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.
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.

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 можно сделать вывод о том, что

После анализа измерений становится ясно, что при очень большом весе задач и малом их количестве на каждом процессе доля и время дисбаланса самые высокие, а также программа работает дольше. При очень малом весе задач и большом их количестве на каждом процессе также наблюдаются большие величины времени и доли дисбаланса. Можно сделать вывод, что эти два параметра должны быть сбалансированы между собой для того, чтобы можно было наблюдать хорошую эффективность. Также наблюдалось, что при итерировании по itersCount сначала наиболее легкие задачи получал первый процесс, а последний – самые сложные, что в последующем менялось в обратную сторону. Из-за того, что процесс с самыми легкими задачами заканчивал их исполнение быстрее остальных, он больше всех обращался к другим процессам за добавочными. Также сделан вывод о том, что наиболее эффективным способом отдачи процессом задач является деление оставшихся пополам.

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 100

#define MIN_AMOUNT_OF_TASKS_TO_SHARE 20
#define TASKS_PER_PROCESS 400

#define TAG_REQUEST 0
#define TAG_REPLY 1

double RES_PER_ITERATION = 0.0;
double GLOABAL_RESULT_SIN = 0.0;

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);

    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;

      // sending number of tasks that shares
      MPI_Send(&tasksToSend, 1, MPI_INT, rankRequestedTasks,
              TAG_REPLY, MPI_COMM_WORLD);

      pthread_mutex_lock(&mutexTasks);
      // sending tasks
      MPI_Send(tasks + amountOfTasksAlreadyExecuted + tasksInRemain -
                  1,
              tasksToSend, MPI_INT, rankRequestedTasks, TAG_REPLY,
              MPI_COMM_WORLD);
      pthread_mutex_unlock(&mutexTasks);
    } else {
      tasksToSend = 0;

      MPI_Send(&tasksToSend, 1, MPI_INT, rankRequestedTasks,
              TAG_REPLY, MPI_COMM_WORLD);
    }
    pthread_mutex_unlock(&mutexTasksInRemain);
  }
  return NULL;
}

void *workerThreadStart(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();

 calculateTask();

 //   requesting tasks from other processes
 for (int currentProc = 0; currentProc < amountOfProcs;
     ++currentProc) {
  if (currentProc == rankOfCurrProc)
   continue;

  // Send to other process that cur proc is free
  // and can compute additional tasks
  MPI_Send(&rankOfCurrProc, 1, MPI_INT, currentProc, TAG_REQUEST,
         MPI_COMM_WORLD);

  // Receive number of additional tasks
  MPI_Recv(&amountOfAdditionalasks, 1, MPI_INT, currentProc,
         TAG_REPLY, MPI_COMM_WORLD, MPI_STATUS_IGNORE);

  if (amountOfAdditionalasks > 0) {
   // Receive tasks
   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);

   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;
    }

    MPI_Barrier(MPI_COMM_WORLD);
    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);
    }
  }

  // Terminate Thread 'Receiver'
  // recv назходится в режиме ожидания сообшения, все процесс к
  // которому ты происодеинен закончил работу (говорим рисеиверу)
  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 createAndStartThreads() {
```

```
  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);

  workerThreadStart(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();
```

```
  createAndStartThreads();
  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

Количество заданий на 1 лист = 300; количество листов = 5;

Замеры программы на 2х процессах:

```
==================================================
Iteration numer: 0
Disbalance time: 0.00581715
Disbalance percentage: 81.0385
------------------------------------------
        Current proc is: 0
Amount of executed tasks: 354
Result of calculating is: 50.0208
Time per iteration: 0.0013611 seconds
        Current proc is: 1
Amount of executed tasks: 246
Result of calculating is: 220.475
Time per iteration: 0.00717825 seconds
==================================================
Iteration numer: 1
Disbalance time: 0.000548627
Disbalance percentage: 11.9314
------------------------------------------
        Current proc is: 0
Amount of executed tasks: 450
Result of calculating is: 183.674
Time per iteration: 0.00459817 seconds
        Current proc is: 1
Amount of executed tasks: 150
Result of calculating is: 355.739
Time per iteration: 0.00404954 seconds
==================================================
Iteration numer: 2
Disbalance time: 1.73009e-05
Disbalance percentage: 0.429308
------------------------------------------
        Current proc is: 0
Amount of executed tasks: 450
Result of calculating is: 317.327
Time per iteration: 0.00402996 seconds
        Current proc is: 1
Amount of executed tasks: 150
Result of calculating is: 491.003
Time per iteration: 0.00401266 seconds
==================================================
Iteration numer: 3
Disbalance time: 8.75802e-05
Disbalance percentage: 2.12398
------------------------------------------
        Current proc is: 0
Amount of executed tasks: 450
```

