МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГБОУ ВО «АЛТАЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт цифровых технологий, электроники и физики (ИЦТЭФ)
Кафедра вычислительной техники и электроники (ВТиЭ)

Анализ характеристик выполнения параллельных приложений в виртуальной среде MPI-машины на примере работы тестовой программы вычисления числа π

Отчет по лабораторной работе №1

Выполнил: студент гр. 5.306М
_____ А. В. Лаптев
Проверил: ст. преп. кафедры ВТиЭ
_____ И. А. Шмаков
«____» _____ 2024г.

Барнаул, 2024г.

СОДЕРЖАНИЕ

Цель работы

Изучить зависимости «коэффициента ускорения» и «латентности» вычислений на «сетевой распределенной ВС» по результатам вычислительных экспериментов, реализуемых с помощью тестовой параллельной MPI-программы по вычислению числа π.

Задание

1. изучить стандарт интерфейса MPI (библиотеке функций как расширение системы программирования C/C++) – описание стандарта находится в Приложении к лабораторной работе в виде файла «стандарт_MPI_Воеводин.doc»;
2. по приведенному ниже тексту MPI-программы восстановить алгоритм вычисления числа и его блок-схему; для проведения вычислительных экспериментов в приложении имеется файл «icpi.c» с исходным текстом программы и исполняемый файл «cpi.exe»;
3. провести вычислительные эксперименты, построить графики и диаграммы, оформить отчет.

Выполнение работы

После знакомства со стандартом интерфейса и изучением приведенного в приложении кода был составлен алгоритм работы программы, на основании которого была создана блок-схема.

Изменения исходного кода

Изначально, для автоматизации программы для расчета значения числа π для различного количества интервалов была частично переписана программа на языке C и также написан bash-скрипт, который является оберткой, в которой выполняется исполняемый файл заданное количество раз с заданными начальными условиями.

Текст измененной программы и bash-скрипт приведены в Приложении.

Описание алгоритма

1. *Начало*
2. Инициализация MPI, определение числа процессов и номера текущего процесса.
3. Прочитать число интервалов для расчёта числа π.
4. Фиксация времени начала вычислений для определения их скорости.
5. *Начало цикла*
6. Для всех значений интервалов в счетчике вычислить аргумент подынтегральной функции.
7. Для всех значений интервалов в счетчике вычислить значение подынтегральной функции.
8. Для всех значений интервалов в счетчике прибавить значение к аккумулятору суммы.
9. *Конец цикла*
10. Вычислить значение $mypi$ для каждого процесса.
11. Сложить значения $mypi$ всех процессов в переменной pi процесса с номером 0.
12. Если номер процесса – 0, зафиксировать время завершения вычислений и вывести значения:
    - полученное значение числа π;
    - ошибку вычисления числа π;
    - время выполнения вычислений в секундах.
13. Возврат к шагу 3.
14. Завершить работу процесса MPI.
15. *Конец*

Блок-схема алгоритма

На рис. 1 представлена блок-схема по приведенному выше алгоритму.

