Московский Государственный Университет имени М. В. Ломоносова
Факультет Вычислительной Математики и Кибернетики

Отчет по практическому заданию по курсу СКиПОД

Воробьев Евгений
328 группа

Москва
2022

Содержание

1 Цель

Реализовать параллельную версию предложенного алгоритма с использованием технологий OpenMP и MPI.

2 Предложенный алгоритм

2.1 Название

Block Red-Black Ordering(Трехмерный)(https://link.springer.com/article/10.1023/A:1021738303840)

2.2 Описание

Параллельное упорядочение называется «блочное красно-черное упорядочение». В этом методе узлы анализируемой сетки делятся на несколько или множество блоков, и к блокам применяется красно-черное упорядочение. Поскольку блоки с одинаковым цветом независимы друг от друга, прямые и обратные замены в итерации ICCG могут быть распараллелены для каждого цвета. Преимущество нового метода состоит в том, что в каждой параллельной подстановке существует только одна точка синхронизации. Для оценки сходимости и параллельного ускорения метода было проведено аналитическое исследование с использованием теории упорядочивающих графов и численных тестов на скалярном параллельном компьютере. Аналитическое исследование показывает, что скорость сходимости улучшается за счет увеличения количества узлов в одном блоке и что легко устанавливается оптимальный размер блока для получения наилучшей скорости сходимости. Численные тесты показывают, что новый метод обеспечивает высокую скорость параллельного ускорения благодаря быстрой сходимости, небольшим затратам на синхронизацию и эффективному использованию кэша данных на скалярном параллельном компьютере.

3 OpenMP

3.1 Изменение кода

- Распараллелены вложенные циклы с помощью *pragma omp parallel*
- Добавлен замер времени с помощью *omp_ get_ wtime()*

(код программы можно посмотреть на https://github.com/user-vo2/-)

3.2 Тестирование

Запуск программы производлся для 1, 2, 4, 8, 32, 64 нитей по 4 раза. В таблицу записывалось среднее значение для каждого количества нитей.(здесь округлено до 3-х знаков после запятой)

	1	2	4	8	16	32	64
150	5.766	3.598	2.827	1.989	1.538	0.954	0.723
162	10.273	3.756	3.025	2.310	1.624	1.467	0.929
200	19.698	12.830	6.606	3.456	2.733	2.383	1.943
242	37.073	23.273	11.374	5.886	4.651	4.213	3.659
258	83.225	62.334	37.765	24.446	17.514	14.574	7.746

