Heidelberg University
Institute of Computer Engineering
Computer Engineering Group

Holger Fröning
Bálint Soproni

Introduction to High-Performance and Distributed Computing
Winter Term 2023/2024

Exercise 3

- **Hand in via Moodle until 09:00 on Monday 20 November, 2023**
- **Include all names on the top sheet. Hand in a single PDF.**
- **Please compress your results into a single archive (`.zip` or `.tar.gz`).**
- **Please employ the following naming convention `hpc<XX>_ex<N>.{zip,tar.gz}`, where `XX` is your group number, and `N` is the number of the current exercise.**
- **A maximum of three students is allowed to collaborate on the exercises.**
- **In case an exercise requires programming:**
  - **include clean and documented code**
  - **include a Makefile for compiling**

3.1 Reading

Read the following paper and provide reviews as explained in the first lecture (see slides):

- Shekhar Borkar and Andrew A. Chien. 2011. The future of microprocessors. Commun. ACM 54, 5 (May 2011), 67-77.

(10 points)

3.2 Matrix multiply – sequential version

Implement a naïve – i.e. non-optimized – sequential version of the matrix multiply operation. Multiply two double-precision floating point matrices. Initialize the matrices using random values. Use appropriate time measurement functions like gettimeofday() or clock() to measure the execution of the multiply operation itself (i.e., without initialization or output).
Execute the program on one idle node. Report the execution time and the achieved GFLOP/s for a matrix multiply of the size 2048x2048 elements.
Explain the huge gap between achieved GFLOP/s and theoretical peak GFLOP/s, in particular which subsystem of this computer is the bottleneck for the execution of this program.
Optimize your program and overcome the locality problem of this program, explain and implement your idea. Execute the improved version, measure again execution time and achieved GFLOP/s.

(25 points)

3.3 Matrix multiply – parallel version using MPI

- Goal is to develop a parallel program performing a matrix multiply operation of the form $C = A \cdot B$. Matrices can be assumed to be square.
- Start with the sequential version implemented in the previous exercise. It is up to you to use the non-optimized or optimized version. Implement an MPI-based parallel version of this program. Ensure the following:
  - Dynamic allocation of all matrices
  - Verify results of parallel execution with sequential execution
- Include instrumentation for time measurement (*MPI_Wtime()*), but do not include initialization in this measurement. The initialization has to be done solely by one master process which then distributes the input data. (no need to distribute C, assume this to be initially 0)

C[i,j]	C[i,0]	C[i,1]	C[i,2]	C[i,3]	C[i,4]
C[0,j]					
C[1,j]					
C[2,j]					
C[3,j]					

- Initialize according to: $A[i,j] = i + j, B[i,j] = i \cdot j$
- Execute with two (worker) processes and report C for an input of 5x5.

(25 points)

3.4 Matrix multiply – scaling process count

- Execute on octane nodes and report execution times and speed-up for 2..16 processes (increment of 2) operating on 2k x 2k matrices. Compare with your sequential (optimized) implementation. Report speed-up and efficiency. Interpret these results!

Processes	Time [s]	Speed-Up	Efficiency
Sequential		1.0	100%
2			
4			
...			
16			

(20 points)

3.5 Matrix multiply – scaling problem size

- Execute on octane nodes with 10 and 16 processes. Vary the problem size accordingly, calculate resulting GFLOP/s and interpret!

Problem size	Time [s]	FLOPS [M/G]	GFLOP/s
128			
256			
512			
1024			
2048			
4096			
8192			

Notes

- Ensure that the computing nodes are quiescent when performing experiments. In particular, look for other participants which could be performing tests. Try not to disturb them; you also might not want to be disturbed. It's strongly recommended to use Slurm!
- Add graphics where helpful for improved interpretation! Often trends are much clearer visible when data is reported graphically.

(20 points)

Holger Fröning
Bálint Soproni

3.6 Willingness to present

Please declare whether you are willing to present any of the previous exercises.
The declaration can be made on a per-exercise basis. Each declaration corresponds to 50% of the exercise points. You can only declare your willingness to present exercises for which you also hand in a solution. If no willingness to present is declared you may still be required to present an exercise for which your group has handed in a solution. This may happen if as example nobody else has declared their willingness to present.

- Reading
- Matrix multiply – sequential version
- Matrix multiply – parallel version using MPI
- Matrix multiply – scaling process count
- Matrix multiply – scaling problem size

(50 points)

Total: 150 points