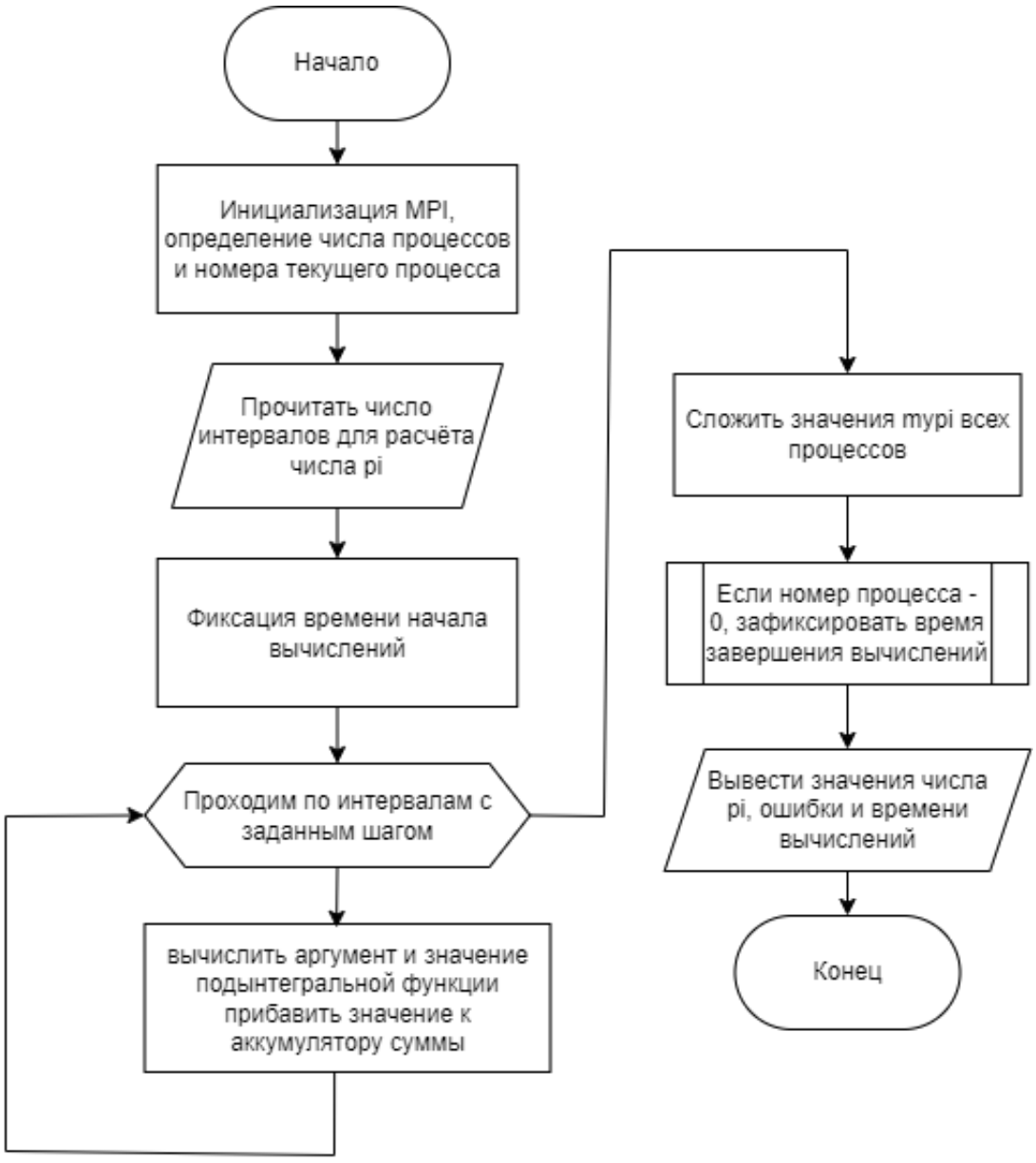

Рис. 1 Блок-схема алгоритма для расчета числа π

Обработка результатов работы программы

В результате работы программы был сформирован текстовый файл со всеми необходимыми исходными данными для построения графиков.

Ниже будет представлено несколько графиков, которые отображают зависимость различных параметров, используемых в расчетах друг от друга.

График, который представлен на рис. 2 отображает зависимость погрешности вычисления числа π от числа интервалов для тестовых замеров, проведенных с использованием архитектуры ARM.

