

# Отчет о выполнении 2 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 плотной матрицей  $A$  и векторами  $x$ ,  $b$ . Алгоритм решения системы уравнений состоит из приведения матрицы к верхнетреугольному виду методом отражений Хаусхолдера, а затем решение получившейся системы методом обратного хода Гаусса.

Метод отражений Хаусхолдера заключается в последовательном умножении матрицы разложения на плотную матрицу  $A$  и на вектор правой части. При этом, матрица разложения не хранится в явном виде, поскольку достаточно на каждой итерации разложения вычислять и хранить один вектор Хаусхолдера. Для более эффективной работы кэша, матрица  $A$  хранится по столбцам.

В программе реализована параллельная версия алгоритма, с использованием интерфейса передачи сообщений.

Приведено сравнение времени работы программы и её ускорения при использовании технологий OpenMPI и MPI.

## 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 `mpixlc` с флагом оптимизации `-O5`. Запуски проводились с привязкой процессов к физическим ядрам с помощью флагов запуска `-map-by core -bind-to core`.

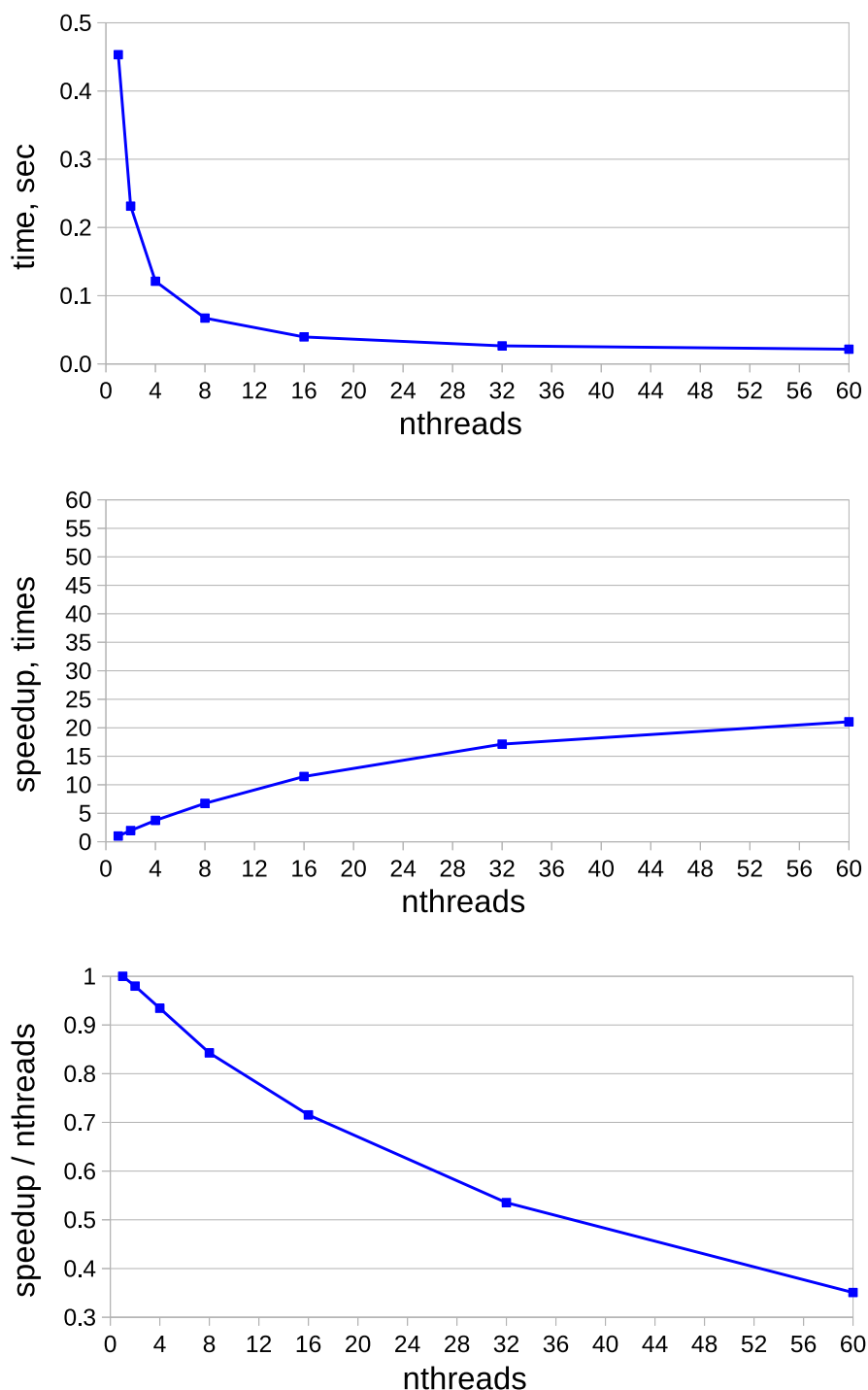
Программа запускалась 6 раз для каждого замера, с выбором наименьшего времени исполнения, чтобы исключить выбросы. Получены результаты для плотных матриц разного размера, при использовании 1, 2, 4, 8, 32, 60 процессов соответственно.

