МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №01

по дисциплине «Параллельные алгоритмы»

Тема: Обмен сообщениями чётных и нечётных процессов

Студент гр. 9303 _____ Колованов Р.А.

Преподаватель _____ Татаринов Ю.С.

Санкт-Петербург

2023

Цель работы.

Написание программы обмена сообщениями чётных и нечётных процессов и определение зависимости времени обмена от длины сообщения. Сравнение двух технологий параллельного программирования MPI и OpenMP.

Формулировка задания.

Написать программу обмена сообщениями чётных и нечётных процессов. Замерить время на одну итерацию обмена и определить зависимость времени обмена от длины сообщения.

Краткое описание алгоритма.

Для начала исходная программа, написанная с использованием технологии MPI, была переписана с использованием технологии OpenMP. Алгоритм обмена сообщениями представляет собой следующую последовательность действий:

- В самом начале цикла определяется длина сообщения, равная 1;
- Для каждого четного процесса генерируется случайная строка, выступающая в роли отправляемого сообщения;
- Запоминается время, с которого начинается обмен сообщениями
- После этого четные процессы осуществляют отправку сообщения соседнему нечетному процессу (процесс 0 отправляет процессу 1, процесс 2 отправляет процессу 3 и так далее);
- Нечетные процессы принимают входящее сообщение;
- Вычисляется время, затраченное на обмен сообщениями;
- Находится максимальное время обмена сообщениями между всеми парами процессов, тем самым находя время обмена сообщениями на одной итерации, и выводится в консоль;
- Увеличивается длина сообщения;
- Осуществляется возврат в начало цикла до тех пор, пока длина сообщения не превысит максимальную длину.

Перед началом отправки или приема сообщений потоки синхронизируются при помощи барьера для того, чтобы отправки и прием начинались одновременно.

Сообщение представляет собой массив символов типа CHAR. В качестве шага между длинами сообщения выбрано значение 10000 символов. Максимальная длина сообщения – 10000000 символов. Программа запускалась с 6 процессами.

Поскольку в OpenMP не предусмотрена функциональность обмена сообщениями между процессами, была реализована собственная система обмена сообщениями.

Структура *Message* представляет собой сообщение, отправляемое от одного процесса другому. Сообщение хранит в себе массив данных, информацию о размере массива и размере его элементов, а также информацию об отправителе.

Структура *ThreadInputStorage* представляет собой хранилище входящих сообщений для потока. Каждый поток имеет собственное хранилище. Хранилище представлено в виде связного списка сообщений *Message*. С хранилищем можно выполнять два действия: добавить в него сообщение (отправить сообщение потоку) и взять из него сообщение (получить сообщение). Для обеспечения потокобезопасности при работе с хранилищем был использован замок *omp_lock_t*.

Для отправки и приема сообщений были созданы две функции *sendData* и *receiveData* соответственно. Функции являются блокирующими.

Формальное описание алгоритма.

Формальное описание алгоритма на сетях Петри для не модифицированной программы представлена на рис. 1.

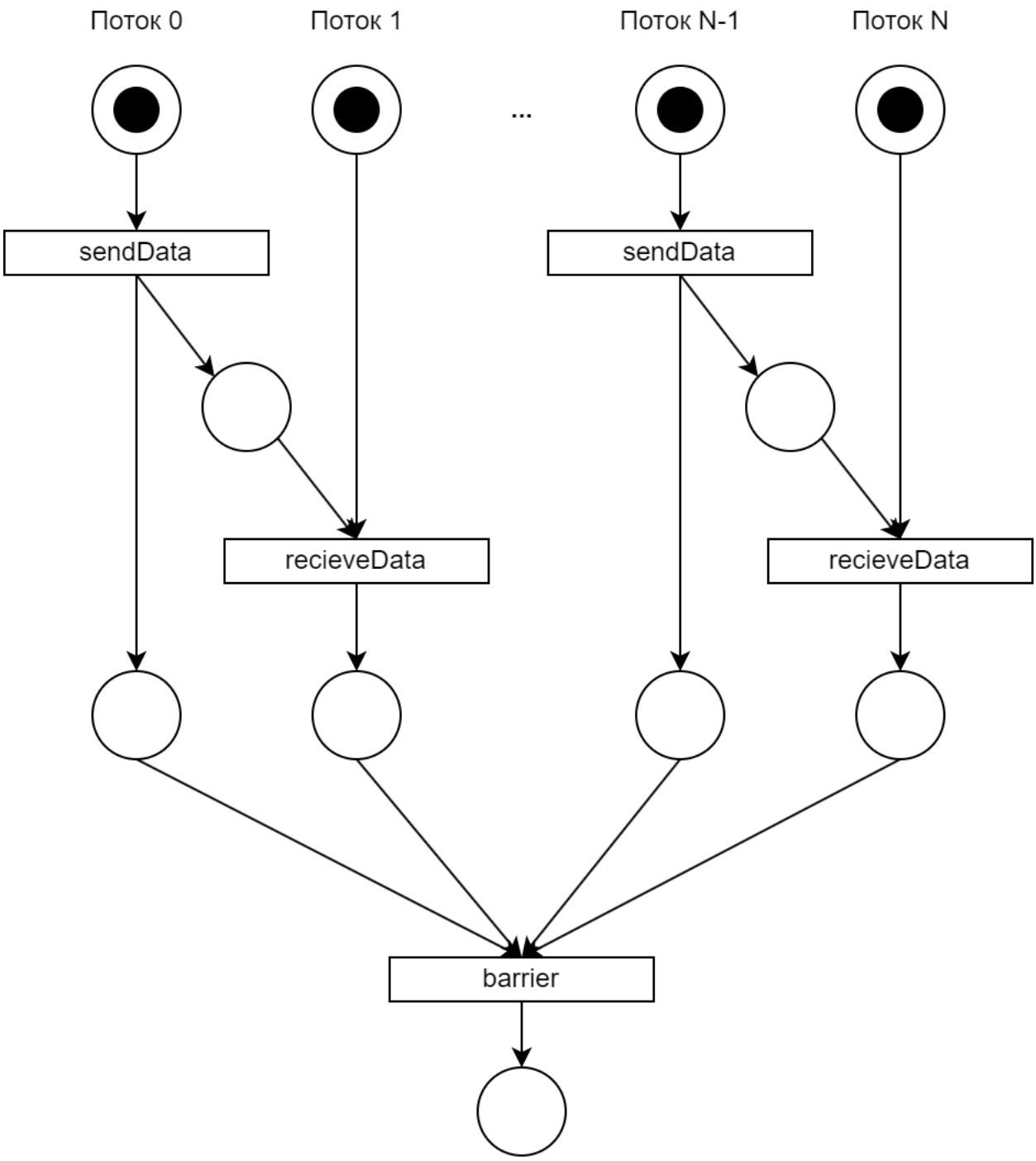

Рисунок 1 – Формальное описание алгоритма на сетях Петри.

