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 ЛАБОРАТОРНОЙ РАБОТЫ

«Программирование многопоточных приложений.
POSIX Threads»

студента 2 курса, 21204 группы

Осипова Александра Александровича

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:
Кандидат технических наук
А. Ю. Власенко

Новосибирск 2023

СОДЕРЖАНИЕ

ЦЕЛЬ

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

ЗАДАНИЕ

1. Формулировка задания представлена на сайте: https://ssd.sscc.ru/ru/content/opplabs/loadbalancing

2. Написать программу, реализующую параллельную обработку списков задач на разных процессах с использованием POSIX Threads API.

3. После каждой итерации обработки списков выводить:

    - количество заданий, выполненных каждый процессом за итерацию
    - значение globalRes
    - совокупное время выполнения заданий на итерации
    - время дисбаланса
    - долю дисбаланса

4. Подобрать параметры так, чтобы параллельная программа работа не меньше 30 секунд.

ОПИСАНИЕ РАБОТЫ

1. Была написана параллельная программа, в которой использовались функции MPI и POSIX Threads API.

1.1. Для организации списка задач был создан класс TaskList (см. Приложение 1)

1.2. В каждом процессе создается по три дополнительных потока:

1) threadWorker – генерирует список задач и обрабатывает его;

2) threadReceiver – принимает новые задачи от других процессов;

3) threadSender – отправляет задачи другим процессам;

1.3. Работа потока threadWorker:

```
void* worker(void* context) {
    ContextMPI *contextmpi = (ContextMPI *) context;

    for (int i = 0; i < ITER_MAX; ++i) {

        generateTaskList(contextmpi, contextmpi->numProc * 1000, i);
        isThreadRunning = true;
        pthread_barrier_wait(&barrier);

        double startTime = MPI_Wtime();
        int countExecutedTasks = 0;

        while (isThreadRunning) {
            while(true) {
                int task = taskList->pop();
                if (task == STOP_WORK) break;
                taskExecute(task);
                ++countExecutedTasks;
            }

            while(taskList->empty() && isThreadRunning) {
                pthread_mutex_lock(&mutex);
                pthread_cond_signal(&condWait);
                pthread_cond_wait(&condWork, &mutex);
                pthread_mutex_unlock(&mutex);
            }
        }
        double endTime = MPI_Wtime();
        printResult(contextmpi, i, countExecutedTasks, endTime - startTime);

    }
    return nullptr;
}
```

1.3.1 Функция, генерирующая новый список задач:

```
void generateTaskList(ContextMPI *contextmpi, int numTasks, int iterCounter)
{
    for (int i = contextmpi->rank * numTasks; i < contextmpi->rank * numTasks
+ numTasks; ++i) {
        int task = abs(50 - (i % 100)) * abs(contextmpi->rank - (iterCounter
```

```
% contextmpi->numProc)) * L;
        taskList->push(task);
    }
}
```

1.4. Работа потока threadReceiver:

```
void* receiver(void* context) {
    ContextMPI *contextmpi = (ContextMPI *) context;

    int recvTask;
    MPI_Status status;

    for (int i = 0; i < ITER_MAX; ++i) {
        pthread_barrier_wait(&barrier);

        while(isThreadRunning) {
            while (!taskList->empty()) {
                pthread_mutex_lock(&mutex);
                pthread_cond_signal(&condWork);
                pthread_cond_wait(&condWait, &mutex);
                pthread_mutex_unlock(&mutex);
            }
            int numCompletedProc = 0;
            for (int i = 0; i < contextmpi->numProc; ++i) {
                if(i == contextmpi->rank) continue;
                MPI_Send(&contextmpi->rank, 1, MPI_INT, i, TAG_REQUEST_TASK,
MPI_COMM_WORLD);
                MPI_Recv(&recvTask, 1, MPI_INT, i, TAG_SEND_TASK,
MPI_COMM_WORLD, MPI_STATUS_IGNORE);
                if (recvTask != PROC_FINISHED) {
                    taskList->push(recvTask);
                } else { ++numCompletedProc; }
            }
            if(numCompletedProc == contextmpi->numProc - 1)
                finishWorkx    (contextmpi, STOP_WORK);
        }
    }
    return nullptr;
}
```

