Лабораторная работа№6 по OS-lite.

Выполнил: Лещев Владимир Олегович

Параметры виртуальной машины.

Детали реализации каждого из алгоритмов.

1. **Эксперимент 1**

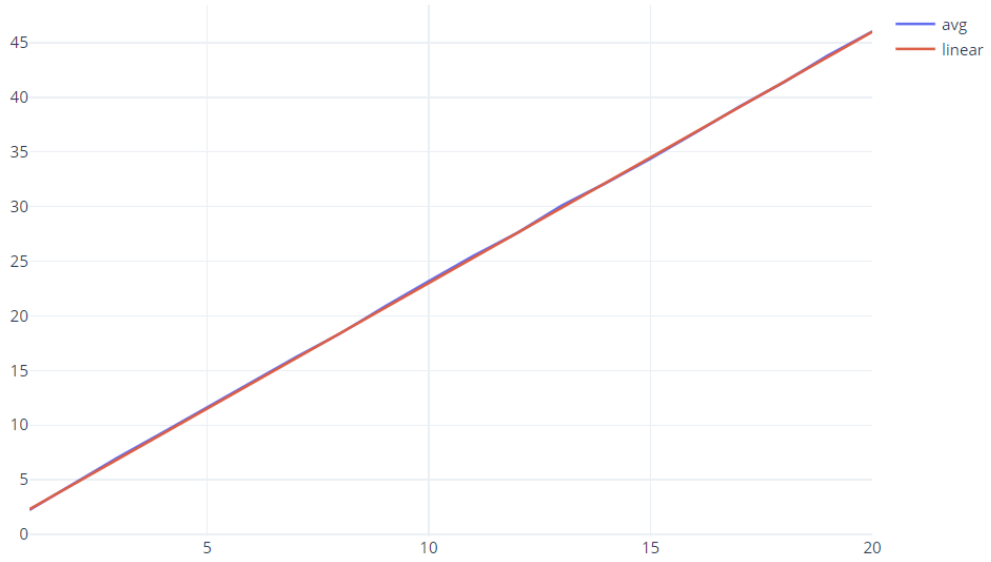
   В нем требовалось реализовать вычисление любой из функции, которая считает в среднем 2-3 секунды. Мной был выбран гиперболический косинус. Я реализовал base1.sh и base2.sh, где первый скрипт запускает последовательно вычисления, а другой параллельно. Потом я просто вызываю их в соответствующих скриптах которые ничем не отличаются, кроме как вызова скриптов base1.sh и base2.sh. Что происходит? Мы просто каждый раз запускаем для

нового N 10 раз скрипт. Для каждого нового N время работы зависит следующим образом (графики ниже.)

Приведем фотографии графиков:

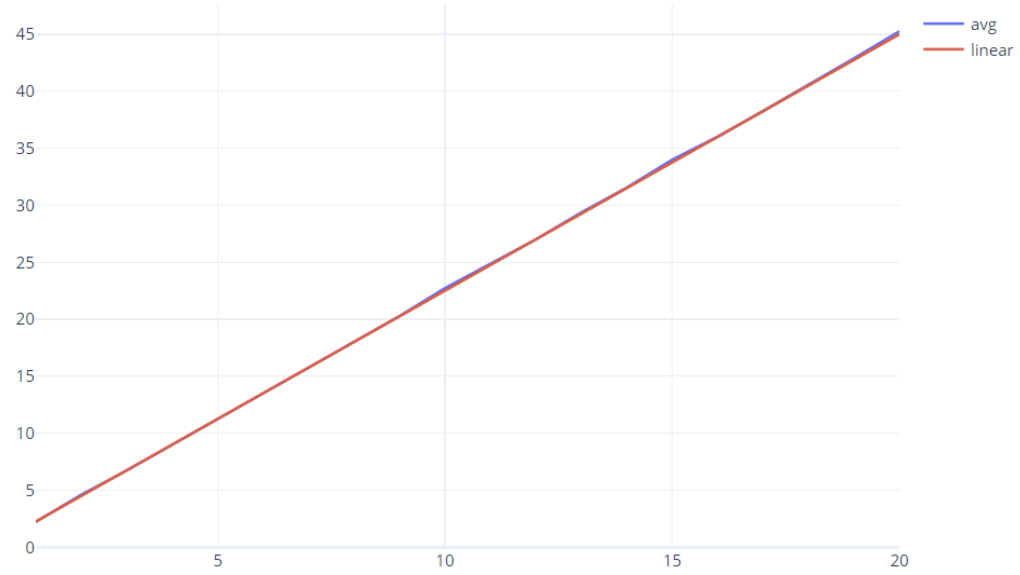
Последовательный запуск.

