МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет информационных технологий

Кафедра параллельных вычислений

ОТЧЕТ

О ВЫПОЛНЕНИИ ПРАКТИЧЕСКОЙ РАБОТЫ

ПРОГРАММИРОВАНИЕ МНОГОПОТОЧНЫХ ПРИЛОЖЕНИЙ. POSIX THREADS.

Студентки 2 курса, группы 21205

Евдокимовой Дари Евгеньевны

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:
Кандидат технических наук, доцент
А.Ю. Власенко

Новосибирск 2023

СОДЕРЖАНИЕ

1.ЦЕЛЬ

1 Освоить разработку многопоточных программ с использованием POSIX Threads API. Познакомиться с задачей динамического распределения работы между процессорами.

2.ЗАДАНИЕ

1 Реализовать программу, выполняющую следующие требования:

Есть список неделимых заданий, каждое из которых может быть выполнено независимо от другого. Задания могут иметь различный вычислительный вес, т.е. требовать при одних и тех же вычислительных ресурсах различного времени для выполнения. Считается, что этот вес нельзя узнать, пока задание не выполнено. После того, как все задания из списка выполнены, появляется новый список заданий. Необходимо организовать параллельную обработку заданий на нескольких компьютерах. Количество заданий существенно превосходит количество процессоров. Программа не должна зависеть от числа компьютеров.

2 Для распараллеливания задачи задания из списка нужно распределять между компьютерами. Так как задания имеют различный вычислительный вес, а список обрабатывается итеративно, и требуется синхронизация перед каждой итерацией, то могут возникать ситуации, когда некоторые процессоры выполнили свою работу, а другие - еще нет. Если ничего не предпринять, первые будут простаивать в ожидании последних.

3 Так возникает задача динамического распределения работы. Для ее решения на каждом процессоре заведем несколько потоков. Как минимум, потоков должно быть 2:

а) поток, который обрабатывает задания и, когда задания закончились, обращается к другим компьютерам за добавкой к работе,

б) поток, ожидающий запросов о работе от других компьютеров

4 Сложность задачи заключается в

а) разработке правильной политики взаимодействия между процессами, когда все посылки (send) запросов и данных и ожидания (receive) приема запросов и данных будут согласованы.

б) организации корректной работы многих потоков с общими структурами данных. Необходимо обеспечивать взаимное исключение потоков при добавлении заданий в список, удалении задач, выборке заданий для выполнения.

5 Количество поочередно обрабатываемых списков сделать таким, чтобы программа выполнялась не менее 30 сек. и списков должно быть не менее 3.

6 После каждой итерации iterCounter (после каждого списка задач) каждый MPI-процесс должен выводить:

- кол-во заданий, выполненных данным процессом за итерацию;
- значение globalRes
- общее время выполнения заданий на этой итерации
- время дисбаланса и долю дисбаланса.

7 Произвести профилирование программы.

8 Составить отчет, содержащий исходные коды разработанных программ и профилирование.

3.ОПИСАНИЕ РАБОТЫ

1. Был реализован алгоритм балансировки: сначала каждый поток выполняет свои задачи. Затем, после завершения, рассылает всем остальным потокам сообщение о том, что поток завершил свою работу и может кому-то помочь. Поток, у которого количество заданий меньше какого-то заданного числа, не сигнализирует о том, что ему нужна помощь. Иначе такой поток отправляет помощнику половину оставшихся заданий.
2. Была написана программа балансировки — см. в Приложение 1.
   Команда для компиляции:
   mpicxx -mt_mpi load_balancer.cpp -o load_balancer
   Команда для запуска:
   mpirun -n ./ load_balancer
3. Минимальное количество заданий у потока, которому нужна помощь = 20.
4. Были произведены замеры времени работы программы — см. Приложение 2.
5. Сделано профилирование на 12 процессах — см. Приложение 3.
   Команды для профилирования:
   mpicxx -mt_mpi load_balancer.cpp -o load_balancer
   mpirun -trace -n 12 ./load_balancer
   traceanalyzer load_balancer.stf

ЗАКЛЮЧЕНИЕ

В ходе работы мы познакомились с таким инструментом программирования многопоточных приложений как Posix Threads.

Нам удалось написать балансировщик нагрузки заданий за счет того, что в каждом процессе стало 2 потока: 1й (main thread) поток отвечает за исполнение задач из своего списка и, в случае выполнения задач своего списка, дополнительного выполнения части задач другого списка; 2й поток отвечает за принятие запроса на выполнение задач другим потоком с последующим отправлением части своих задач из текущего списка.