Result of calculating is: 450.98
Time per iteration: 0.00412339 seconds
Current proc is: 1
Amount of executed tasks: 150
Result of calculating is: 626.266
Time per iteration: 0.00403581 seconds
==========
Iteration numer: 4
Disbalance time: 0.000108329
Disbalance percentage: 2.63034

Current proc is: 0
Amount of executed tasks: 450
Result of calculating is: 584.633
Time per iteration: 0.00411843 seconds
Current proc is: 1
Amount of executed tasks: 150
Result of calculating is: 761.53
Time per iteration: 0.0040101 seconds
==========
Time for all lists: 0.020211seconds

Замеры программы на 4х процессах:
==========
Iteration numer: 0
Disbalance time: 71.2276
Disbalance percentage: 99.9831

Current proc is: 0
Amount of executed tasks: 474
Result of calculating is: 159.627
Time per iteration: 0.0120456 seconds
Current proc is: 1
Amount of executed tasks: 358
Result of calculating is: 313.176
Time per iteration: 0.0139392 seconds
Current proc is: 2
Amount of executed tasks: 167
Result of calculating is: 151.979
Time per iteration: 71.2397 seconds
Current proc is: 3
Amount of executed tasks: 201
Result of calculating is: 155.611
Time per iteration: 0.269782 seconds
==========
Iteration numer: 1
Disbalance time: 71.1188
Disbalance percentage: 99.9649

Current proc is: 0
Amount of executed tasks: 600
Result of calculating is: 434.091

Time per iteration: 0.0249635 seconds
Current proc is: 1
Amount of executed tasks: 173
Result of calculating is: 459.615
Time per iteration: 71.1438 seconds
Current proc is: 2
Amount of executed tasks: 216
Result of calculating is: 342.134
Time per iteration: 0.253715 seconds
Current proc is: 3
Amount of executed tasks: 211
Result of calculating is: 348.79
Time per iteration: 0.0252623 seconds
==========
Iteration numer: 2
Disbalance time: 2.79046
Disbalance percentage: 99.61

Current proc is: 0
Amount of executed tasks: 563
Result of calculating is: 673.825
Time per iteration: 0.0109246 seconds
Current proc is: 1
Amount of executed tasks: 180
Result of calculating is: 607.558
Time per iteration: 0.677388 seconds
Current proc is: 2
Amount of executed tasks: 218
Result of calculating is: 524.44
Time per iteration: 2.80138 seconds
Current proc is: 3
Amount of executed tasks: 239
Result of calculating is: 568.417
Time per iteration: 0.0165311 seconds
==========
Iteration numer: 3
Disbalance time: 48.7167
Disbalance percentage: 99.9775

Current proc is: 0
Amount of executed tasks: 564
Result of calculating is: 913.546
Time per iteration: 0.0109803 seconds
Current proc is: 1
Amount of executed tasks: 179
Result of calculating is: 755.199
Time per iteration: 0.669292 seconds
Current proc is: 2
Amount of executed tasks: 219
Result of calculating is: 707.263
Time per iteration: 48.7276 seconds
Current proc is: 3

Amount of executed tasks: 238
Result of calculating is: 787.235
Time per iteration: 0.0160928 seconds
==========
Iteration numer: 4
Disbalance time: 71.1158
Disbalance percentage: 99.9757

Current proc is: 0
Amount of executed tasks: 577
Result of calculating is: 1166.31
Time per iteration: 0.0196805 seconds
Current proc is: 1
Amount of executed tasks: 173
Result of calculating is: 901.637
Time per iteration: 71.1331 seconds
Current proc is: 2
Amount of executed tasks: 217
Result of calculating is: 897.419
Time per iteration: 0.291782 seconds
Current proc is: 3
Amount of executed tasks: 233
Result of calculating is: 1001.75
Time per iteration: 0.0173112 seconds
==========
Time for all lists: 265.045seconds

