Laboratory for Parallel Programming
Prof. Dr. Felix Wolf

TECHNISCHE UNIVERSITÄT DARMSTADT

Aamer Shah, shah@cs.tu-darmstadt.de

# Large-Scale Parallel Computing (WS 15/16)

# Exercise 3

A solution will be presented on November 24th, 2015. Attendance is optional. The exercise solutions will not be corrected and graded.

## Task 1

In this task, we will look at an MPI program and see how it scales when using multiple processors. We will also look at how to perform basic optimization of parallel programs.
The exercise has an archive attached, which has programs that perform matrix multiplication. The program `matmul_seq.c` performs a sequential matrix multiplication. The programs

1. `matmul_mpi.c`
2. `matmul_trans.c`
3. `matmul_cont.c`

are improvements to the original sequential program. The improved programs use MPI to distribute the workload among different processes. Each program outputs the time it takes in various steps of execution. At the end it outputs the total time spent by the program.
To compile the programs, use the Makefile in the archive. There is also a sample batch script in the archive. As a first step, before running the Makefile, execute the following commands to load the required modules:

- `module load gcc`
- `module load intel`
- `module load openmpi/intel`

Now to run an MPI program, use the following command
`mpiexec -np <NP> ./program_name`
where `NP` is the number of processes desired. For further information about the MPI functions being used in the programs, kindly visit: `http://www.mpi-forum.org/docs/docs.html`
Step-wise: perform the following subtasks:

**a)** Compile and execute the sequential program on the Lichtenberg cluster. Note: execute the program on a compute node using a batch script. As this is a sequential program, there is no need for `mpiexec`.

**b)** Compile and execute the `matmul_mpi.c` program on the cluster. Use, in increasing order, `2, 4, 8, 16, 32 and 64` processes for execution. Note the time spent in different sections of the program

**c)** Make a graph to show the speedup of the `matmul_mpi` program. Use the sequential program as the baseline performance. How is the performance compared to the sequential program?

**d)** Make a graph to show the speedup of the matrix multiplication section. How is the performance compared to the sequential program?

**e)** Look at the scalability graphs of other sections of the `matmul_mpi` program. Can you identify the section hindering program scalability? How can this section be improved?

**f)** Look at the `matmul_trans.c` program and see how it improves the hindering program section.

**g)** Compile and execute the `matmul_trans` program on `2, 4, 8, 16, 32 and 64` number of processes. Note the time spent in different sections of the program.

**h)** Make a graph to show the speedup of the `matmul_trans` program. How is the performance compared to the previous two programs?

**i)** Make a graph to show the speedup of the matrix multiplication section. How is the performance compared to the previous two programs?

**j)** Look at the scalability graphs of other sections of the `matmul_trans` program. Can the program still be improved?

**k)** Compile and execute the `matmul_cont.c` program on `2, 4, 8, 16, 32 and 64` number of processes. Note the time spent in different sections of the program.

**l)** Does the performance of the program improve? What improvements have been made in the program compared to `matmul_trans.c` program?