1. На одном процессоре.



Вывод: Из экспериментальных данных становится очевидным, что зависимость времени работы алгоритма от N приближенно равна линейной зависимости. Почему это так? При возрастании N на единицу, время работы алгоритма base_i_.sh увеличивается на среднее время работы алгоритма вычисления функции.

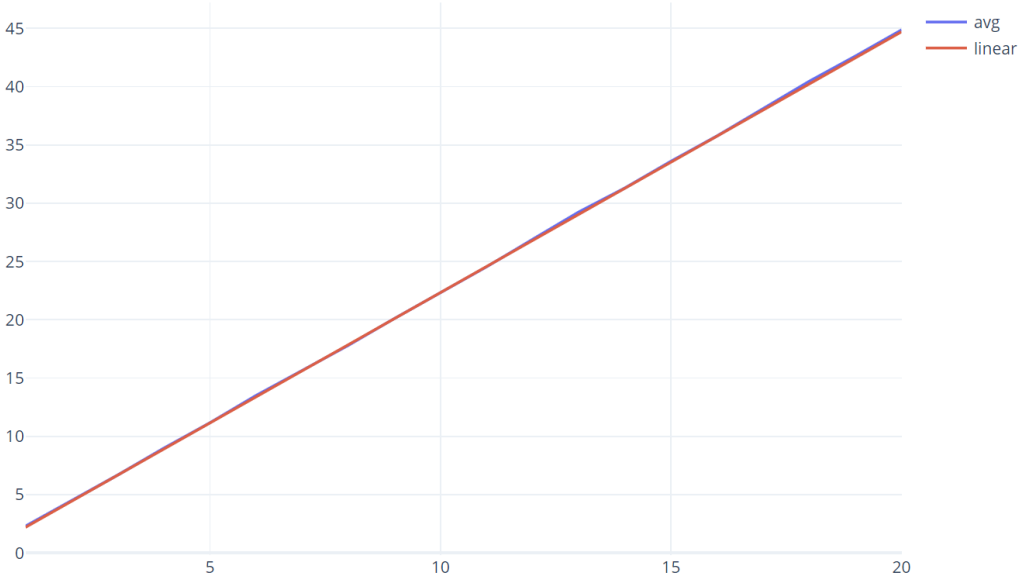
1. На двух.



Вывод: Ситуация почти никак (с учетом погрешности) не изменилась по сравнению с запуском на 1 процессоре. Почему? Когда процесс начинает выполняться, он это делает на 1 из 2 доступных ему процессорах. Один будет свободен, но не сможет запустить следующий процесс, в силу того, что каждый из следующих процессов ожидает завершение работы предыдуще запущенного.