## 3. Оценки эффективности MPI программы

Для каждого набор входных данных приведены графики общего времени решения системы, ускорения и эффективности.

Для небольшой матрицы, размером 1000 строк, максимальное ускорение достига-

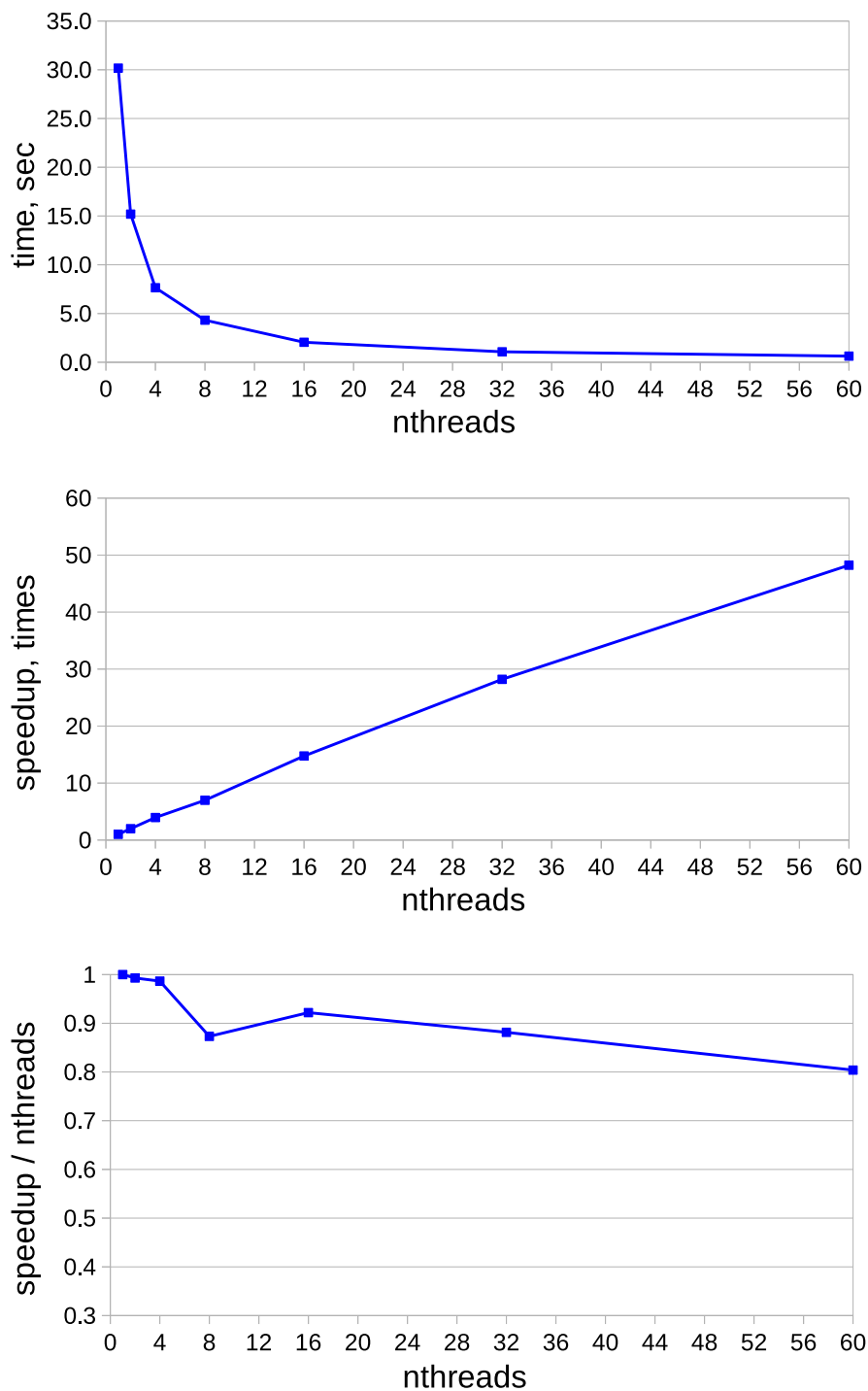
ется на 60 процессах, при этом падение эффективности, как и рост ускорения – носит достаточно линейных характер (Графики 1). Низкая эффективность при большом числе процессов обусловлена большой долей коммуникаций между процессами на фоне сравнительно небольших входных данных.