Листинг программы.

Исходная программа, использующая технологию MPI, представлена в листинге 1.

Листинг 1. Программа на основе MPI.

```
#include <iostream>
#include <thread>
```

```
#include <mpi.h>

void fillArrayWithData(char* data, int count)
{
  for (int i = 0; i < count; ++i)
  {
      data[i] = i % 256;
  }
}

int main(int argc, char** argv) {
  const int messageMaxLength = 10000000, lengthStep = 10000;
  int processNumber, processRank;
  MPI_Status status;

  MPI_Init(&argc, &argv);
  MPI_Comm_size(MPI_COMM_WORLD, &processNumber);
  MPI_Comm_rank(MPI_COMM_WORLD, &processRank);

  for (int length = 1; length <= messageMaxLength; length += lengthStep) {
      char* buffer = new char[length];

      if (processRank % 2 == 0) {
        fillArrayWithData(buffer, length);
      }

      MPI_Barrier(MPI_COMM_WORLD);
      double startTime = MPI_Wtime();

      if (processRank % 2 == 0) {
        if (processRank < processNumber - processNumber % 2) {
            MPI_Send(buffer,   length,   MPI_CHAR,   processRank   +   1,   0,
MPI_COMM_WORLD);
        }
      }
      else {
        MPI_Recv(buffer,  length,  MPI_CHAR,  processRank  -  1,  MPI_ANY_TAG,
MPI_COMM_WORLD, &status);
      }

      double deltaTime = MPI_Wtime() - startTime;
      delete[] buffer;

      double maxTime;
      MPI_Reduce(&deltaTime,    &maxTime,    1,    MPI_DOUBLE,    MPI_MAX,    0,
MPI_COMM_WORLD);

      if (processRank == 0) {
        printf("%.7f, ", maxTime);
        std::this_thread::sleep_for(std::chrono::milliseconds(50));
      }

      MPI_Barrier(MPI_COMM_WORLD);
  }

  MPI_Finalize();

  return 0;
}
```

Переписанная программа, использующая технологию OpenMP, представлена в листинге 2.

Листинг 2. Программа на основе OpenMP.

```
#include <iostream>
#include <array>
#include <list>
#include <thread>
#include <omp.h>

/*!
 * \brief Сообщение. Содержит данные и информацию об отправителе.
 */
struct Message
{
  static const int ANY_THREAD = -1;

  Message() :
      data{ nullptr },
      count{ 0 },
      typeSize{ 0 },
      senderId{ ANY_THREAD }
  {}

  Message(const Message&) = delete;
  Message& operator=(const Message&) = delete;
  Message(Message&&) = delete;
  Message& operator=(Message&&) = delete;

  ~Message()
  {
      reset();
  }

  void setData(void* data, int count, int typeSize, int senderId = ANY_THREAD)
  {
      reset();
      this->senderId = senderId;
      this->count = count;
      this->typeSize = typeSize;
      this->data = new char[count * typeSize];
      std::memcpy(this->data, data, count * typeSize);
  }

  void reset()
  {
      if (data != nullptr)
      {
        delete[] data;
        data = nullptr;
      }
      count = 0;
      typeSize = 0;
      senderId = ANY_THREAD;
  }

  void* data;             // Указатель на данные
  size_t count;           // Количество данных
  size_t typeSize;        // Размер типа данных
  short int senderId; // ID потока-отправителя
};
```

```
/*!
 * \brief Хранилище сообщений. Хранит сообщения для определенного потока в виде
связного списка.
 */
struct ThreadInputStorage
{
  explicit ThreadInputStorage() :
      messages{},
      storageLock{ nullptr }
  {
      omp_init_lock(&storageLock);
  }

  ~ThreadInputStorage()
  {
      omp_destroy_lock(&storageLock);
  }

  void pushMessage(Message* message)
  {
      omp_set_lock(&storageLock);
      messages.push_back(message);
      omp_unset_lock(&storageLock);
  }

  Message* popMessage(int senderId = Message::ANY_THREAD)
  {
      Message* result = nullptr;

      omp_set_lock(&storageLock);
      if (!messages.empty())
      {
        for (auto it = messages.cbegin(); it != messages.cend(); ++it)
        {
            if (senderId == Message::ANY_THREAD || senderId == (*it)->senderId)
            {
              result = *it;
              messages.erase(it);
              break;
            }
        }
      }
      omp_unset_lock(&storageLock);

      return result;
  }

  omp_lock_t storageLock;           // Мьютекс на доступ к списку сообщений
  std::list<Message*> messages;     // Связный список сообщений
};

namespace
{
  constexpr int THREADS = 6;
  std::array<ThreadInputStorage, THREADS> INPUT_STORAGES;
}

/*!
 * \brief Функция отправки сообщения другому потоку.
 *
 * Функция является блокирующей - освобождается после того, как данные из
входного буффера будут скопированы и отправлены.
```

```
 *
 * \param data Указатель на массив данных, который необходимо отправить.
 * \param count Количество элементов в массиве данных
 * \param typeSize Размер одного элемента массива в байтах
 * \param destination ID потока, которому необходимо отправить сообщение
 */
void sendData(void* data, int count, int typeSize, int destination)
{
  if (destination < 0 || destination >= THREADS)
  {
      return;
  }

  auto& storage = INPUT_STORAGES.at(destination);
  auto* message = new Message;
  message->setData(data, count, typeSize, omp_get_thread_num());
  storage.pushMessage(message);
}

/*!
 * \brief Функция приема сообщения от другого потока.
 *
 * Функция является блокирующей - освобождается после того, как данные из
сообщения буду получены.
 *
 * \param data Указатель на массив данных, куда необходимо записать полученные
данные.
 * \param count Количество элементов в массиве данных
 * \param typeSize Размер одного элемента массива в байтах
 * \param source ID потока, от которого необходимо получить сообщение
 */
Message* recieveData(void* data, int count, int typeSize, int source =
Message::ANY_THREAD)
{
  auto& storage = INPUT_STORAGES.at(omp_get_thread_num());
  auto* message = storage.popMessage(source);

  while (message == nullptr)
  {
      message = storage.popMessage(source);
  }

  int size = count * typeSize;
  if (size > message->count * message->typeSize)
  {
      size = message->count * message->typeSize;
  }

  std::memcpy(data, message->data, size);

  delete message;
}

void fillArrayWithData(char* data, int count)
{

  for (int i = 0; i < count; ++i)
  {
      data[i] = i % 256;
  }
}

int main()
```

```
{
  const int messageMaxLength = 10000000, lengthStep = 10000;

  for (int length = 1; length < messageMaxLength; length += lengthStep)
  {
      double maxTime = -1;

      #pragma omp parallel num_threads(THREADS)
      {
        const auto threadId = omp_get_thread_num();
        char* buffer = new char[length];

        if (threadId % 2 == 0)
        {
            fillArrayWithData(buffer, length);
        }

        #pragma omp barrier
        double startTime = omp_get_wtime();

        if (threadId % 2 == 0) {
            if (threadId < THREADS - THREADS % 2) {
              sendData(buffer, length, sizeof(char), threadId + 1);
            }
        }
        else
        {
            recieveData(buffer, length, sizeof(char), threadId - 1);
        }

        double elapsedTime = omp_get_wtime() - startTime;
        delete[] buffer;

        #pragma omp critical
        {
            #pragma omp flush(maxTime)
            if (elapsedTime > maxTime)
            {
              maxTime = elapsedTime;
            }
        }

        #pragma omp barrier
      }

      printf("%.7f, ", maxTime);
      std::this_thread::sleep_for(std::chrono::milliseconds(50));
  }

  return 0;
}
```

Результаты работы программы.

