Отчет о выполнении 3 задания практикума кафедры СКИ

Р.М. Куприй, 323 группа

Факультет ВМК МГУ имени М.В. Ломоносова

1. Задание

По заданию написана программа для решения системы линейных уравнений $Ax = b$, с разреженной матрицей A и плотными векторами x, b. Система решается итерационным методом сопряженных градиентов (Conjugate Gradient Method) с предобуславливателем Якоби.

Итерационный метод решения заключается в последовательном уменьшении невязки и вычислении поправок к выбранному начальному приближению решения. Для повышения обусловленности матрицы и ускорения сходимости используется предобуславливатель Якоби. Предобуславливатель простой формы, записывает скалярное произведение $diag(A) * r$ в вектор z.

В программе реализована параллельная версия алгоритма, с использованием технологий OpenMP.

2. Методика тестирования

В программе замеряется время разложения матрицы и время решение системы обратным ходом Гаусса, общее время есть сумма этих двух составляющих. Для верификации результатов вычисляется невязка решения: $||Ax-b||$, а также при известном решении системы – невязка ошибки: $||x - solution||$. Примером известного решения может быть единичный вектор x, который возникает тогда, когда вектор b составлен из сумм строк матрицы A.

Для тестирования производительности использовалась параллельная вычислительная система Polus, с 3 вычислительными узлами, в каждом из которых 2 10 ядерных процессора IBM POWER8. Для компиляции использовался компилятор `xlc++` с флагом опимизации -O5. Запуски проводились с равномерным выделением ядер для программы и их закреплением с помощью команды `#BSUB -R affinity(core=x)` и специального предоставленного скрипта для запуска.

Для исключения выбросов и получения равномерной оценки по запускам, программа запускалась 6 раз для каждого замера, по 50 раз решая заданную систему уравнений. По прогонам внутри каждого запуска выбиралось среднее время, а среди всех запусков выбиралось минимальное. Получены резльтаты для разреженных матриц разного размера, при использовании 1, 2, 4, 8, 32, 64 нитей соотвественно.

3. Оценки эффективности MPI программы

При измерениях использовались расчетные сетки по кубикам со стороной 25, 50, 100, 250, 500. Соответственно, размер матрицы это третья степень размера стороны куба. Для каждого набора входных данных приведены графики общего времени

решения системы, ускорения и эффективности.

Для маленькой сетки, на кубике со стороной 25, максимальное ускорение достигается на 8 нитях, при этом эффективность ускорения резко падает с ростом числа нитей (Графики 1). Низкая эффективность при большом числе процессов обусловлена большой долей коммуникаций между процессами на фоне сравнительно небольших входных данных.