При анализе профилирования и результатов вычислений можно сделать вывод о том, что при увеличении числа списков самые легкие задачи получал первый процесс, а последний — самые сложные . Затем они менялись, потому что (по формуле заполнения веса задачи) процесс с более легкими задачами завершал исполнение своих задачи быстрее остальных, а затем — исполнял части задачи других процессов.

СПИСОК ЛИТЕРАТУРЫ

1. Онлайн учебник по Posix Threads [Электронный ресурс]. URL: https://hpc-tutorials.llnl.gov/posix/
2. Видео-лекция по Posix Threads [Электронный ресурс]. URL: https://www.youtube.com/watch?v=TCfM5deMD4Y&t=1487s

Приложение 1. Листинг параллельной программы

```
#include <iostream>
#include <math.h>
#include <mpi.h>
#include <pthread.h>

#define AMOUNT_OF_LISTS 5
#define WEIGHT_COEFFICIENT 5000

#define MIN_AMOUNT_OF_TASKS_TO_SHARE 20
#define TASKS_PER_PROCESS 2400

#define TAG_REQUEST 0
#define TAG_REPLY 1

double RES_PER_ITERATION = 0;
double GLOABAL_RESULT_SIN = 0;

int rankOfCurrProc, amountOfProcs;

int *tasks;
int tasksInRemain;
int amountOfTasksAlreadyExecuted;

pthread_mutex_t mutexTasks;
pthread_mutex_t mutexTasksInRemain;

pthread_t recvThread;

void initTasksWeight() {
  pthread_mutex_lock(&mutexTasks);
  for (int i = 0; i < TASKS_PER_PROCESS; ++i) {
    tasks[i] =
        abs(50 - i % 100) *
        abs(rankOfCurrProc - (TASKS_PER_PROCESS % amountOfProcs)) *
        WEIGHT_COEFFICIENT;
  }
  pthread_mutex_unlock(&mutexTasks);
}

void calculateTask() {
  pthread_mutex_lock(&mutexTasksInRemain);

  for (int i = 0; tasksInRemain; ++i, tasksInRemain--) {
    pthread_mutex_unlock(&mutexTasksInRemain);

    // когда отдается часть заданий, функция не должна выоплнять часть
    // заздания которая возможна будет прередана
    pthread_mutex_lock(&mutexTasks);
```

```
    int task_weight = tasks[i];
    pthread_mutex_unlock(&mutexTasks);

    for (int j = 0; j < task_weight; ++j) {
      RES_PER_ITERATION += sin(j);
    }

    ++amountOfTasksAlreadyExecuted;

    pthread_mutex_lock(&mutexTasksInRemain);
  }
  pthread_mutex_unlock(&mutexTasksInRemain);
}

void *receiverThreadGo(void *args) {
  int tasksToSend;
  int rankRequestedTasks;

  while (true) {
    // receiving process rank that requests tasks
    MPI_Recv(&rankRequestedTasks, 1, MPI_INT, MPI_ANY_SOURCE,
            TAG_REQUEST, MPI_COMM_WORLD, MPI_STATUS_IGNORE);

    if (rankRequestedTasks == rankOfCurrProc)
      break;

    pthread_mutex_lock(&mutexTasksInRemain);
    if (tasksInRemain >= MIN_AMOUNT_OF_TASKS_TO_SHARE) {
      tasksToSend = tasksInRemain / 2;
      tasksInRemain -= tasksToSend;

      // отправляем только КОЛИЧЕСТВО такок
      MPI_Send(&tasksToSend, 1, MPI_INT, rankRequestedTasks,
              TAG_REPLY, MPI_COMM_WORLD);

      pthread_mutex_lock(&mutexTasks);

      // отправялем сами таски
      MPI_Send(tasks + amountOfTasksAlreadyExecuted + tasksInRemain -
                  1,
              tasksToSend, MPI_INT, rankRequestedTasks, TAG_REPLY,
              MPI_COMM_WORLD);
      pthread_mutex_unlock(&mutexTasksInRemain);

      pthread_mutex_unlock(&mutexTasks);
    } else {
      tasksToSend = 0;

      MPI_Send(&tasksToSend, 1, MPI_INT, rankRequestedTasks,
              TAG_REPLY, MPI_COMM_WORLD);
    }
```

```
  }
  return NULL;
}

void *workerThreadGo(void *args) {
  tasks = new int[TASKS_PER_PROCESS];

  double startt;
  double minTime, maxTime;

  for (int iterCounter = 0; iterCounter < AMOUNT_OF_LISTS;
      ++iterCounter) {
    initTasksWeight();

    pthread_mutex_lock(&mutexTasksInRemain);
    tasksInRemain = TASKS_PER_PROCESS;
    pthread_mutex_unlock(&mutexTasksInRemain);
    amountOfTasksAlreadyExecuted = 0;
    int amountOfAdditionalasks;

    startt = MPI_Wtime();

    /*
      процесс сначала считает свои таски, и когда закончил, рассылает
      остальным сообщение о том, что свободен и может посчитать часть
      тасок (т.е. это уже дополнительные таски) с других процессов
    */

    calculateTask();

    // запрашиваем таски с других процессов
    for (int currentProc = 0; currentProc < amountOfProcs;
        ++currentProc) {
      if (currentProc == rankOfCurrProc)
        continue;

      // процесс, который закончил свои вычисления сигнализирует
      // сообщением другим процессам, что он свободен и может принять
      // такси с других процессов на исполнение
      MPI_Send(&rankOfCurrProc, 1, MPI_INT, currentProc, TAG_REQUEST,
          MPI_COMM_WORLD);

      // получаем количество(!) ДОПОЛНИТЕЛЬНЫХ тасок
      MPI_Recv(&amountOfAdditionalasks, 1, MPI_INT, currentProc,
          TAG_REPLY, MPI_COMM_WORLD, MPI_STATUS_IGNORE);

      // если есть эти дополнительные таски, то принимаем их и
      // начинаем их исполнять
      if (amountOfAdditionalasks > 0) {
        // принимаем сами ТАСКИ
        MPI_Recv(tasks, amountOfAdditionalasks, MPI_INT, currentProc,
```

```
            TAG_REPLY, MPI_COMM_WORLD, MPI_STATUS_IGNORE);

    pthread_mutex_lock(&mutexTasksInRemain);
    tasksInRemain = amountOfAdditionalasks;
    pthread_mutex_unlock(&mutexTasksInRemain);

    // исполняем их
    calculateTask();
   }
  }

  double endt = MPI_Wtime();
  double resTime = endt - startt;

  MPI_Allreduce(&resTime, &minTime, 1, MPI_DOUBLE, MPI_MIN,
            MPI_COMM_WORLD);

  MPI_Allreduce(&resTime, &maxTime, 1, MPI_DOUBLE, MPI_MAX,
            MPI_COMM_WORLD);

  if (rankOfCurrProc == 0) {
   std::cout << "================================================="
          << std::endl;

   std::cout << "Iteration numer: " << iterCounter << std::endl;
   std::cout << "Disbalance time: " << maxTime - minTime
          << std::endl;
   std::cout << "Disbalance percentage: "
          << (maxTime - minTime) / maxTime * 100 << std::endl;
   std::cout << "-------------------------------------------"
          << std::endl;
  }

  for (int currentProc = 0; currentProc < amountOfProcs;
      currentProc++) {
   if (rankOfCurrProc == currentProc) {
    std::cout << "\t\tCurrent proc is: " << rankOfCurrProc
           << std::endl;
    std::cout << "Amount of executed tasks: "
           << amountOfTasksAlreadyExecuted << std::endl;
    std::cout << "Result of calculating is: " << RES_PER_ITERATION
           << std::endl;
    std::cout << "Time per iteration: " << resTime << " seconds"
           << std::endl;
   }
   MPI_Barrier(MPI_COMM_WORLD);
  }
 }

// recv назходится в режиме ожидания сообшения о том, все процесс к
// которому он происодеинен закончил работу (говорим рисиверу)
```

```
  MPI_Send(&rankOfCurrProc, 1, MPI_INT, rankOfCurrProc, 0,
         MPI_COMM_WORLD);

  MPI_Allreduce(&RES_PER_ITERATION, &GLOABAL_RESULT_SIN, 1,
             MPI_DOUBLE, MPI_SUM, MPI_COMM_WORLD);

  delete tasks;

  return NULL;
}

void createAndGoThreads() {
  pthread_mutex_init(&mutexTasks, NULL);
  pthread_mutex_init(&mutexTasksInRemain, NULL);

  pthread_attr_t attributes;
  if (pthread_attr_init(&attributes) != 0) {
    MPI_Finalize();
    perror("Can't init attributes");
    abort();
  }

  if (pthread_attr_setdetachstate(&attributes,
                            PTHREAD_CREATE_JOINABLE) != 0) {
    MPI_Finalize();
    perror("Error in setting attributes");
    abort();
  }

  if (pthread_create(&recvThread, &attributes, receiverThreadGo,
                  NULL) != 0) {
    MPI_Finalize();
    perror("Can't create thread");
    abort();
  }

  pthread_attr_destroy(&attributes);

  workerThreadGo(NULL); // it's main thread

  // main thread is waitng for recieiver thread to finish
  pthread_join(recvThread, NULL);

  pthread_mutex_destroy(&mutexTasks);
  pthread_mutex_destroy(&mutexTasksInRemain);
}

int main(int argc, char **argv) {
  int reqiredLevel =
     MPI_THREAD_MULTIPLE; // we want this level of supporting threads
  int providedLevel;       // real level of supporting threads
```

```
  MPI_Init_thread(&argc, &argv, reqiredLevel, &providedLevel);
  if (providedLevel != reqiredLevel) {
    MPI_Finalize();
    perror("Can't load reqired level");
    return 0;
  }

  MPI_Comm_size(MPI_COMM_WORLD, &amountOfProcs);
  MPI_Comm_rank(MPI_COMM_WORLD, &rankOfCurrProc);

  double startt = MPI_Wtime();
  createAndGoThreads();
  double endt = MPI_Wtime();

  double resTime = endt - startt;

  if (rankOfCurrProc == 0) {
    std::cout << "========================================="
              << std::endl;
    std::cout << "Time for all lists: " << resTime << "seconds"
              << std::endl;
  }

  MPI_Finalize();
  return 0;
}
```

Приложение 2. Результаты замеров выполнения работы

Замеры программы на 2х процессах:

```
==================================================
Iteration numer: 0
Disbalance time: 0.00374473
Disbalance percentage: 0.0564269
------------------------------------------
        Current proc is: 0
Amount of executed tasks: 3600
Result of calculating is: 1093.15
Time per iteration: 6.63642 seconds
        Current proc is: 1
Amount of executed tasks: 1200
Result of calculating is: 1093.15
Time per iteration: 6.63267 seconds
==================================================
Iteration numer: 1
Disbalance time: 0.39021
Disbalance percentage: 5.59894
------------------------------------------
        Current proc is: 0
Amount of executed tasks: 3600
Result of calculating is: 2186.29
Time per iteration: 6.57914 seconds
        Current proc is: 1
Amount of executed tasks: 1200
Result of calculating is: 2186.29
Time per iteration: 6.96935 seconds
==================================================
Iteration numer: 2
Disbalance time: 0.0400798
Disbalance percentage: 0.571657
------------------------------------------
        Current proc is: 0
Amount of executed tasks: 3600
Result of calculating is: 3279.44
Time per iteration: 7.01115 seconds
        Current proc is: 1
Amount of executed tasks: 1200
Result of calculating is: 3279.44
Time per iteration: 6.97107 seconds
==================================================
Iteration numer: 3
Disbalance time: 0.464463
Disbalance percentage: 7.14123
------------------------------------------
        Current proc is: 0
Amount of executed tasks: 3600
```

Result of calculating is: 4372.58
Time per iteration: 6.0395 seconds
Current proc is: 1
Amount of executed tasks: 1200
Result of calculating is: 4372.58
Time per iteration: 6.50397 seconds
==========
Iteration numer: 4
Disbalance time: 1.34304
Disbalance percentage: 19.1715

Current proc is: 0
Amount of executed tasks: 3600
Result of calculating is: 5465.73
Time per iteration: 5.66237 seconds
Current proc is: 1
Amount of executed tasks: 1200
Result of calculating is: 5465.73
Time per iteration: 7.00541 seconds
==========
Time for all lists: 34.129seconds

Замеры программы на 4х процессах:
==========
Iteration numer: 0
Disbalance time: 1.37784
Disbalance percentage: 9.88175

Current proc is: 0
Amount of executed tasks: 4262
Result of calculating is: 1697.18
Time per iteration: 12.5654 seconds
Current proc is: 1
Amount of executed tasks: 2434
Result of calculating is: 1809.11
Time per iteration: 13.9433 seconds
Current proc is: 2
Amount of executed tasks: 1805
Result of calculating is: 1268.71
Time per iteration: 13.8929 seconds
Current proc is: 3
Amount of executed tasks: 1099
Result of calculating is: 909.279
Time per iteration: 13.3444 seconds
==========
Iteration numer: 1
Disbalance time: 1.3943
Disbalance percentage: 10.1267

Current proc is: 0
Amount of executed tasks: 4262
Result of calculating is: 3394.35

Time per iteration: 12.3743 seconds
Current proc is: 1
Amount of executed tasks: 2434
Result of calculating is: 3618.23
Time per iteration: 13.7686 seconds
Current proc is: 2
Amount of executed tasks: 1805
Result of calculating is: 2537.42
Time per iteration: 13.7135 seconds
Current proc is: 3
Amount of executed tasks: 1099
Result of calculating is: 1818.56
Time per iteration: 13.1717 seconds
==========
Iteration numer: 2
Disbalance time: 1.37989
Disbalance percentage: 10.0349

Current proc is: 0
Amount of executed tasks: 4262
Result of calculating is: 5091.53
Time per iteration: 12.3711 seconds
Current proc is: 1
Amount of executed tasks: 2433
Result of calculating is: 5426.15
Time per iteration: 13.751 seconds
Current proc is: 2
Amount of executed tasks: 1805
Result of calculating is: 3806.13
Time per iteration: 13.719 seconds
Current proc is: 3
Amount of executed tasks: 1100
Result of calculating is: 2727.84
Time per iteration: 13.1662 seconds
==========
Iteration numer: 3
Disbalance time: 1.38471
Disbalance percentage: 10.0689

Current proc is: 0
Amount of executed tasks: 4262
Result of calculating is: 6788.71
Time per iteration: 12.3676 seconds
Current proc is: 1
Amount of executed tasks: 2433
Result of calculating is: 7234.07
Time per iteration: 13.7523 seconds
Current proc is: 2
Amount of executed tasks: 1805
Result of calculating is: 5074.84
Time per iteration: 13.7185 seconds
Current proc is: 3

Amount of executed tasks: 1100
Result of calculating is: 3637.12
Time per iteration: 13.2574 seconds
==========
Iteration numer: 4
Disbalance time: 2.00041
Disbalance percentage: 12.398

Current proc is: 0
Amount of executed tasks: 4284
Result of calculating is: 8506.01
Time per iteration: 14.1345 seconds
Current proc is: 1
Amount of executed tasks: 2444
Result of calculating is: 9052.46
Time per iteration: 16.135 seconds
Current proc is: 2
Amount of executed tasks: 1845
Result of calculating is: 6222.89
Time per iteration: 14.3033 seconds
Current proc is: 3
Amount of executed tasks: 1027
Result of calculating is: 4476.64
Time per iteration: 14.437 seconds
==========
Time for all lists: 71.358seconds

Замеры программы на 8и процессах

==========
Iteration numer: 0
Disbalance time: 8.03557
Disbalance percentage: 12.3998

Current proc is: 0
Amount of executed tasks: 4617
Result of calculating is: 2020.38
Time per iteration: 60.706 seconds
Current proc is: 1
Amount of executed tasks: 2793
Result of calculating is: 2451.71
Time per iteration: 58.6466 seconds
Current proc is: 2
Amount of executed tasks: 2050
Result of calculating is: 1848.39
Time per iteration: 58.139 seconds
Current proc is: 3
Amount of executed tasks: 2725
Result of calculating is: 2271.18
Time per iteration: 59.1675 seconds
Current proc is: 4
Amount of executed tasks: 1879

Result of calculating is: 1046.64
Time per iteration: 56.7682 seconds
Current proc is: 5
Amount of executed tasks: 1910
Result of calculating is: 1240.42
Time per iteration: 64.8038 seconds
Current proc is: 6
Amount of executed tasks: 1698
Result of calculating is: 1314.36
Time per iteration: 62.7397 seconds
Current proc is: 7
Amount of executed tasks: 1528
Result of calculating is: 1329
Time per iteration: 62.9998 seconds
==========
Iteration numer: 1
Disbalance time: 5.77234
Disbalance percentage: 8.57288

Current proc is: 0
Amount of executed tasks: 4065
Result of calculating is: 3545.78
Time per iteration: 64.6019 seconds
Current proc is: 1
Amount of executed tasks: 3052
Result of calculating is: 5190.18
Time per iteration: 63.2161 seconds
Current proc is: 2
Amount of executed tasks: 2724
Result of calculating is: 4171.08
Time per iteration: 62.0223 seconds
Current proc is: 3
Amount of executed tasks: 2764
Result of calculating is: 4338.38
Time per iteration: 67.3325 seconds
Current proc is: 4
Amount of executed tasks: 1851
Result of calculating is: 2110.06
Time per iteration: 64.4165 seconds
Current proc is: 5
Amount of executed tasks: 1804
Result of calculating is: 2332.28
Time per iteration: 62.7489 seconds
Current proc is: 6
Amount of executed tasks: 1407
Result of calculating is: 2105.74
Time per iteration: 62.2321 seconds
Current proc is: 7
Amount of executed tasks: 1533
Result of calculating is: 2590.46
Time per iteration: 61.5602 seconds
==========

Iteration numer: 2
Disbalance time: 11.0799
Disbalance percentage: 18.7182

Current proc is: 0
Amount of executed tasks: 3876
Result of calculating is: 4892.02
Time per iteration: 52.752 seconds
Current proc is: 1
Amount of executed tasks: 2902
Result of calculating is: 7784.63
Time per iteration: 48.1137 seconds
Current proc is: 2
Amount of executed tasks: 3042
Result of calculating is: 6589.25
Time per iteration: 52.4781 seconds
Current proc is: 3
Amount of executed tasks: 2771
Result of calculating is: 6689.98
Time per iteration: 59.1936 seconds
Current proc is: 4
Amount of executed tasks: 1831
Result of calculating is: 3261.87
Time per iteration: 49.4711 seconds
Current proc is: 5
Amount of executed tasks: 1416
Result of calculating is: 3541.15
Time per iteration: 49.5041 seconds
Current proc is: 6
Amount of executed tasks: 1799
Result of calculating is: 3190.8
Time per iteration: 53.4589 seconds
Current proc is: 7
Amount of executed tasks: 1563
Result of calculating is: 3565.23
Time per iteration: 50.9743 seconds
==========
Iteration numer: 3
Disbalance time: 14.8619
Disbalance percentage: 32.8366

Current proc is: 0
Amount of executed tasks: 4957
Result of calculating is: 7223.92
Time per iteration: 32.8195 seconds
Current proc is: 1
Amount of executed tasks: 2737
Result of calculating is: 10284.9
Time per iteration: 34.4466 seconds
Current proc is: 2
Amount of executed tasks: 2836
Result of calculating is: 8818.4

Time per iteration: 45.2601 seconds
Current proc is: 3
Amount of executed tasks: 2326
Result of calculating is: 8365.57
Time per iteration: 30.3982 seconds
Current proc is: 4
Amount of executed tasks: 1897
Result of calculating is: 4530.51
Time per iteration: 34.2258 seconds
Current proc is: 5
Amount of executed tasks: 1511
Result of calculating is: 4728.7
Time per iteration: 32.624 seconds
Current proc is: 6
Amount of executed tasks: 1624
Result of calculating is: 4377.03
Time per iteration: 32.967 seconds
Current proc is: 7
Amount of executed tasks: 1312
Result of calculating is: 4350.04
Time per iteration: 32.8387 seconds
==========
Iteration numer: 4
Disbalance time: 4.5282
Disbalance percentage: 13.0535

Current proc is: 0
Amount of executed tasks: 4903
Result of calculating is: 9506.47
Time per iteration: 31.6986 seconds
Current proc is: 1
Amount of executed tasks: 2784
Result of calculating is: 12751.1
Time per iteration: 32.1398 seconds
Current proc is: 2
Amount of executed tasks: 2530
Result of calculating is: 11095.5
Time per iteration: 34.6896 seconds
Current proc is: 3
Amount of executed tasks: 2268
Result of calculating is: 9799.66
Time per iteration: 30.1614 seconds
Current proc is: 4
Amount of executed tasks: 1956
Result of calculating is: 5793.47
Time per iteration: 32.0775 seconds
Current proc is: 5
Amount of executed tasks: 1902
Result of calculating is: 6002.8
Time per iteration: 31.8462 seconds
Current proc is: 6
Amount of executed tasks: 1649

Result of calculating is: 5346.96
Time per iteration: 31.7106 seconds
Current proc is: 7
Amount of executed tasks: 1208
Result of calculating is: 5261.57
Time per iteration: 31.9846 seconds
==========
Time for all lists: 272.599seconds