1.5. Функция, оповещающая потоки о прекращении работы на данной итерации:

```
void finishWork(ContextMPI *contextmpi, const int recvTask) {
       MPI_Barrier(MPI_COMM_WORLD);
       isThreadRunning = false;

       MPI_Send(&recvTask, 1, MPI_INT, contextmpi->rank, TAG_REQUEST_TASK,
       MPI_COMM_WORLD);

       pthread_mutex_lock(&mutex);
       pthread_cond_signal(&condWork);
       pthread_mutex_unlock(&mutex)
}
```

1.6. Работа потока threadSender:

```
     void* sender(void* context) {
    ContextMPI *contextmpi = (ContextMPI *) context;

    MPI_Status status;
    int rankSender;
```

```
    for (int i = 0; i < ITER_MAX; ++i) {
        pthread_barrier_wait(&barrier);

        while(isThreadRunning) {
            MPI_Recv(&rankSender, 1, MPI_INT, MPI_ANY_SOURCE,
TAG_REQUEST_TASK, MPI_COMM_WORLD, &status);
            if(rankSender == STOP_WORK) continue;
            int task = taskList->pop();
            MPI_Send(&task, 1, MPI_INT, rankSender, TAG_SEND_TASK,
MPI_COMM_WORLD);
        }
    }
    return nullptr;
}
```

1.7. Подобранные значения параметров:

- *L = 800*
- *ITER_MAX = 20*
- numTasks = numProc * 1000

1.8. Результат работы программы был записан в файл result.txt (см. Приложение 3).

1.8.1 Результат первых пяти итераций:

countIter	rank	countExecutedTasks	globalRes	timeToIter
0	0	7832	2686.697886	2.141083
	1	6087	5527.064144	2.141092
	2	4675	4350.329954	2.141090
	3	3596	3444.159663	2.141178
	4	2810	2546.918546	2.141191
	Imbalance: 0.000108		Proportion: 0.005051%	
1	0	5399	7669.465568	1.376918
	1	7872	8191.785782	1.377361
	2	5399	9303.082122	1.376950
	3	3701	6857.638877	1.376964
	4	2629	5032.045248	1.377024
	Imbalance: 0.000444		Proportion: 0.032215%	
2	0	3302	10724.976638	1.174694
	1	5160	12891.152037	1.174706
	2	8044	12075.206857	1.174784
	3	5157	11572.648794	1.174698
	4	3337	8093.450231	1.174703
	Imbalance: 0.000090		Proportion: 0.007685%	
3	0	2599	13182.277504	1.374558
	1	3686	16294.402539	1.374565
	2	5415	17004.127968	1.374560

	3	7899	14326.695269	1.374584
	4	5401	13048.778681	1.374587
	Imbalance: 0.000029		Proportion: 0.002116 %	
4	0	2779	15716.533399	1.954926
	1	3579	19718.586461	1.954925
	2	4663	21344.768855	1.954919
	3	6102	19891.651052	1.954894
	4	7877	15739.912389	1.954920
	Imbalance: 0.000033		Proportion: 0.001677%	

ЗАКЛЮЧЕНИЕ

В рамках данной лабораторной работы была разобрана задача динамического распределения работы между процессами с использованием POSIX Threads API. Этот инструмент позволил выделить на каждом процессе дополнительные три потока, каждый из которых занимался специально отведенной для него функцией.

Приложение 1. tasklist.h

```
#pragma once

#include <atomic>
#include <queue>
#include <random>
#include <utility>

enum {
    TAG_REQUEST_TASK = 0,
    TAG_SEND_TASK = 1,
    ROOT = 0,
    STOP_WORK = -1,
    PROC_FINISHED = -1
};


class TaskList {
private:
    std::queue<int> queue;
    pthread_mutex_t* mutex;
    pthread_cond_t* cond;

public:
    TaskList(pthread_mutex_t* mutex, pthread_cond_t* cond) {
        this->mutex = mutex;
        this->cond = cond;
    }

    void push(const int &task) {
        pthread_mutex_lock(mutex);
        queue.push(task);
        pthread_cond_signal(cond);
        pthread_mutex_unlock(mutex);
    }

    int pop() {
        int task;
        pthread_mutex_lock(mutex);
        if (!queue.empty()) {
            task = queue.front();
            queue.pop();
        } else {
            task = PROC_FINISHED;
        }
        pthread_mutex_unlock(mutex);
        return task;
    }

    bool empty() {
        pthread_mutex_lock(mutex);
        bool isEmpty = queue.empty();
        pthread_mutex_unlock(mutex);

        return isEmpty;
    }
};
```