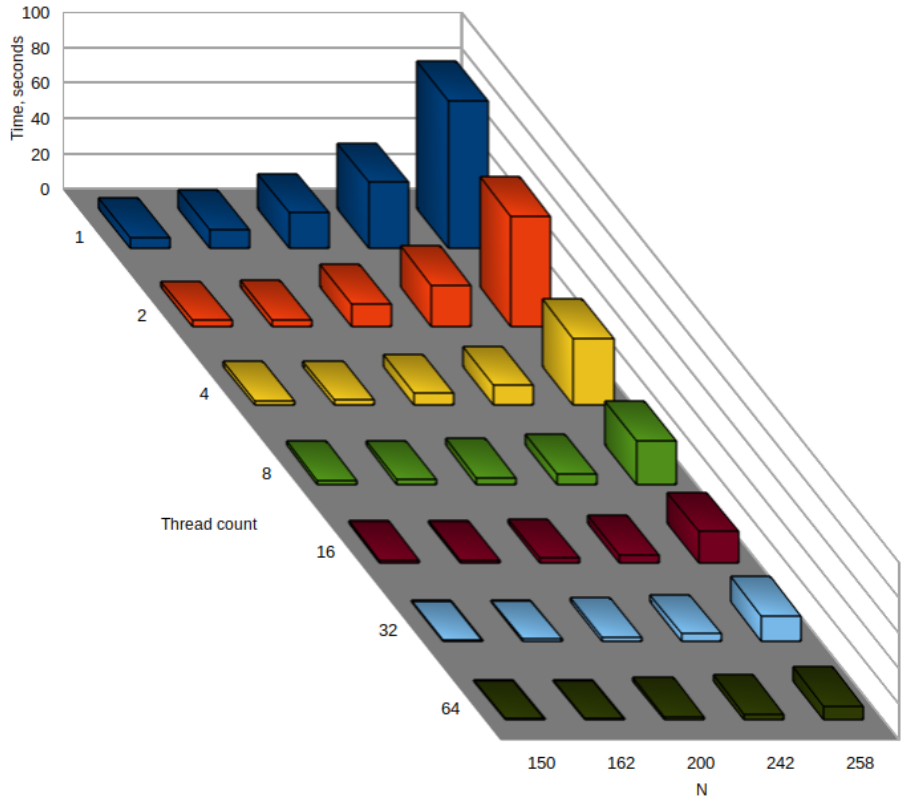

Рис. 1: График для OMP

4 MPI

4.1 Изменение кода

- Добавлено разделение матрицы по строкам в зависимости от количества процессов. Теперь каждый процесс обрабатывает только свою часть матрицы.
- Реализованы функции *pass_first_row()*, *pass_last_row()* и *wait_all()* для взаимодействия процессов.
- Добавлен замер времени с помощью *MPI_Wtime()*

(код программы можно посмотреть на https://github.com/user-vo2/-)

4.2 Тестирование

Запуск программы производлся для 1, 2, 4, 8, 32, 64 процессов по 5 раз. В таблицу записывалось среднее значение для каждого количества процессов.(здесь округлено до 3-х знаков после запятой)

	1	2	4	8	16	32	64
114	5.471	2.922	1.639	0.968	0.651	0.582	1.384
130	8.491	4.313	2.215	1.194	0.842	0.859	2.169
162	15.834	8.021	4.067	2.133	1.363	1.472	4.035
242	55.071	27.765	14.034	7.259	4.298	4.301	14.056
258	73.385	36.965	18.719	9.799	5.502	5.981	18.619

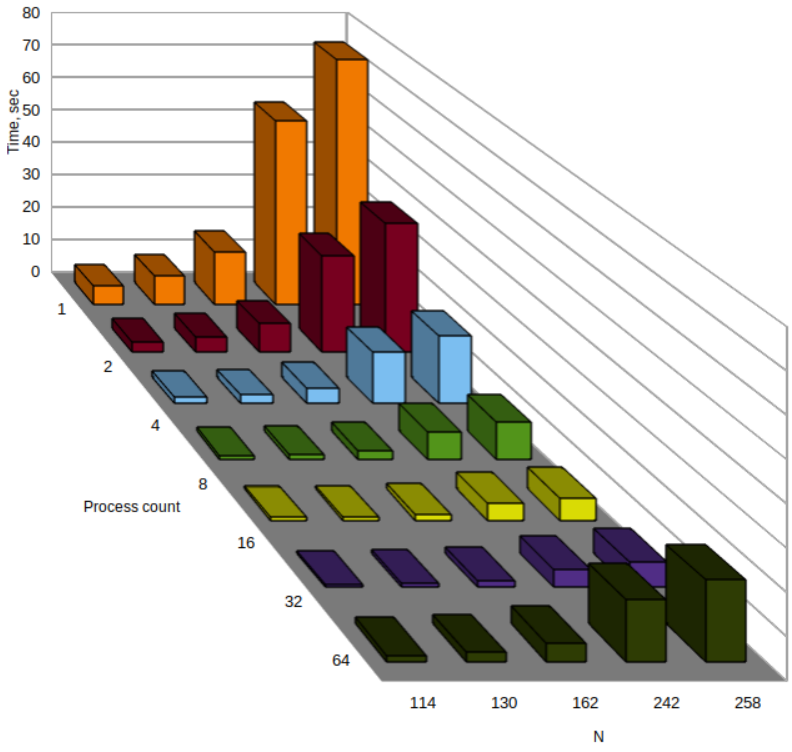

Рис. 2: График для MPI

5 Вывод

Выполнена работа по разработке параллельной версии алгоритма ленточного умножения матриц. Изучены технологии написания параллельных алгоритмов OpenMP и MPI. Проанализировано время выполнения алгоритмов на различных вычислительных системах.

Технология OpenMP крайне удобна в использовании, причем дает колоссальный прирост производительности на рассчитанных на многопоточные вычисления системах.

MPI можно назвать более низкоуровневой технологией: разработка MPI-программы знакомит с основами взаимодействия вычислительных узлов суперкомпьютера.

В общем зачете OpenMP показала лучшие результаты чем MPI. Это связано с накладными расходами на распараллеливание.

6 Ссылки

- https://github.com/user-vo2/-
- https://link.springer.com/article/10.1023/A:1021738303840