Замеры программы на 4х процессах:
==========
Iteration numer: 0
Disbalance time: 121.333
Disbalance percentage: 99.9821

Current proc is: 0
Amount of executed tasks: 154
Result of calculating is: 140.151
Time per iteration: 0.0233712 seconds
Current proc is: 1
Amount of executed tasks: 222
Result of calculating is: 189.188
Time per iteration: 116.134 seconds
Current proc is: 2
Amount of executed tasks: 211
Result of calculating is: 182.399
Time per iteration: 0.0250057 seconds
Current proc is: 3
Amount of executed tasks: 412
Result of calculating is: 362.324
Time per iteration: 0.0217192 seconds
Current proc is: 4
Amount of executed tasks: 407

Result of calculating is: 94.328
Time per iteration: 67.1958 seconds
Current proc is: 5
Amount of executed tasks: 395
Result of calculating is: 349.588
Time per iteration: 121.355 seconds
Current proc is: 6
Amount of executed tasks: 411
Result of calculating is: 379.732
Time per iteration: 0.0216977 seconds
Current proc is: 7
Amount of executed tasks: 188
Result of calculating is: 168.018
Time per iteration: 0.0312756 seconds
==========
Iteration numer: 1
Disbalance time: 102.815
Disbalance percentage: 99.9812

Current proc is: 0
Amount of executed tasks: 176
Result of calculating is: 283.853
Time per iteration: 0.0193203 seconds
Current proc is: 1
Amount of executed tasks: 137
Result of calculating is: 307.005
Time per iteration: 100.699 seconds
Current proc is: 2
Amount of executed tasks: 319
Result of calculating is: 470.011
Time per iteration: 102.834 seconds
Current proc is: 3
Amount of executed tasks: 346
Result of calculating is: 661.001
Time per iteration: 0.0211031 seconds
Current proc is: 4
Amount of executed tasks: 432
Result of calculating is: 215.853
Time per iteration: 0.0195265 seconds
Current proc is: 5
Amount of executed tasks: 480
Result of calculating is: 783.817
Time per iteration: 0.0210558 seconds
Current proc is: 6
Amount of executed tasks: 228
Result of calculating is: 590.764
Time per iteration: 0.0211428 seconds
Current proc is: 7
Amount of executed tasks: 282
Result of calculating is: 429.824
Time per iteration: 0.0201558 seconds
==========

Iteration numer: 2
Disbalance time: 114.406
Disbalance percentage: 99.9826

Current proc is: 0
Amount of executed tasks: 151
Result of calculating is: 396.706
Time per iteration: 0.0310065 seconds
Current proc is: 1
Amount of executed tasks: 153
Result of calculating is: 433.103
Time per iteration: 99.2947 seconds
Current proc is: 2
Amount of executed tasks: 333
Result of calculating is: 770.591
Time per iteration: 39.8124 seconds
Current proc is: 3
Amount of executed tasks: 356
Result of calculating is: 972.262
Time per iteration: 93.7522 seconds
Current proc is: 4
Amount of executed tasks: 442
Result of calculating is: 346.896
Time per iteration: 0.0212338 seconds
Current proc is: 5
Amount of executed tasks: 443
Result of calculating is: 1142.06
Time per iteration: 114.426 seconds
Current proc is: 6
Amount of executed tasks: 231
Result of calculating is: 805.964
Time per iteration: 0.0213947 seconds
Current proc is: 7
Amount of executed tasks: 291
Result of calculating is: 701.471
Time per iteration: 0.0199669 seconds
==========
Iteration numer: 3
Disbalance time: 64.117
Disbalance percentage: 99.9705

Current proc is: 0
Amount of executed tasks: 114
Result of calculating is: 496.788
Time per iteration: 61.2887 seconds
Current proc is: 1
Amount of executed tasks: 229
Result of calculating is: 639.832
Time per iteration: 0.0213077 seconds
Current proc is: 2
Amount of executed tasks: 225
Result of calculating is: 969.065