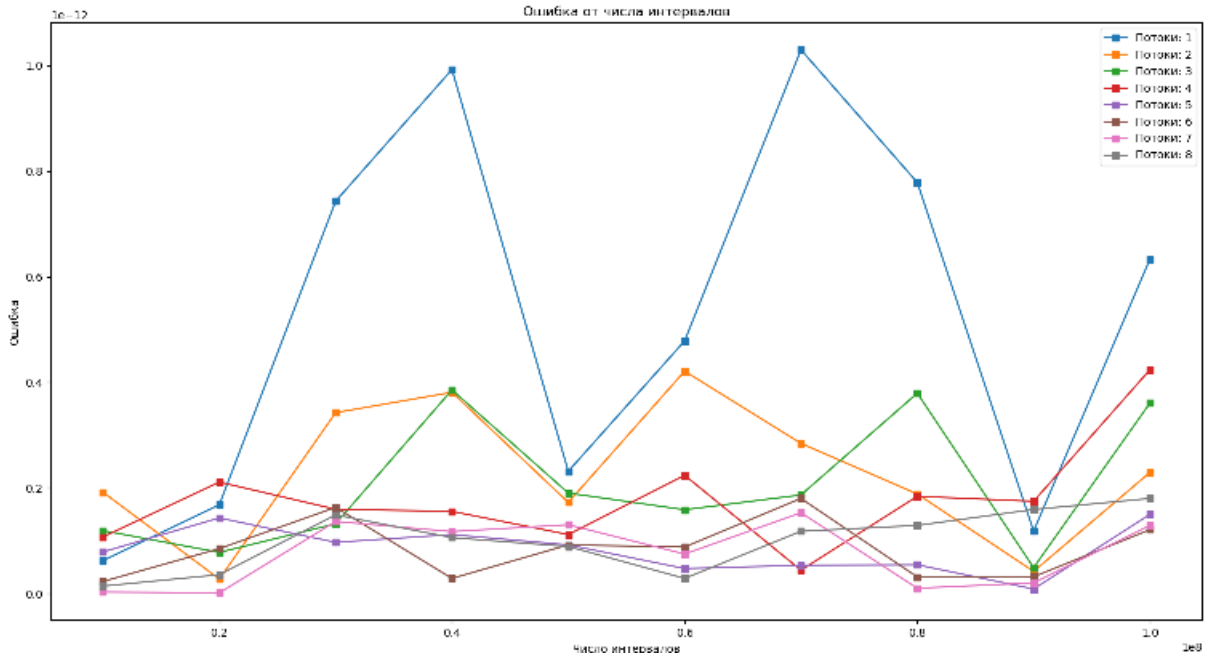

Рис. 2 Зависимость погрешности вычисления числа π от числа интервалов

Для расчета следующей зависимости требуется найти ускорение, с которым обрабатываются одни и те же данные на разном количестве компьютеров. Это можно сделать по формуле (график, отображающий эту зависимость приведен на рис. 3):