Приложение 2. Листинг программы, main.cpp

```
#include <iostream>
#include <mpi.h>
#include <pthread.h>
#include <memory>
#include <math.h>
#include "tasklist.h"

using namespace std;

#define L 800
#define ITER_MAX 20

pthread_barrier_t barrier;

typedef struct ContextMPI{
    int rank;
    int numProc;
} ContextMPI;

pthread_mutex_t mutex;

pthread_cond_t condWait;
pthread_cond_t condWork;

bool isThreadRunning;

std::unique_ptr<TaskList> taskList;

double globalRes = 0;

void generateTaskList(ContextMPI *contextmpi, int numTasks, int iterCounter) {
    for (int i = contextmpi->rank * numTasks; i < contextmpi->rank * numTasks +
numTasks; ++i) {
        int task = abs(50 - (i % 100)) * abs(contextmpi->rank - (iterCounter %
contextmpi->numProc)) * L;
        taskList->push(task);
    }
}

void* sender(void* context) {
    ContextMPI *contextmpi = (ContextMPI *) context;

    MPI_Status status;
    int rankSender;

    for (int i = 0; i < ITER_MAX; ++i) {
        pthread_barrier_wait(&barrier);

        while(isThreadRunning) {
            MPI_Recv(&rankSender, 1, MPI_INT, MPI_ANY_SOURCE, TAG_REQUEST_TASK,
MPI_COMM_WORLD, &status);
            if(rankSender == STOP_WORK) continue;
            int task = taskList->pop();
            MPI_Send(&task, 1, MPI_INT, rankSender, TAG_SEND_TASK, MPI_COMM_WORLD);
        }
    }
    return nullptr;
}

void finishWork(ContextMPI *contextmpi, const int recvTask) {
    MPI_Barrier(MPI_COMM_WORLD);
    isThreadRunning = false;
```

```
    MPI_Send(&recvTask, 1, MPI_INT, contextmpi->rank, TAG_REQUEST_TASK, MPI_COMM_WORLD);

    pthread_mutex_lock(&mutex);
    pthread_cond_signal(&condWork);
    pthread_mutex_unlock(&mutex);
}

void* receiver(void* context) {
    ContextMPI *contextmpi = (ContextMPI *) context;

    int recvTask;
    MPI_Status status;

    for (int i = 0; i < ITER_MAX; ++i) {
        pthread_barrier_wait(&barrier);

        while(isThreadRunning) {
            while (!taskList->empty()) {
                pthread_mutex_lock(&mutex);
                pthread_cond_signal(&condWork);
                pthread_cond_wait(&condWait, &mutex);
                pthread_mutex_unlock(&mutex);
            }
            int numCompletedProc = 0;
            for (int i = 0; i < contextmpi->numProc; ++i) {
                if(i == contextmpi->rank) continue;
                MPI_Send(&contextmpi->rank, 1, MPI_INT, i, TAG_REQUEST_TASK,
MPI_COMM_WORLD);
                MPI_Recv(&recvTask, 1, MPI_INT, i, TAG_SEND_TASK, MPI_COMM_WORLD,
MPI_STATUS_IGNORE);
                if (recvTask != PROC_FINISHED) {
                    taskList->push(recvTask);
                } else { ++numCompletedProc; }
            }
            if(numCompletedProc == contextmpi->numProc - 1)
                finishWork(contextmpi, STOP_WORK);
        }
    }
    return nullptr;
}

void taskExecute(int repeatNum) {
    for (int i = 0; i < repeatNum; ++i) {
        globalRes += sin(i);
    }
}

void printResult(ContextMPI *contextmpi, int i, int countExecutedTasks, double
timeToIter) {