Time per iteration: 0.0224778 seconds
Current proc is: 3
Amount of executed tasks: 400
Result of calculating is: 1326.27
Time per iteration: 64.1359 seconds
Current proc is: 4
Amount of executed tasks: 502
Result of calculating is: 531.49
Time per iteration: 0.0189124 seconds
Current proc is: 5
Amount of executed tasks: 376
Result of calculating is: 1481.4
Time per iteration: 0.0226606 seconds
Current proc is: 6
Amount of executed tasks: 379
Result of calculating is: 1147.38
Time per iteration: 2.5732 seconds
Current proc is: 7
Amount of executed tasks: 175
Result of calculating is: 863.423
Time per iteration: 0.0214478 seconds
==========
Iteration numer: 4
Disbalance time: 110.73
Disbalance percentage: 99.9807

Current proc is: 0
Amount of executed tasks: 149
Result of calculating is: 622
Time per iteration: 0.021358 seconds
Current proc is: 1
Amount of executed tasks: 267
Result of calculating is: 880.029
Time per iteration: 110.752 seconds
Current proc is: 2
Amount of executed tasks: 227
Result of calculating is: 1180.2
Time per iteration: 0.0214163 seconds
Current proc is: 3
Amount of executed tasks: 382
Result of calculating is: 1672.74
Time per iteration: 0.0215451 seconds
Current proc is: 4
Amount of executed tasks: 439
Result of calculating is: 660.243
Time per iteration: 0.0218922 seconds
Current proc is: 5
Amount of executed tasks: 377
Result of calculating is: 1820.14
Time per iteration: 0.0230446 seconds
Current proc is: 6
Amount of executed tasks: 259

Result of calculating is: 1388.43
Time per iteration: 0.0215487 seconds
Current proc is: 7
Amount of executed tasks: 300
Result of calculating is: 1142.52
Time per iteration: 0.0219782 seconds
==========
Time for all lists: 513.531seconds

