Университет ИТМО

Факультет программной инженерии и компьютерной техники

Высокопроизводительные системы

Лабораторная работа 1

Параллелизация алгоритмов сортировки

Вариант 40

Выполнил: Чепрасов Иван

Группа: P34022

г. Санкт-Петербург

2021 год.

Цель работы

Реализовать алгоритмы сортировки в соответствии с вариантом задания. Параллелизовать реализованные алгоритмы с помощью OpenMP, применяя для каждого общий набор из разных подходов для распределения частей работы и объединения результатов. Собрать статистику о времени работы реализованных алгоритмов в целом и отдельных этапов в реализациях для различных наборов входных данныхв сочетании с различными степенями параллелизма. Сравнить все полученные реализации: разные алгоритмы попарно для каждого подхода и отдельно все реализации каждого алгоритма между собой. Объяснить результаты.

Вариант 40

1. Tree Sort
2. Smooth Sort

Задачи

1. Изучить алгоритмы сортировки по варианту
2. Изучить директивы OpenMP
3. Реализовать генерацию массивов заданного размера
4. Реализовать алгоритмы в нескольких вариациях (по 2 в моём случае)
5. Реализовать разбиение массива на примерно равные части и последующее склеивание частей
6. Внедрить использование директив OpenMP для паралеллизации
7. Убрать привязку к кол-ву используемых потоков и кратности кол-ва потоков и размера сортируемого массива
8. Внедрить сбор метрик
9. Вынести общие, повторяемые операции в отдельные функции

Описание программы и аспекты реализации

Программа состоит из трёх модулей (treeSort, smoothSort, utils). Реализация алгоритмов находится в первых двух модулях. В модуле utils находятся вспомогательные функции и структуры, разнесённые по смыслу на файлы: linked_list, array_utils, test_utils.

Все тестовые операции и параллелизация вынесены в отдельные функции, принимающие алгоритм сортировки в качестве входных аргументов.

Вывод программы:

Multi-threaded Tree Sort

Starting 5 test(s) on 5000000 elements...

Test 1 finished in 1.61s and used 8.30s of cpu time.

Test 2 finished in 2.51s and used 13.64s of cpu time.

Test 3 finished in 2.69s and used 14.09s of cpu time.

Test 4 finished in 3.11s and used 14.97s of cpu time.

Test 5 finished in 2.95s and used 14.84s of cpu time.

Checking accuracy...

Great Job! No errors occurred

Mean real time is: 2.57 seconds

Mean cpu time is: 13.17 seconds

Total acceleration acceleration_rate is 5.12!

Tree Sort with Parallel Traverse

Starting 5 test(s) on 5000000 elements...

Test 1 finished in 14.85s and used 15.07s of cpu time.

Test 2 finished in 16.75s and used 16.99s of cpu time.

Test 3 finished in 17.65s and used 17.89s of cpu time.

Test 4 finished in 18.45s and used 18.66s of cpu time.

Test 5 finished in 17.20s and used 17.45s of cpu time.

Checking accuracy...

Great Job! No errors occurred

Mean real time is: 16.98 seconds

Mean cpu time is: 17.21 seconds

Total acceleration acceleration_rate is 1.01!

Single-threaded Tree Sort

Starting 5 test(s) on 5000000 elements...

Test 1 finished in 17.31s and used 17.30s of cpu time.

Test 2 finished in 18.07s and used 17.89s of cpu time.

Test 3 finished in 17.87s and used 17.84s of cpu time.

Test 4 finished in 17.23s and used 17.23s of cpu time.

Test 5 finished in 17.90s and used 17.86s of cpu time.

Checking accuracy...

Great Job! No errors occurred

Mean real time is: 17.67 seconds

Mean cpu time is: 17.62 seconds

Total acceleration acceleration_rate is 1.00!

Multi-threaded Smooth Sort

Starting 5 test(s) on 5000000 elements...

Test 1 finished in 0.68s and used 2.96s of cpu time.

Test 2 finished in 0.61s and used 2.82s of cpu time.

Test 3 finished in 0.60s and used 2.81s of cpu time.

Test 4 finished in 0.60s and used 2.82s of cpu time.

Test 5 finished in 0.62s and used 2.88s of cpu time.

Checking accuracy...

Great Job! No errors occurred

Mean real time is: 0.62 seconds

Mean cpu time is: 2.86 seconds

Total acceleration acceleration_rate is 4.60!

Single-threaded Smooth Sort

Starting 5 test(s) on 5000000 elements...

Test 1 finished in 3.02s and used 3.02s of cpu time.

Test 2 finished in 2.99s and used 2.99s of cpu time.

Test 3 finished in 3.01s and used 3.01s of cpu time.

Test 4 finished in 2.99s and used 2.99s of cpu time.

Test 5 finished in 3.01s and used 3.01s of cpu time.

Checking accuracy...

Great Job! No errors occurred

Mean real time is: 3.00 seconds

Mean cpu time is: 3.00 seconds

Total acceleration acceleration_rate is 1.00!

Alternative Smooth Sort

Starting 5 test(s) on 5000000 elements...

Test 1 finished in 5.58s and used 6.19s of cpu time.

Test 2 finished in 5.01s and used 5.71s of cpu time.

Test 3 finished in 5.00s and used 5.76s of cpu time.

Test 4 finished in 5.01s and used 5.66s of cpu time.

Test 5 finished in 4.91s and used 5.59s of cpu time.

Checking accuracy...

Great Job! No errors occurred

Mean real time is: 5.10 seconds

Mean cpu time is: 5.78 seconds

Total acceleration acceleration_rate is 1.13!

Исходный код программы

https://github.com/ivancheprasov/highPerformance1

Результаты

Всё из списка задач было сделано.

Выводы

Tree Sort менее эффективен по времени, чем Smooth Sort.

Разбиение исходного массива значительно уменьшает время сортировки.

Параллелизация сбора результатов (формирование результирующего массива) из дерева, полученного одним потоком при сортировке tree sort, не принесло желаемых результатов и ускорило процесс только на треть секунды при 5 млн элементах.

Реализованный (альтернативный, урезанный) алгоритм Smooth Sort можно применять только на массивах малого размера. Для использования на больших массивах требуется дополнительно применить оригинальный алгоритм. Smooth Sort выполняется быстрее в частично отсортированном массиве, однако в данном случае неотсортированные элементы не

меняются местами, а просто идут со сдвигом. Поэтому затрачиваемое время больше чем в оригинальном алгоритме.