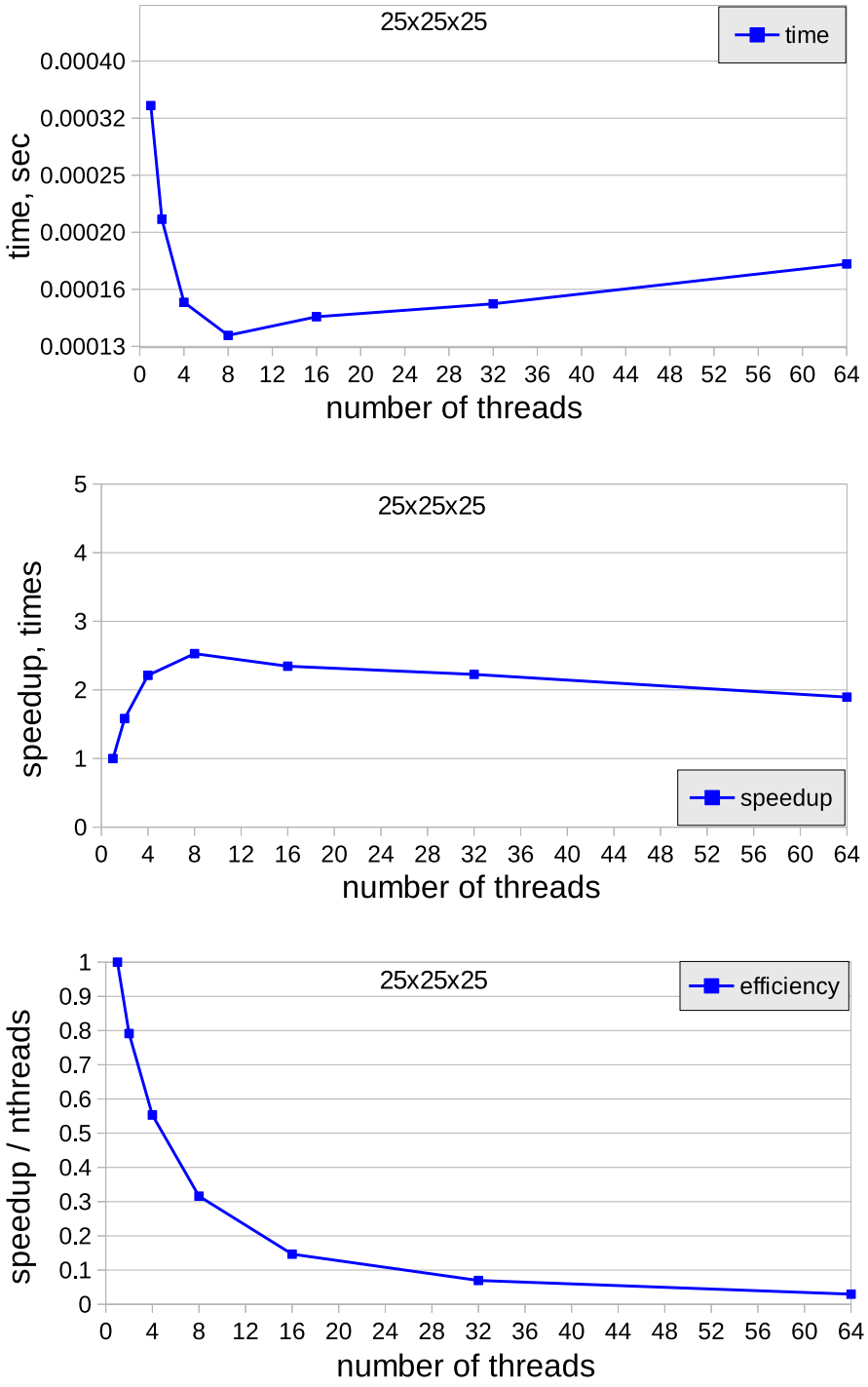

Рис. 1. Результаты распараллеливания программы для матрицы размером 25^3; верхний график – времена исполнения, срений – достигаемое ускорение, нижний – эффективность ускорения работы программы.

В следующем наборе измерений использовалась сетка кубика со стороной 50. С ростом объема данных растёт ресурс параллелизма, который можно использовать. Это повышает эффективность распараллеливания для малого числа нитей. Максимальное ускорение также достигается на 8 нитях (Графики 2).

