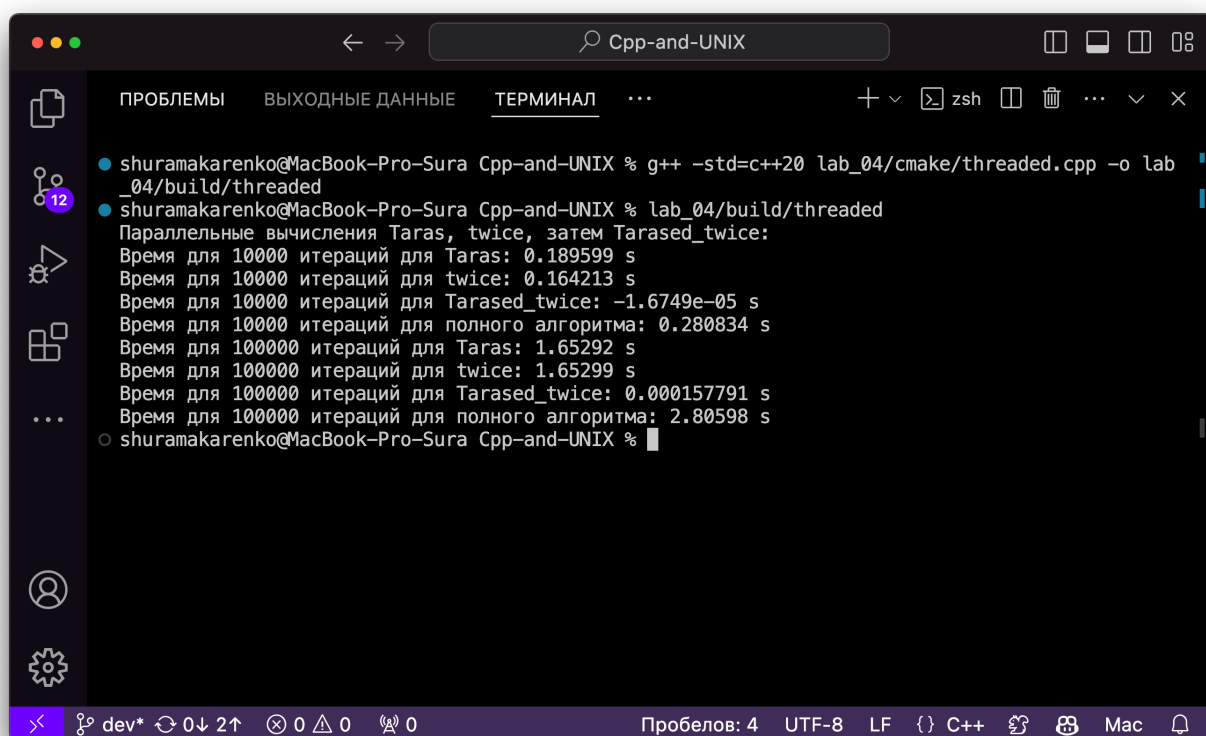
Формула 1: $f(x) = x^2 - x^2 + x^4 - x^5 + x + x$

Формула 2: $f(x) = x + x$

Вывести длительность выполнения всех 10 000 итераций и 100 000 итераций в сек. в разбивке по шагам вычислений 1, 2 и 3

Решение

Видно, что суммарное время меньше чем сумма времен



```
shuramakarenko@MacBook-Pro-Sura Cpp-and-UNIX % g++ -std=c++20 lab_04/cmake/threaded.cpp -o lab_04/build/threaded
shuramakarenko@MacBook-Pro-Sura Cpp-and-UNIX % lab_04/build/threaded
Параллельные вычисления Taras, twice, затем Tarased_twice:
Время для 10000 итераций для Taras: 0.189599 s
Время для 10000 итераций для twice: 0.164213 s
Время для 10000 итераций для Tarased_twice: -1.6749e-05 s
Время для 10000 итераций для полного алгоритма: 0.280834 s
Время для 100000 итераций для Taras: 1.65292 s
Время для 100000 итераций для twice: 1.65299 s
Время для 100000 итераций для Tarased_twice: 0.000157791 s
Время для 100000 итераций для полного алгоритма: 2.80598 s
shuramakarenko@MacBook-Pro-Sura Cpp-and-UNIX %
```

3. [C++ PROCESS] Параллельные вычисления через процессы

Требуется параллельно (насколько возможно с помощью процессов) выполнить вычисления по формуле 1, вычисления по формуле 2, после чего выполнить вычисления по формуле 3, которые выглядят следующим образом: результат вычислений 1 + результат вычислений 2 - результат вычислений 1s

Выполнить последовательно на 10 000 итераций и 100 000 итераций

Формула 1: $f(x) = x^2 - x^2 + x^4 - x^5 + x + x$

Формула 2: $f(x) = x + x$

Вывести длительность выполнения всех 10 000 итераций и 100 000 итераций в сек. в разбивке по шагам вычислений 1, 2 и 3

