Universität Heidelberg
Institut für Technische Informatik

Holger Fröning
Benjamin Baumann

Introduction to High Performance Computing
Term 2014/2015 (Winter)

Exercise 4

- **Return electronically until Tuesday, 17.11.2014 23:55 Uhr**
- **Include name on the top sheet. Stich several sheets together.**
- **A maximum of two students is allowed to work jointly on the exercises.**

4.1 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.

4.2 Matrix multiply – parallel version using MPI

- Goal is to develop a parallel program performing a matrix multiply operation of the form C = A * 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)
  - Initialize according to: A[i,j] = i+j, B[i,j] = i*j
- Execute with 2 (worker) processes and report C for an input of 5x5.

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

4.3 Matrix multiply – scaling process count

- Execute on creek[01-08] 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!

NP	Time [s]	Speed-up	Efficiency
Sequential		1.0	100%
2			
4			
6			
8			
..			
16			

4.4 Matrix multiply – scaling problem size

- Execute on creek[01-08] with 16 processes. Vary the problem size accordingly, calculate resulting GFLOP/s and interpret!

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

Last 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.
- Add graphics where helpful for improved interpretation! Often trends are much clearer visible when data is reported graphically.

(15+40+20+20 points)

Total: 95 points