$$k_i = \frac{t_1}{t_i}.$$

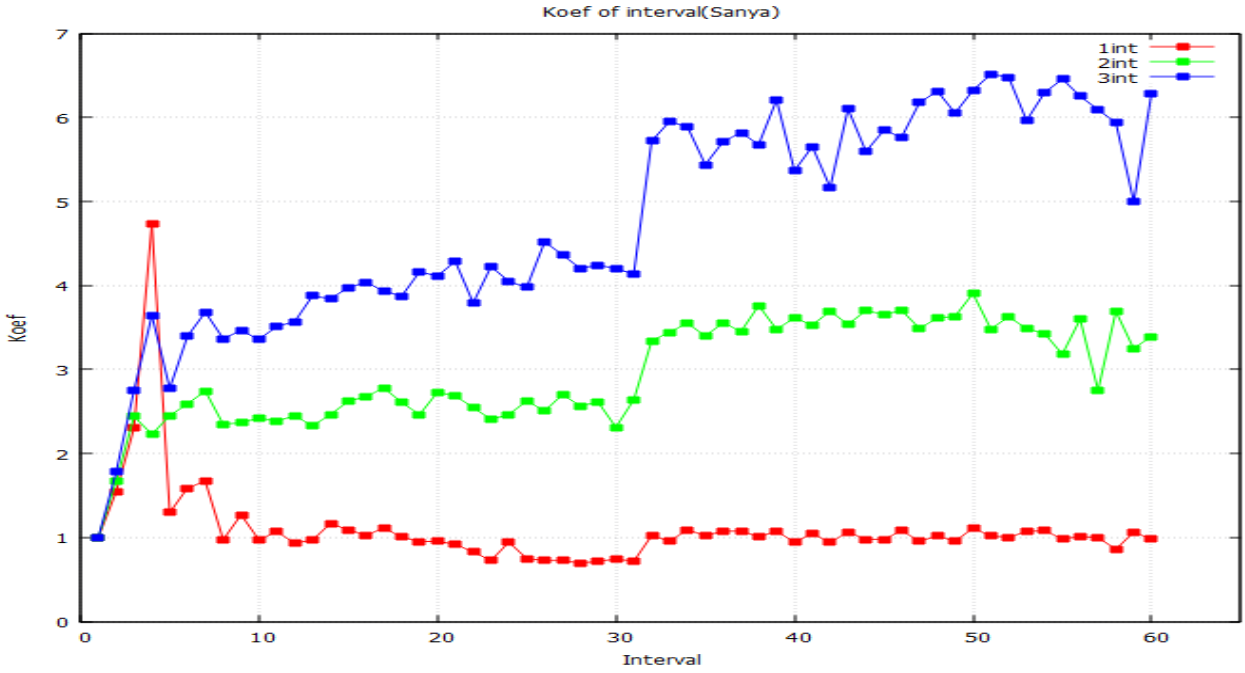

Рис. 3 Коэффициент ускорения при вычислении числа π на разном количестве компьютеров

Ниже на рис. 4 для сравнения приведен график, который показывает такую же зависимость для архитектуры ARM.

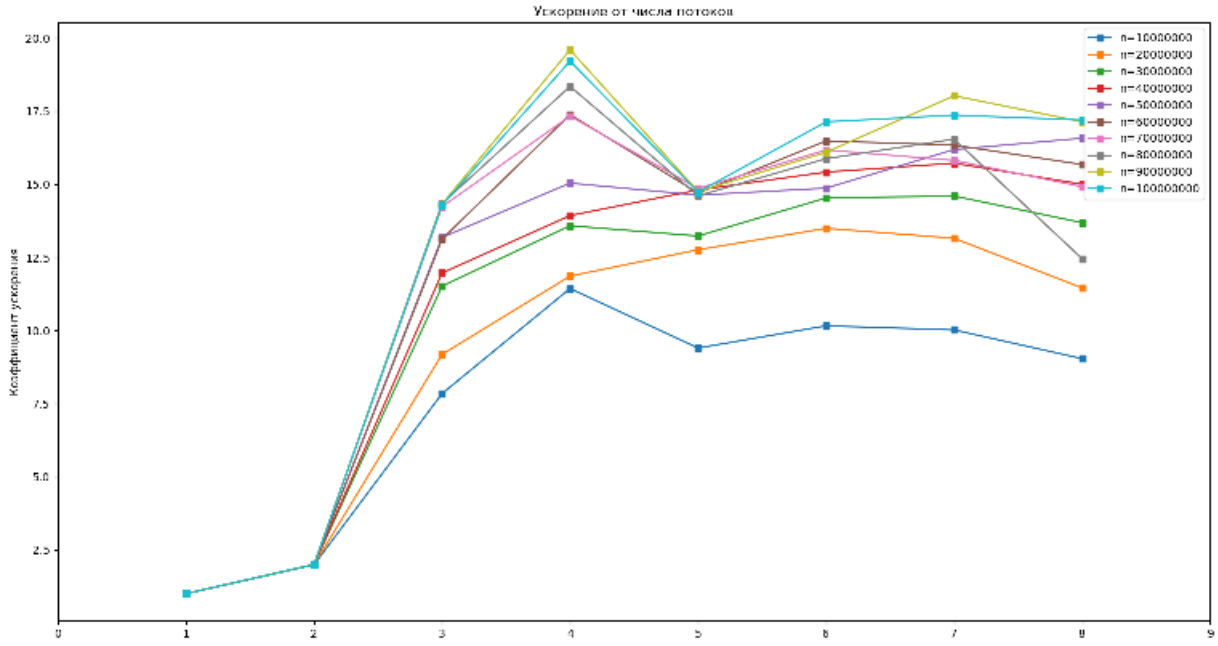

Рис. 4 Коэффициент ускорения при вычислении числа π на разном количестве компьютеров для ARM

На рис. 5 и рис. 6 представлены сравнительные графики для зависимости среднего времени вычисления числа π от количества интервалов.

