Национальный исследовательский ядерный университет «МИФИ» (Московский Инженерно–Физический Институт)
Кафедра №42 «Криптология и кибербезопасность»

Лабораторная работа №7: «Совместное применение OpenMP и MPI»

Рекомендуемая дата защиты: 14.12.2023

Цель работы

Приобрести навыки совместного применения технологий OpenMP и MPI для решения масштабных вычислительных задач.

Теоретическая подготовка

В рамках данной работы необходимо организовать поиск простых чисел в заданном диапазоне. Существует несколько методов решения этой задачи, здесь рассматривается простейший. Для доказательства простоты числа N можно прямым перебором убедиться, что оно не имеет делителей. Важно: для этого достаточно рассмотреть делители в диапазоне от 2 до $\left\lfloor \sqrt{N} \right\rfloor$ – если в разложении числа на множители один больше этого числа, значит, второй меньше, и уже должен был быть найден.

Так, для каждого числа задача сводится к перебору в цикле потенциальных делителей от 2 до $\left\lfloor \sqrt{N} \right\rfloor$ и нахождению остатка от деления этого числа на каждый из них. Если остаток нулевой, можно немедленно переходить к следующему числу. Если ни один из потенциальных делителей не разделил N без остатка – число простое.

Входными данными для алгоритма являются границы области поиска: $N_{start}..N_{end}$. Предлагается с помощью MPI разделить работу по просмотру этой области поиска. Корневой процессор MPI рассчитывает границы каждой рабочей области и рассылает их рабочим процессам MPI. Каждый рабочий процесс в цикле проверяет все потенциальные делители каждого из чисел в своей рабочей области: здесь уместно применить OpenMP.

При нахождении простого числа, рабочий процессор MPI должен записать его в память (важно обеспечить синхронизацию по записи для OpenMP). Наконец, корневой процессор MPI должен собрать результаты с рабочих процессоров и вывести проверенный диапазон и все найденные в нём простые числа.

Ход работы

1. Реализовать описанную в предыдущем разделе программу с использованием технологий MPI и OpenMP;
2. Подобрать диапазон поиска таким образом, чтобы время работы программы было не менее 30 секунд; чем больше – тем лучше;
3. Измерить время работы ускорение и эффективность реализации. Пусть число вычислительных ядер в системе обозначено c, число поддерживаемых потоков обозначено p. Если $p > c$, передать диспетчеру MPI число процессоров c и убедиться, что OpenMP использует не более p/c потоков. Если $p == c$, передать диспетчеру MPI число процессоров $c/2$ и убедиться, что OpenMP использует два потока. Необходимо с помощью соответствующих функций убедиться, что OpenMP и MPI используют корректное число процессоров. Замечание: разделение нагрузки таким образом не гарантирует, что операционная система выделит по одному ядру на каждый процесс MPI;
4. Измерить время работы каждого процесса MPI. Составить диаграмму распределения вычислительной нагрузки: столбчатая диаграмма, которая показывает время работы каждого из процессоров при решении задачи;
5. Оформить отчёт.

Оформление отчёта

1. Титульный лист: название института, название лабораторной работы, имя, фамилия, номер группы, год,...
2. Блок-схема и описание работы алгоритма;
3. Таблица входных данных: число процессов MPI, число потоков OpenMP, размер рабочей области, принцип её разделения;
4. Таблица экспериментальных данных: время работы последовательного алгоритма; время работы, ускорение, эффективность параллельного алгоритма. Время работы каждого из процессоров при параллельном решении задачи;
5. Диаграмма распределения вычислительной нагрузки;
6. Заключение: краткое описание проделанной работы.
7. Приложение: разработанные программные коды;