Результаты работы программы, использующей технологию MPI, представлена в листинге 3.

Листинг 3. Результаты работы программы (MPI).

```
0.0000349,  0.0000145,  0.0000093,  0.0000096,  0.0000111,  0.0000113,  0.0000134,
0.0000269,  0.0000307,  0.0000322,  0.0000361,  0.0000408,  0.0000193,  0.0000232,
0.0000209,  0.0000232,  0.0000561,  0.0000249,  0.0000269,  0.0000307,  0.0000707,
0.0000304,  0.0000323,  0.0000351,  0.0000350,  0.0000318,  0.0000316,  0.0000370,
0.0000533,  0.0000421,  0.0000456,  0.0000445,  0.0000400,  0.0000465,  0.0000468,
0.0000442,  0.0000504,  0.0000631,  0.0000456,  0.0000426,  0.0000498,  0.0000601,
0.0000611,  0.0000627,  0.0000639,  0.0000599,  0.0000610,  0.0000644,  0.0000728,
0.0001633,  0.0000551,  0.0000594,  0.0001641,  0.0001812,  0.0000589,  0.0001974,
0.0000597,  0.0001650,  0.0000847,  0.0001766,  0.0000806,  0.0001877,  0.0000798,
0.0001951,  0.0000820,  0.0001354,  0.0000941,  0.0002133,  0.0000849,  0.0002245,
0.0000821,  0.0001816,  0.0000986,  0.0002017,  0.0001097,  0.0003189,  0.0000946,
0.0002905,  0.0001000,  0.0002400,  0.0001270,  0.0001794,  0.0000980,  0.0002205,
0.0000851,  0.0002413,  0.0000908,  0.0002253,  0.0001101,  0.0002433,  0.0000933,
0.0002562,  0.0001273,  0.0003792,  0.0001385,  0.0002101,  0.0001666,  0.0003046,
0.0000994,  0.0002553,  0.0002700,  0.0003061,  0.0004461,  0.0002358,  0.0002727,
0.0002250,  0.0002155,  0.0002368,  0.0001928,  0.0001949,  0.0002712,  0.0002931,
0.0004818,  0.0003998,  0.0003462,  0.0003209,  0.0003362,  0.0003237,  0.0003044,
0.0003104,  0.0007269,  0.0003366,  0.0002144,  0.0002145,  0.0002640,  0.0002755,
0.0003255,  0.0003297,  0.0004497,  0.0003504,  0.0003632,  0.0003577,  0.0003366,
0.0003700,  0.0004049,  0.0007675,  0.0003802,  0.0003641,  0.0004032,  0.0004174,
0.0003677,  0.0003548,  0.0002450,  0.0002958,  0.0002603,  0.0003218,  0.0003678,
0.0003768,  0.0004065,  0.0003846,  0.0004263,  0.0004328,  0.0004130,  0.0007738,
0.0004139,  0.0004403,  0.0004355,  0.0004191,  0.0004377,  0.0007443,  0.0004877,
0.0004276,  0.0004465,  0.0004211,  0.0007783,  0.0002730,  0.0004403,  0.0003608,
0.0003666,  0.0004326,  0.0005935,  0.0004593,  0.0004848,  0.0004478,  0.0004941,
0.0008737,  0.0004679,  0.0004731,  0.0004736,  0.0004884,  0.0007909,  0.0006045,
0.0005616,  0.0004787,  0.0004631,  0.0008012,  0.0004890,  0.0003271,  0.0003676,
0.0003970,  0.0004227,  0.0007722,  0.0005945,  0.0005359,  0.0005182,  0.0005767,
0.0008152,  0.0006485,  0.0005549,  0.0005458,  0.0008560,  0.0005598,  0.0005700,
0.0005820,  0.0005413,  0.0004357,  0.0004316,  0.0004730,  0.0005052,  0.0006941,
0.0006533,  0.0006223,  0.0006112,  0.0006546,  0.0006873,  0.0006500,  0.0008584,
0.0006210,  0.0006127,  0.0009971,  0.0006630,  0.0006044,  0.0006847,  0.0010035,
0.0006654,  0.0004316,  0.0004386,  0.0005208,  0.0007572,  0.0008265,  0.0006810,
0.0007468,  0.0009650,  0.0007454,  0.0005519,  0.0006125,  0.0009233,  0.0006661,
0.0007533,  0.0007799,  0.0011085,  0.0007658,  0.0006819,  0.0006189,  0.0004741,
0.0006093,  0.0006017,  0.0008157,  0.0007552,  0.0007929,  0.0006901,  0.0012189,
0.0006762,  0.0006934,  0.0007419,  0.0011141,  0.0011761,  0.0008269,  0.0010454,
0.0007564,  0.0007512,  0.0007465,  0.0010158,  0.0006284,  0.0005577,  0.0006554,
0.0009077,  0.0008448,  0.0007548,  0.0010529,  0.0008442,  0.0008310,  0.0008411,
0.0012971,  0.0008642,  0.0008060,  0.0008256,  0.0011870,  0.0007950,  0.0008236,
0.0008762,  0.0012144,  0.0008873,  0.0008400,  0.0006262,  0.0006667,  0.0010285,
0.0008873,  0.0009713,  0.0008706,  0.0012290,  0.0008357,  0.0007946,  0.0007645,
0.0011526,  0.0008900,  0.0009144,  0.0009682,  0.0009382,  0.0009027,  0.0010537,
0.0008727,  0.0008766,  0.0010224,  0.0008831,  0.0010740,  0.0006212,  0.0006754,
0.0011797,  0.0010884,  0.0009717,  0.0009510,  0.0012039,  0.0009258,  0.0011435,
0.0012418,  0.0008982,  0.0010231,  0.0012118,  0.0009068,  0.0008722,  0.0013742,
0.0009865,  0.0009635,  0.0012512,  0.0007001,  0.0008251,  0.0011307,  0.0010698,
0.0009496,  0.0013070,  0.0010255,  0.0010113,  0.0012925,  0.0010800,  0.0010215,
0.0013577,  0.0009866,  0.0009334,  0.0012443,  0.0009156,  0.0009591,  0.0012515,
0.0009513,  0.0011427,  0.0011913,  0.0009972,  0.0014823,  0.0006389,  0.0008877,
0.0011958,  0.0011843,  0.0010755,  0.0013399,  0.0010342,  0.0009443,  0.0013633,
0.0010215,  0.0012014,  0.0013342,  0.0011137,  0.0014274,  0.0011255,  0.0010713,
0.0013831,  0.0011095,  0.0011994,  0.0007285,  0.0009137,  0.0012685,  0.0011092,
0.0010956,  0.0016565,  0.0012092,  0.0011664,  0.0014720,  0.0010777,  0.0012936,
0.0011420,  0.0011358,  0.0014150,  0.0011555,  0.0010075,  0.0016448,  0.0011563,
0.0015174,  0.0011649,  0.0012725,  0.0007502,  0.0009464,  0.0013450,  0.0012349,
0.0011810,  0.0014197,  0.0011691,  0.0013348,  0.0012310,  0.0015410,  0.0010830,
0.0012001,  0.0015764,  0.0012904,  0.0012185,  0.0013917,  0.0012833,  0.0013555,
0.0012141,  0.0013304,  0.0016580,  0.0009026,  0.0011048,  0.0012666,  0.0012737,
0.0016301,  0.0012483,  0.0011824,  0.0014905,  0.0014455,  0.0015616,  0.0012907,
0.0014435,  0.0013398,  0.0015011,  0.0011665,  0.0018061,  0.0014661,  0.0012229,
```

```
0.0016993,  0.0012826,  0.0012726,  0.0008971,  0.0012398,  0.0020223,  0.0015400,
0.0014759,  0.0016717,  0.0013198,  0.0012691,  0.0018589,  0.0013691,  0.0014842,
0.0012504,  0.0016969,  0.0014878,  0.0012399,  0.0014194,  0.0015131,  0.0016011,
0.0014247,  0.0016160,  0.0016768,  0.0016668,  0.0009612,  0.0012945,  0.0014579,
0.0014115,  0.0016850,  0.0014451,  0.0017524,  0.0014884,  0.0016052,  0.0014439,
0.0015900,  0.0015094,  0.0014206,  0.0018012,  0.0014426,  0.0017492,  0.0014967,
0.0017512,  0.0014505,  0.0010481,  0.0014418,  0.0015871,  0.0018585,  0.0014557,
0.0018370,  0.0014622,  0.0021698,  0.0014057,  0.0017074,  0.0014557,  0.0019047,
0.0014938,  0.0018542,  0.0016163,  0.0018520,  0.0014512,  0.0017594,  0.0015554,
0.0013590,  0.0015343,  0.0016861,  0.0016547,  0.0019123,  0.0017519,  0.0019221,