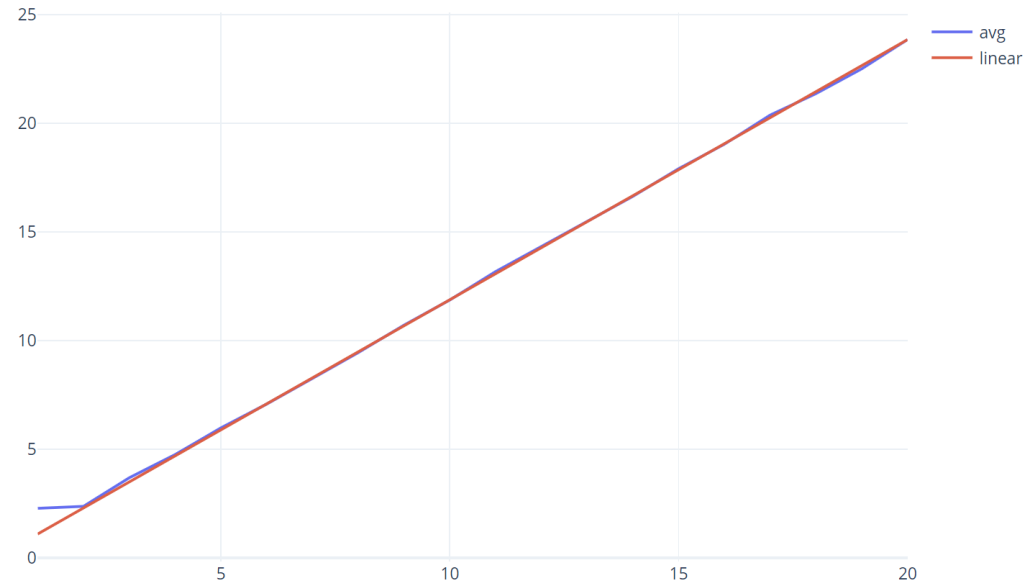
Параллельный

1. На одном процессоре.



Вывод: Заметим, что ситуация опять же схожа на выше рассмотренные. Так как процессор только 1, процессы выстраиваются в очередь, где при завершении одного из них, запускается другой. В сумме получается такой же результат как и предыдущие.

1. На двух:



Вывод: Ситуация отличается от предыдущих. Это связанно с тем, что у нас изначально доступно 2 процессора. При параллельном запуске процессы можно сказать распалогаются уже в две очереди (на 1 процессор 1 очередь). Таким образом время выполения алгоритма у нас уменьшается соответсвенно в два раза.

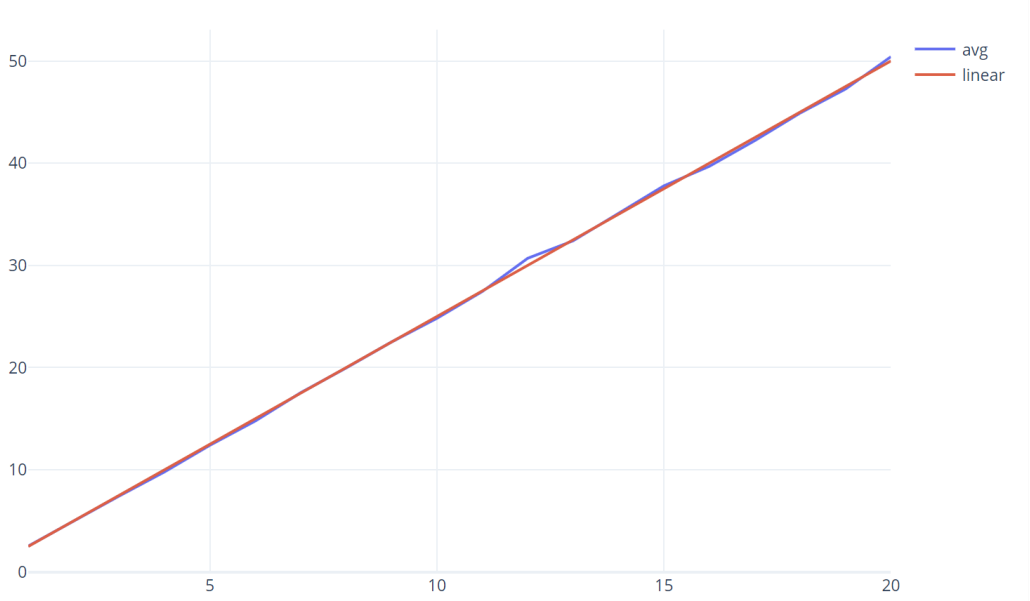
2. **Эксперимент 2**

В нем требовалось реализовать чтение и запись в фаил. Среднее время работы 2-3 секунды. Написал алгоритм генерации файлов, в зависимости от N. Зачем? Проблема возникала с параллельным запуском на двух процессорах, когда требуемый фаил был занят уже на другом процессоре. Реализовал base1.sh и base2.sh (аналогично первому эксперименту). Первый скрипт запускает последовательно

алгоритм чтения-записи, а другой параллельно. Потом я просто вызываю их в соответствующих скриптах которые ничем не отличаются, кроме как вызова скриптов base1.sh и base2.sh.