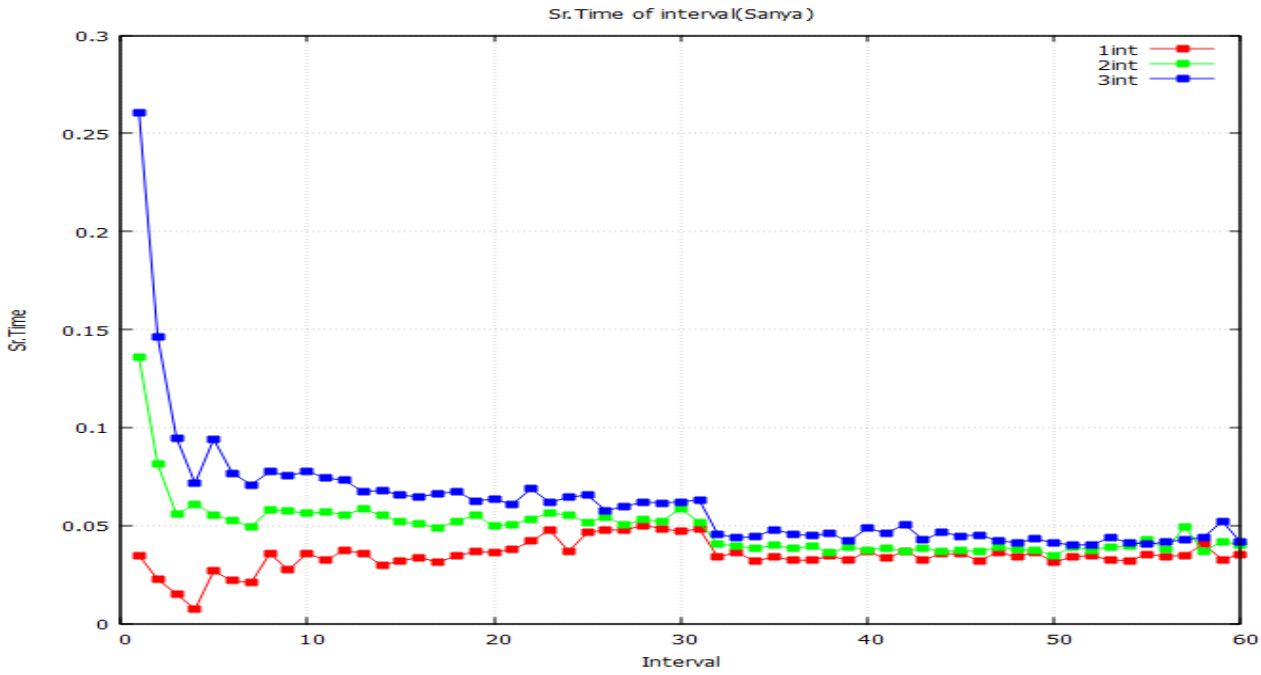

Рис. 5 Зависимость времени счета от количества интервалов

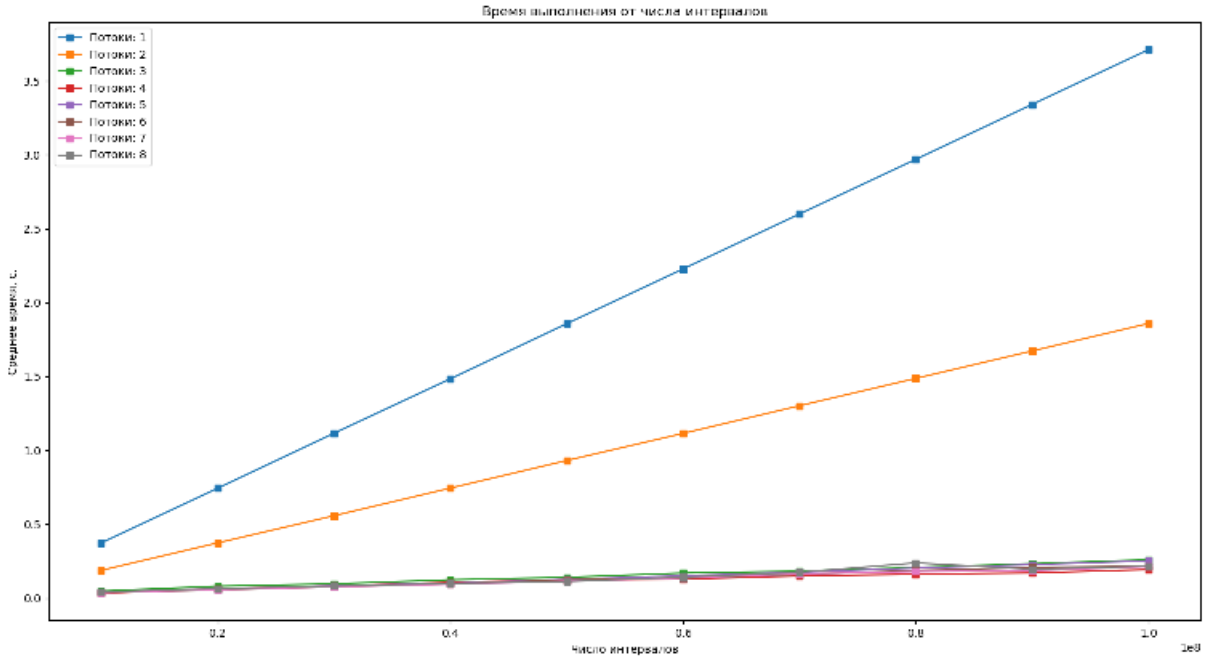

Рис. 6 Зависимость времени счета от количества интервалов для ARM

На рис. 7 показана зависимость среднего времени расчетов от различного количества потоков.

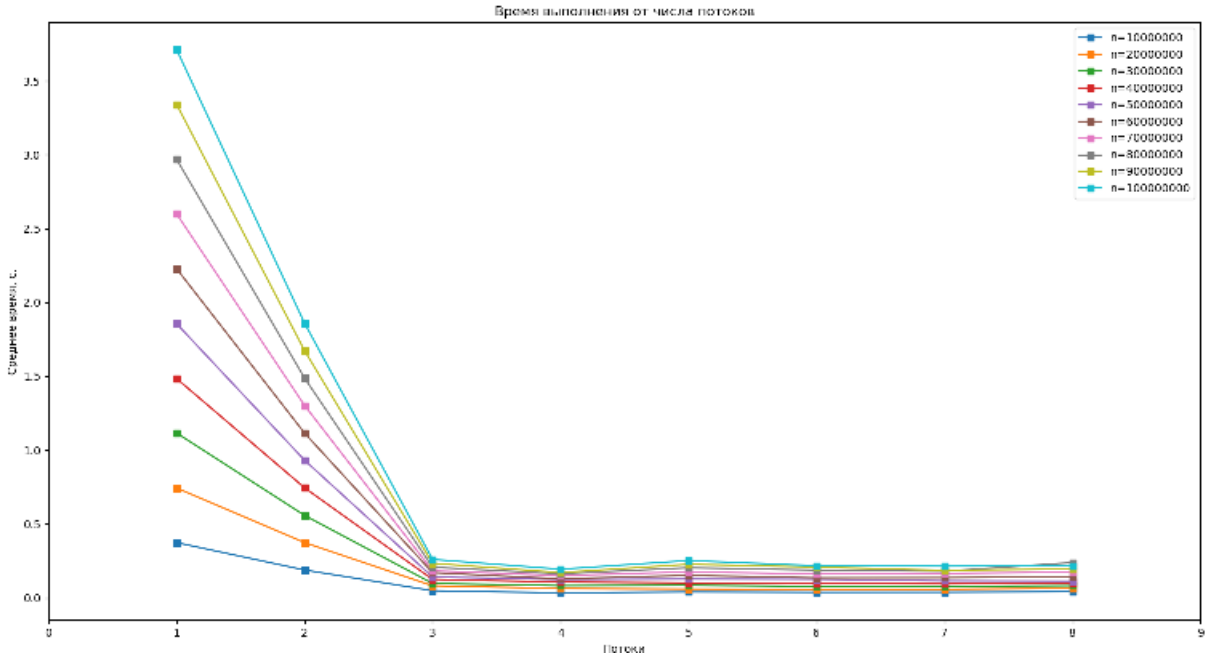

Рис. 7 Зависимость времени счета от количества потоков для ARM

Вывод

Анализ вычислительных экспериментов показывает, что с увеличением в составе «сетевой распределенной ЭВМ» числа процессоров за счет увеличения обменов сообщениями между процессорами по «комбинаторному закону» увеличивается латентность в процессе выполнения параллельной программы (увеличиваются временные задержки), что приводит к снижению пропорционального роста ускорения вычислений.