0.0015756,  0.0017154,  0.0016151,  0.0017527,  0.0016323,  0.0017630,  0.0014775,
0.0020139,  0.0015115,  0.0020450,  0.0016894,  0.0011992,  0.0016294,  0.0016753,
0.0016073,  0.0020310,  0.0016205,  0.0018372,  0.0017010,  0.0019033,  0.0016095,
0.0018975,  0.0016095,  0.0019013,  0.0015965,  0.0019021,  0.0015117,  0.0018718,
0.0017906,  0.0019631,  0.0021032,  0.0016238,  0.0019210,  0.0010262,  0.0017351,
0.0016313,  0.0020223,  0.0016965,  0.0019738,  0.0017438,  0.0018453,  0.0016733,
0.0020736,  0.0018995,  0.0017190,  0.0020206,  0.0017889,  0.0020197,  0.0017499,
0.0019607,  0.0017876,  0.0017886,  0.0014768,  0.0018624,  0.0017494,  0.0020231,
0.0018754,  0.0020036,  0.0017285,  0.0020657,  0.0017963,  0.0021041,  0.0019000,
0.0018156,  0.0020767,  0.0022088,  0.0020972,  0.0021274,  0.0018587,  0.0020503,
0.0018135,  0.0021374,  0.0012651,  0.0017984,  0.0018505,  0.0020734,  0.0019376,
0.0019854,  0.0021281,  0.0018044,  0.0025286,  0.0019452,  0.0020657,  0.0020989,
0.0019158,  0.0020373,  0.0019041,  0.0021253,  0.0020809,  0.0018798,  0.0020894,
0.0019351,  0.0015238,  0.0019327,  0.0019282,  0.0021604,  0.0022371,  0.0019291,
0.0022014,  0.0022418,  0.0018937,  0.0022776,  0.0019413,  0.0021758,  0.0021163,
0.0018698,  0.0022277,  0.0020810,  0.0021320,  0.0023121,  0.0019388,  0.0014695,
0.0020477,  0.0019612,  0.0023955,  0.0023941,  0.0020314,  0.0022944,  0.0021401,
0.0019986,  0.0023195,  0.0022453,  0.0022442,  0.0021173,  0.0024625,  0.0023564,
0.0020126,  0.0024388,  0.0020567,  0.0022349,  0.0022553,  0.0015582,  0.0021931,
0.0021019,  0.0023626,  0.0021087,  0.0020720,  0.0022795,  0.0022834,  0.0022562,
0.0024203,  0.0022902,  0.0021669,  0.0022578,  0.0022844,  0.0020730,  0.0022939,
0.0024806,  0.0020854,  0.0023057,  0.0020302,  0.0020445,  0.0022011,  0.0023482,
0.0023115,  0.0021177,  0.0024217,  0.0023461,  0.0022012,  0.0023645,  0.0023128,
0.0021967,  0.0023812,  0.0023729,  0.0021681,  0.0025343,  0.0024641,  0.0022152,
0.0024100,  0.0025293,  0.0020543,  0.0014453,  0.0022335,  0.0022244,  0.0024744,
0.0021805,  0.0026029,  0.0022216,  0.0023967,  0.0023087,  0.0021805,  0.0025429,
0.0023906,  0.0022360,  0.0026844,  0.0023680,  0.0023006,  0.0024856,  0.0023346,
0.0026252,  0.0022806,  0.0026444,  0.0018234,  0.0023367,  0.0024407,  0.0025935,
0.0030918,  0.0025335,  0.0021782,  0.0027958,  0.0024795,  0.0022603,  0.0025267,
0.0025221,  0.0025942,  0.0023199,  0.0027080,  0.0025446,  0.0022986,  0.0026809,
0.0024932,  0.0023548,  0.0026516,  0.0017143,  0.0024099,  0.0025535,  0.0024805,
0.0025850,  0.0026117,  0.0025658,  0.0022950,  0.0026799,  0.0025036,  0.0027272,
0.0023352,  0.0027575,  0.0026851,  0.0026650,  0.0025705,  0.0027082,  0.0026491,
0.0024286,  0.0026066,  0.0019482,  0.0024661,  0.0026474,  0.0026343,  0.0023671,
0.0028194,  0.0026459,  0.0026918,  0.0026908,  0.0027522,  0.0024853,  0.0028992,
0.0029912,  0.0027553,  0.0024934,  0.0029264,  0.0027334,  0.0027667,  0.0024830,
0.0027927,  0.0027061,  0.0028465,  0.0025601,  0.0016995,  0.0025604,  0.0027774,
0.0027425,  0.0024354,  0.0029206,  0.0027656,  0.0028009,  0.0026266,  0.0029796,
0.0027653,  0.0028726,  0.0026627,  0.0026417,  0.0030193,  0.0027408,  0.0027991,
0.0030521,  0.0025195,  0.0020237,  0.0027314,  0.0029152,  0.0029010,  0.0025972,
0.0030050,  0.0027562,  0.0028683,  0.0028180,  0.0027152,  0.0029363,  0.0028490,
0.0028410,  0.0028852,  0.0027172,  0.0028928,  0.0027226,  0.0029872,  0.0028384,
0.0027217,  0.0031388,  0.0018773,  0.0025769,  0.0028295,  0.0028760,  0.0026244,
0.0030459,  0.0030076,  0.0029360,  0.0026165,  0.0032777,  0.0028855,  0.0029678,
0.0030967,  0.0028516,  0.0028885,  0.0029798,  0.0027875,  0.0030036,  0.0029840,
0.0027923,  0.0018704,  0.0028782,  0.0029956,  0.0029899,  0.0030283,  0.0027366,
0.0030583,  0.0027630,  0.0028708,  0.0030913,  0.0030210,  0.0030380,  0.0027025,
0.0031150,  0.0028277,  0.0030027,  0.0032076,  0.0029316,  0.0031216,  0.0019877,
0.0027628,  0.0031096,  0.0029155,  0.0028912,  0.0031083,  0.0032668,  0.0028671,
0.0031635,  0.0030372,  0.0030423,  0.0030864,  0.0030502,  0.0030565,  0.0030546,
0.0031246,  0.0030193,  0.0031430,  0.0031367,  0.0031108,  0.0031952,  0.0031728,
0.0030150,  0.0034388,  0.0028933,  0.0032720,  0.0030425,  0.0031832,  0.0032326,
0.0031886,  0.0034765,  0.0031046,  0.0028539,  0.0029359,  0.0033410,  0.0033927,
```

```
0.0030151,  0.0030209,  0.0031468,  0.0031609,  0.0021757,  0.0029309,  0.0030208,
0.0029859,  0.0030565,  0.0033202,  0.0031458,  0.0031233,  0.0028227,  0.0034197,
0.0028458,  0.0033103,  0.0031333,  0.0031653,  0.0030229,  0.0029414,  0.0028800,
0.0033082,  0.0031233,  0.0033457,  0.0031499,  0.0020404,  0.0028413,  0.0031073,
0.0031772,  0.0030482,  0.0032209,  0.0032697,  0.0031207,  0.0032151,  0.0031979,
0.0033567,  0.0033648,  0.0029316,  0.0033834,  0.0032925,  0.0031714,  0.0031206,
0.0031617,  0.0031764,  0.0032008,  0.0022923,  0.0028703,  0.0028428,  0.0032108,
0.0030027,  0.0030918,  0.0030953,  0.0031119,  0.0030822,  0.0031886,  0.0032099,
0.0032292,  0.0032049,  0.0030995,  0.0030548,  0.0031173,  0.0029388,  0.0030904,
0.0030391,  0.0030800,  0.0030341,  0.0021157,  0.0031332,  0.0032913,  0.0029887,
0.0031505,  0.0031498,  0.0037496,  0.0032036,  0.0035727,  0.0031648,  0.0032885,
0.0032306,  0.0032066,  0.0031644,  0.0032604,  0.0033403,  0.0032178,  0.0032295,
0.0032889,  0.0032337,  0.0025542,  0.0029232,  0.0030008,  0.0036558,  0.0030375,
0.0027782,  0.0035353,  0.0030757,  0.0033911,  0.0030918,  0.0032649,  0.0034075,
0.0033375,  0.0035893,  0.0033013,  0.0033523,  0.0033755,  0.0035806,  0.0032820,
0.0030342,  0.0030873,  0.0024062,  0.0028201,  0.0030129,  0.0030678,  0.0030855,
0.0030384,  0.0031906,  0.0030652,  0.0032076,  0.0029029,  0.0033678,  0.0029931,
0.0033936,  0.0034343,  0.0032933,  0.0032346,  0.0033346,  0.0030427,  0.0031148,
0.0029991, 0.0024253, 0.0030285, 0.0031448, 0.0028891, 0.0029480
```