    double maxTime = 0;
    double minTime = 0;

    MPI_Reduce(&timeToIter, &maxTime, 1, MPI_DOUBLE, MPI_MAX, 0, MPI_COMM_WORLD);
    MPI_Reduce(&timeToIter, &minTime, 1, MPI_DOUBLE, MPI_MIN, 0, MPI_COMM_WORLD);

    double imbalance = maxTime - minTime;

    printf("ITER: %d, RANK: %d, countExecutedTasks: %d, globalRes: %f, timeToIter:
%f\n", i, contextmpi->rank, countExecutedTasks, globalRes, timeToIter);
    if (contextmpi->rank == ROOT) {
        printf("ITER: %d, imbalance: %f, proportion of imbalance: %f %\n", i, imbalance,
(imbalance / maxTime) * 100);
    }
```

```
}

void* worker(void* context) {
    ContextMPI *contextmpi = (ContextMPI *) context;

    for (int i = 0; i < ITER_MAX; ++i) {

        generateTaskList(contextmpi, contextmpi->numProc * 1000, i);
        isThreadRunning = true;
        pthread_barrier_wait(&barrier);

        double startTime = MPI_Wtime();
        int countExecutedTasks = 0;

        while (isThreadRunning) {
            while(true) {
                int task = taskList->pop();
                if (task == STOP_WORK) break;
                taskExecute(task);
                ++countExecutedTasks;
            }

            while(taskList->empty() && isThreadRunning) {
                pthread_mutex_lock(&mutex);
                pthread_cond_signal(&condWait);
                pthread_cond_wait(&condWork, &mutex);
                pthread_mutex_unlock(&mutex);
            }
        }
        double endTime = MPI_Wtime();
        printResult(contextmpi, i, countExecutedTasks, endTime - startTime);

    }
    return nullptr;
}

void run(ContextMPI *contextmpi) {
    pthread_mutex_init(&mutex, nullptr);

    pthread_barrier_init(&barrier, NULL, 3);

    pthread_attr_t attrs;
    pthread_attr_init(&attrs);
    pthread_attr_setdetachstate(&attrs, PTHREAD_CREATE_JOINABLE);

    pthread_t threadWorker;
    pthread_t threadReceiver;
    pthread_t threadSender;

    pthread_cond_init(&condWait, nullptr);
    pthread_cond_init(&condWork, nullptr);

    taskList = std::make_unique<TaskList>(&mutex, &condWork);

    MPI_Barrier(MPI_COMM_WORLD);

    pthread_create(&threadWorker, &attrs, worker, contextmpi);
    pthread_create(&threadReceiver, &attrs, receiver, contextmpi);
    pthread_create(&threadSender, &attrs, sender, contextmpi);

    pthread_join(threadWorker, nullptr);
    pthread_join(threadReceiver, nullptr);
    pthread_join(threadSender, nullptr);
```

```
    pthread_attr_destroy(&attrs);
    pthread_cond_destroy(&condWait);
}

int main(int argc, char **argv) {
    int provider;
    MPI_Init_thread(&argc, &argv, MPI_THREAD_MULTIPLE, &provider);

    ContextMPI contextmpi;
    MPI_Comm_rank(MPI_COMM_WORLD, &contextmpi.rank);
    MPI_Comm_size(MPI_COMM_WORLD, &contextmpi.numProc);

    double start_time = MPI_Wtime();

    run(&contextmpi);

    double end_time = MPI_Wtime();

    if (contextmpi.rank == ROOT)
        std::cout << std::endl << "TIME: " << end_time - start_time << " seconds" <<
std::endl;

    MPI_Finalize();
    return 0;
}
```