Решение

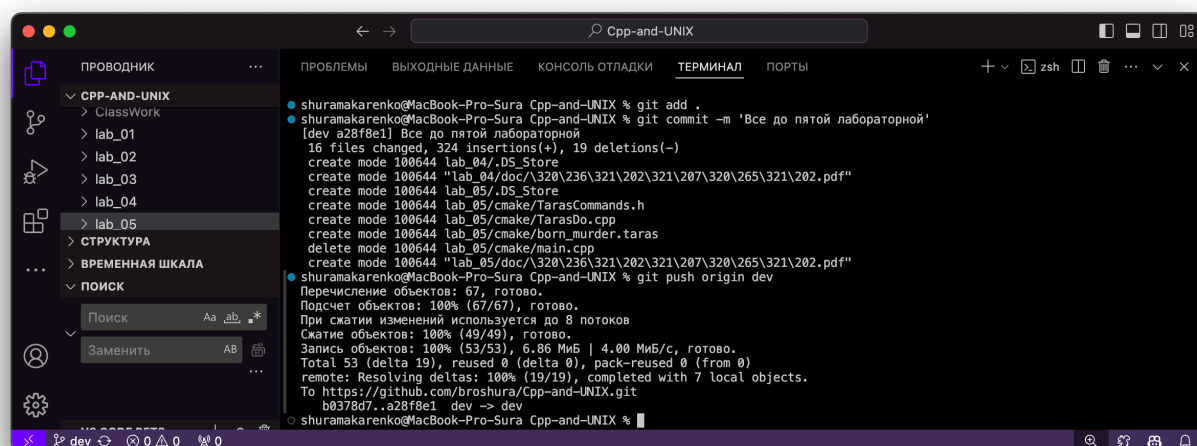
```
shuramakarenko@MacBook-Pro-Sura Cpp-and-UNIX % g++ -std=c++20 lab_04/cmake/p
rocessed.cpp -o lab_04/build/processed
shuramakarenko@MacBook-Pro-Sura Cpp-and-UNIX % lab_04/build/processed

Параллельные вычисления Taras, twice, затем Tarased_twice:
Время для 10000 итераций для Taras: 2.69912 s
Время для 10000 итераций для twice: 2.67566 s
Время для 10000 итераций для Tarased_twice: -1.5042e-05 s
Время для 10000 итераций для полного алгоритма: 3.77362 s
Время для 100000 итераций для Taras: 26.9789 s
Время для 100000 итераций для twice: 27.7593 s
Время для 100000 итераций для Tarased_twice: 0.000160875 s
Время для 100000 итераций для полного алгоритма: 38.1251 s
shuramakarenko@MacBook-Pro-Sura Cpp-and-UNIX %
```

4. [LOG] Результат всех вышеперечисленных шагов сохранить в репозиторий (+ отчет по данной ЛР в папку doc)

Фиксацию ревизий производить строго через ветку `dev`. С помощью скриптов накатить ревизии на `stg` и на `prd`. Скрипты разместить в корне репозитория. Также создать скрипты по возврату к виду текущей ревизии (даже если в папке имеются несохраненные изменения + новые файлы).

Решение



```
shuramakarenko@MacBook-Pro-Sura Cpp-and-UNIX % git add .
shuramakarenko@MacBook-Pro-Sura Cpp-and-UNIX % git commit -m 'Все до пятой лабораторной'
[dev a28f8e1] Все до пятой лабораторной
16 files changed, 324 insertions(+), 19 deletions(-)
create mode 100644 lab_04/.DS_Store
create mode 100644 "lab_04/doc/\320\236\321\202\321\207\320\265\321\202.pdf"
create mode 100644 lab_05/.DS_Store
create mode 100644 lab_05/cmake/TarasCommands.h
create mode 100644 lab_05/cmake/TarasDo.cpp
create mode 100644 lab_05/cmake/born_murder.taras
delete mode 100644 lab_05/cmake/main.cpp
create mode 100644 "lab_05/doc/\320\236\321\202\321\207\320\265\321\202.pdf"
shuramakarenko@MacBook-Pro-Sura Cpp-and-UNIX % git push origin dev
Перечисление объектов: 67, готово.
Подсчет объектов: 100% (67/67), готово.
При сжатии изменений используется до 8 потоков
Сжатие объектов: 100% (49/49), готово.
Запись объектов: 100% (53/53), 6.86 Миб | 4.00 Миб/с, готово.
Total 53 (delta 19), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (19/19), completed with 7 local objects.
To https://github.com/broshura/Cpp-and-UNIX.git
b037807...a28f8e1 dev -> dev
shuramakarenko@MacBook-Pro-Sura Cpp-and-UNIX %
```

Выводы

1. Сравнили скорость вычисления как при организации параллельных вычислений с помощью потоков и процессов, так и при последовательных вычислениях, научились реализовывать каждый вариант.
2. Последовательные вычисления справились лучше всего - нет никаких лишних затрат на создание и администрирование параллельных процессов.
3. Вычисление через потоки примерно в 1000 раз медленнее, потому что их создание и уничтожение каждую итерацию цикла стоит много времени.
4. Вычисления через процессы примерно в 10000 раз медленнее, потому что организация процесса через `fork()` еще и копирует окружение, что еще больше замедляется процесс на каждой итерации.
5. Вычисления через процессы примерно экономит 30% относительно суммарного времени, а потоки - 12%. Если правильно организовать процессы и потоки, не создавая их такое количество раз, выигрыш действительно можно получить примерно такой для двух потоков/процессов, а для большего еще лучше.