**Рис. 1.** Результаты распараллеливания программы для матрицы размером 1000 строк; верхний график – времена исполнения, средний – достигаемое ускорение, нижний – эффективность ускорения работы программы.

В следующем наборе измерений использовалась матрица размером 4000 строк. С

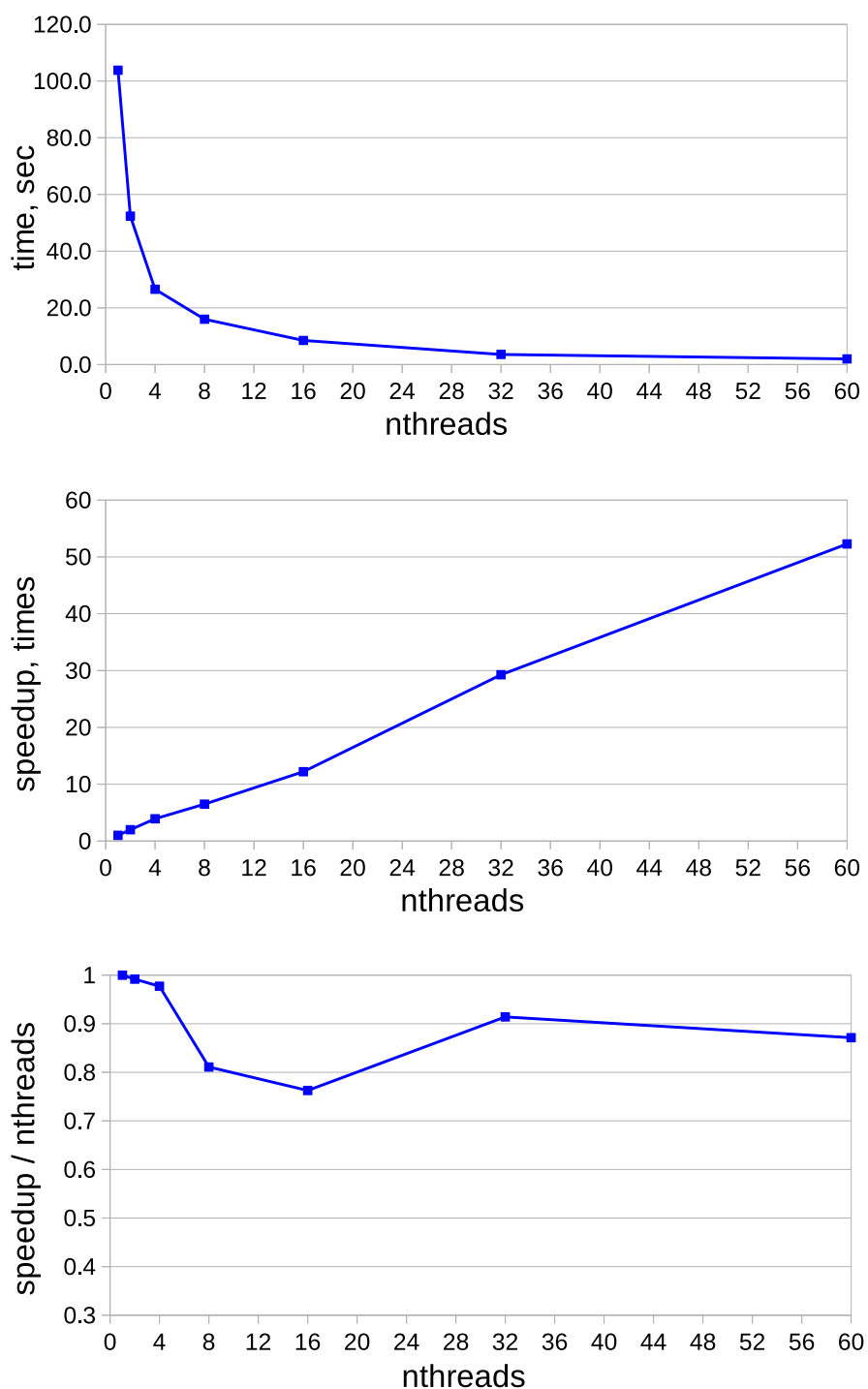
ростом объема данных растёт ресурс параллелизма, который можно использовать. Это повышает эффективность распараллеливания. Ускорение растёт почти линейно, а эффективность не опускается ниже 80%. (Графики 2).



