

Министерство образования и науки Российской Федерации
Санкт-Петербургский политехнический университет Петра Великого

—
Институт компьютерных наук и технологий
Кафедра «Информационная безопасность компьютерных систем»

Лабораторная работа №4

«Вычислительные кластеры»

по дисциплине «Операционные системы»

Выполнил
студент гр. 23508/4

Проценко Е.Г.

<подпись>

преподаватель

Резединова Е. Ю.

<подпись>

Санкт-Петербург
2016

ФОРМУЛИРОВКА ЗАДАНИЯ


Цель работы — исследование параллельных вычислений, изучение использования MPI и работы с вычислительными кластерами LAM/MPI и MPICH.


Вариант 18 – вычисление транспонированной матрицы.


РЕЗУЛЬТАТЫ РАБОТЫ

Созданы и объединены в сеть следующие системы:

ОС	IP-адрес
Ubuntu 14	192.168.100.100
Ubuntu 14	192.168.100.101
Ubuntu 14	192.168.100.102
Ubuntu 14	192.168.100.103

 **Xubuntu_Masta** (ssh_nfs_mpich2_done)
➡ Работает

 **Xubuntu_Slave1** (ssh_nfs_mpich2_done)
➡ Работает

 **Xubuntu_Slave2** (ssh_nfs_mpich2_done)
➡ Работает

 **Xubuntu_Slave3** (ssh_nfs_mpich2_done)
➡ Работает

Для взаимодействия были подключены SSH и NFS-сервер, настроена синхронизация папок между вычислительными узлами – для обмена информацией. Ссылки на видеоматериалы по которым настраивались SSH, NFS, MPICH в приложении.

Написана на языке C программа, вычисляющая транспонированную матрицу. Исходный код находится в приложении.

```
japroc@Proland:~$ sudo pico /home/japroc/tp_matrix.c
japroc@Proland:~$ mpicc /home/japroc/tp_matrix.c -o /home/japroc/tp_matrix
japroc@Proland:~$ sudo cp /home/japroc/tp_matrix /mirror/
japroc@Proland:~$ mpiexec -n 4 -f /mirror/hosts /mirror/tp_matrix
0 4 8 12
1 5 9 13
2 6 10 14
3 7 11 15
```

Тем временем LAM/MPI – является одно из вариаций MPICH. Исключением является то, что файл hosts настраивается заранее, поэтому этот файл не вызывается как параметр при запуске mpirun. Вызов программы:

```
japroc@Proland:/$ mpirun -np 3 '/mirror/matrix'|
```

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Кластер может уменьшать быстродействие работы программы в том случае, если время передачи между узлами информации, необходимой для проведения одной итерации, становится большим по сравнению со временем счета одной итерации. Данный факт возможен при слишком малом объеме исходных данных.

2. MPI (интерфейс передачи сообщений)— это стандарт на программный инструментарий для обеспечения связи между отдельными процессами параллельной задачи. MPI предоставляет программисту единый механизм взаимодействия процессов внутри параллельно исполняемой задачи независимо от машинной архитектуры, взаимного расположения процессов и API операционной системы.

3. Назначение функции MPI_Comm_rank(MPI_Comm comm, int* rank) заключается в определении номера процесса rank в группе comm.

4. Парадигма программирования Single Program - Multiple Data заключается в том, что одна программа должна описывать поведение для всех процессов текущего кластера.

5. Блокирующие функции используются для синхронного завершения текущей операции всеми процессами.

ВЫВОДЫ

В ходе работы были изучены принципы распределения вычислений на нескольких машинах. Это необходимо для вычислений, требующих больших объемов вычислительных ресурсов. Реализована программа, высчитывающая степень матрицы. Причем, чем больше узлов включено в работу, тем эффективнее будет решаться задача. Для этого необходимо грамотно и равномерно распределить рабочие данные между узлами кластера. Эффект от количества узлов объясняется тем, что несколько машин работают над разными участками матрицы параллельно.

ПРИЛОЖЕНИЕ

Исходный код:

```
#include <stdio.h>
#include "mpi.h"
#define N 4

int main(int argc, char **argv)
{
    int rank;
    int size;
    int i, j;
    int matrix[N][N];
    int tp_matrix[N][N];
    for (i = 0; i < N; i++)
    {
        for (j = 0; j < N; j++)
        {
            matrix[i][j] = i * N + j;
        }
    }

    MPI_Init(0, 0);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    for (i = 0; i < N; i++)
    {
        tp_matrix[rank][i] = matrix[i][rank];
    }
    printf("%d %d %d %d\n", tp_matrix[rank][0], tp_matrix[rank][1],
        tp_matrix[rank][2], tp_matrix[rank][3]);

    MPI_Finalize();
    return 0;
}
```

Ссылки на видеоматериалы:

SSH - <https://www.youtube.com/watch?v=0jQrhBplCBY>

NFS - <https://www.youtube.com/watch?v=2sr3hN-qCVU>

MPICH - <https://www.youtube.com/watch?v=nmyrBKrXlnE>