Факультет программной инженерии и компьютерной техники

Параллельные вычисления

Лабораторная работа №5

Параллельное программирование с использованием стандарта POSIX Threads

Преподаватель: Жданов Андрей Дмитриевич

Выполнил: студент: Кульбако Артемий Юрьевич, P4115

Санкт-Петербург

2023

ОГЛАВЛЕНИЕ

ОГЛАВЛЕНИЕ 2
ЗАДАНИЕ.....3
ВЫПОЛНЕНИЕ.....4
ИСХОДНЫЙ КОД.....7

ЗАДАНИЕ

1. Взять в качестве исходной OpenMP-программу из ЛР-4, в которой распараллелены все этапы вычисления. Убедиться, что в этой программе корректно реализован одновременный доступ к общей переменной, используемой для вывода в консоль процента завершения программы.
2. Изменить исходную программу так, чтобы вместо OpenMP-директив применялся стандарт POSIX Threads:
    - для получения оценки «3» достаточно изменить только один этап (`Generate, Map, Merge, Sort`), который является узким местом (`bottle neck`), а также функцию вывода в консоль процента завершения программы
    - для получения оценки «4» и «5» необходимо изменить всю программу, но допускается в качестве расписания циклов использовать `schedule static` - для получения оценки «5» необходимо хотя бы один цикл
    - для получения оценки «5» необходимо хотя бы один цикл распараллелить, реализовав вручную расписание `schedule dynamic` или `schedule guided`.

3. Провести эксперименты и по результатам выполнить сравнение работы двух параллельных программ (`OpenMP` и `POSIX Threads`), которое должно описывать следующие аспекты работы обеих программ (для различных N):

- полное время решения задачи
- параллельное ускорение
- количество строк кода, добавленных при распараллеливании, а также грубая оценка времени, потраченного на распараллеливание (накладные расходы программиста)
- остальные аспекты, которые вы выяснили самостоятельно (**Обязательный пункт**)

ВЫПОЛНЕНИЕ

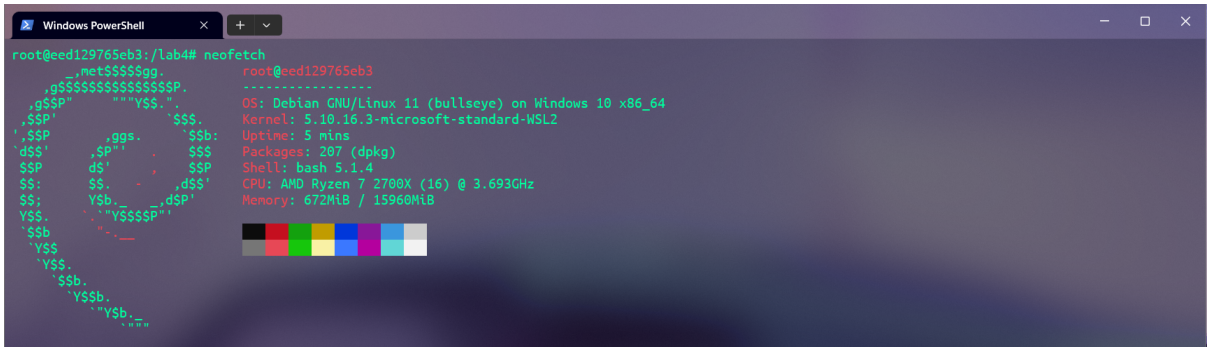
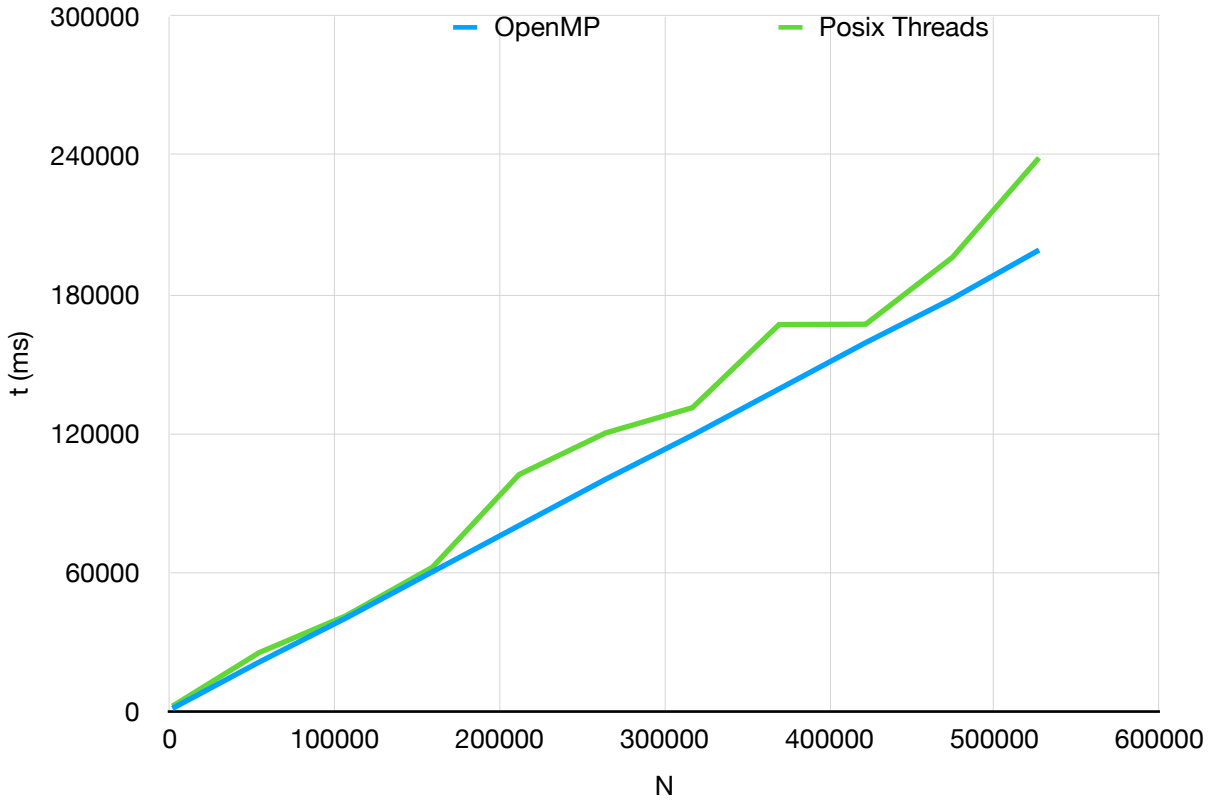