Приложение 3. Скрины из traceanalyzer

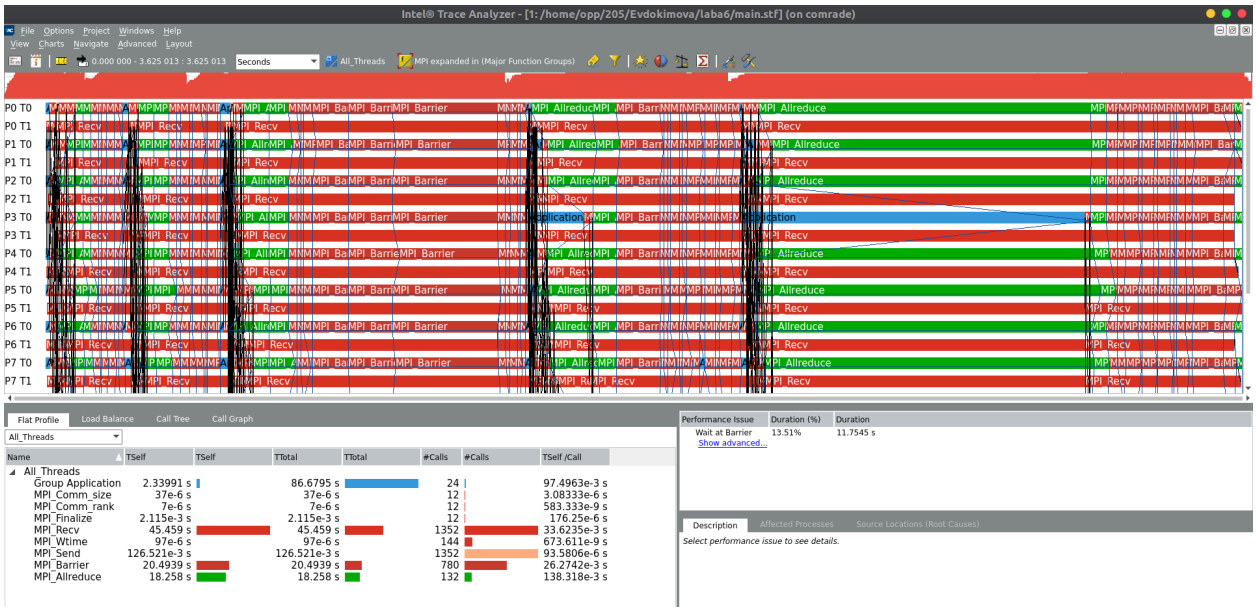

Рис. 1. Общий работы всей программы и общее время работы функций

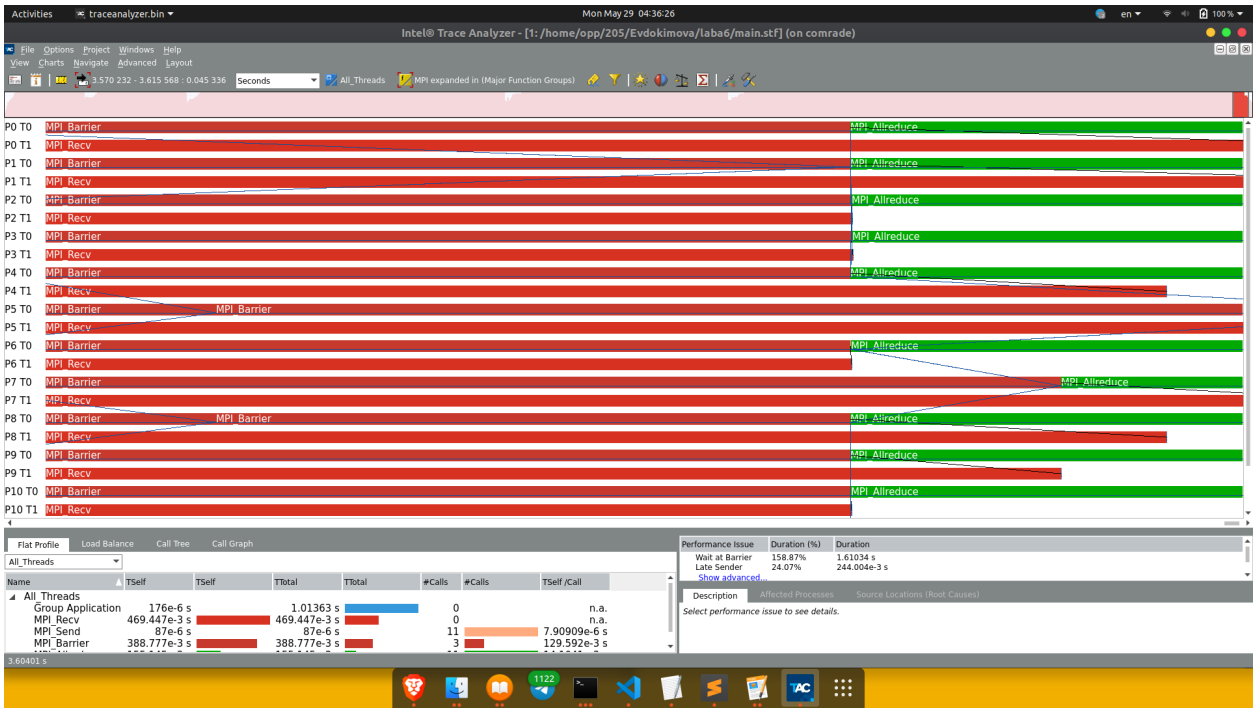