Приложение 3. Скрины из traceanalyzer

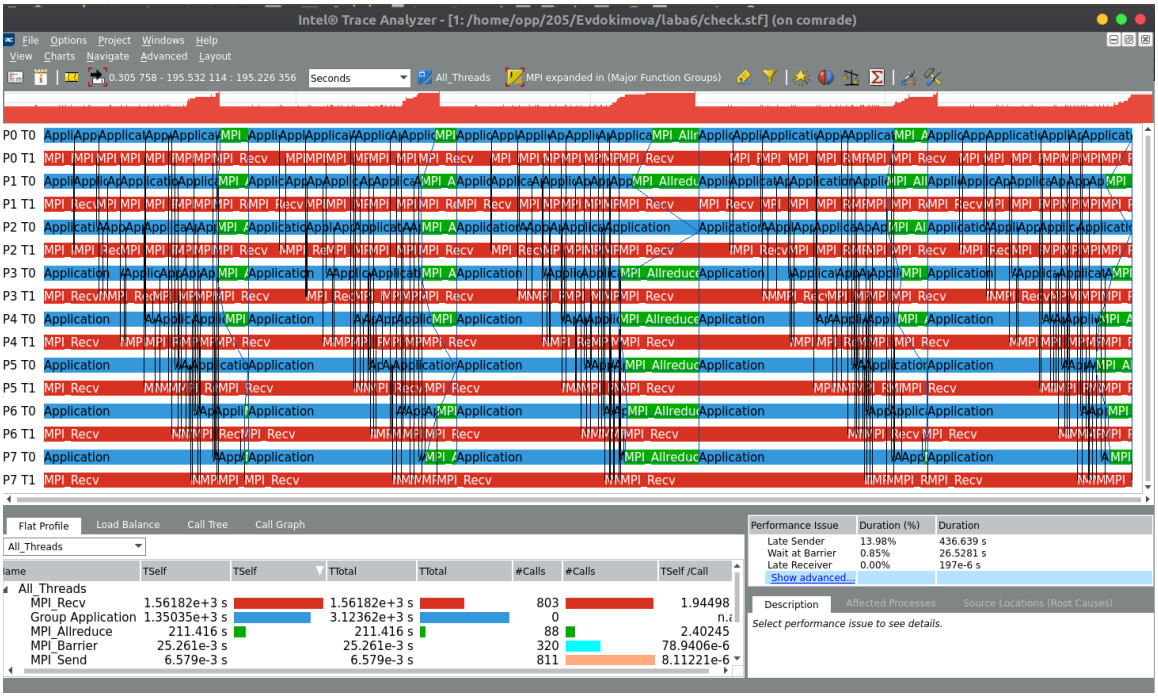

Рис. 1. Общий работы всей программы и общее время работы функций

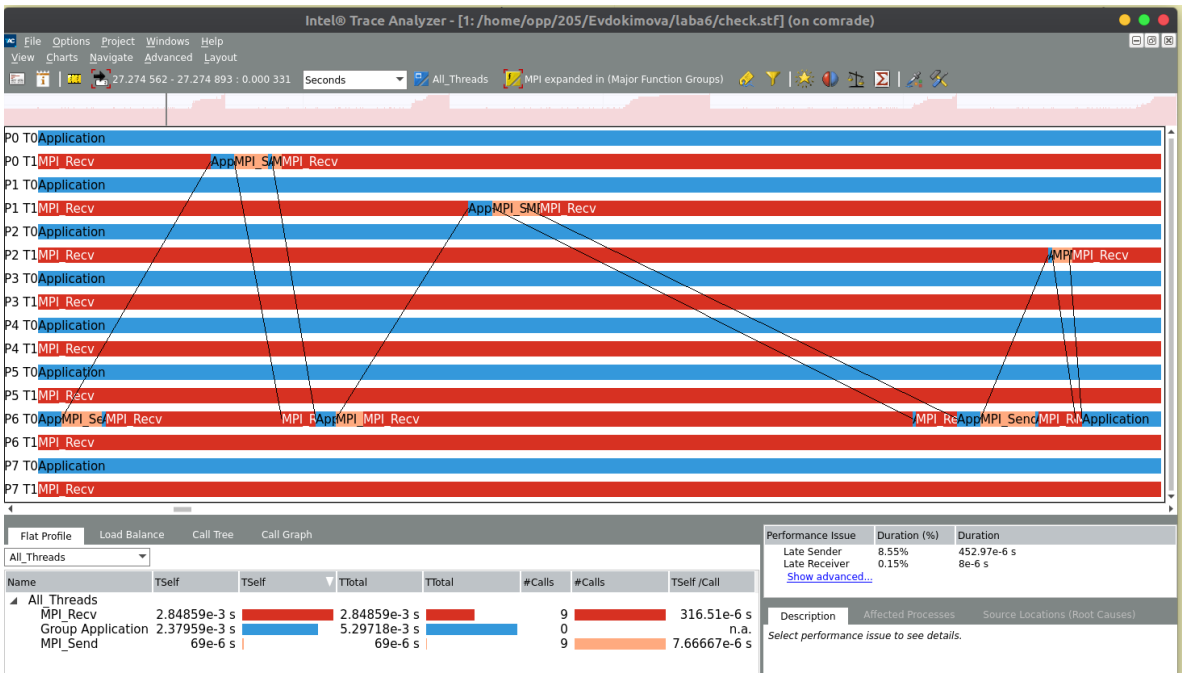