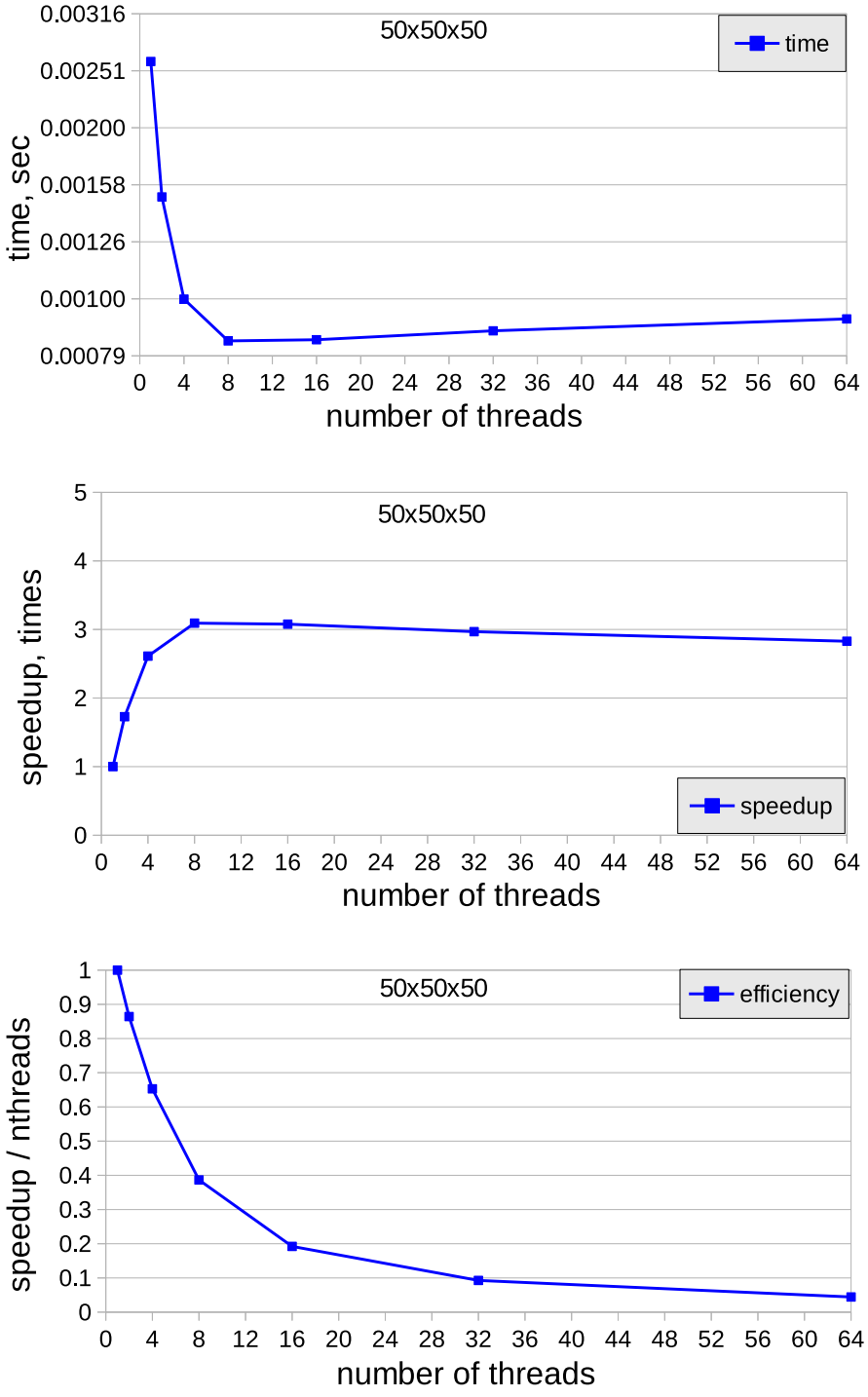

Рис. 2. Результаты распараллеливания программы для матрицы размером 50^3; верхний график – времена исполнения, срений – достигаемое ускорение, нижний – эффективность ускорения работы программы.

Набор данных входных данных среднего размера использует для построения си-

стемы сетку кубика со стороной 100. На этом датасете достигается максимальное ускорение – в 7 раз на 16 процессах. При переходе от 1 нити к 2 нитям число итераций у солвера сокращается, поэтому ускорение изменяется нелинейно (Графики 3).