Приложение 3. Результат работы программы, result.txt

```
ITER: 0, RANK: 0, countExecutedTasks: 7832, globalRes: 2686.697886, timeToIter: 2.141083
ITER: 0, imbalance: 0.000108, proportion of imbalance: 0.005051 %
ITER: 0, RANK: 1, countExecutedTasks: 6087, globalRes: 5527.064144, timeToIter: 2.141092
ITER: 0, RANK: 2, countExecutedTasks: 4675, globalRes: 4350.329954, timeToIter: 2.141090
ITER: 0, RANK: 3, countExecutedTasks: 3596, globalRes: 3444.159663, timeToIter: 2.141178
ITER: 0, RANK: 4, countExecutedTasks: 2810, globalRes: 2546.918546, timeToIter: 2.141191
ITER: 1, RANK: 0, countExecutedTasks: 5399, globalRes: 7669.465568, timeToIter: 1.376918
ITER: 1, imbalance: 0.000444, proportion of imbalance: 0.032215 %
ITER: 1, RANK: 1, countExecutedTasks: 7872, globalRes: 8191.785782, timeToIter: 1.377361
ITER: 1, RANK: 2, countExecutedTasks: 5399, globalRes: 9303.082122, timeToIter: 1.376950
ITER: 1, RANK: 3, countExecutedTasks: 3701, globalRes: 6857.638877, timeToIter: 1.376964
ITER: 1, RANK: 4, countExecutedTasks: 2629, globalRes: 5032.045248, timeToIter: 1.377024
ITER: 2, RANK: 0, countExecutedTasks: 3302, globalRes: 10724.976638, timeToIter:
1.174694
ITER: 2, imbalance: 0.000090, proportion of imbalance: 0.007685 %
ITER: 2, RANK: 1, countExecutedTasks: 5160, globalRes: 12891.152037, timeToIter:
1.174706
ITER: 2, RANK: 2, countExecutedTasks: 8044, globalRes: 12075.206857, timeToIter:
1.174784
ITER: 2, RANK: 3, countExecutedTasks: 5157, globalRes: 11572.648794, timeToIter:
1.174698
ITER: 2, RANK: 4, countExecutedTasks: 3337, globalRes: 8093.450231, timeToIter: 1.174703
ITER: 3, RANK: 0, countExecutedTasks: 2599, globalRes: 13182.277504, timeToIter:
1.374558
ITER: 3, imbalance: 0.000029, proportion of imbalance: 0.002116 %
ITER: 3, RANK: 1, countExecutedTasks: 3686, globalRes: 16294.402539, timeToIter:
1.374565
ITER: 3, RANK: 2, countExecutedTasks: 5415, globalRes: 17004.127968, timeToIter:
1.374560
ITER: 3, RANK: 3, countExecutedTasks: 7899, globalRes: 14326.695269, timeToIter:
1.374584
ITER: 3, RANK: 4, countExecutedTasks: 5401, globalRes: 13048.778681, timeToIter:
1.374587
ITER: 4, RANK: 0, countExecutedTasks: 2779, globalRes: 15716.533399, timeToIter:
1.954926
ITER: 4, imbalance: 0.000033, proportion of imbalance: 0.001677 %
```

```
ITER: 4, RANK: 1, countExecutedTasks: 3579, globalRes: 19718.586461, timeToIter:
1.954925
ITER: 4, RANK: 2, countExecutedTasks: 4663, globalRes: 21344.768855, timeToIter:
1.954919
ITER: 4, RANK: 3, countExecutedTasks: 6102, globalRes: 19891.651052, timeToIter:
1.954894
ITER: 4, RANK: 4, countExecutedTasks: 7877, globalRes: 15739.912389, timeToIter:
1.954920
ITER: 5, RANK: 0, countExecutedTasks: 7883, globalRes: 18391.719547, timeToIter:
1.956015
ITER: 5, imbalance: 0.000024, proportion of imbalance: 0.001249 %
ITER: 5, RANK: 1, countExecutedTasks: 6088, globalRes: 25319.295178, timeToIter:
1.956029
ITER: 5, RANK: 2, countExecutedTasks: 4644, globalRes: 25635.762946, timeToIter:
1.956032
ITER: 5, RANK: 3, countExecutedTasks: 3590, globalRes: 23313.079906, timeToIter:
1.956029
ITER: 5, RANK: 4, countExecutedTasks: 2795, globalRes: 18306.764772, timeToIter:
1.956008
ITER: 6, RANK: 0, countExecutedTasks: 5764, globalRes: 23680.603204, timeToIter:
1.495846
ITER: 6, imbalance: 0.000036, proportion of imbalance: 0.002389 %
ITER: 6, RANK: 1, countExecutedTasks: 8221, globalRes: 28320.457385, timeToIter:
1.495882
ITER: 6, RANK: 2, countExecutedTasks: 4204, globalRes: 29474.707797, timeToIter:
1.495848