Результаты работы программы, использующей технологию OpenMP, представлена в листинге 4.

Листинг 4. Результаты работы программы (OpenMP).

```
0.0000166,  0.0000298,  0.0000675,  0.0001014,  0.0001171,  0.0001083,  0.0001226,
0.0001495,  0.0001528,  0.0001685,  0.0001739,  0.0001710,  0.0002006,  0.0002053,
0.0002465,  0.0002679,  0.0002543,  0.0002520,  0.0002070,  0.0002222,  0.0002387,
0.0002570,  0.0002451,  0.0002580,  0.0003164,  0.0002478,  0.0003017,  0.0006251,
0.0003008,  0.0003339,  0.0002885,  0.0003109,  0.0002540,  0.0003235,  0.0003341,
0.0003604,  0.0003556,  0.0003933,  0.0005361,  0.0004150,  0.0004038,  0.0003975,
0.0003724,  0.0004949,  0.0006327,  0.0005087,  0.0003613,  0.0004860,  0.0003640,
0.0004727,  0.0004570,  0.0004723,  0.0005630,  0.0005179,  0.0004523,  0.0005320,
0.0005852,  0.0006446,  0.0005051,  0.0005602,  0.0005618,  0.0005457,  0.0006038,
0.0006849,  0.0005761,  0.0005537,  0.0005383,  0.0006131,  0.0005641,  0.0005463,
0.0006002,  0.0009023,  0.0006700,  0.0008006,  0.0007336,  0.0008044,  0.0006810,
0.0007525,  0.0007110,  0.0007004,  0.0007063,  0.0006601,  0.0007124,  0.0008826,
0.0007774,  0.0008713,  0.0009327,  0.0007151,  0.0010481,  0.0008069,  0.0008374,
0.0009178,  0.0008399,  0.0011315,  0.0009317,  0.0009232,  0.0008912,  0.0009629,
0.0008010,  0.0008882,  0.0008773,  0.0009606,  0.0010814,  0.0009123,  0.0008848,
0.0006804,  0.0006916,  0.0006824,  0.0006657,  0.0006232,  0.0007340,  0.0008413,
0.0005739,  0.0006278,  0.0006174,  0.0006868,  0.0006774,  0.0008034,  0.0009724,
0.0008612,  0.0008420,  0.0009353,  0.0008657,  0.0008964,  0.0010133,  0.0008494,
0.0009766,  0.0013396,  0.0006558,  0.0006811,  0.0007770,  0.0007587,  0.0008014,
0.0013607,  0.0009342,  0.0009440,  0.0009110,  0.0009911,  0.0009523,  0.0009489,
0.0010732,  0.0009333,  0.0010367,  0.0007429,  0.0008212,  0.0008498,  0.0008090,
0.0008969,  0.0011231,  0.0010023,  0.0010309,  0.0012024,  0.0009426,  0.0009786,
0.0009657,  0.0010014,  0.0011486,  0.0011166,  0.0010955,  0.0008949,  0.0008278,
0.0009025,  0.0008161,  0.0011170,  0.0012525,  0.0012349,  0.0013087,  0.0013736,
0.0012558,  0.0012623,  0.0012467,  0.0013128,  0.0013503,  0.0012331,  0.0014043,
0.0010330,  0.0009638,  0.0010556,  0.0010449,  0.0014917,  0.0013961,  0.0012261,
0.0013908,  0.0012942,  0.0013907,  0.0013174,  0.0013284,  0.0014587,  0.0013614,
0.0014234,  0.0014915,  0.0011467,  0.0010604,  0.0010445,  0.0020757,  0.0014800,
0.0014007,  0.0016106,  0.0015235,  0.0014468,  0.0015426,  0.0014504,  0.0015527,
0.0014952,  0.0015628,  0.0016643,  0.0016078,  0.0010849,  0.0011844,  0.0012574,
0.0021606,  0.0015015,  0.0015046,  0.0015384,  0.0015037,  0.0014765,  0.0016073,
0.0017119,  0.0016009,  0.0015618,  0.0015400,  0.0016115,  0.0016044,  0.0012183,
0.0012404,  0.0012004,  0.0017047,  0.0018801,  0.0016399,  0.0016102,  0.0015633,
0.0017456,  0.0016960,  0.0016576,  0.0016697,  0.0017134,  0.0017013,  0.0018256,
0.0016840,  0.0012985,  0.0013561,  0.0012027,  0.0016500,  0.0018115,  0.0018432,
```