Рис. 2. Кусок выполнения

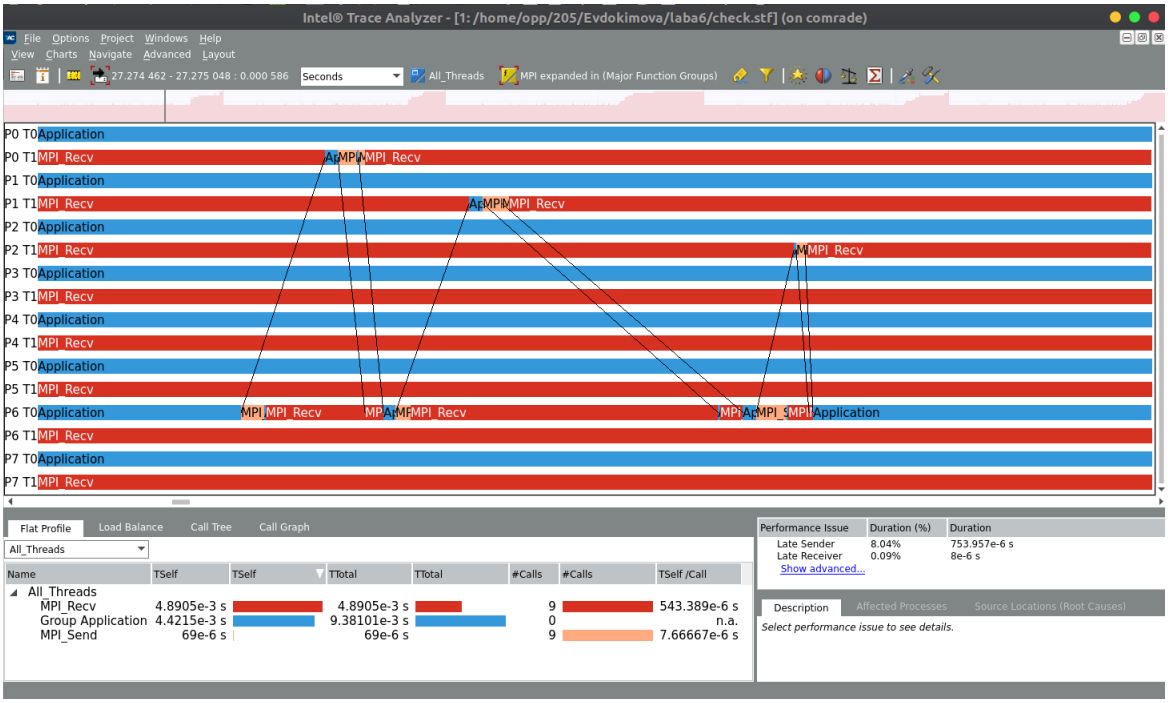

Рис. 3. Еще один кусок работы программы

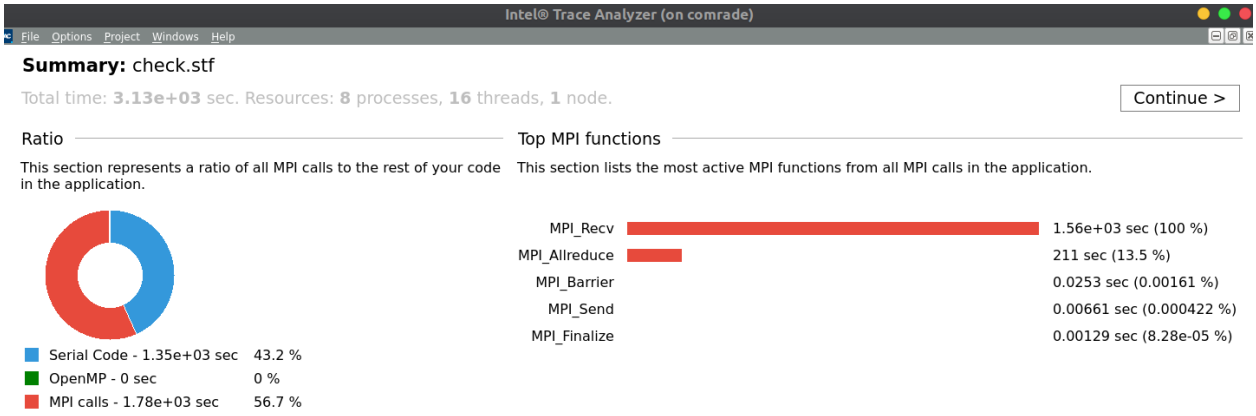

Рис. 4. Общее время работы функций на диаграмме