Рис. 2. Начало работы программы

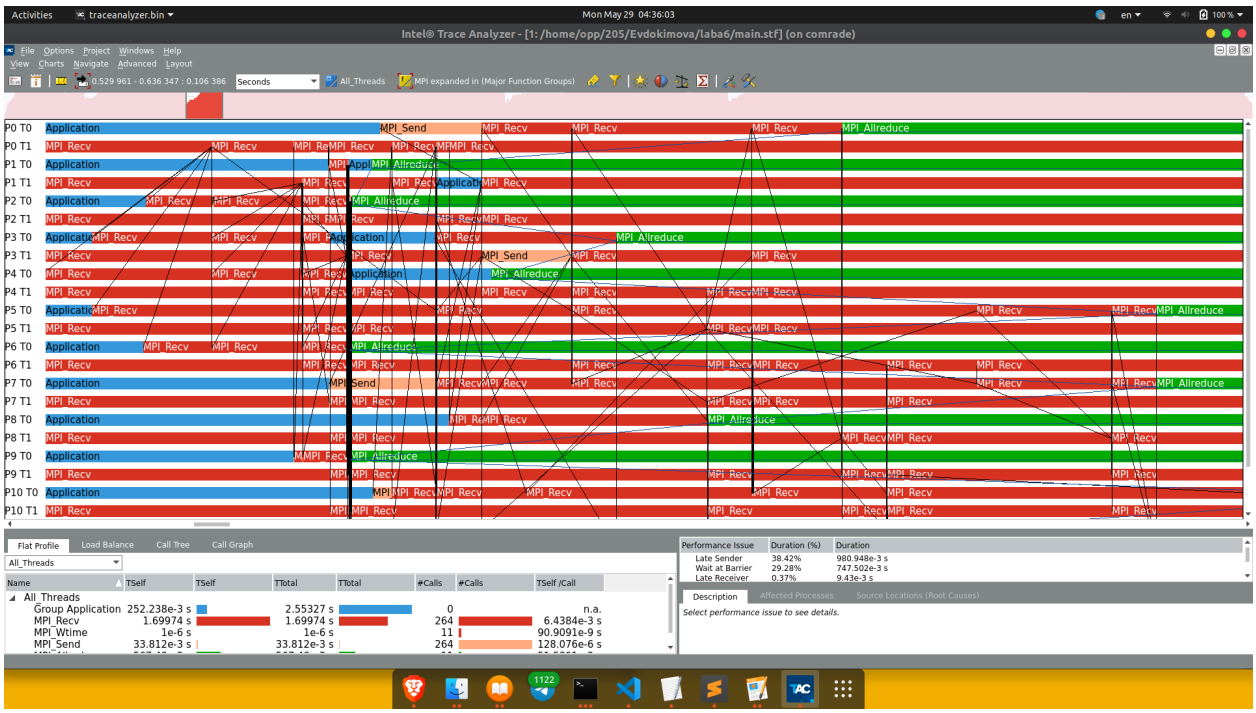

Рис. 3. Нулевой список

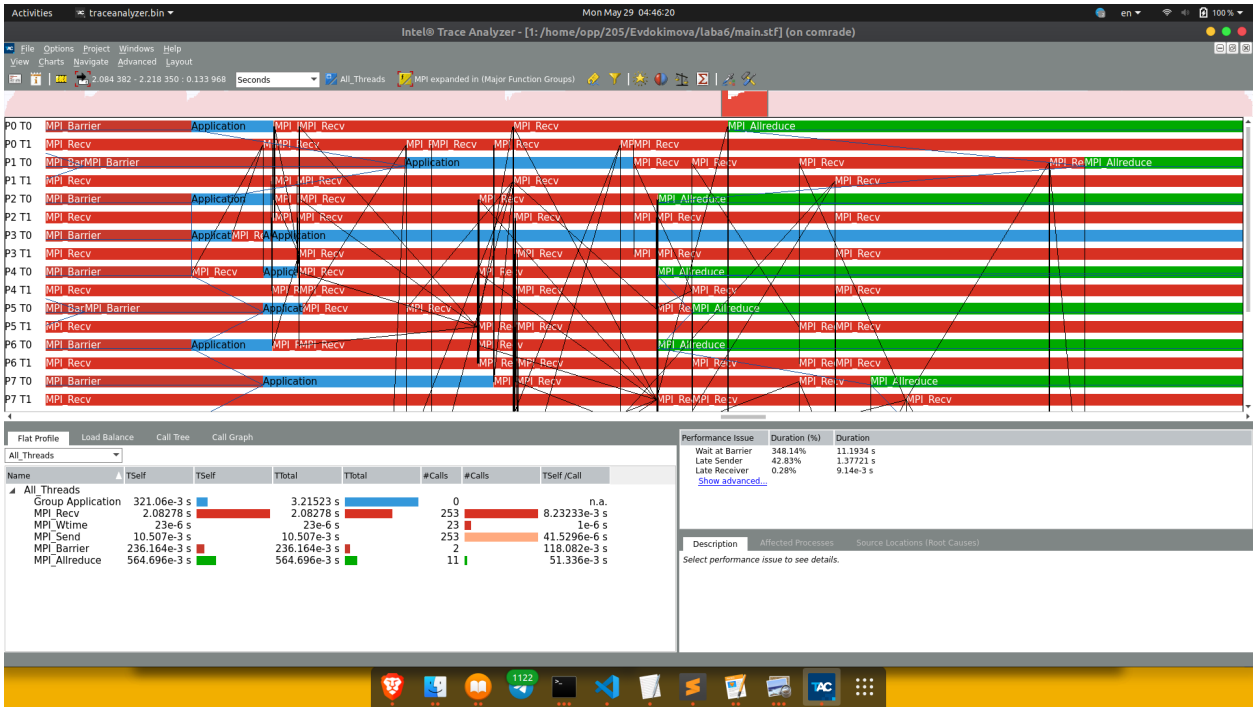

Рис. 4. Последний список