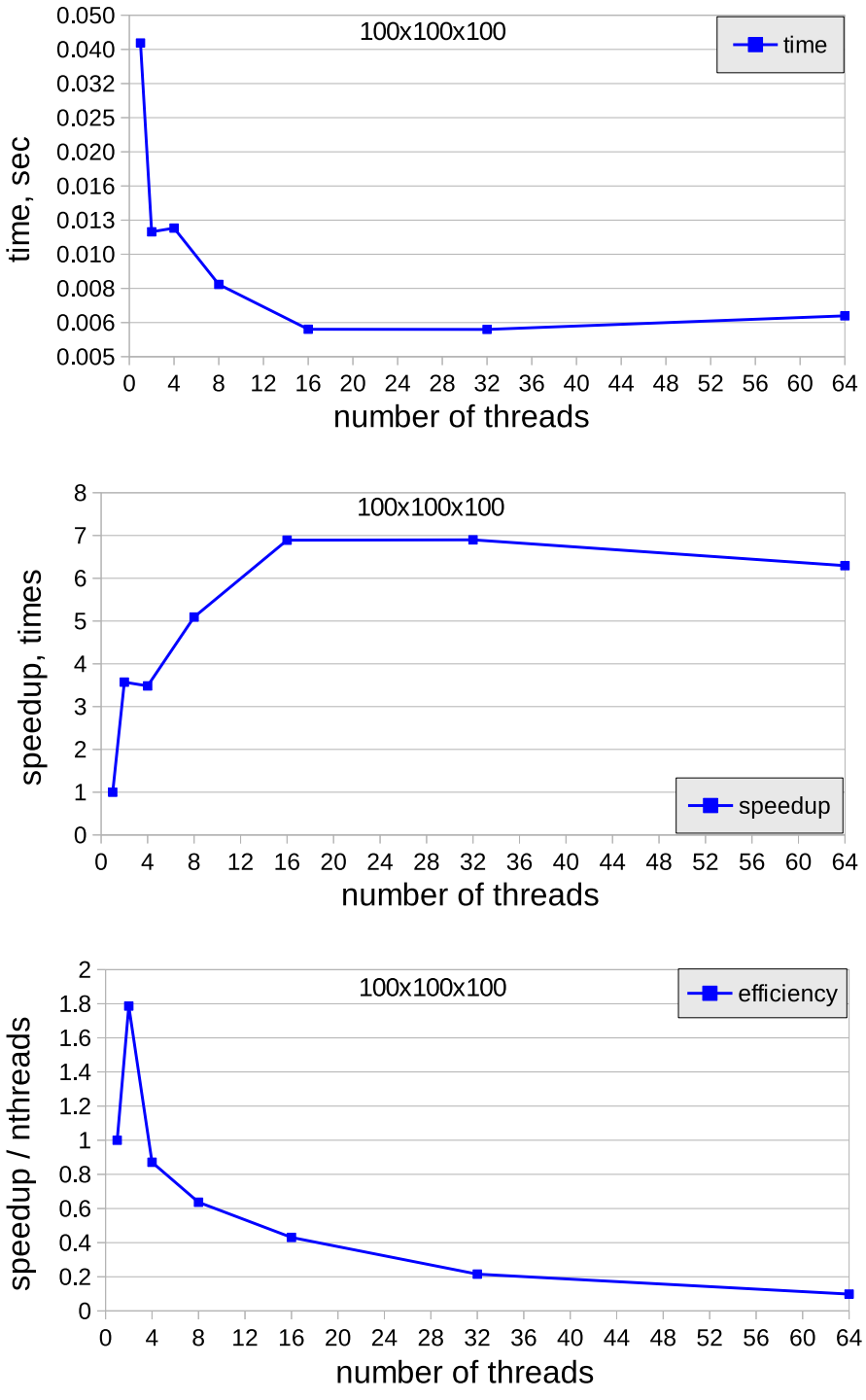

Рис. 3. Результаты распараллеливания программы для матрицы размером 100^3; верхний график – времена исполнения, срений – достигаемое ускорение, нижний – эффективность ускорения работы программы.

Для крупного набора входных данных используется кубик со стороной 250. На этом датасете максимальное ускорение достигается на 16 процессах (Графики 4).

Ускорение снижается на этом датасете в сравнении с предыдущим.