0.0017309, 0.0016812, 0.0016532, 0.0018529, 0.0019193, 0.0017105, 0.0018091,
0.0017694, 0.0018152, 0.0020501, 0.0011798, 0.0013265, 0.0017753, 0.0018609,
0.0018631, 0.0018672, 0.0018660, 0.0019625, 0.0020323, 0.0018704, 0.0020386,
0.0029884, 0.0018601, 0.0018491, 0.0020212, 0.0019442, 0.0014495, 0.0014494,
0.0018682, 0.0020191, 0.0019705, 0.0020449, 0.0019295, 0.0018914, 0.0019833,
0.0019813, 0.0021360, 0.0020572, 0.0020137, 0.0020927, 0.0019553, 0.0014961,
0.0016072, 0.0019964, 0.0020336, 0.0020998, 0.0021135, 0.0020287, 0.0019673,
0.0019458, 0.0020927, 0.0020346, 0.0021642, 0.0022308, 0.0019846, 0.0019744,
0.0022041, 0.0015109, 0.0016627, 0.0021485, 0.0022115, 0.0021605, 0.0020980,
0.0021698, 0.0021994, 0.0022008, 0.0020842, 0.0022231, 0.0022443, 0.0021844,
0.0023816, 0.0022378, 0.0041647, 0.0014985, 0.0015709, 0.0023452, 0.0022855,
0.0023487, 0.0024218, 0.0023677, 0.0023500, 0.0022977, 0.0024247, 0.0023464,
0.0022735, 0.0022062, 0.0024381, 0.0022387, 0.0023834, 0.0017002, 0.0019207,
0.0023474, 0.0023986, 0.0023577, 0.0023851, 0.0023554, 0.0024911, 0.0024227,
0.0022808, 0.0024509, 0.0023096, 0.0024607, 0.0026059, 0.0023395, 0.0024304,
0.0016322, 0.0018689, 0.0024496, 0.0024973, 0.0026089, 0.0025363, 0.0024892,
0.0024162, 0.0024805, 0.0024853, 0.0024608, 0.0040524, 0.0024956, 0.0023972,
0.0024456, 0.0024493, 0.0019218, 0.0025559, 0.0025719, 0.0026235, 0.0025993,
0.0025242, 0.0026257, 0.0025050, 0.0025907, 0.0026510, 0.0024844, 0.0026063,
0.0026875, 0.0026432, 0.0026635, 0.0029704, 0.0017996, 0.0042438, 0.0027189,
0.0026920, 0.0026555, 0.0027313, 0.0029596, 0.0029577, 0.0028435, 0.0026254,
0.0028492, 0.0029036, 0.0028148, 0.0026254, 0.0026530, 0.0026046, 0.0045025,
0.0029095, 0.0030883, 0.0028031, 0.0028569, 0.0031475, 0.0028399, 0.0027942,
0.0028298, 0.0029127, 0.0029866, 0.0027670, 0.0028378, 0.0030107, 0.0028416,
0.0019581, 0.0045054, 0.0029589, 0.0029886, 0.0028410, 0.0028790, 0.0031246,
0.0029837, 0.0028980, 0.0029520, 0.0029304, 0.0031434, 0.0030632, 0.0030085,
0.0029409, 0.0029908, 0.0021757, 0.0055450, 0.0031010, 0.0029884, 0.0031422,
0.0029648, 0.0031611, 0.0029162, 0.0030795, 0.0029794, 0.0029597, 0.0030866,
0.0031094, 0.0031070, 0.0030722, 0.0031531, 0.0021405, 0.0046918, 0.0031484,
0.0032863, 0.0032326, 0.0031932, 0.0031276, 0.0032115, 0.0032183, 0.0032398,
0.0031384, 0.0032754, 0.0031778, 0.0032404, 0.0032082, 0.0031039, 0.0022192,
0.0028729, 0.0031819, 0.0032726, 0.0063349, 0.0032707, 0.0032784, 0.0034041,
0.0031061, 0.0034117, 0.0033989, 0.0032844, 0.0033154, 0.0032859, 0.0032616,
0.0032645, 0.0022420, 0.0029634, 0.0032363, 0.0034960, 0.0033585, 0.0032458,
0.0033554, 0.0033494, 0.0034160, 0.0033858, 0.0034262, 0.0033444, 0.0034666,
0.0035884, 0.0036152, 0.0034493, 0.0025504, 0.0034736, 0.0033087, 0.0033990,
0.0034475, 0.0035245, 0.0033282, 0.0034317, 0.0033777, 0.0034548, 0.0035155,
0.0036336, 0.0033696, 0.0034682, 0.0036381, 0.0035389, 0.0024425, 0.0066182,
0.0035577, 0.0034990, 0.0035999, 0.0036490, 0.0035257, 0.0035482, 0.0035744,
0.0036067, 0.0035377, 0.0037324, 0.0034306, 0.0038386, 0.0037095, 0.0039306,
0.0025952, 0.0035289, 0.0036901, 0.0037028, 0.0036417, 0.0037861, 0.0036752,
0.0035612, 0.0037852, 0.0036595, 0.0036036, 0.0038268, 0.0035756, 0.0038085,
0.0036437, 0.0032936, 0.0037056, 0.0036579, 0.0036561, 0.0037379, 0.0037342,
0.0037251, 0.0037031, 0.0037095, 0.0038223, 0.0036801, 0.0038600, 0.0038317,
0.0039575, 0.0038584, 0.0040023, 0.0047475, 0.0027620, 0.0039135, 0.0037698,
0.0038471, 0.0039603, 0.0039164, 0.0038839, 0.0038920, 0.0039071, 0.0039172,
0.0039433, 0.0039077, 0.0039622, 0.0040106, 0.0073414, 0.0026877, 0.0038161,
0.0039644, 0.0039568, 0.0039966, 0.0039733, 0.0041612, 0.0041747, 0.0041007,
0.0039014, 0.0040898, 0.0040206, 0.0039508, 0.0038433, 0.0041412, 0.0040746,
0.0030448, 0.0041401, 0.0042820, 0.0040542, 0.0041485, 0.0041148, 0.0042459,
0.0055407, 0.0038732, 0.0040412, 0.0041498, 0.0039946, 0.0044019, 0.0042622,
0.0041734, 0.0040519, 0.0029017, 0.0043326, 0.0042129, 0.0041868, 0.0043064,
0.0042485, 0.0043327, 0.0041389, 0.0043065, 0.0043125, 0.0041603, 0.0042333,
0.0042104, 0.0073132, 0.0041997, 0.0043351, 0.0028388, 0.0045947, 0.0042064,
0.0041183, 0.0043751, 0.0043264, 0.0041804, 0.0042331, 0.0043071, 0.0043060,
0.0043716, 0.0041276, 0.0042975, 0.0041966, 0.0044304, 0.0042841, 0.0030751,
0.0042979, 0.0044408, 0.0044394, 0.0044231, 0.0043794, 0.0044008, 0.0044213,
0.0044083, 0.0045162, 0.0045890, 0.0044324, 0.0044202, 0.0045443, 0.0045144,
0.0045206, 0.0073857, 0.0043464, 0.0044877, 0.0042869, 0.0045109, 0.0044864,
0.0046553, 0.0060472, 0.0044762, 0.0046769, 0.0043755, 0.0047220, 0.0044981,
0.0049104, 0.0046568, 0.0045617, 0.0030517, 0.0045040, 0.0044880, 0.0046736,
0.0048070, 0.0046311, 0.0045136, 0.0046817, 0.0046498, 0.0044940, 0.0046009,
0.0073211, 0.0046997, 0.0047016, 0.0046869, 0.0049855, 0.0043678, 0.0046235,

```
0.0046500, 0.0046278, 0.0047253, 0.0048412, 0.0047302, 0.0048199, 0.0046494,
0.0047704, 0.0047389, 0.0047744, 0.0050122, 0.0048218, 0.0088206, 0.0034622,
0.0052002, 0.0049503, 0.0046692, 0.0048545, 0.0049563, 0.0047590, 0.0048697,
0.0047509, 0.0049182, 0.0045517, 0.0050752, 0.0049474, 0.0047679, 0.0049842,
0.0048855, 0.0039202, 0.0046445, 0.0050444, 0.0049082, 0.0048493, 0.0049091,
0.0048935, 0.0049737, 0.0048629, 0.0050421, 0.0049082, 0.0050699, 0.0051861,
0.0051283, 0.0052837, 0.0053008, 0.0049044, 0.0049104, 0.0049642, 0.0054179,
0.0049540, 0.0049259, 0.0050284, 0.0050213, 0.0050187, 0.0051220, 0.0050176,
0.0051180, 0.0049808, 0.0051664, 0.0053615, 0.0054473, 0.0082099, 0.0053965,
0.0050071, 0.0050270, 0.0051320, 0.0051193, 0.0068070, 0.0051216, 0.0050774,
0.0051271, 0.0051719, 0.0052771, 0.0052478, 0.0053366, 0.0053174, 0.0071213,
0.0053291, 0.0051564, 0.0051482, 0.0053927, 0.0053038, 0.0051405, 0.0055327,
0.0074650, 0.0053173, 0.0052177, 0.0052747, 0.0051966, 0.0053705, 0.0053405,
0.0050339, 0.0064818, 0.0056175, 0.0053886, 0.0054042, 0.0053942, 0.0054031,
0.0052421, 0.0054050, 0.0054184, 0.0078128, 0.0054778, 0.0052772, 0.0054003,
0.0054962, 0.0054186, 0.0054562, 0.0045128, 0.0053689, 0.0053011, 0.0053790,
0.0051777, 0.0053955, 0.0055779, 0.0052305, 0.0053670, 0.0055331, 0.0053817,
0.0054148, 0.0055907, 0.0056208, 0.0055445, 0.0056342, 0.0062004, 0.0053963,
0.0055299, 0.0054571, 0.0055057, 0.0057081, 0.0055499, 0.0054906, 0.0054435,
0.0057403, 0.0055344, 0.0055784, 0.0057057, 0.0055892, 0.0055343, 0.0075799,
0.0054726, 0.0055017, 0.0055676, 0.0056805, 0.0055838, 0.0056876, 0.0056594,
0.0056518, 0.0055600, 0.0087540, 0.0055951, 0.0055501, 0.0057782, 0.0058254,
0.0056350, 0.0048377, 0.0056999, 0.0057864, 0.0057229, 0.0057569, 0.0055560,
0.0057597, 0.0057702, 0.0056018, 0.0057042, 0.0057268, 0.0058500, 0.0058183,
0.0058401, 0.0060197, 0.0057871, 0.0064779, 0.0058504, 0.0058511, 0.0058723,
0.0057082, 0.0057505, 0.0058289, 0.0059507, 0.0059889, 0.0087579, 0.0060597,
0.0059920, 0.0058353, 0.0058035, 0.0058243, 0.0050501, 0.0056307, 0.0060902,
0.0060318, 0.0059064, 0.0059141, 0.0058118, 0.0060397, 0.0058088, 0.0105880,
0.0060374, 0.0059794, 0.0061132, 0.0060982, 0.0062869, 0.0062552, 0.0073021,
0.0060852, 0.0059517, 0.0061470, 0.0059672, 0.0060273, 0.0060710, 0.0060580,
0.0060786, 0.0060086, 0.0060055, 0.0061688, 0.0059633, 0.0060035, 0.0058729,
0.0076930, 0.0060533, 0.0060367, 0.0061336, 0.0060997, 0.0061100, 0.0061236,
0.0083001, 0.0063875, 0.0061009, 0.0060731, 0.0062790, 0.0061892, 0.0061031,
0.0060967, 0.0062965, 0.0085982, 0.0060509, 0.0060932, 0.0060999, 0.0061525,
0.0064999, 0.0062623, 0.0066569, 0.0063106, 0.0064042, 0.0063401, 0.0060032,
0.0063658, 0.0063895, 0.0062319, 0.0084371, 0.0061850, 0.0061900, 0.0062934,
0.0062986, 0.0062513, 0.0062513, 0.0062573, 0.0063245, 0.0062379, 0.0064694,
0.0063288, 0.0063701, 0.0064365, 0.0065358, 0.0066416, 0.0062067, 0.0063603,
0.0062404, 0.0063011, 0.0063390, 0.0066107, 0.0063601, 0.0066657, 0.0065400,
0.0066338, 0.0066798, 0.0069264, 0.0068272, 0.0067890, 0.0101948, 0.0063502,
0.0063143, 0.0063597, 0.0062727, 0.0064254, 0.0064068, 0.0063822, 0.0066096,
0.0064711, 0.0064059, 0.0061251, 0.0064646, 0.0108370, 0.0068151, 0.0056250,
0.0074436, 0.0065368, 0.0063315, 0.0065643, 0.0066542, 0.0065217, 0.0064794,
0.0065649, 0.0065321, 0.0068283, 0.0067914, 0.0106402, 0.0068501, 0.0069757,
0.0056529, 0.0065496, 0.0064846, 0.0065557, 0.0066167, 0.0065383
```