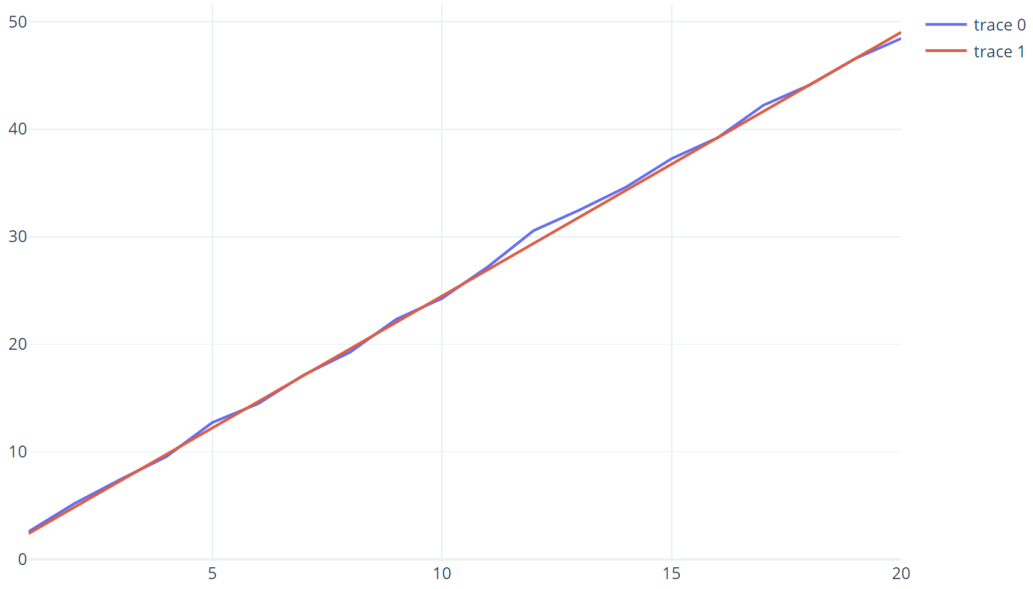
Последовательный запуск.

1. На одном процессоре.



Вывод: При последовательном запуске на одном процессоре, мы получаем примерно линейную зависимость. Оно и понятно, при каждом новом запуске мы увеличиваем N на 1, и при этом время работы алгоритма увеличивается примерно на среднее время выполнение одного запуска. По сути аналогично последовательному запуску на 1 процессоре из 1 эксперимента. (только тут имеется погрешность на открытие и закрытие файла)

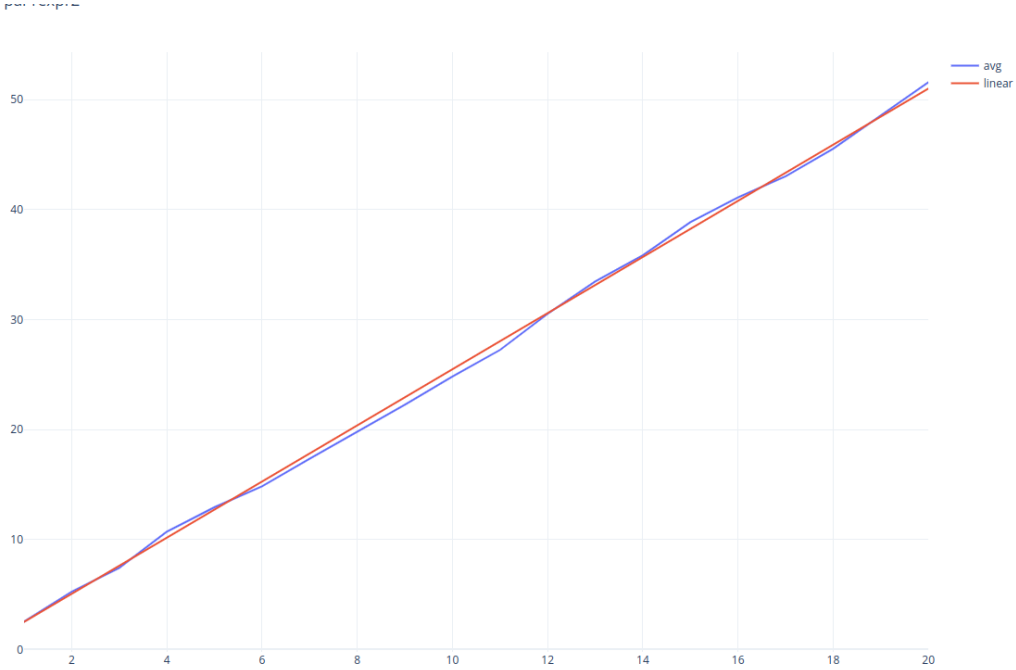
1. На двух.