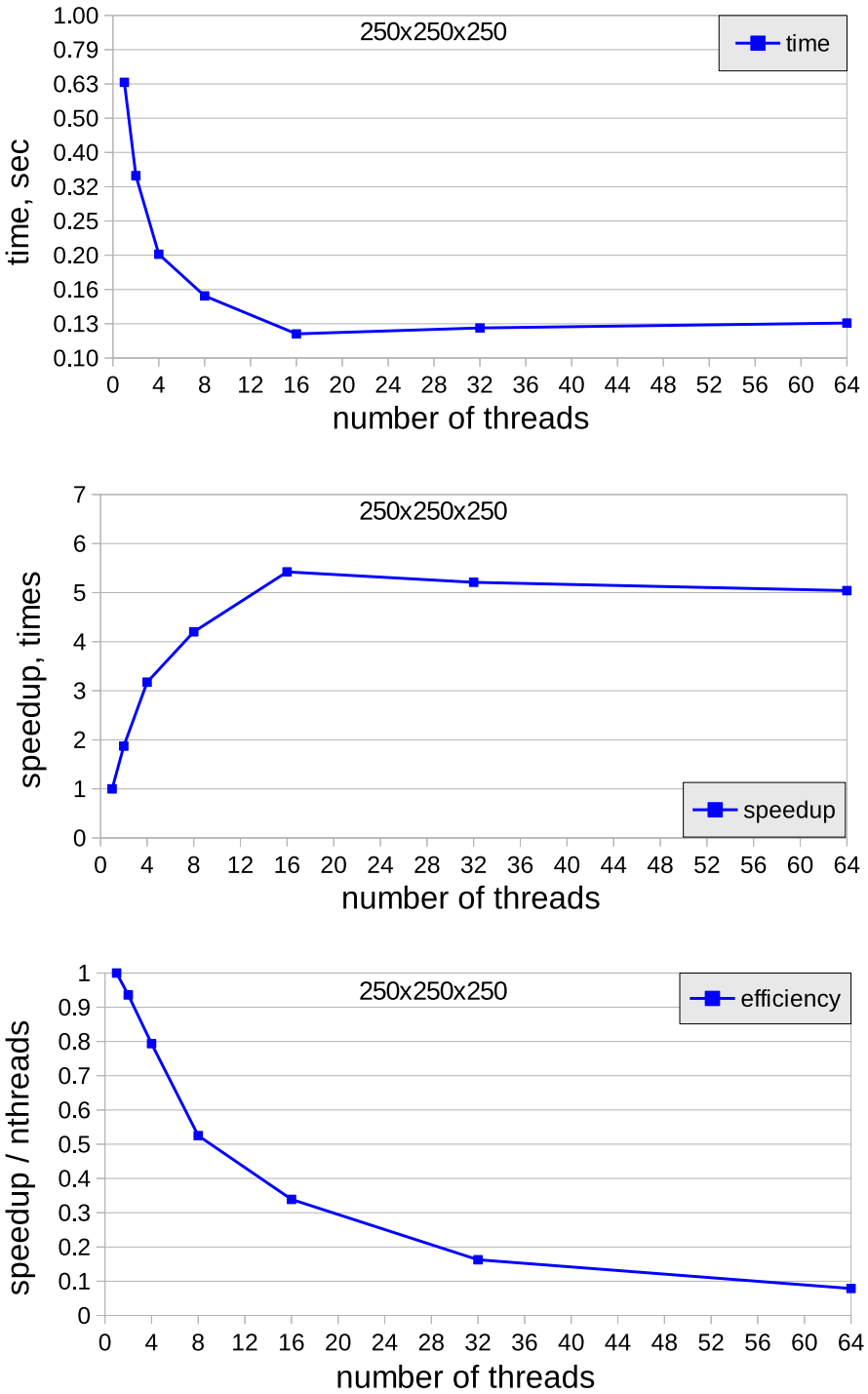

Рис. 4. Результаты распараллеливания программы для матрицы размером 250^3; верхний график – времена исполнения, срений – достигаемое ускорение, нижний – эффективность ускорения работы программы.

Для очень крупного набора входных данных используется кубик со стороной 500. Максимальное ускорение достигается на 16 и 32 нитях (Графики 5). Ускорение немного ниже, чем на крупном датасете.