График зависимости времени обмена от размера сообщения.

Графики зависимости времени обмена от размера сообщения для технологий MPI и OpenMP представлен на рис. 2.

Как видно из графика, время обмена сообщениями для обеих технологий линейно зависит от размера отправляемых сообщений. Чем больше размер сообщения, тем больше времени затрачивается на его передачу. Помимо этого, MPI затрачивает меньше времени на обмен сообщениями, чем OpenMP.

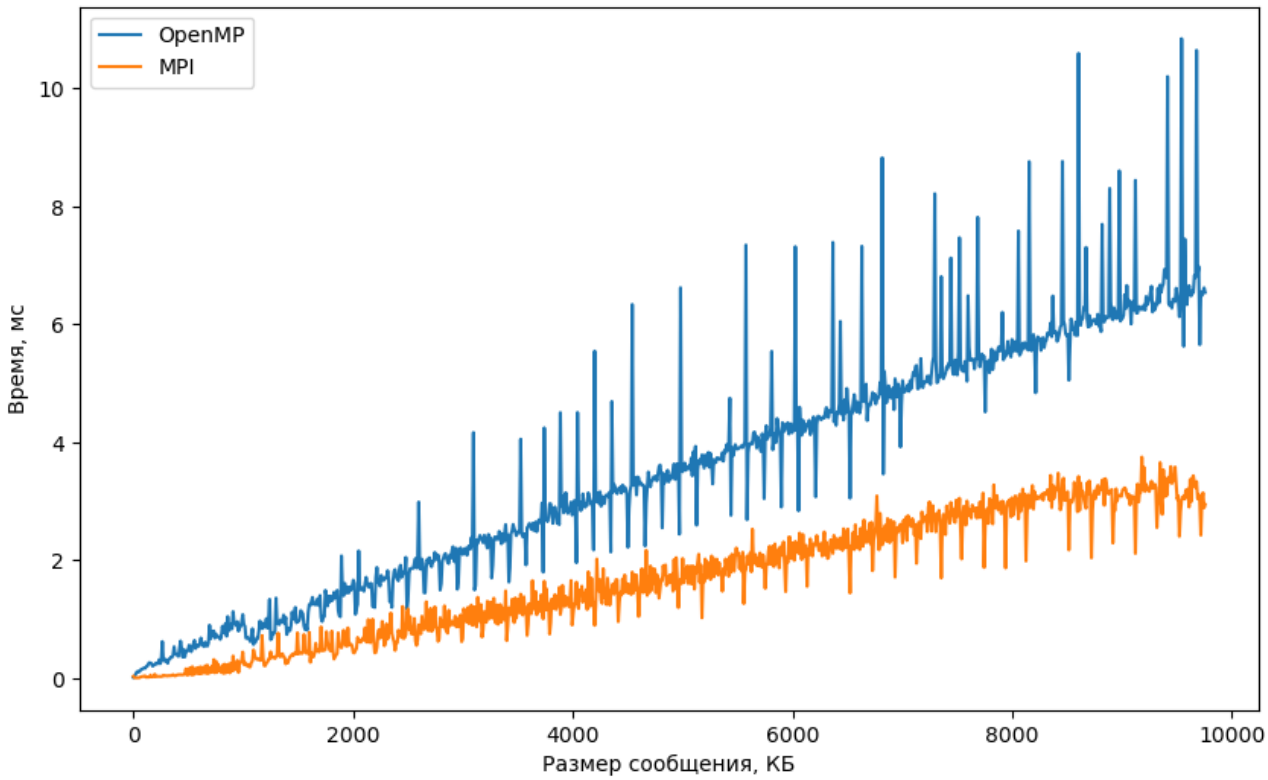

Рисунок 2 – Графики зависимости времени обмена от размера сообщения (6 процессов/потоков).

Замеры проводились на ПК со следующими характеристиками:

- Процессор AMD Ryzen 5 5600X (6 физических и 12 логических ядер);
- Оперативная память 32 ГБ 3200 МГц;
- Видеокарта GeForce RTX 3060 Ti.

Выводы.

Была написана программа, осуществляющая обмен сообщениями между четными и нечетными процессами. Было выполнено измерение времени обмена сообщениями при различных размерах сообщения.

Исходя из полученных графиков видно, время обмена сообщениями для обеих технологий линейно зависит от размера отправляемых сообщений. Чем больше размер сообщения, тем больше времени затрачивается на его передачу. Помимо этого, MPI затрачивает меньше времени на обмен сообщениями, чем OpenMP. Это может объясняться тем, что механизм обмена сообщениями в MPI

более оптимизирован, чем самостоятельно реализованный механизм в OpenMP. OpenMP не включает в себя готовый механизм обмена сообщениями между потоками.