ПРИЛОЖЕНИЕ

Листинг 1 Исходный код для расчета числа π

```
/* This is an interactive version of cpi */
#include "mpi.h"
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

double f(double);

double f(double a)
{
    return (4.0 / (1.0 + a*a));
}

int main(int argc,char *argv[])
{
    int n, myid, numprocs, i;
    double PI25DT = 3.14159265358979323846264643;
    double mypi, pi, h, sum, x;
    double startwtime = 0.0, endwtime;
    int  namelen;
    char processor_name[MPI_MAX_PROCESSOR_NAME];

    MPI_Init(&argc,&argv);
    MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD,&myid);
    MPI_Get_processor_name(processor_name,&namelen);

    /*
    fprintf(stdout,"Process %d of %d is on %s\n",
        myid, numprocs, processor_name);
    fflush(stdout);
```

```
32      */
33
34      if (myid == 0) {
35          n = atoi(argv[1]);
36          int count_thread = atoi(argv[2]);
37          // Расчет ускорения
38          //spped =
39          fflush(stdout);
40          startwtime = MPI_Wtime();
41      }
42      MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);
43      h   = 1.0 / (double) n;
44      sum = 0.0;
45      for (i = myid + 1; i <= n; i += numprocs) {
46          x = h * ((double)i - 0.5);
47          sum += f(x);
48      }
49      mypi = h * sum;
50      MPI_Reduce(&mypi, &pi, 1, MPI_DOUBLE, MPI_SUM, 0,
        ↪  MPI_COMM_WORLD);
51
52      if (myid == 0) {
53          printf("%.16f, %.16f, ",
54                 pi, fabs(pi - PI25DT));
55          endwtime = MPI_Wtime();
56          printf("%f\n", endwtime-startwtime);
57          fflush( stdout );
58
59          FILE *error;
60          error = fopen("error_graphic.txt", "a");
61          FILE *coef_speed;
62          coef_speed = fopen("coef_speed.txt", "a");
63          FILE *time;
64          time = fopen("time.txt", "a");
65
```

```
66          fprintf(error, "%f %d\n", fabs(pi - PI25DT), n);
67          //fprintf(coef_speed, "%d %d\n", speed, count_thread);
68          fprintf(error, "%f %d\n", endwtime-startwtime, n);
69
70          fclose(error);
71          fclose(coef_speed);
72          fclose(time);
73      }
74      MPI_Finalize();
75      return 0;
76  }
```

Листинг 2 Bash-скрипт для запуска исполняемого файла

```
1  #!/bin/bash
2  # Подключение ко всем машинам
3  for ((cnt_machine = 13; cnt_machine < 14; cnt_machine++))
4  do
5  ssh-copy-id user@kc203-"$cnt_machine".mc.asu.ru
6  rsync -avP /home/user/Документы/Laptev
   ↪  kc203-"$cnt_machine".mc.asu.ru:/home/user/Документы
7  rsync -avP /home/user/.ssh/id_rsa.pub
   ↪  kc203-"$cnt_machine".mc.asu.ru:/home/user/.ssh/
8  # Компиляция исполняемого файла
9  mpicc.mpich -o run_icpi icpi.c
10 done
11 # Создание пустых текстовых файлов для сохранения данных для
   ↪  графиков
12 > error_graphic.txt
13 > coef_speed.txt
14 > time.txt
15 # Цикл для многократного запуска на выполнение файла для расчета
   ↪  числа Пи
16 first_number_of_intervals=3000000
17 last_number_of_intervals=59000000
```

```
18 step=28000000
19 threads=1
20 for ((thread = 1; thread <= threads; thread++))
21 do
22 for ((cnt = $first_number_of_intervals; cnt <=
   ↪  $last_number_of_intervals; cnt += step))
23 do
24 mpirun.mpich -f host.txt -n $thread ./run_icpi $cnt $thread
25 done
26 done
```