ITER: 6, RANK: 3, countExecutedTasks: 3974, globalRes: 26996.416159, timeToIter:
1.495878
ITER: 6, RANK: 4, countExecutedTasks: 2837, globalRes: 20993.285209, timeToIter:
1.495876
ITER: 7, RANK: 0, countExecutedTasks: 3299, globalRes: 26761.978578, timeToIter:
1.174287
ITER: 7, imbalance: 0.000025, proportion of imbalance: 0.002116 %
ITER: 7, RANK: 1, countExecutedTasks: 5155, globalRes: 32993.838783, timeToIter:
1.174300
ITER: 7, RANK: 2, countExecutedTasks: 8042, globalRes: 32218.170903, timeToIter:
1.174275
ITER: 7, RANK: 3, countExecutedTasks: 5166, globalRes: 31723.901359, timeToIter:
1.174298
ITER: 7, RANK: 4, countExecutedTasks: 3338, globalRes: 24070.997092, timeToIter:
1.174293
ITER: 8, RANK: 0, countExecutedTasks: 2600, globalRes: 29229.829240, timeToIter:
1.372342
ITER: 8, imbalance: 0.000046, proportion of imbalance: 0.003380 %
ITER: 8, RANK: 1, countExecutedTasks: 3691, globalRes: 36412.949257, timeToIter:
1.372296
ITER: 8, RANK: 2, countExecutedTasks: 5408, globalRes: 37161.955101, timeToIter:
1.372297
ITER: 8, RANK: 3, countExecutedTasks: 7883, globalRes: 34433.134811, timeToIter:
1.372305
ITER: 8, RANK: 4, countExecutedTasks: 5418, globalRes: 29029.865710, timeToIter:
1.372304
ITER: 9, RANK: 0, countExecutedTasks: 2786, globalRes: 31792.342069, timeToIter:
1.963685
ITER: 9, imbalance: 0.000057, proportion of imbalance: 0.002928 %
ITER: 9, RANK: 1, countExecutedTasks: 3601, globalRes: 39851.339087, timeToIter:
1.963701
ITER: 9, RANK: 2, countExecutedTasks: 4663, globalRes: 41477.931354, timeToIter:
1.963656
ITER: 9, RANK: 3, countExecutedTasks: 6042, globalRes: 39952.828798, timeToIter:
1.963649
ITER: 9, RANK: 4, countExecutedTasks: 7908, globalRes: 31748.463001, timeToIter:
1.963643
ITER: 10, RANK: 0, countExecutedTasks: 7897, globalRes: 34510.987703, timeToIter:
```

```
1.965477
ITER: 10, imbalance: 0.000026, proportion of imbalance: 0.001337 %
ITER: 10, RANK: 1, countExecutedTasks: 6027, globalRes: 45346.356287, timeToIter:
1.965493
ITER: 10, RANK: 2, countExecutedTasks: 4662, globalRes: 45782.878638, timeToIter:
1.965489
ITER: 10, RANK: 3, countExecutedTasks: 3602, globalRes: 43397.639015, timeToIter:
1.965503
ITER: 10, RANK: 4, countExecutedTasks: 2812, globalRes: 34340.212856, timeToIter:
1.965482
ITER: 11, RANK: 0, countExecutedTasks: 5380, globalRes: 39407.268367, timeToIter:
1.385124
ITER: 11, imbalance: 0.000558, proportion of imbalance: 0.040272 %
ITER: 11, RANK: 1, countExecutedTasks: 7834, globalRes: 47987.204056, timeToIter:
1.385682
ITER: 11, RANK: 2, countExecutedTasks: 5429, globalRes: 50781.665097, timeToIter:
1.385135
ITER: 11, RANK: 3, countExecutedTasks: 3719, globalRes: 46870.110519, timeToIter:
1.385143
ITER: 11, RANK: 4, countExecutedTasks: 2638, globalRes: 36830.673862, timeToIter:
1.385146
ITER: 12, RANK: 0, countExecutedTasks: 3511, globalRes: 42648.696159, timeToIter:
1.260518
ITER: 12, imbalance: 0.000122, proportion of imbalance: 0.009712 %
ITER: 12, RANK: 1, countExecutedTasks: 5420, globalRes: 52910.955311, timeToIter:
1.260640
ITER: 12, RANK: 2, countExecutedTasks: 7681, globalRes: 53255.565927, timeToIter:
1.260601
ITER: 12, RANK: 3, countExecutedTasks: 5133, globalRes: 51560.038342, timeToIter:
1.260595
ITER: 12, RANK: 4, countExecutedTasks: 3255, globalRes: 39805.083121, timeToIter:
1.260597
ITER: 13, RANK: 0, countExecutedTasks: 2574, globalRes: 45085.733134, timeToIter:
1.477870
ITER: 13, imbalance: 0.000372, proportion of imbalance: 0.025164 %
