

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Самарский национальный исследовательский университет
имени академика С.П. Королева»
(Самарский университет)

Институт информатики, математики и электроники
Факультет информатики
Кафедра технической кибернетики

ЛАБОРАТОРНАЯ РАБОТА №1

Введение в параллельное программирование

по курсу
Параллельное программирование

Студент гр. 6407 _____ В.С. Гринина
Преподаватель,
к.ф.-м.н. _____ Е.С. Козлова

ЗАДАНИЕ

В данной лабораторной работе было необходимо произвести запуск программ «Hello world», которые используют технологии MPI и OpenMP, на различном количестве процессоров (потоков). В ходе анализа работы программы оценить время ее выполнения на различном количестве исполняющих нитей (процессов), для чего в заготовках произвести изменения. [1]

ВВЕДЕНИЕ

Долгий период времени IT специалисты делали упор на совершенствование вычислительной мощности компьютера. Был даже выведен закон Мура, согласно которому производительность процессоров удваивается каждые 18 месяцев.

Не так давно этот закон перестал выполняться. Следовало выбрать новую область для совершенствования скорости выполнения программ. Так появилось на свет параллельное программирование, в основе которого лежит организация вычислений таким образом, что программа разрабатывается как набор взаимодействующих вычислительных процессов, которые работают одновременно. [2]

Огромный блок задач представляет собой всевозможные матричные вычисления. Так как размер матриц часто достигает достаточно большого значения, при котором нельзя себе позволить запустить задачу на обычном персональном компьютере, параллельные вычисления в таких задачах позволяют решить ее в гораздо более короткий срок.

Данная лабораторная работа представляет собой знакомство с работой кластера, взаимодействием компьютера с кластером, а также с такими технологиями как OpenMP и MPI, которые являются стандартными для параллельных вычислений. [1] Для этих целей запустим классическую программу «Hello World» на параллельный лад.

1 Ход выполнения работы

Для удаленного доступа к командной строке сервера будем использовать протокол SSH. Для простоты будем использовать командную строку. Чтобы войти на нужный кластер исполним команду [1]:

```
ssh user01@sk.ssau.ru
```

Для синхронизации файлов между компьютером и сервером используем протокол SSHFS, который позволяет примонтировать удаленную директорию. Редактировать код будем в SublimeText.

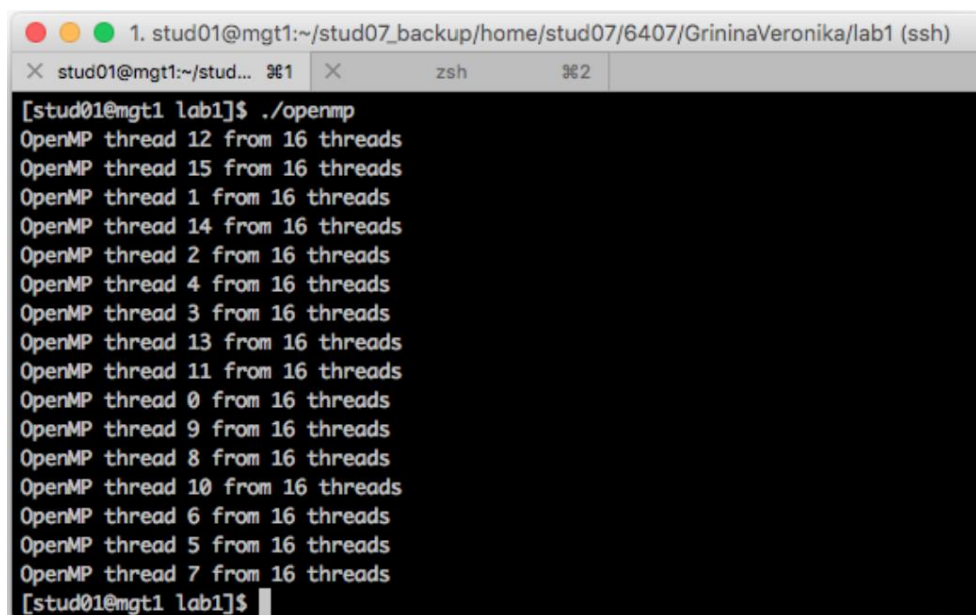
1.1 Технология OpenMP

Для того, чтобы запустить программу, использующую технологию OpenMP, прежде всего нужно подгрузить необходимый модуль и скомпилировать исходный файл программы. Компилировать будем с помощью gcc компилятора следующим образом [3]:

```
module load intel/icc16  
gcc openmp.c -o openmp -fopenmp
```

Непосредственно запуск программы производится просто запуском скомпилированного файла.

На рисунке 1 показан результат выполнения программы с использованием 16 нитей исполнения.



```
1. stud01@mgt1:~/stud07_backup/home/stud07/6407/GrininaVeronika/lab1 (ssh)  
stud01@mgt1:~/stud... %1  zsh %2  
[stud01@mgt1 lab1]$ ./openmp  
OpenMP thread 12 from 16 threads  
OpenMP thread 15 from 16 threads  
OpenMP thread 1 from 16 threads  
OpenMP thread 14 from 16 threads  
OpenMP thread 2 from 16 threads  
OpenMP thread 4 from 16 threads  
OpenMP thread 3 from 16 threads  
OpenMP thread 13 from 16 threads  
OpenMP thread 11 from 16 threads  
OpenMP thread 0 from 16 threads  
OpenMP thread 9 from 16 threads  
OpenMP thread 8 from 16 threads  
OpenMP thread 10 from 16 threads  
OpenMP thread 6 from 16 threads  
OpenMP thread 5 from 16 threads  
OpenMP thread 7 from 16 threads  
[stud01@mgt1 lab1]$
```

Рисунок 1 – Результат выполнения программы с OpenMP на 16 нитях

Из рисунка 1 видно, что процесс был запущен параллельно, так как номера нитей не упорядочены.

Код программы, использующей технологию OpenMP, приведен в приложении.

1.2 Технология MPI

Для компиляции и запуска программ с технологией MPI следует сначала загрузить необходимый модуль:

```
module load impi/4
```

Компилировать программу будем уже с помощью другого компилятора mpicc следующим образом [4]:

```
mpicc -o mpi mpi.c
```

Для запуска программы на кластере нам потребуется использовать технологию PBS Torque. Основные ее команды: qsub (для отправления задачи в очередь), qstat (для проверки состояния программы в очереди) и qdel (для удаления программы из очереди) [1].

В приложениях приведен код программы и скрипта для запуска задачи на кластере. На рисунке 2 представлены запуск программы на кластере, проверка состояния задачи и содержимое выходного файла.

```
1. stud01@mgt1:~/stud07_backup/home/stud07/6407/GrininaVeronika/lab1 (ssh)
X stud01@mgt1:~/stud... %1 X zsh %2
[stud01@mgt1 lab1]$ qsub script
233178.mgt1
[stud01@mgt1 lab1]$ qstat 233178
Job ID          Name          User          Time Use S Queue
-----
233178.mgt1     nika_helloworld stud01         00:00:00 C batch

[stud01@mgt1 lab1]$ cat nika_helloworld.o233178
===== Begin prologue script =====
===== End prologue script =====
Hello world from process 8 from total number of 16
Hello world from process 5 from total number of 16
Hello world from process 4 from total number of 16
Hello world from process 13 from total number of 16
Hello world from process 6 from total number of 16
Hello world from process 15 from total number of 16
Hello world from process 11 from total number of 16
Hello world from process 10 from total number of 16
Hello world from process 0 from total number of 16
Hello world from process 2 from total number of 16
Hello world from process 12 from total number of 16
Hello world from process 7 from total number of 16
Hello world from process 3 from total number of 16
Hello world from process 9 from total number of 16
Hello world from process 14 from total number of 16
Hello world from process 1 from total number of 16
[stud01@mgt1 lab1]$
```

Рисунок 1 — Запуск и результат выполнения программы MPI на кластере

1.3 Время выполнения программ

Для дополнительного анализа работы программ засечем время их выполнения и внесем эти данные в таблицу 1. Очевидно, что данные программы не будут давать выигрыша в скорости выполнения при увеличении количества исполняемых нитей (процессов), так как количество выполняемых операций увеличивается с увеличением нитей (процессов). Однако, можно заметить, что программа, основанная на технологии MPI работает быстрее. Возможно, это связано с тем, что OpenMP создает новые нити, в то время как MPI – новые процессы, а также с тем, что в программе с использованием технологии OpenMP был реализован вывод в консоль, в то время как программа MPI записывала результат в отдельный файл.

Таблица 1 – Время выполнения программ при различных параметрах, с

Количество нитей (процессов)	1	2	4	8	16
OpenMP	0.0000570	0.0001270	0.0007579	0.0030680	0.0177980
MPI	0.0000119	0.0000460	0.0001709	0.0002439	0.0004458

ЗАКЛЮЧЕНИЕ

В ходе проведения данной лабораторной работы было установлено соединение с кластером, на нем были скомпилированы и запущены на разном количестве нитей (процессов) две программы «Hello world» с использованием технологий для параллельного программирования OpenMP и MPI. Для дополнительного анализа работ программ было засечено и проанализировано время исполнения данных программ. На основании результатов данной лабораторной работы можно сделать вывод, что технология MPI работает быстрее, чем OpenMP. Однако, технология MPI является менее простой в использовании, по сравнению с OpenMP.

Были получены базовые знания, необходимые для работы с кластером и запуска параллельных программ на нем. Было выявлено, что распараллеливание программы не всегда приводит к более быстрому ее выполнению.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Козлова Е.С. Методические указания к лабораторной работе №1 по курсу «Параллельное программирование» Методические указания к лабораторным работам [Текст]/ Сост. Козлова Е.С. – Самара, 2017. 12 с.

[2] Параллельные вычисления [Электронный ресурс] // Википедия: свободная энцикл. – Электрон. дан. – [Б. м.], 2017. – URL: https://en.wikipedia.org/wiki/Concurrent_computing (дата обращения: 10.09.2018).

[3] Справочное руководство по OpenMP [Электронный ресурс] // OpenMP: – Электрон. дан. – [Б. м.], 2015. – URL: <https://www.openmp.org/resources/refguides/> (дата обращения: 08.10.2018).

[4] Справочное руководство в примерах по MPI [Электронный ресурс] // MPI Tutorials: – Электрон. дан. – [Б. м.], 2018. – URL: <http://mpitutorial.com/tutorials/> (дата обращения: 10.10.2018).

[5] Страуструп, Б. Язык программирования C++ [Текст]/ / Б. Страуструп. – М:Бином, 2011 –1136 с

ПРИЛОЖЕНИЕ А

КОД ПРОГРАММЫ OPENMP

```
#include <math.h>
#include <omp.h>
#include <time.h>
#include <stdlib.h>
#include <locale.h>
#include <stdio.h>

int main(int argc, char* argv[]) {
    omp_set_num_threads(16);
    int nTheads, theadNum;
    #pragma omp parallel private(nTheads, theadNum)
    {
        nTheads = omp_get_num_threads();
        theadNum = omp_get_thread_num();
        printf("OpenMP thread %d from %d threads \n",
theadNum, nTheads);
    }
    return 0;
}
```

ПРИЛОЖЕНИЕ Б

КОД ПРОГРАММЫ MPI

```
#include "mpi.h"
#include "stdio.h"
#include <time.h>
#include <math.h>
#include <omp.h>

int main(int argc, char* argv[]) {
    int rank, ranksize, i;
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &ranksize);
    printf("Hello world from process %d from total number of
%d\n", rank, ranksize);
    MPI_Finalize();
    return 0;
}
```

ПРИЛОЖЕНИЕ В

КОД СКРИПТА ЗАПУСКА MPI

```
#!/bin/bash
#PBS -N MPI_mapo_8
#PBS -l walltime=00:01:10
#PBS -l nodes=1:ppn=8
#PBS -j oe
#PBS -A tk

cd $PBS_O_WORKDIR

module load impi/4
export I_MPI_DEVICE=rdma
export I_MPI_DEBUG=0
export I_MPI_FALLBACK_DEVICE=disable
mpirun -r ssh -machinefile $PBS_NODEFILE -np $PBS_NP ./MPI
```