Критерием остановки итерационного алгоритма была оценка вычисленной невяз-

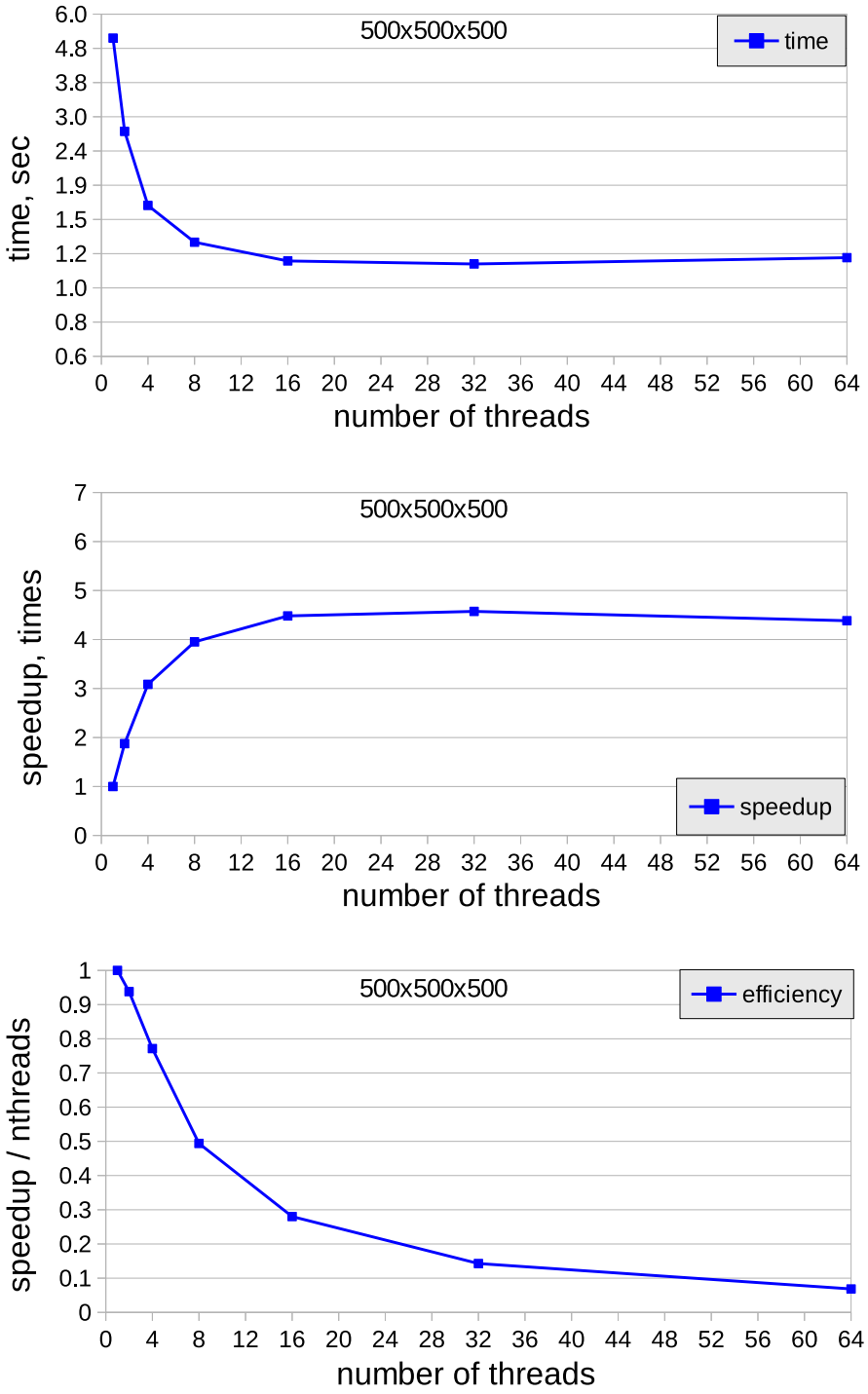

Рис. 5. Результаты распараллеливания программы для матрицы размером 500^3; верхний график – времена исполнения, срений – достигаемое ускорение, нижний – эффективность ускорения работы программы.

ки: при достижении невязки порядка 10^{-12} или меньше, алгоритм останавливается. Вторым ограничением является заданное максимальное число итераций, впрочем, на всех рассматриваемых матрицах алгоритм сходился (т.е. достигал заданного значения невязки). Также для верификации результатов вычислены невязки ошибки и решения, которые составили порядок $10^{-10} - 10^{-12}$.

4. Анализ полученных результатов

В результате всех измерений наблюдаются одинаковое поведение ускорения и эффективности. Эффективность резко падает с ростом числа нитей. Однако, для крупных размеров входных данных эффективность снижается плавнее.

Изменение ускорения напоминает логарифмический характер. При этом, после достижения максимального ускорения на 8 или 16 нитях, его значение не сильно меняется при росте числа нитей (до 32 или 64).

По результатам дополнительных тестов, производительность операции SpMV составляет тысячную долю от пиковой производительности кластера – 3.7 $Gflops$. Её вычислительная интенсивность составляет порядка 0.2. Принимая за максимальную пропускную способность шины памяти значение в 20 GB/s, можно получить, что TBP (Theoretical Bounded Performance) составляет около 4 $GFlops$. Это 0.01 процента от пиковой производительности. Порядок эффективности других базовых операций в сущности такой же.

Производительность операции скалярного умножения векторов составила 1.6 $GFlops$. Вычислительная интенсивность равно 0.125, соответственно значение TBP равно 2.5 $GFlops$.

Производительность операции `axpby` достигает 2.77 $GFlops$ по результатам замеров. Вычислительная производительность составляет 0.1875, тогда TBP будет достигать 3.75 $GFlops$.

Это довольно грубые оценки, которые тем не менее позволяют оценить порядок эффективности выполнения операций на рассматриваемой системе.