**Рис. 2.** Результаты распараллеливания программы для матрицы размером 4000 строк; верхний график – времена исполнения, средний – достигаемое ускорение, нижний – эффективность ускорения работы программы.

Последний набор данных использует матрицу размером 6000 строк. На нём также эффективность работы программы превышает 80%, а ускорение растёт почти линейно.

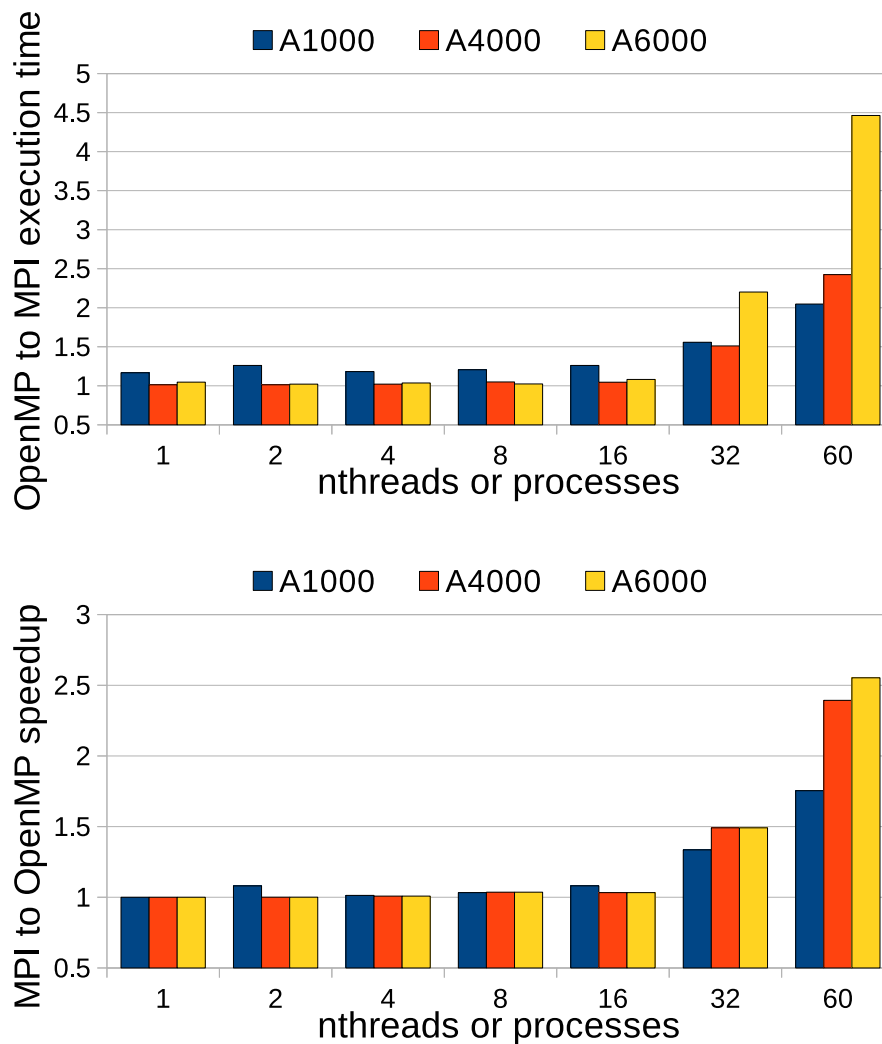
но с ростом числа процессов. На этом датасете достигается максимальное ускорение в 52 раза при использовании 60 процессов (Графики 3).



**Рис. 3.** Результаты распараллеливания программы для матрицы размером 6000 строк; верхний график – времена исполнения, средний – достигаемое ускорение, нижний – эффективность ускорения работы программы.

## 4. Сравнение OpenMPI и MPI версий программ

При использовании нитей больше, чем физических ядер на процессорах одного узла, эффективность распараллеливания OpenMP резко снижается, в отличие от MPI, где эффективно распределяются процессы между узлами вычислительной системы. На крупных датасетах разница в эффективности особо заметна. Так, при максимальном числе процессов, достигаемое ускорение MPI программы в 2.5 раза выше, чем при использовании OpenMP (Графики 4).



**Рис. 4.** Результаты сравнения OpenMP и MPI программ; верхний график – сравнение времени работы OpenMP и MPI программ (во сколько MPI быстрее), нижний график – сравнение достигаемого ускорения MPI и OpenMP программ (во сколько раз ускорение MPI выше чем ускорение OpenMP).