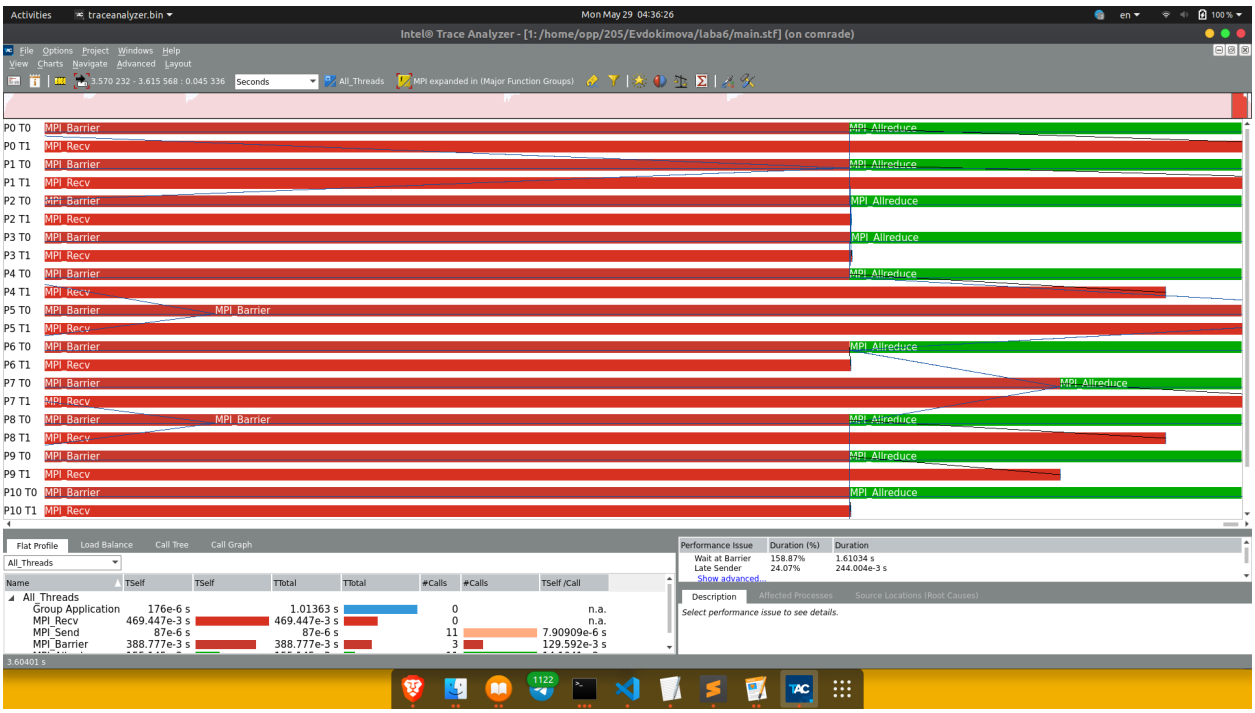

Рис. 4. Окончание работы программы

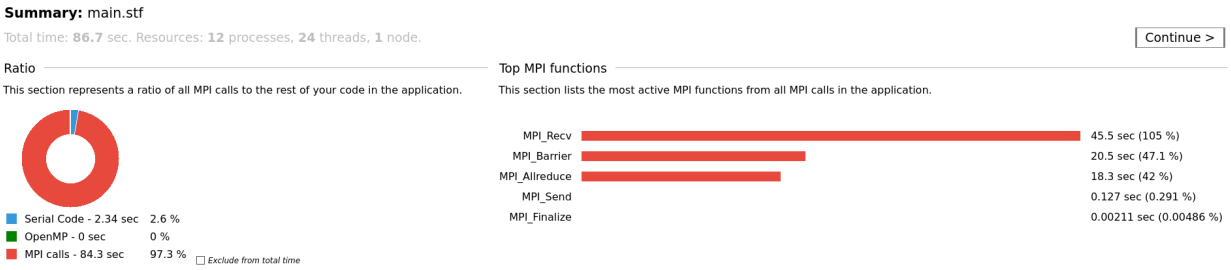

Рис. 5. Общее время работы функций на диаграмме