Рис. 1 - Характеристики машины

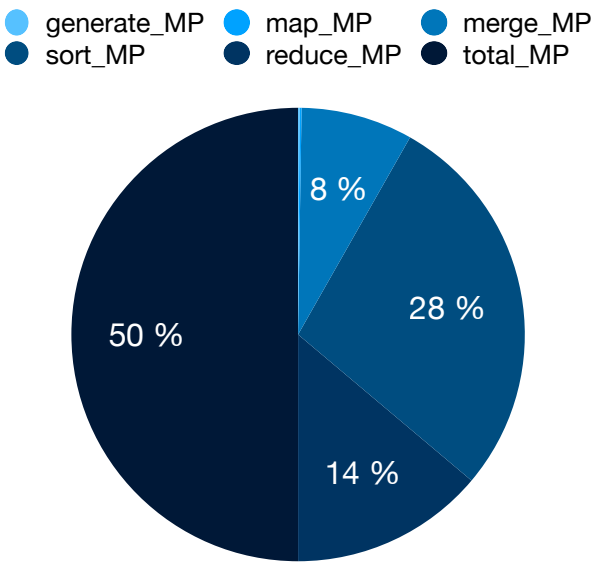
Запустим две программы: параллелизм первой осуществляется за счёт библиотеки OpenMP с использованием расписания static, параллелизм второй за счёт самостоятельно разработанного аналога на Posix Threads.
Можно увидеть, что моё решение сработало медленнее чем OpenMP, но в среднем не сильно, на 15%.

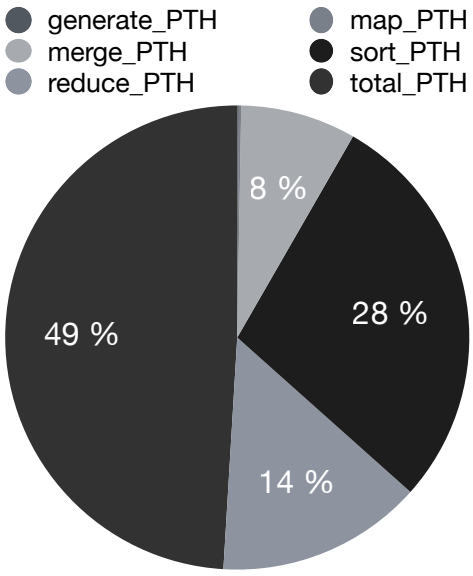


В жёлтой таблице представлена статистика замедления программы в мс в зависимости от количества обрабатываемых элементов и среднее значение падения производительности.

N	total_MP	total_PTH	PARALLEL SPEEDUP
1600	1030	2000	-49 %
54139	21032	25119	-16 %
106678	40035	41003	-2 %
159217	60036	62006	-3 %
211756	80038	102053	-22 %
264295	100040	120048	-17 %
316834	119042	130946	-9 %
369373	139043	166851	-17 %
421912	159046	166998	-5 %
474451	178048	195852	-9 %
526990	199053	238863	-17 %
СУММ	1096443	1251739	-12 %
			-15 %

На диаграммах распределения можно увидеть, что больше всего времени занимает этап сортировки, что ожидаемо, так как все остальные этапа имеют сложность O(n), а сортировка куда сложнее.





Кодовая база проекта увеличилась практически в 2 раза (с 271 строки к 518), и это при условии реализации только одного типа расписания, поэтому стоит сказать авторам OpenMP большое спасибо за столь полезный и лёгкий в использовании фреймворк.
Накладные расходы программиста составили ±12 часов (но стоит учитывать, что я не силён в C и много экспериментировал с макросами).

ИСХОДНЫЙ КОД

Таблицы в формате csv и исходной код программы и скриптом для тестирования доступен на: https://github.com/testpassword/Parallel-computing/tree/master/lab5-12.03.23: