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 _____ А.С. Анурин
Преподаватель,
к.ф.-м.н. _____ Е.С. Козлова

САМАРА 2018

ЗАДАНИЕ

В данной лабораторной работе было необходимо произвести запуск пробной программы, которая выводит «Hello world» на суперкомпьютерном кластере «Сергей Королев». Доступ к серверу необходимо было получить по протоколу SSH. Пробная программа должна быть реализована с использованием технологий OpenMP и MPI и запущена на разном количестве потоков [1].

ВВЕДЕНИЕ

Некоторое время назад компьютерная индустрия окончательно осознала провал закона Мура и сместила фокус в наращивании вычислительных мощностей с роста производительности одного ядра на рост количества самих ядер. Этот факт обязал разработчиков программного обеспечения соответственно адаптировать свои продукты, что сильно повысило актуальность такой области информатики, как параллельные алгоритмы. На данный момент большинство научных задач, на выполнение которых требуются значимые вычислительные мощности, реализуются с использованием параллелизма.

Данная работа предполагает знакомство с двумя ключевыми технологиями в этой сфере: OpenMP и MPI, и с интерфейсом взаимодействия с кластером «Сергей Королев.»

1 Ход выполнения работы

Доступ к кластеру был осуществлен по протоколу SSH с использованием пользовательского аккаунта, предоставленного преподавателем. Обмен файлами был осуществлен с помощью технологии SSHFS [2]. Благодаря монтированию удаленной файловой системы, редактирование исходного кода может проводиться в комфорте локальной системы.

1.1 Технология OpenMP

Основной единицей параллельного исполнения в технологии OpenMP является нить исполнения (thread) [3]. В данной работе нам необходимо запустить исполнение на 16 нитях, каждая из которых должна вывести свой номер. Компиляция проводилась стандартным компилятором gcc с директивой -fopenmp. В приложениях приведен код самой программы и скрипта запуска.

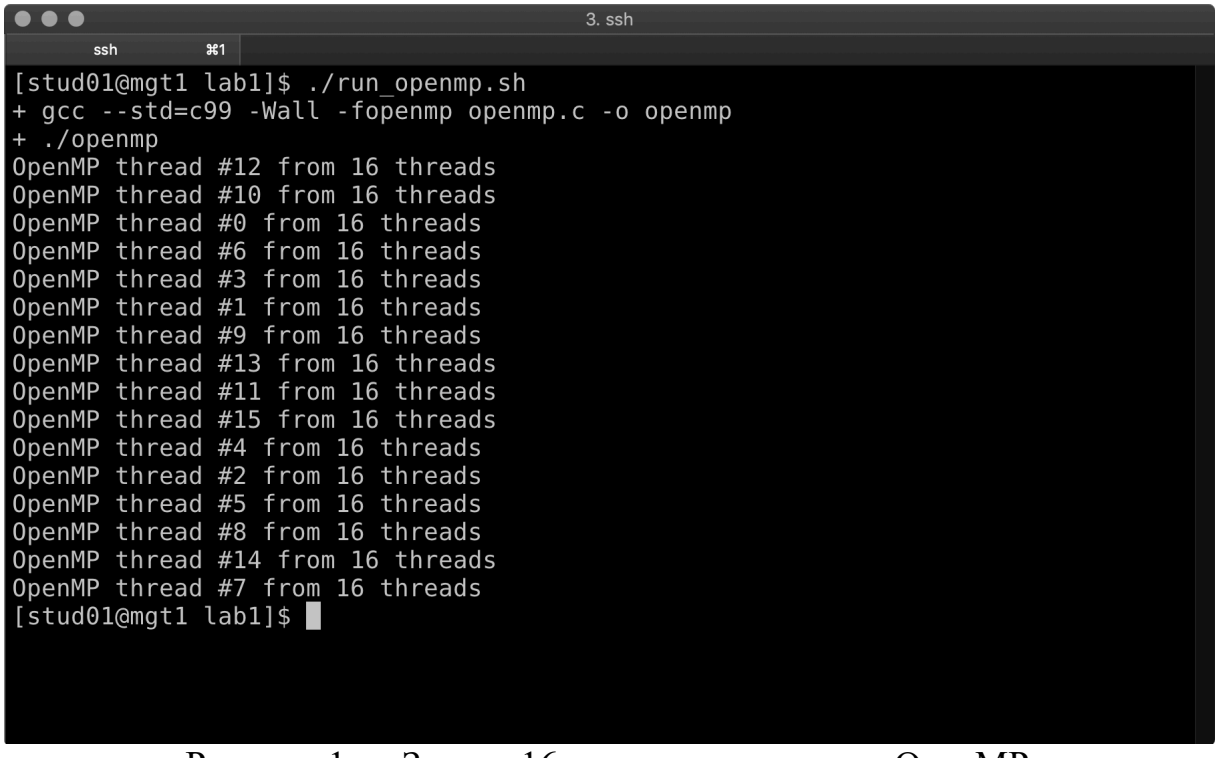

Рисунок 1 — Запуск 16 потоков с помощью OpenMP

Рисунок 1 иллюстрирует процесс запуска и исполнения программы, основанной на технологии OpenMP. Видно, что нити запускаются

неупорядоченно, что обосновывается тем, что OpenMP не предоставляет гарантий на порядок исполнения нитей.

1.2 Технология MPI

Несколько менее тривиальной по сравнению с OpenMP является технология MPI. Нам придется воспользоваться другим компилятором (mpicc) и технологией Torque для того, чтобы послать нашу задачу на исполнение. В этом нам помогут утилиты qsub и qstat.

Немного большая запутанность процесса запуска по сравнению с OpenMP связана с тем, что основным примитивом исполнения MPI является процесс [4]. Более того, процессы MPI могут быть запущены более чем на одном узле, что предполагает наличие специализированной среды синхронизации и коммуникации процессов по сети.

В приложениях приведен код программы, скрипта посылки и скрипта запуска для Torque. Рисунок 2 иллюстрирует процесс посылки задачи в систему исполнения Torque.

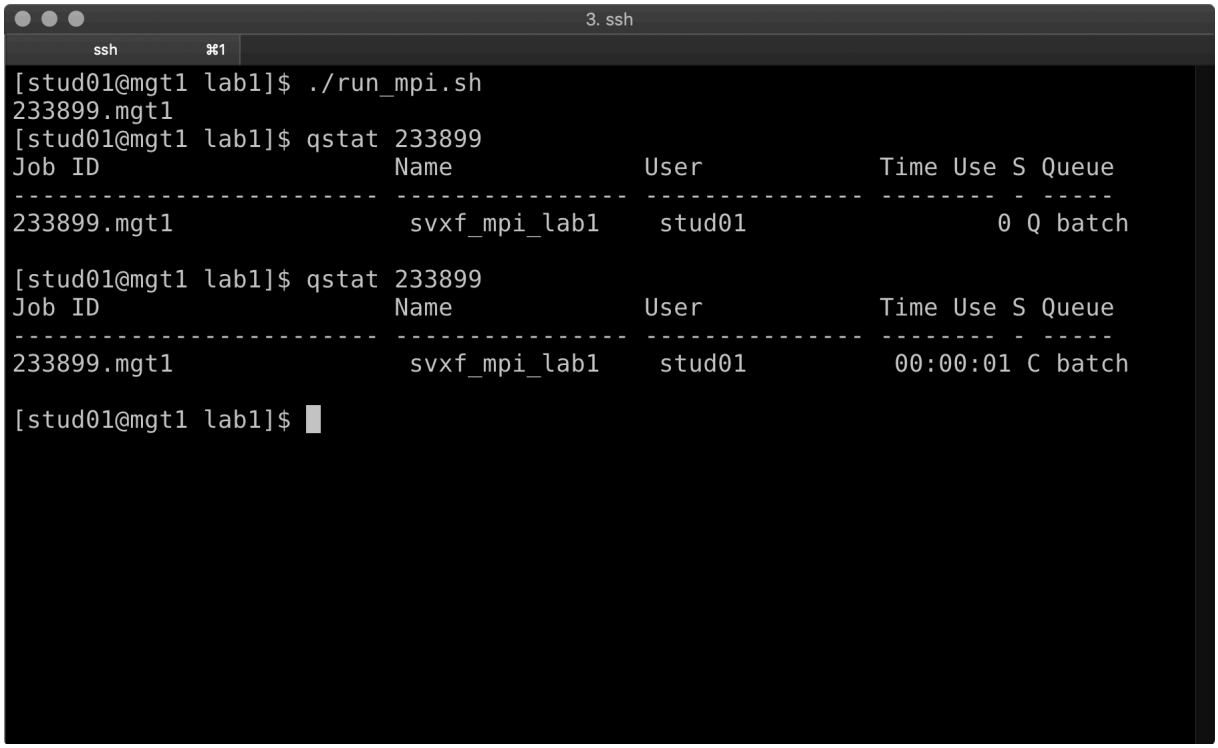

Рисунок 2 — Посылка программы MPI в систему исполнения Torque

Проведя какое-то время в очереди, наша задача выполнится на узле, соответствующем требованиям к ресурсам, которые мы указали. По завершению исполнения мы получим файл с выводом программы. Рисунок 3 показывает результат работы нашей программы.