Вывод: Ситуация почти не отличается от запуска на 1 процессоре, так как мы все равно запускаем алгоритм последовательно, т.е. ждем пока он завершится на любом из процессоров, по сути получаем те же самые результаты.

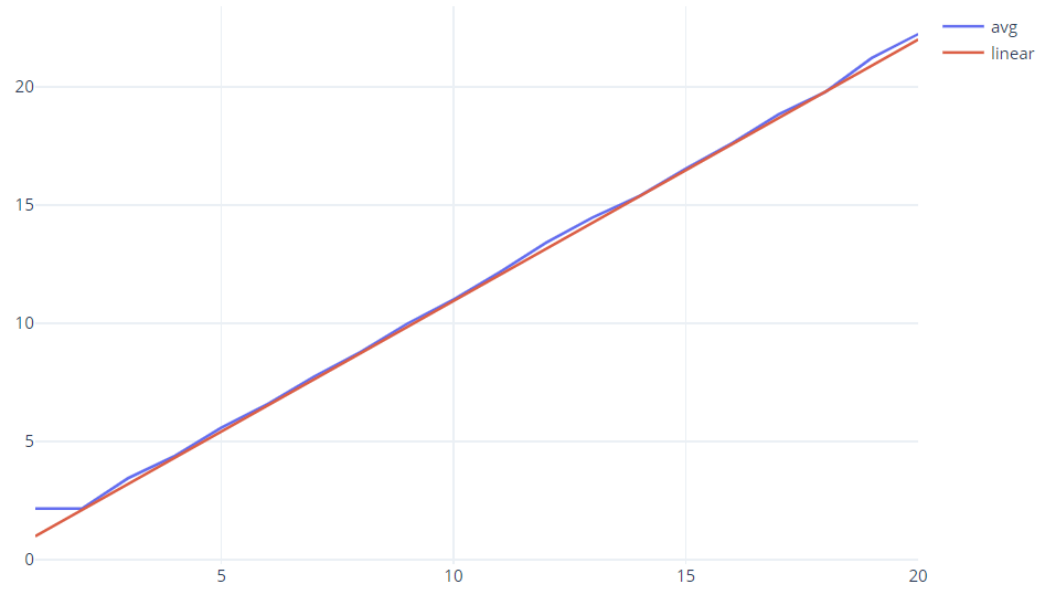
Параллельный

1. На одном процессоре.



Вывод: Так как процессор один, то процессы выстраиваются в очередь, и ожидают пока текущий исполняемы процесс завершится.

1. На двух:



Вывод: При параллельном запуске на двух мы видим, что процессы распределяются на два процессора. По сути мы получаем что то на подобии двух очередей, где в каждой примерно одинковое кол-во запусков. А это означает что наш график просто получит угол наклона в 2 раза меньше. Так же стоит сказать, что в условии реализации алгоритма записи в фаил, эксперимент проводился так: на каждый процесс был свой фаил для записи.

Детали реализации алгоритмов

1. **Первый эксперимент.**

Основной алгоритм есть вычисление нашей функции в точке - evalCh.c. Мы получаем точку и находим в ней значение по формуле Тейлора.

Далее мы работаем с алгоритмами base1.sh и base2.sh. Они просто запускают наш алгоритм то число раз, которое мы им

передали. Их отличие друг от друга в том, что один запускает последовательно процессы, а другой параллельно.

2. **Второй эксперимент.**

Основной алгоритм есть запись в фаил writeFile.c и writeFile.sh (writeFile.sh - скрипт компилирующий .c и запускающий его на переданном значении). Так же есть createFile.c и createFile.sh создающие для каждого алгоритма свой фаил для чтения-записи.

Далее мы работаем с алгоритмами base1.sh и base2.sh. Они просто запускают наш алгоритм (writeFile.sh) то число раз, которое мы им передали. Их отличие друг от друга в том, что один запускает последовательно процессы, а другой параллельно.

Вывод

Таким образом, мы увидели как процессоры функционируют в ОС в различных ситуациях (вычисление, запись-чтение - на разном числе процессов), а так же выявили закономерность в том, что все наблюдаемые нами зависимости в экспериментах примерно линейны.