ITER: 13, RANK: 1, countExecutedTasks: 3635, globalRes: 56254.419459, timeToIter:
1.477889
ITER: 13, RANK: 2, countExecutedTasks: 5358, globalRes: 58163.624423, timeToIter:
1.478242
ITER: 13, RANK: 3, countExecutedTasks: 7946, globalRes: 54363.279119, timeToIter:
1.477892
ITER: 13, RANK: 4, countExecutedTasks: 5487, globalRes: 44812.130127, timeToIter:
1.477900
ITER: 14, RANK: 0, countExecutedTasks: 2760, globalRes: 47610.233094, timeToIter:
1.981799
ITER: 14, imbalance: 0.000024, proportion of imbalance: 0.001192 %
ITER: 14, RANK: 1, countExecutedTasks: 3550, globalRes: 59653.338460, timeToIter:
1.981808
ITER: 14, RANK: 2, countExecutedTasks: 4626, globalRes: 62445.622738, timeToIter:
1.981798
ITER: 14, RANK: 3, countExecutedTasks: 6134, globalRes: 59977.556131, timeToIter:
1.981822
ITER: 14, RANK: 4, countExecutedTasks: 7930, globalRes: 47547.606028, timeToIter:
1.981815
ITER: 15, RANK: 0, countExecutedTasks: 7869, globalRes: 50231.219083, timeToIter:
1.956129
ITER: 15, imbalance: 0.000674, proportion of imbalance: 0.034456 %
ITER: 15, RANK: 1, countExecutedTasks: 6076, globalRes: 65300.994003, timeToIter:
1.956122
ITER: 15, RANK: 2, countExecutedTasks: 4658, globalRes: 66743.267764, timeToIter:
1.956125
ITER: 15, RANK: 3, countExecutedTasks: 3598, globalRes: 63391.665567, timeToIter:
1.956122
ITER: 15, RANK: 4, countExecutedTasks: 2799, globalRes: 50122.380224, timeToIter:
```

```
1.956796
ITER: 16, RANK: 0, countExecutedTasks: 5404, globalRes: 55170.415097, timeToIter:
1.375752
ITER: 16, imbalance: 0.000693, proportion of imbalance: 0.050316 %
ITER: 16, RANK: 1, countExecutedTasks: 7885, globalRes: 68011.785947, timeToIter:
1.375791
ITER: 16, RANK: 2, countExecutedTasks: 5384, globalRes: 71644.360019, timeToIter:
1.375776
ITER: 16, RANK: 3, countExecutedTasks: 3693, globalRes: 66860.322092, timeToIter:
1.376445
ITER: 16, RANK: 4, countExecutedTasks: 2634, globalRes: 52601.490885, timeToIter:
1.375771
ITER: 17, RANK: 0, countExecutedTasks: 3320, globalRes: 58222.277548, timeToIter:
1.174811
ITER: 17, imbalance: 0.000043, proportion of imbalance: 0.003637 %
ITER: 17, RANK: 1, countExecutedTasks: 5144, globalRes: 72707.025735, timeToIter:
1.174772
ITER: 17, RANK: 2, countExecutedTasks: 8007, globalRes: 74349.643336, timeToIter:
1.174768
ITER: 17, RANK: 3, countExecutedTasks: 5166, globalRes: 71592.119265, timeToIter:
1.174768
ITER: 17, RANK: 4, countExecutedTasks: 3363, globalRes: 55720.725110, timeToIter:
1.174768
ITER: 18, RANK: 0, countExecutedTasks: 2602, globalRes: 60680.993171, timeToIter:
1.374983
ITER: 18, imbalance: 0.000045, proportion of imbalance: 0.003278 %
ITER: 18, RANK: 1, countExecutedTasks: 3674, globalRes: 76137.843660, timeToIter:
1.374951
ITER: 18, RANK: 2, countExecutedTasks: 5415, globalRes: 79335.150429, timeToIter:
1.374938
ITER: 18, RANK: 3, countExecutedTasks: 7897, globalRes: 74239.570415, timeToIter:
1.374940
ITER: 18, RANK: 4, countExecutedTasks: 5412, globalRes: 60697.080718, timeToIter:
1.374952
ITER: 19, RANK: 0, countExecutedTasks: 2789, globalRes: 63251.378430, timeToIter:
1.963486
ITER: 19, imbalance: 0.000057, proportion of imbalance: 0.002899 %
ITER: 19, RANK: 1, countExecutedTasks: 3590, globalRes: 79570.103907, timeToIter:
1.963453
ITER: 19, RANK: 2, countExecutedTasks: 4665, globalRes: 83600.801398, timeToIter:
1.963429
ITER: 19, RANK: 3, countExecutedTasks: 6059, globalRes: 79794.875105, timeToIter:
1.963439
ITER: 19, RANK: 4, countExecutedTasks: 7897, globalRes: 63428.649740, timeToIter:
1.963432

TIME: 31.9076 seconds
```