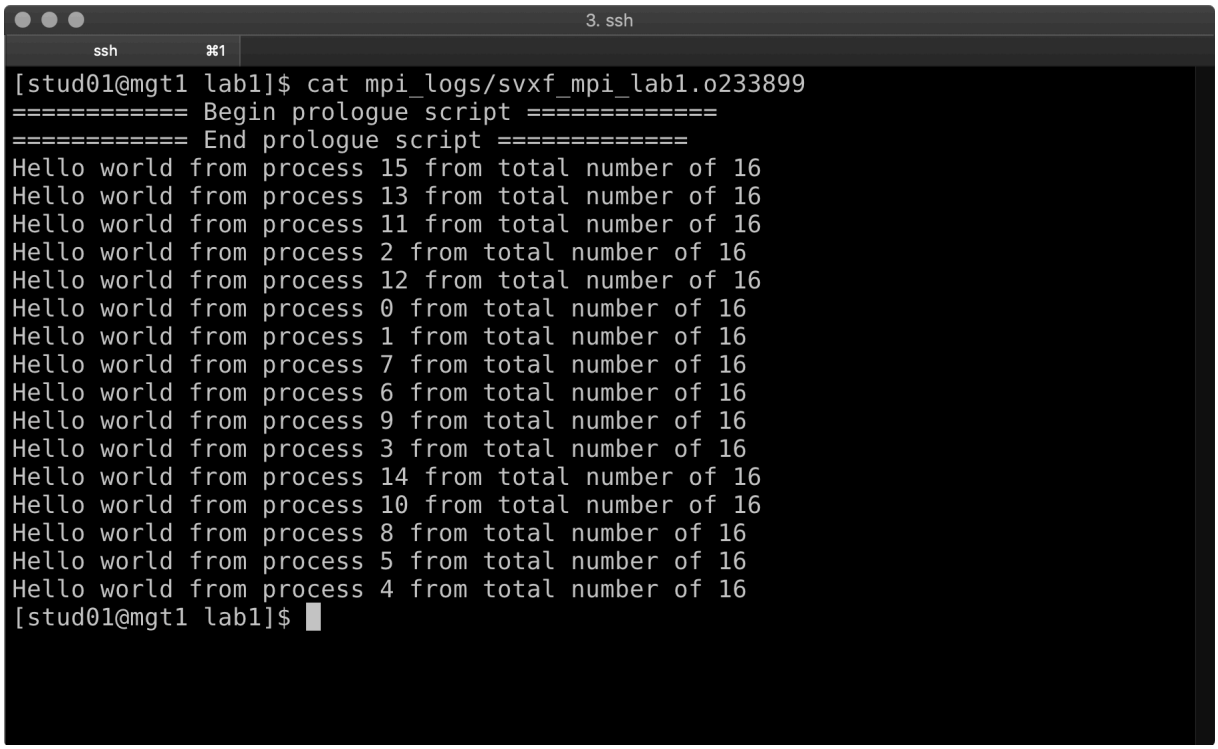

Рисунок 3 — Результат исполнения программы MPI

Можно заметить, что как и в случае OpenMP, MPI не дает гарантий на порядок исполнения процессов.

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы было осуществлено соединение с суперкомпьютерным кластером «Сергей Королев», проведено знакомство с интерфейсом взаимодействия с кластером и выполнения команд на нем. Кроме этого, были реализованы две пробные программы на технологиях параллельного программирования OpenMP и MPI. Для каждой из технологий был проведен короткий, но полный цикл разработки: написание, компиляция, развертка в среде исполнение, получение результата. Обе технологии потребовали использование специфичных инструментов (gcc для OpenMP, mpicc/qsub/qstat для MPI).

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Козлова Е.С. Методические указания к лабораторным работам. Самара: Самарский государственный аэрокосмический университет имени академика С.П. Королева (национальный исследовательский университет), 2017. 65 с.

2 libfuse/sshfs: A network filesystem client to connect to SSH servers [Электронный ресурс] // – 2018. – URL: https://github.com/libfuse/sshfs (дата обращения: 22.12.2018).

3 Home - OpenMP [Электронный ресурс] // – 2018. – URL: https:// www.openmp.org/ (дата обращения: 22.12.2018).

4. Barker B. Message passing interface (mpi). Workshop: High Performance Computing on Stampede, 2015.

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 thread_count, thread_num;

  #pragma omp parallel private(thread_count, thread_num)
  {
    thread_count = omp_get_num_threads();
    thread_num = omp_get_thread_num();
    printf("OpenMP thread #%d from %d threads \n", thread_num,
thread_count);
  }
  return 0;
}
```

ПРИЛОЖЕНИЕ Б
КОД СКРИПТА ЗАПУСКА OPENMP

```
#!/bin/bash

set -euxo pipefail

gcc --std=c99 -Wall -fopenmp openmp.c -o openmp

./openmp
```

ПРИЛОЖЕНИЕ В
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);//
  //Определяем свой номер в группе:
  MPI_Comm_rank(MPI_COMM_WORLD, &rank); //
  //Определяем размер группы:
  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 Г
КОД СКРИПТА ПОСЫЛКИ MPI

```
#!/bin/bash

set -euo pipefail

module load impi/4

mpicc --std=c99 mpi.c -o mpi

export NODES=1
export PPN=16

qsub \
      -V \
      -l "nodes=${NODES}:ppn=${PPN}" \
      -l walltime=00:00:20 \
      -N "svxf_mpi_lab1" \
      -j oe \
      -A tk \
      -o ./mpi_logs \
      mpi_qsub_script.sh
```

ПРИЛОЖЕНИЕ Д
КОД СКРИПТА ЗАПУСКА MPI

```
#!/bin/bash

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
```